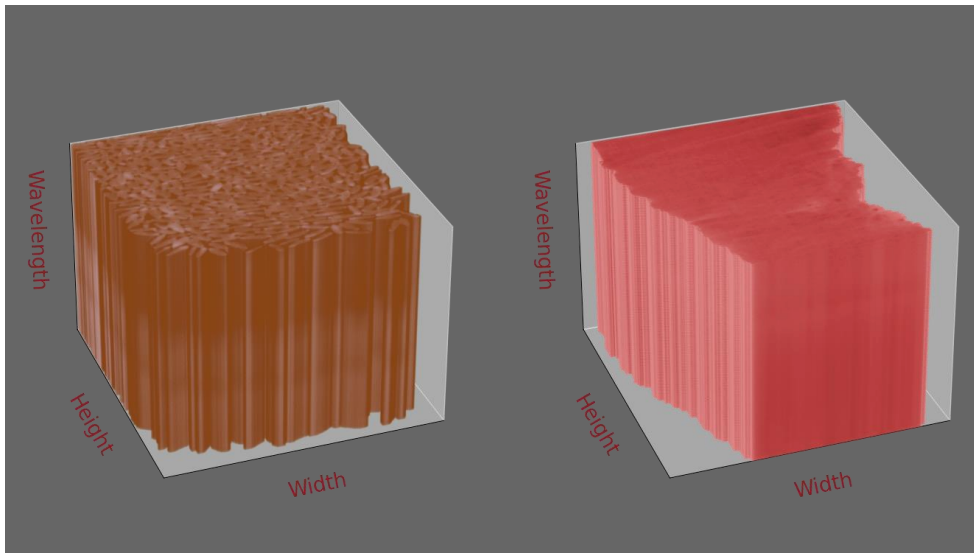




PhD thesis

Near-Infrared Hyperspectral Imaging Applications in Food Analysis

Improving Algorithms and Methodologies

Ole-Christian Galbo Engstrøm

Principal supervisor: Kim Steenstrup Pedersen

Submitted: May 31st, 2025

This thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen

Author

Ole-Christian Galbo Engstrøm, PhD Student, MSc

FOSS Analytical A/S, Denmark

Department of Computer Science, University of Copenhagen, Denmark

Department of Food Science, University of Copenhagen, Denmark

Principal supervisor

Kim Steenstrup Pedersen, Professor, PhD

Department of Computer Science, University of Copenhagen, Denmark

Natural History Museum of Denmark, University of Copenhagen, Denmark

Industrial supervisor

Erik Schou Dreier, Senior Scientist, PhD

FOSS Analytical A/S, Denmark

Academic co-supervisor

Birthe Møller Jespersen, Docent, PhD

UCL University College, Denmark¹

Industrial co-supervisor

Toke Lund-Hansen, Director, PhD

FOSS Analytical A/S, Denmark

¹During the first year of this PhD project, Birthe Møller Jespersen was an associate professor at The Department of Food Science, University of Copenhagen, Denmark.

Acknowledgments

Kim, we first met when I took your course, Robot Lab, as an undergraduate student interested in image analysis and computer vision. I enjoyed your teaching style and personality, so I contacted you again while searching for a supervisor for my bachelor's project. I wanted to do something with deep learning and image analysis. We decided to do a project about implementing the YOLO object detector and using it on the Cityscapes dataset. I had no experience with deep learning, but thanks to your mentorship, I learned how to mathematically model the object detection task. When the time came for my master's project, I knew I wanted you as my supervisor again. Once again, I learned a great deal through your mentorship. This time, we built something new, and based on that work, we wrote an article together. It became my first publication and laid the foundation for my PhD project, where you again agreed to supervise me. Kim, I am extremely grateful for everything you have taught me. It has been incredibly fun and educational. Thank you.

Erik, we met when I was looking for a master's project with Kim as a supervisor. Kim was involved in about ten potential projects, and about half of them were yours concerning your postdoctoral research. You had one project about so-called hyperspectral images, which I had never heard about. It sounded pretty cool, and you seemed both friendly and intelligent. So, I decided to work on your project. It was the single best choice I have ever made. Because of our collaboration, so many doors have opened for me, and this PhD project is a direct consequence. Working with you has been an incredible educational, professional, and personal experience. I can not put a price on your mentorship. Thank you.

Birthe, you have taught me a great deal about food science. You have fantastic knowledge about grain, and I have benefited particularly from your insights into the barley germination process. I have always found our conversations uplifting, encouraging, and insightful. This project would not have been possible without you or your knowledge. I am very grateful that you agreed to supervise me. Thank you.

Toke, you have been my manager for the duration of this project. I have always felt thankful that you let me work my odd hours and allowed me to be wherever I felt best, whether at FOSS, DIKU, or home. At the same time, you made me feel like a valuable employee of FOSS, not only by actively showing interest in my research but by inviting me to give talks about it in front of both specialists and leadership. I am pleased about the way you have acted as my manager. Thank you.

Martin, when I had written the first proof related to fast cross-validation,

you offered to proofread it. When you were done, you had come up with an alternative proof. It looked like something that had come straight out of a textbook on algorithms. Immediately, I knew that your expertise could take the work to another level. I am grateful you agreed to work on the project with me, and I am delighted about how it turned out. Thank you.

Puneet and Yamine, you invited me to stay with you at WUR. I am grateful for the hospitality you showed me and the work we did together. It was a great experience. Thank you.

Maria and Michela, thank you for your help with the work on chemical maps. Your insights significantly improved the quality of the work. I am happy that I got to work with you.

I want to thank Thomas Nikolajsen for taking a chance on me by employing me at FOSS Analytical A/S to write a grant proposal for The Innovation Fund Denmark to get funding for this project. At the same time, I want to thank Nicolai Bork for helping me write the grant proposal and Tommy Bysted for signing off on it on behalf of FOSS.

I also want to thank my employer, FOSS Analytical A/S, and The Innovation Fund Denmark for funding this project.

Thanks to all my friends and colleagues at FOSS and DIKU for scientific discussion and social gatherings and for welcoming me into your communities. You made this journey a joyful experience.

Thanks to Uncle Bo for insightful and cozy mathematical sparring sessions.

Thanks to Dad, Mom, Micky, Mads, and Niels for your unwavering support.

Abstract - Dansk

Denne afhandling undersøger anvendelsen af nær-infrarød hyperspektral billeddannelse (NIR-HSI) til analyse af fødevarekvalitet. Undersøgelsen gennemføres gennem fire studier, der arbejder med fem forskningshypoteser. Studierne sammenligner modeller baseret på såkaldte convolutional neural networks (CNN'er) og såkaldt partial least squares (PLS) til en række analyser.

Generelt overgår fælles spatio-spektral analyse med CNN'er både rumlig analyse med CNN'er og spektral analyse med PLS, når der modelleres parametre, hvor kemisk og fysisk visuel information er relevant. Ved modellering af kemiske parametre med et todimensionelt (2D) CNN forbedres den prædiktive ydeevne væsentligt, hvis CNN'et udvides med et indledende lag dedikeret til spektral foldning. Dette lag lærer en spektral forbehandling svarende til den, som kemometrikere anvender. PLS-baseret spektral modellering klarer sig dog lige så godt ved analyse af gennemsnitsindholdet af kemiske parametre i fødevarer og anbefales derfor som tilgang.

Grundet de høje omkostninger forbundet med indsamling af referenceværdier for enkelte fødevarer undersøger afhandlingen, om det er muligt at træne modeller med gennemsnitlige referenceværdier fra mængder af fødevarer sammen med et stort antal NIR-HSI-billedudsnit fra disse mængder. Modeleringsresultaterne var præcise for både klassifikation af homogene prøver og regression af heterogene prøver. I sidstnævnte tilfælde skal gennemsnitsprædiktionerne dog justeres additivt og multiplikativt — dette gælder både CNN'er og PLS-baserede modeller.

Modelering af den rumlige fordeling af kemiske parametre med NIR-HSI er typisk begrænset af mulighederne for at anskaffe rumligt fordelte referenceværdier. Derfor anvendte et studie gennemsnitlige mængdereferenceværdier til at generere kemiske kort over fedtindhold i svinemaver. En PLS-baseret tilgang resulterede i betydelige problemer med ujævne kemiske kort og pixelvise prædiktioner uden for det fysisk mulige område mellem 0–100% fedtindhold. Omvendt lykkedes det et 2D-CNN, udvidet med det førnævnte lag til spektral foldning, at generere kemiske kort direkte fra det originale NIR-HSI-billede og afhjælpe de problemer, der opstod med PLS.

Det sidste studie forsøgte at modellere bygs spiringsevne ved at analysere NIR-spektre, RGB-billeder og NIR-HSI-billeder. Resultaterne var dog inkonklusive grundet datasættets lave spiringsgrad.

Derudover har denne afhandling ført til udviklingen af to open source Python-pakker. Den første muliggør hurtig PLS-baseret modellering, mens den anden muliggør ekstremt hurtig krydsvalidering af PLS og andre klassiske maskinlæringsmodeller baseret på et nyt banebrydende algoritmisk resultat.

Abstract - English

THIS thesis investigates the application of near-infrared hyperspectral imaging (NIR-HSI) for food quality analysis. The investigation is conducted through four studies operating with five research hypotheses. For several analyses, the studies compare models based on convolutional neural networks (CNNs) and partial least squares (PLS).

Generally, joint spatio-spectral analysis with CNNs outperforms spatial analysis with CNNs and spectral analysis with PLS when modeling parameters where chemical and physical visual information are relevant. When modeling chemical parameters with a 2-dimensional (2D) CNN, augmenting the CNN with an initial layer dedicated to performing spectral convolution significantly enhances its predictive performance by learning a spectral preprocessing similar to that applied by chemometric domain experts. Still, PLS-based spectral modeling performs equally well for analysis of the mean content of chemical parameters in food samples and is the recommended approach.

Due to the expensive nature of obtaining reference values for individual food samples, this thesis investigates the feasibility of training models with bulk mean reference values paired with large quantities of NIR-HSI image crops from said bulk sample. The results were accurate for both the classification of homogeneous samples and the regression of heterogeneous samples. In the latter case, however, a bias and scale correction must be applied to the bulk-wise mean predictions of image crops for both PLS- and CNN-based models.

Modeling the spatial distribution of chemical parameters with NIR-HSI is typically limited by the ability to obtain spatially resolved reference values. Therefore, a study used bulk mean references for chemical map generation of fat content in pork bellies. A PLS-based approach yielded significant issues with non-smooth chemical maps and pixel-wise predictions outside the physically possible range of 0-100%. Conversely, a 2D CNN augmented with the previously mentioned spectral convolution layer successfully generated chemical maps directly from the input NIR-HSI image and mitigated the issues arising with PLS.

The final study attempted to model barley’s germinative capacity by analyzing NIR spectra, RGB images, and NIR-HSI images. However, the results were inconclusive due to the dataset’s low degree of germination.

Additionally, this thesis has led to the development of two open-sourced Python packages. The first facilitates fast PLS-based modeling, while the second facilitates *extremely* fast cross-validation of PLS and other classical machine learning models based on a new groundbreaking algorithmic result.

Preface

THIS thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen. It is structured as a synopsis of published works, including three peer-reviewed articles, one manuscript currently under review, two non-peer-reviewed technical reports, two software packages, and one dataset. It also contains a chapter describing a previously unpublished study. Table 1 summarizes the works. Additionally, in continuation of Ole-Christian Galbo Engstrøm’s master’s project, he, Erik Schou Dreier, and Kim Steenstrup Pedersen conducted some preliminary work on adapting convolutional neural networks (CNNs) for protein content regression analysis on hyperspectral images of wheat grain kernels (Engstrøm et al. 2021). That work contains relevant context to this PhD thesis but is not part thereof.

Work	Type	Peer reviewed	Year
Engstrøm et al., 2023b	Conference workshop	Yes	2023
Engstrøm et al., 2024	Journal	Yes	2024
Engstrøm and Jensen, 2025	Journal	Yes	2025
Engstrøm et al., 2025a	Journal	Under review	2025
Engstrøm et al., 2023a	Technical report	No	2023
Engstrøm et al., 2025b	Technical report	No	2025
Barley Germination	Chapter in this thesis	No	2025
ikpls	Software package	Yes	2024
cvmatrix	Software package	No	2024
Barley Germination Dataset	Dataset	No	2025

Table 1: Works included in this PhD thesis.

Ole-Christian Galbo Engstrøm is the first author and primary contributor to all the works included in this thesis. While the research hypotheses were developed with supervisors and co-authors, Ole-Christian Galbo Engstrøm carried out most of the software development, experimental work, data analysis, and writing.

Contents

Contents	9
1 Introduction	11
I Fast Partial Least Squares	21
2 Improved Kernel Partial Least Squares	23
3 Fast Cross-Validation	33
II Near-Infrared Hyperspectral Image Analysis	51
4 Convolutional Neural Networks for Near-Infrared Hyperspectral Image Analysis	53
5 Bulk References for Subsample Predictions	65
6 Chemical Maps	73
7 Barley Germination	99
III Final Remarks	111
8 Discussion	113
9 Conclusion	121
Bibliography	123
IV Appendices	133
A Technical Report: Bulk References	135

B Technical Report: Barley Germination Dataset	157
---	------------

Chapter 1

Introduction

NEAR-INFRARED hyperspectral imaging (NIR-HSI) (also known as spectral imaging Polder and Gowen 2020) platforms can capture NIR-HSI images which, like regular color images, are three-dimensional cuboids with height, width, and depth. Unlike regular color images, a NIR-HSI image’s depth dimension contains an entire NIR wavelength spectrum at each spatial pixel, typically with more than 100 uniformly distributed wavelength channels. The NIR spectral dimension is beneficial for food analysis as contents of chemical quality parameters such as protein and fat correlate with light absorption¹ in the near-infrared part of the electromagnetic spectrum (Osborne 2006; Manley and Williams 2021). While the spatial dimensions enable analysis of a food sample’s physical and morphological features, their combination with the spectral dimension enables the modeling of the spatial distribution of chemical parameters. Thus, NIR-HSI is especially well-suited for food analysis (Amigo et al. 2013; Ishikawa et al. 2021; Medina–García et al. 2025).

A NIR-HSI platform enables rapid image acquisition. When coupled with, for example, a segmentation or object detection model, individual elements or smaller subsamples can be extracted from the original images to generate large datasets for subsequent analysis quickly. This is particularly useful for powerful deep learning models that require much more data to train than, for example, linear regression models that are otherwise widely used in chemometrics, the multidisciplinary field concerned with the study of extracting chemically relevant information from data (S. Wold 1995). These large datasets, however, must still be annotated to enable supervised learning. In classical chemometrics, bulk samples are analyzed with spectrometers to get a mean spectrum, which is then matched with a bulk mean reference value, which is easy and cheap to obtain. Conversely, there is generally no cheap or easy way to obtain subsample reference values. Consider, for example, estimating the protein content in grain or the fat content in pork bellies. Here, it is easy to

¹The NIR spectra from a NIR-HSI image are reflectance spectra but can be transformed to pseudo-absorbance by negating their logarithm.

obtain reference values for the mean protein content in a grain batch or the fat content in an entire pork belly. Obtaining reference values for individual grains or a tiny subsection of a pork belly is a much more costly and time-consuming affair. Therefore, it is convenient if the cheap bulk mean references can be paired with the subsamples and still enable accurate modeling.

Another benefit of a NIR-HSI platform is the combination and potential synergies of spectral and spatial information. The expressive power of a spatio-spectral model should be at least the same as that of a strictly spatial or spectral model. In particular, for problems where both spatial and spectral features are relevant, a spatio-spectral model should outperform the alternatives. It may also be the case that a property exists that requires both spatial and spectral information to be modeled accurately. An NIR-HSI image contains vast amounts of data, not all of which may be relevant to a given task. Classical chemometrics relies on dimensionality reduction techniques such as principal component analysis (Pearson 1901; Hotelling 1933) or partial least squares (PLS, H. Wold 1966) in attempts to suppress noise and build models based on the essential information.

Additionally, as NIR spectra are noisy and convoluted signals, manual preprocessing is often applied to extract the underlying chemical information (Rinnan et al. 2009). Instead of using these classical methods that require manual tuning of hyperparameters and preprocessing, deep learning models should be able to simultaneously extract the relevant information directly from the NIR-HSI image and learn to reveal the chemical information hidden in the NIR spectra without any manual preprocessing. All these observations and ideas lay the foundation for the following research hypotheses.

1. NIR-HSI images can improve the modeling of properties that can already be modeled with spectral or spatial information alone.
2. Deep learning facilitates the learning of spectral preprocessing that is otherwise manually selected in chemometrics.
3. Models trained using bulk mean references for subsamples can estimate the bulk references when averaging subsample predictions.
4. Using bulk mean references, NIR-HSI images can be used to model the spatial distribution of chemical parameters throughout the images.
5. A food quality parameter exists that can be accurately modeled with neither color (RGB) imaging nor NIR spectroscopy but can be accurately modeled with NIR-HSI.

Four different projects investigate the hypotheses. The first project develops and analyzes models for grain variety classification and protein content regression in wheat. It compares end-to-end deep learning with classical chemometrics and focuses primarily on learnable preprocessing for deep

learning models. The second project builds on the first study’s results and investigates the implications of using bulk mean references for subsamples. The third project also uses bulk mean references and is concerned with modeling the spatial distribution of fat in pork bellies to generate chemical maps; those are pixel-wise fat predictions across the entire NIR-HSI image. Finally, the fourth project attempts to analyze barley’s germinative capacity ahead of the germination process. It compares models relying on RGB images, NIR spectra, and NIR-HSI images for this task.

In addition to those four projects, this thesis presents two projects on fast PLS-based modeling. The first project concerns the open-source implementation of a fast and numerically stable PLS model in Python. The second project is concerned with developing a new algorithm that can be used in unison with, among other models, PLS to significantly speed up cross-validation (Stone 1974; Hastie et al. 2009).

1.1 Modeling

As mentioned at the beginning of this introduction, the structure of NIR-HSI images is similar to that of regular color images. Therefore, the works in this thesis extensively use convolutional neural networks (CNNs) for their analyses. CNNs have, since the CNN AlexNet (Krizhevsky et al. 2012) won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC, Russakovsky et al. 2015), been the model of choice for various image analysis tasks including, but not limited to, classification and regression (LeCun et al. 2015). More recently, Vision Transformers (ViTs, Dosovitskiy et al. 2020) have gained traction for image analysis and can even surpass the performance of CNNs. However, ViTs require more data to train than CNNs. Therefore, given the relatively small datasets used in this thesis, the works will use CNNs rather than ViTs. At the same time, PLS is a primary tool for the analysis of NIR spectra of food in chemometrics (Brereton et al. 2018; Sørensen et al. 2021). Therefore, all studies in this thesis will compare CNN- and PLS-based models.

1.1.1 Partial Least Squares

A dataset of NIR spectra can be organized into a matrix $\mathbf{X} \in \mathbb{R}^{N \times K}$ where N is the number of spectra used in the analysis, and K is the number of wavelength channels in each spectrum. These can be matched with another matrix $\mathbf{Y} \in \mathbb{R}^{N \times M}$ containing the parameters of interest, where M is the number of parameters associated with each spectrum. If the values in $\mathbf{Y}$ correspond to some chemical parameters, such as, for example, protein or fat, linear regression can be used to model the relationship between absorbance NIR spectra in $\mathbf{X}$ and the chemical parameters in $\mathbf{Y}$. Generally, ordinary least squares (OLS) can be used to model such a relationship. OLS computes the

regression matrix, $\mathbf{B} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$, that minimizes the sum of squared residuals between predicted values, $\mathbf{XB}$, and actual values, $\mathbf{Y}$.

However, NIR spectra suffer from high multicollinearity in the wavelength channels (Osborne 2006). Additionally, spectrometers can measure thousands of wavelength channels, meaning datasets of NIR spectra often have more wavelengths, K , than samples, N . In both cases, $\mathbf{X}^T \mathbf{X}$ becomes rank deficient, implying it has no inverse, so OLS fails to compute $\mathbf{B}$. Several dimensionality reduction methods can mitigate this. Perhaps the most common dimensionality reduction method is principal component analysis (PCA), where OLS can then subsequently model the relationship from dimensionality-reduced $\mathbf{X}$ to $\mathbf{Y}$, a technique known as principal component regression (PCR, Hotelling 1957; Kendall 1957).

In the supervised setting, however, there is generally no guarantee that the variance retained by PCA is relevant when used to predict $\mathbf{Y}$. Here, PLS is a more well-suited alternative. PLS is a dimensionality reduction and regression method, initially developed by H. Wold (1966) with the Nonlinear Iterative Partial Least Squares (NIPALS) algorithm. Despite the name, PLS is a linear method that models a linear relationship between $\mathbf{X}$ and $\mathbf{Y}$. PLS performs a dimensionality reduction of $\mathbf{X}$ and $\mathbf{Y}$ to so-called score spaces. The score spaces are determined to maximize covariance between $\mathbf{X}$ scores and $\mathbf{Y}$ scores while keeping $\mathbf{X}$ scores orthogonal to each other. From this reduced space, a linear regression can be computed from the $\mathbf{X}$ -scores to the original $\mathbf{Y}$ -values. The corresponding regression coefficients are then projected back to the original $\mathbf{X}$ -space to express the regression directly from $\mathbf{X}$ to $\mathbf{Y}$.

PLS was initially designed for regression (PLS-R, H. Wold 1966; S. Wold et al. 2001) where $\mathbf{Y}$ contains continuous values. However, it can also be used for classification (commonly known as PLS discriminant analysis, PLS-DA, Sjöström et al. 1986; Stähle and S. Wold 1987; Barker and Rayens 2003; Brereton and Lloyd 2014) where $\mathbf{Y}$ contains discrete values. When $\mathbf{Y}$ is a vector ($M = 1$), PLS is commonly called PLS1. Otherwise, if $\mathbf{Y}$ is a matrix with $M > 1$, PLS is commonly called PLS2 (even if $M > 2$). When clear from context, this thesis will use the general term PLS instead of PLS1 and PLS2 and will not discriminate between PLS-R and PLS-DA.

1.1.2 Convolutional Neural Networks

CNNs (LeCun et al. 1989; LeCun et al. 2002; LeCun et al. 2015) are neural networks that perform discrete convolutions between the input data and kernels with learnable weights.² The convolution operation strides a typically small kernel across the input image and, at each location, computes the sum of

²In practice, cross-correlation is used in place of convolution. This is because convolution is cross-correlation with a flipped kernel. Since the kernel’s weights are learned, the flipped kernel might as well be learned directly, circumventing the need for and computational cost associated with flipping the kernel.

the element-wise multiplication between the kernel and the section of the image with which it overlaps. At each location, the result of the sum, a so-called feature, is stored at the corresponding position in the output array, a so-called feature map. Figure 1.1 shows a basic illustration of a two-dimensional (2D) convolution. Convolutions are generalized to any number of dimensions, but CNNs for image analysis typically use the 2D variant. Convolutions are linear operations, and since linearity is closed under composition, they are often combined with non-linear activation functions to facilitate learning non-linear feature maps. While this is the basic idea behind a convolution, there are many additional details when used in the context of CNNs, the explanations of which are out of scope for this thesis.

Perhaps the most popular CNN variant of all time is ResNet (K. He et al. 2016)³, an ensemble of which was also the first to surpass human performance on ILSVRC. Like many other CNNs, a ResNet is structured as a sequence of convolution layers with interleaving non-linearities forming a feature extractor. This is followed by an average pooling that transforms the extracted feature map to a vector, which is input to a final, fully connected layer that performs linear regression. Thus, a ResNet is, in essence, a convolution-based feature extractor followed by a linear regression. When used for analysis in NIR-HSI, ResNet can extract and combine features across spatial and spectral locations and subsequently perform a regression on these. Combined with an appropriate loss (objective) function, gradient descent can be used to optimize the parameters of the entire ResNet for a given task directly from the input data. This thesis extensively uses ResNet-18, the smallest ResNet, for regression and classification tasks.

1.2 Related Work

NIR-HSI has seen widespread application in food analysis research in the previous decades (Gowen et al. 2007; Amigo et al. 2013; Wieme et al. 2022; Medina-García et al. 2025). To limit the scope of this section, it focuses on work related to the five research hypotheses enumerated earlier in this chapter.

As mentioned, a NIR-HSI image can be analyzed by looking at its spectral or spatial dimensions alone or both in unison. Models trained jointly on both spatial and spectral modalities should be able to perform at least as well as models trained on either modality alone. Indeed, it seems that spatio-spectral modeling may outperform or perform as well as either spatial and spectral modeling in cases where both modalities contain signals relevant to the task (Dreier et al. 2022; Pourdarbani et al. 2023; Ebrahimi et al. 2023). Other times, however, spectral modeling outperforms joint spatio-spectral modeling (Engstrøm et al. 2021). In particular, chemical parameters may be more chal-

³At the time of writing, the ResNet article has accumulated over 275,000 citations on Google Scholar.

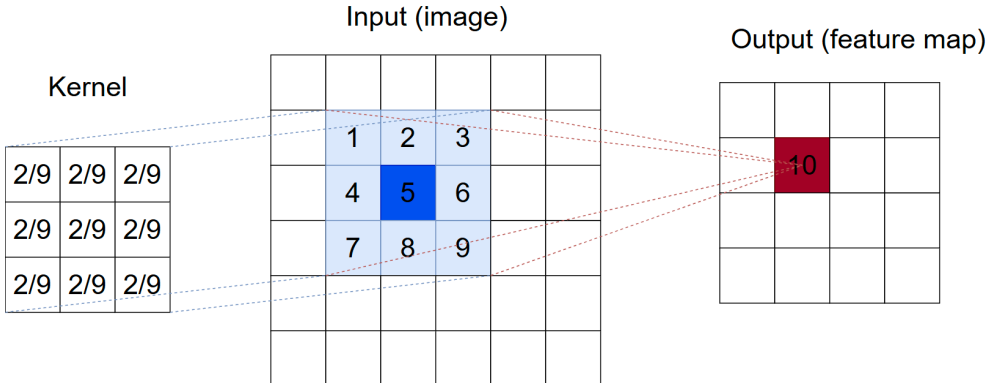


Figure 1.1: An illustration of the 2D convolution (actually cross-correlation) operation. The kernel is placed with its center on the blue 5. Element-wise multiplication follows, and the result is summed up and placed at the associated location (illustrated in red) in the output feature map. This kernel computes a feature equal to twice the average of the overlapped part of the input image.

lenging to model with joint analysis of the entire NIR-HSI images than spectra. This may seem peculiar, as all the chemical information in the spectrum is also in the NIR-HSI image. However, when using PLS to analyze chemical parameters, it is common to apply some combination of preprocessing that includes smoothing and taking derivatives of the spectral dimension. This reduces the physical effects of light scattering in the spectrum and reveals the underlying chemical information (Rinnan et al. 2009). Likewise, when training 1D CNNs for spectral analysis, such preprocessing is not necessarily applied as these can learn to mimic the spectral preprocessing in their first layer and may perform as well or better than preprocessed PLS models (Acquarelli et al. 2017; Cui and Fearn 2018; Helin et al. 2022; Mishra et al. 2022).

On the other hand, 2D and 3D CNNs are commonly used to analyze the entire NIR-HSI images (Zhu et al. 2020; Z. Su et al. 2021; Engstrøm et al. 2021; Dreier et al. 2022; Pourdarbani et al. 2023; Ebrahimi et al. 2023). While they can also learn such preprocessing, it requires tuning many more parameters than for 1D CNNs and is thus more challenging to achieve in practice. This phenomenon may contribute to joint spatio-spectral modeling sometimes being outperformed by spectral modeling for chemical analysis. Indeed, our preliminary work shows that adding a specialized spectral convolution layer prior to a 2D ResNet significantly enhances its performance on wheat protein content regression from NIR-HSI images (Engstrøm et al. 2021). We explore and improve this idea in Chapter 4.

Another caveat for CNNs is that they require more training data than simpler models such as PLS. Unfortunately, large NIR-HSI ImageNet-like datasets for food analysis are unavailable (Mishra et al. 2022). Therefore, practitioners

and researchers must often generate datasets themselves. NIR-HSI images can be divided by, for example, cropping into subsamples to inflate the amount of images in the dataset. However, these subsamples must be accompanied by reference values for supervised learning. These will often be both expensive and time-consuming to obtain. Therefore, a strategy is to take the reference value of the entire NIR-HSI image and assign that to every subsample. This approach is entirely valid for classifying homogeneous images, as the true value for the subsamples will be identical to that of the original image (Dreier et al. 2022).

On the other hand, if the reference value is a mean of a heterogeneous sample, as it will be for regression of a chemical parameter, it may not be accurate for every subsample. Still, by definition, the errors will average out when considering all subsamples. Thus, regression modeling with such a dataset may yield an accurate model (Engstrøm et al. 2021). However, as we shall see in Chapter 5, this approach introduces subtle yet significant biases in the trained model, regardless of whether it is CNN- or PLS-based.

Another use of NIR-HSI imaging is chemical map generation. As previously mentioned, chemical map generation is the modeling of a chemical parameter’s spatial distribution throughout the NIR-HSI image. Here, a model assigns a prediction to every pixel of the NIR-HSI image. For over a decade, researchers handled this task with PLS-models trained on mean spectra with associated reference values (Burger and Geladi 2006; ElMasry et al. 2013; Kamruzzaman et al. 2012; Barbin et al. 2013; Kamruzzaman et al. 2016; C. Ma et al. 2021; Florián-Huamán et al. 2022; Cruz-Tirado et al. 2023; H.-J. He et al. 2023; Adesokan et al. 2024; Albano-Gaglio et al. 2025). As deep learning has become more commonly used in chemometrics (Mishra et al. 2022), 1D CNNs have also been applied to this task (Cai et al. 2024; Shi et al. 2024; Xi et al. 2025). PLS models and 1D CNNs may seem well-suited for this task as they naturally operate on the pixel-wise level. Indeed, a 2D CNN that consumes the entire NIR-HSI image to make a single prediction, such as, for example, ResNet, can not be used to generate pixel-wise predictions. However, as we shall see in Chapter 6, not considering the context of the entire NIR-HSI may lead to significant faults in the generated chemical maps. Indeed, Barbin et al. (2013) note that their PLS model gives reasonable predictions on mean spectra but that the chemical maps generated by the same model are unsatisfactory.

Researching this topic revealed no works that generate chemical maps directly from the input NIR-HSI image and associated mean reference value without relying on 1D models for pixel-wise predictions. However, Shi et al. (2024) train a linear regression model on mean spectra for quantification of nutrients in wheat. Then, they train a generative adversarial network (GAN) to generate chemical maps. The GAN requires reference chemical maps for the training process. To the best of Ole-Christian Galbo Engstrøm’s understanding, the reference chemical maps are generated by the linear regression

model, and, as such, the approach still directly relies on the pixel-wise predictions of a 1D model. Indeed, if this interpretation is correct, then the GAN is directly optimized to predict chemical maps similar to the ones generated by their pixel-wise linear regression model and, as such, is simply a more computationally expensive path to the same result. In any case, their GAN approach requires a reference chemical map, which we circumvent in Chapter 6.

As mentioned in this section, some parameters are best modeled with joint spatio-spectral analysis of entire NIR-HSI images. Generally, however, these parameters can also be modeled fairly accurately using either spatial or spectral modalities alone. It seems challenging to find a parameter for which utilization of the entire NIR-HSI image not only improves performance but is required to get performance surpassing random guessing. Chapter 7 argues why barley’s germinative capacity may be one such parameter. That is the task of predicting if and when a given barley kernel will germinate after exposure to moisture. This task has not received much attention from the research community. Indeed, Orth et al. (2025) offers a review on the closely related task of pre-harvest germination detection in barley, where the latest study referenced is around two years old and uses mean spectra from NIR-HSI images to classify germinated and ungerminated barley kernels (Orth et al. 2023). The review mentions that both physical and chemical parameters are necessary for accurate detection. Their review shows that analysis through NIR-HSI may be a promising direction, although it is not yet established as a viable alternative to the slow and destructive standard methods. Indeed, Arngren et al. (2011) have attempted to model barley germination time. Their accuracy, however, is relatively low for individual kernels, and they must rely on bulk averages and reduced granularity in germination time intervals between classes to achieve better performance. Their study was conducted before CNNs became mainstream for image analysis, so they relied on manual feature engineering to extract both spatial and spectral features for modeling. In Chapter 7, we use CNNs to model barley’s germinative capacity.

1.3 Structure of Thesis

The remainder of this thesis is structured in parts. Part I contains work related to fast PLS modeling. Chapter 2 elaborates on the need for a fast and versatile PLS model implementation. Chapter 3 shows how it can be improved further when used in a cross-validation setting. Part II contains work related to the five research hypotheses presented earlier. Chapter 4 improves 2D ResNet’s capability to perform regression analysis on chemical parameters; Chapter 5 analyzes the results further and identifies caveats when using bulk references for subsamples in the context of regression analysis. Chapter 6 improves chemical map modeling by identifying and addressing the issues arising when generating chemical maps directly from PLS trained on

mean spectra. Chapter 7 presents a previously unpublished study regarding modeling barley's germinative capacity using both RGB and NIR-HSI image modalities. Part III contains a discussion and conclusion upon the previous two parts in the context of the research hypotheses. Finally, Part IV consists of appendices with technical reports containing additional information regarding the works in Chapters 5 and 7.

Part I

Fast Partial Least Squares

Chapter 2

Improved Kernel Partial Least Squares

PARTIAL least squares (PLS) models a linear relationship between $\mathbf{X} \in \mathbb{R}^{N \times K}$ and $\mathbf{Y} \in \mathbb{R}^{N \times M}$ and can be implemented by a plethora of algorithms. For example, $\mathbf{X}$ could be N spectra with K wavelengths, and $\mathbf{Y}$ could be the associated M reference values for each spectrum. In HSI, we are particularly interested in cases where $N > K$ as samples can rapidly be imaged. In contrast, the number of wavelength channels is constant and typically in the hundreds, thus quickly dominated by the number of samples. In chemometrics, when using spectrometers with many wavelength channels to measure smaller datasets, it may be the case that $N < K$. Although not applicable in this thesis, Chapter 8 comments on what to do in this case.

The execution time and numerical stability of the different PLS algorithms vary, as shown by Alin (2009) and Andersson (2009). While the original PLS algorithm, NIPALS (H. Wold 1966), is the most numerically stable option, it is, unfortunately, also the slowest. When $N > K$, it seems that Improved Kernel PLS (IKPLS, Dayal and MacGregor 1997) is the best choice of PLS algorithm as it is extremely fast and also highly numerically stable. In some cases, however, it can be shown to be numerically unstable for ill-conditioned $\mathbf{X}$ (Björck and Indahl 2017). IKPLS comes in two variants, Algorithm 1 and Algorithm 2. If $N \gg K$, Algorithm 2 is faster than Algorithm 1. Andersson (2009) analyzed Algorithm 1 for numerical analysis in his study. However, empirical studies have shown that both algorithms almost always give the same result, as evidenced by the test suite by Engström et al. (2024).

NIPALS, however, is the only algorithm implemented by scikit-learn (Pedregosa et al. 2011), the ubiquitous machine learning library for Python. While forcing users to use the slow NIPALS algorithm, the scikit-learn implementation has several additional inconveniences. First, there are 16 combinations of centering and scaling for $\mathbf{X}$ and $\mathbf{Y}$. scikit-learn’s implementation always enforces centering of both $\mathbf{X}$ and $\mathbf{Y}$ while allowing the user to deter-

mine whether to scale both $\mathbf{X}$ and $\mathbf{Y}$ or neither but not just one of them. So, scikit-learn allows only two of the 16 combinations.

Additionally, scikit-learn requires a user to input the exact number of components, A , and will only output predictions for this exact number of components. However, to get a PLS model with A components, one must first compute the initial $1, \dots, A - 1$ components. This means that when a PLS model with A components has been trained (calibrated), all predictions for that PLS model using one up to and including A components are readily available. Getting all these predictions is required when determining the optimal number of components to use with, e.g., cross-validation. Getting all A predictions from scikit-learn's implementation is not impossible but requires extensive knowledge and hacking of their implementation, which will still be slower than either IKPLS algorithm. Thus, to make PLS modeling convenient and straightforward for our studies and other practitioners, we developed and open-sourced IKPLS Algorithm 1 and Algorithm 2 in Python (Engström et al. 2024).

Dayal and MacGregor (1997) have made an error, or perhaps a typo, in step 2 of their description of the IKPLS algorithm. They write that if $M < K$, $\mathbf{q}$ can be computed as the eigenvector corresponding to the largest eigenvalue of $\mathbf{Y}^T \mathbf{X}^T \mathbf{X} \mathbf{Y}$. In reality, the matrix product should be $\mathbf{Y}^T \mathbf{X} \mathbf{X}^T \mathbf{Y}$ and is correctly implemented by Engström et al. (2024). While the latter is the correct matrix product, it is likely also what Dayal and MacGregor (1997) intended to write as the former matrix product is generally undefined due to a dimensionality mismatch unless $\mathbf{X}$ is a square matrix, that is, $N = K$.

Later in this PhD project, after having implemented the original IKPLS formulation by Dayal and MacGregor (1997), PLS had to be used for the imbalanced barley germination dataset that was curated for this PhD thesis (Engström et al. 2025b). To address the class imbalance, the original IKPLS implementation was augmented to accommodate sample weighting as described by Becker and Ismail (2016). When using weighted PLS and scaling is desired, it should be carried out with the weighted sample standard deviation, which is the square root of the weighted sample variance. In their Equation 4, Becker and Ismail (2016) define the weighted sample variance row vector of $\mathbf{X}$ as

$$\sigma_{\mathbf{w}}^2 = \frac{\sum_{i=1}^N \mathbf{w}_i (\mathbf{X}_i - \mu_{\mathbf{w}})^{\circ 2}}{\left(\sum_{i=1}^N \mathbf{w}_i\right) - 1}. \quad (2.1)$$

Here, $\mathbf{w}$ is a length N vector of sample weights where $\mathbf{w}_i \geq 0$ for all $i \in \{1, \dots, N\}$, $\mu_{\mathbf{w}}$ is the weighted mean row vector of $\mathbf{X}$, and $\circ$ denotes Hadamard (element-wise) exponentiation. However, this formulation is not well defined in general. Consider, for example, the case where the weights are normalized to sum to one, $\sum_{i=1}^N \mathbf{w}_i = 1$. Then, the denominator in the above expression becomes zero, leading to an undefined weighted sample variance.

Therefore, the IKPLS implementation (Engstrøm et al. 2024) instead uses the NIST formulation for weighted sample variance (National Institute of Standards and Technology 2013b) and its square root, weighted sample standard deviation (National Institute of Standards and Technology 2013a). The NIST weighted sample variance is given by

$$\sigma_{\mathbf{w}}^2 = \frac{N' \sum_{i=1}^N \mathbf{w}_i (\mathbf{X}_i - \mu_{\mathbf{w}})^{\circ 2}}{(N' - 1) \sum_{i=1}^N \mathbf{w}_i}, \quad (2.2)$$

where N' is the number of non-zero weights. Here, scaling the weights by a constant factor does not change the weighted sample variance; therefore, this expression works seamlessly with weights normalized to one. Note, that if $\sum_{i=1}^N \mathbf{w}_i = N'$, then Equation (2.1) and Equation (2.2) are equal. This will, for example, be the case when using balanced class and sample weights from scikit-learn (Pedregosa et al. 2011). These weights are inspired by King and Zeng (2001) and are the only source of class and sample weights in this thesis.

What follows is the article by Engstrøm et al. (2024) about the Python software package `ikpls`¹. While the article is peer-reviewed, most of the review process focused on reviewing the source code. The code review is linked in the article and [here](#)² for convenience.

¹<https://github.com/sm00thix/ikpls>

²<https://github.com/openjournals/joss-reviews/issues/6533>

IKPLS: Improved Kernel Partial Least Squares and Fast Cross-Validation Algorithms for Python with CPU and GPU Implementations Using NumPy and JAX

Ole-Christian Galbo Engstrøm ^{1,2,3¶}, Erik Schou Dreier ¹, Birthe Møller Jespersen ⁴, and Kim Steenstrup Pedersen ^{2,5}

¹ FOSS Analytical A/S, Denmark ² Department of Computer Science (DIKU), University of Copenhagen, Denmark ³ Department of Food Science (UCPH FOOD), University of Copenhagen, Denmark ⁴ UCL University College, Denmark ⁵ Natural History Museum of Denmark (NHMD), University of Copenhagen, Denmark ¶ Corresponding author

DOI: [10.21105/joss.06533](https://doi.org/10.21105/joss.06533)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: Sébastien Boisgérault 

Reviewers:

- [@parmentelat](#)
- [@basileMarchand](#)

Submitted: 25 January 2024

Published: 23 July 2024

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

The `ikpls` software package provides fast and efficient tools for PLS (Partial Least Squares) modeling. This package is designed to help researchers and practitioners handle PLS modeling faster than previously possible – particularly on large datasets. The PLS implementations in `ikpls` use the fast IKPLS (Improved Kernel PLS) algorithms (Dayal & MacGregor, 1997), providing a substantial speedup compared to `scikit-learn`'s (Pedregosa et al., 2011) PLS implementation, which is based on NIPALS (Nonlinear Iterative Partial Least Squares) (H. Wold, 1966). The `ikpls` package also offers an implementation of IKPLS combined with the fast cross-validation algorithm by O.-C. G. Engstrøm (2024), significantly accelerating cross-validation of PLS models – especially when using a large number of cross-validation splits.

`ikpls` offers NumPy-based CPU and JAX-based CPU/GPU/TPU implementations. The JAX implementations are also differentiable, allowing seamless integration with deep learning techniques. This versatility enables users to handle diverse data dimensions efficiently.

In conclusion, `ikpls` empowers researchers and practitioners in machine learning, chemometrics, and related fields with efficient, scalable, and end-to-end differentiable tools for PLS modeling, facilitating optimal component selection and preprocessing decisions by offering implementations of

1. both variants of IKPLS for CPUs;
2. both variants of IKPLS for GPUs, both of which are end-to-end differentiable, allowing integration with deep learning models;
3. IKPLS combined with a cross-validation algorithm that yields a substantial speedup compared to the classical cross-validation algorithm.

Statement of need

PLS (H. Wold, 1966) is a standard method in machine learning and chemometrics. PLS can be used as a regression model, PLS-R (PLS regression) (S. Wold et al., 1983, 2001), or a classification model, PLS-DA (PLS discriminant analysis) (Barker & Rayens, 2003). PLS takes as input a matrix $\mathbf{X}$ with dimension (N, K) of predictor variables and a matrix $\mathbf{Y}$ with dimension (N, M) of response variables. PLS decomposes $\mathbf{X}$ and $\mathbf{Y}$ into A latent variables (also called components), which are linear combinations of the original $\mathbf{X}$ and $\mathbf{Y}$. Choosing the optimal number of components, A , depends on the input data and varies from task to task. Additionally, selecting the optimal preprocessing method is challenging to assess before model

validation Sørensen et al. (2021) but is required for achieving optimal performance (Du et al., 2022). The optimal number of components and the optimal preprocessing method are typically chosen by cross-validation, which may be very computationally expensive. The implementations of the fast cross-validation algorithm (O.-C. G. Engstrøm, 2024) will significantly reduce the computational cost of cross-validation.

This work introduces the Python software package, `ikpls`, with novel, fast implementations of IKPLS Algorithm #1 and Algorithm #2 by Dayal & MacGregor (1997), which have previously been compared with other PLS algorithms and shown to be fast (Alin, 2009) and numerically stable (Andersson, 2009). The implementations introduced in this work use NumPy (Harris et al., 2020) and JAX (Bradbury et al., 2018). The NumPy implementations can be executed on CPUs, and the JAX implementations can be executed on CPUs, GPUs, and TPUs. The JAX implementations are also end-to-end differentiable, allowing integration into deep learning methods. This work compares the execution time of the implementations on input data of varying dimensions. It reveals that choosing the implementation that best fits the data will yield orders of magnitude faster execution than the common NIPALS (H. Wold, 1966) implementation of PLS, which is the one implemented by scikit-learn (Pedregosa et al., 2011), an extensive machine learning library for Python. With the implementations introduced in this work, choosing the optimal number of components and the optimal preprocessing becomes much more feasible than previously. Indeed, derivatives of this work have previously been applied to do this precisely (O.-C. G. Engstrøm et al., 2023a, 2023b).

Other implementations of other PLS algorithms with NumPy and scikit-learn exist, even for more specialized tasks such as multiblock PLS (Baum & Vermue, 2019). These implementations, however, are not as fast as IKPLS (Alin, 2009). Implementations of IKPLS exist in R and MATLAB. To the best of the authors' knowledge, however, there are no Python implementations of IKPLS that simultaneously correctly handle all possible dimensions of $\mathbf{X}$ and $\mathbf{Y}$. To the best of the authors' knowledge, no other PLS algorithms exist in JAX, nor do implementations of IKPLS in other frameworks with automatic differentiation.

Implementations

IKPLS (Dayal & MacGregor, 1997) comes in two variants: Algorithm #1 and Algorithm #2. The implementations compute internal matrices $\mathbf{W}$ ($\mathbf{X}$ weights) of dimension (K, A) , $\mathbf{P}$ ($\mathbf{X}$ loadings) of dimension (K, A) , $\mathbf{Q}$ ($\mathbf{Y}$ loadings) of dimension (M, A) , $\mathbf{R}$ ($\mathbf{X}$ rotations) of dimension (K, A) and a tensor $\mathbf{B}$ (regression coefficients) of dimension (A, K, M) . Algorithm #1 also computes $\mathbf{T}$ ($\mathbf{X}$ scores) of dimension (N, A) .

The `ikpls` package has been rigorously tested for equivalence against scikit-learn's NIPALS using NIR spectra data from Dreier et al. (2022) and scikit-learn's PLS test-suite. Examples are provided for core functionalities, demonstrating fitting, predicting, cross-validating on CPU and GPU, and gradient propagation through PLS fitting.

NumPy

`ikpls` includes a Python class implementing both NumPy-based CPU IKPLS algorithms. It subclasses scikit-learn's `BaseEstimator`, facilitating integration with functions like `cross_validate`. Another class with IKPLS and fast cross-validation (O.-C. G. Engstrøm, 2024) is available.

JAX

For GPU/TPU acceleration, `ikpls` provides Python classes for each IKPLS algorithm using JAX. JAX combines Autograd (Maclaurin et al., 2015) with XLA (Accelerated Linear Algebra) for high-performance computation on various hardware. Automatic differentiation in forward and backward modes enables seamless integration with deep learning techniques, supporting user-defined metric functions.

Benchmarks

Benchmarks compare ikpls implementations with scikit-learn’s NIPALS across varying data dimensions and component numbers. Single fits and leave-one-out cross-validation (LOOCV) scenarios are explored. To estimate execution time in a realistic scenario, the reported execution times for LOOCV include calibration of the PLS models and computation of the root mean squared error on the validation sample for all components from 1 to A .

The benchmarks use randomly generated data with fixed seeds for consistency. Default parameters are $N = 10,000$, $K = 500$, and $A = 30$, testing both single-target (PLS1) and multi-target (PLS2) scenarios.

The results in [Figure 1](#) suggest CPU IKPLS for single fits, with a preference for IKPLS #2 if $N \gg K$. GPU usage is advised for larger datasets. In cross-validation, IKPLS options consistently outperform scikit-learn’s NIPALS, with CPU IKPLS #2 (fast cross-validation) excelling, especially for large datasets. GPU IKPLS #1 is optimal in specific cases, considering preprocessing constraints. Fast cross-validation delivers significant speedup, more pronounced for IKPLS #2, especially when dealing with a larger number of target variables (M) ([O.-C. G. Engstrøm, 2024](#)).

In an attempt to give guidelines for algorithm choice for the most common use cases, we report the execution time of the implementations with varying values for each of the parameters above. Specifically, we define a list of values for each parameter to take while the rest of the parameters maintain their default settings. We use

$N \in [10^1, 10^2, 10^3, 10^4, 10^5, 10^6]$, $K \in [30, 50, 10^2, 5 \cdot 10^2, 10^3, 5 \cdot 10^3, 10^4]$, $A \in [10, 20, 30, 50, 100, 200, 500]$, and $M \in [1, 10]$.

All the experiments are executed on the hardware shown in [Table 1](#) on a machine running Ubuntu 22.04 Jammy Jellyfish.

Table 1: Hardware used in the execution time experiments.

Component	Name
Motherboard	ASUS PRO WS X570-ACE
CPU	AMD Ryzen 9 5950X
CPU Cooler	NZXT Kraken X73
GPU	NVIDIA GeForce RTX3090 Ti, CUDA 11.8
RAM	4x32GB, DDR4, 3.2GHz, C16

ikpls version: 1.0.1

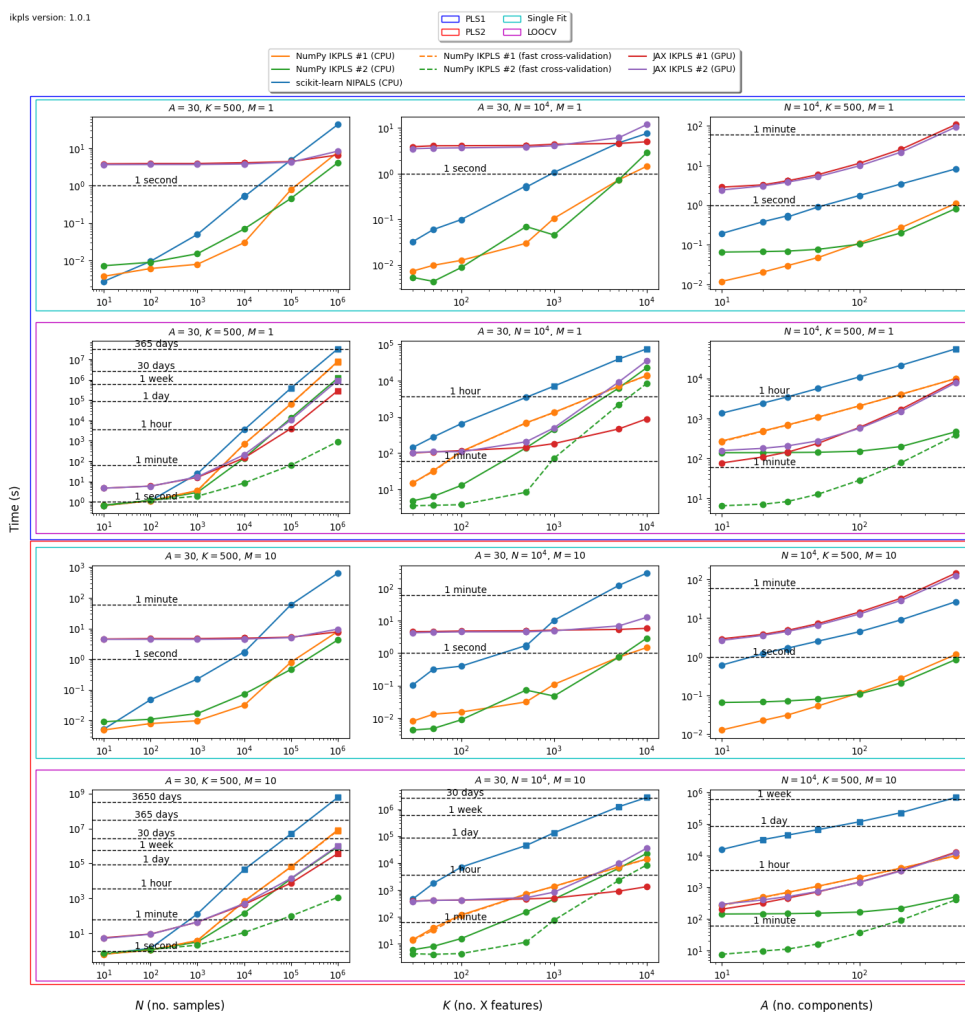


Figure 1: Results of our timing experiments. We vary N , K , and A in the first, second, and third columns. The first two rows are PLS1. The last two rows are PLS2. The first and third rows are single-fit. The second and fourth rows are leave-one-out cross-validation, computing the mean squared error and best number of components for each validation split. A circle indicates that the experiment was run until the end, and the time reported is exact. A square means that the experiment was run until the time per iteration had stabilized and used to forecast the time usage if the experiment was run to completion.

Acknowledgements

This work is part of an industrial Ph.D. project receiving funding from FOSS Analytical A/S and The Innovation Fund Denmark. Grant Number: 1044-00108B.

References

- Alin, A. (2009). Comparison of PLS algorithms when number of objects is much larger than number of variables. *Statistical Papers*, 50(4), 711. <https://doi.org/10.1007/s00362-009-0251-7>
- Andersson, M. (2009). A comparison of nine PLS1 algorithms. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 23(10), 518–529. <https://doi.org/10.1002/cem.1248>

- Barker, M., & Rayens, W. (2003). Partial least squares for discrimination. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(3), 166–173. <https://doi.org/10.1002/cem.785>
- Baum, A., & Vermue, L. (2019). Multiblock PLS: Block dependent prediction modeling for python. *Journal of Open Source Software*, 4(34), 1190. <https://doi.org/10.21105/joss.01190>
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). JAX: Composable transformations of Python+NumPy programs (Version 0.3.13). <http://github.com/google/jax>
- Dayal, B. S., & MacGregor, J. F. (1997). Improved PLS algorithms. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 11(1), 73–85. [https://doi.org/10.1002/\(SICI\)1099-128X\(199701\)11:1%3C73::AID-CEM435%3E3.0.CO;2-%23](https://doi.org/10.1002/(SICI)1099-128X(199701)11:1%3C73::AID-CEM435%3E3.0.CO;2-%23)
- Dreier, E. S., Sørensen, K. M., Lund-Hansen, T., Jespersen, B. M., & Pedersen, K. S. (2022). Hyperspectral imaging for classification of bulk grain samples with deep convolutional neural networks. *Journal of Near Infrared Spectroscopy*, 30(3), 107–121. <https://doi.org/10.1177/09670335221078356>
- Du, Z., Tian, W., Tilley, M., Wang, D., Zhang, G., & Li, Y. (2022). Quantitative assessment of wheat quality using near-infrared spectroscopy: A comprehensive review. *Comprehensive Reviews in Food Science and Food Safety*, 21(3), 2956–3009. <https://doi.org/10.1111/1541-4337.12958>
- Engstrøm, O.-C. G. (2024). *Shortcutting cross-validation: Efficiently deriving column-wise centered and scaled training set $X^T X$ and $X^T Y$ without full recomputation of matrix products or statistical moments*. <https://doi.org/10.48550/arXiv.2401.13185>
- Engstrøm, O.-C. G., Dreier, E. S., Jespersen, B. M., & Pedersen, K. S. (2023a). Improving deep learning on hyperspectral images of grain by incorporating domain knowledge from chemometrics. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 485–494. <https://doi.org/10.1109/iccvw60793.2023.00055>
- Engstrøm, O.-C. G., Dreier, E. S., Jespersen, B. M., & Pedersen, K. S. (2023b). Analyzing near-infrared hyperspectral imaging for protein content regression and grain variety classification using bulk references and varying grain-to-background ratios. *arXiv Preprint arXiv:2311.04042*. <https://doi.org/10.48550/arXiv.2311.04042>
- Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., & others. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- Maclaurin, D., Duvenaud, D., & Adams, R. P. (2015). Autograd: Effortless gradients in numpy. *ICML 2015 AutoML Workshop*, 238, 5.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Rinnan, Å., Berg, F. van den, & Engelsen, S. B. (2009). Review of the most common pre-processing techniques for near-infrared spectra. *TrAC Trends in Analytical Chemistry*, 28(10), 1201–1222. <https://doi.org/10.1016/j.trac.2009.07.007>
- Sørensen, K. M., Berg, F. van den, & Engelsen, S. B. (2021). NIR data exploration and regression by chemometrics—a primer. *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*, 127–189. https://doi.org/10.1007/978-981-15-8648-4_7
- Wold, H. (1966). Estimation of principal components and related models by iterative least

squares. *Multivariate Analysis*, 391–420.

Wold, S., Albano, C., Dunn, W. J., Esbensen, K., Hellberg, S., Johansson, E., Sjöström, M., Martens, H., & Russwurm, J. (1983). Food research and data analysis. *London: H. Martens and H. Russwurm Jr.*

Wold, S., Sjöström, M., & Eriksson, L. (2001). PLS-regression: A basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58(2), 109–130. [https://doi.org/10.1016/s0169-7439\(01\)00155-1](https://doi.org/10.1016/s0169-7439(01)00155-1)

Chapter 3

Fast Cross-Validation

IN NIR-HSI, it is often the case that $N \gg K$ and $N \gg M$, that is, $\mathbf{X}$ and $\mathbf{Y}$ have significantly more rows than columns, that is, there are more spectra than both wavelength channels and reference values per spectrum. Here, IKPLS Algorithm 2 is the fastest choice for the PLS model, and most of the computational cost is due to the computation of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. These matrix products are computed initially, and the remainder of the PLS fit uses these two matrix products without looking at $\mathbf{X}$ and $\mathbf{Y}$ ever again. This implies that the only part of the PLS fit with a runtime complexity dependent on N is the computation of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. Thus, if the computation of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ can be sped up, the total computation time can be drastically improved.

During cross-validation, $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ must be computed independently for each training partition so as not to introduce data leakage from the corresponding validation partitions. However, as training partitions overlap significantly between the rows of $\mathbf{X}$ and $\mathbf{Y}$, they also overlap significantly in the computations required to compute training partition-wise $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. For P partitions (that is, P -fold cross-validation), the computational complexity of computing all training partition-wise $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ is $\Theta(PNK(K + M))$. Engström and Jensen (2025) devise algorithms that address and eliminate the redundant calculations and decrease this computational complexity by a factor of P to $\Theta(NK(K + M))$ for all 16 combinations of centering and scaling of $\mathbf{X}$ and $\mathbf{Y}$. This means that all P training partition-wise $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ can be computed in the same time as computing the total $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ once. We achieve this speed-up without increasing the space complexity. Combining these algorithms with IKPLS Algorithm 2 makes cross-validation of PLS models on datasets with $N \gg K$ *extremely* fast. This is because the entire training partition-wise $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ need not be recomputed for each training partition, and these computations would otherwise dominate the total runtime for the PLS cross-validation as shown by Engström et al. (2024) and devised in the following article by Engström and Jensen, 2025.

RESEARCH ARTICLE OPEN ACCESS

Fast Partition-Based Cross-Validation With Centering and Scaling for $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ Ole-Christian Galbo Engstrøm^{1,2,3} | Martin Holm Jensen¹¹Research and Development, FOSS Analytical A/S, Hillerød, Denmark | ²Department of Computer Science, University of Copenhagen, Copenhagen, Denmark | ³Department of Food Science, University of Copenhagen, Frederiksberg, Denmark**Correspondence:** Ole-Christian Galbo Engstrøm (ole.e@di.ku.dk)**Received:** 11 September 2024 | **Revised:** 27 January 2025 | **Accepted:** 28 January 2025**Funding:** This research was supported by the Innovation Fund Denmark and FOSS Analytical A/S, grant number: 1044-00108B.**Keywords:** algorithm design | centering and scaling | computational complexity | cross-validation | leakage by preprocessing

ABSTRACT

We present algorithms that substantially accelerate partition-based cross-validation for machine learning models that require matrix products $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. Our algorithms have applications in model selection for, for example, principal component analysis (PCA), principal component regression (PCR), ridge regression (RR), ordinary least squares (OLS), and partial least squares (PLS). Our algorithms support all combinations of column-wise centering and scaling of $\mathbf{X}$ and $\mathbf{Y}$, and we demonstrate in our accompanying implementation that this adds only a manageable, practical constant over efficient variants without preprocessing. We prove the correctness of our algorithms under a fold-based partitioning scheme and show that the running time is independent of the number of folds; that is, they have the same time complexity as that of computing $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ and space complexity equivalent to storing $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{X}^T\mathbf{X}$, and $\mathbf{X}^T\mathbf{Y}$. Importantly, unlike alternatives found in the literature, we avoid data leakage due to preprocessing. We achieve these results by eliminating redundant computations in the overlap between training partitions. Concretely, we show how to manipulate $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ using only samples from the validation partition to obtain the preprocessed training partition-wise $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. To our knowledge, we are the first to derive correct and efficient cross-validation algorithms for any of the 16 combinations of column-wise centering and scaling, for which we also prove only 12 give distinct matrix products.

1 | Introduction

In machine learning and predictive modeling, the objective is often to discover a latent representation, relationship (model), or both for a set of data represented by data set matrices $\mathbf{X}$ and $\mathbf{Y}$. Both matrices have N rows, where each row represents a (data) sample. The $\mathbf{X}$ matrix contains K features (variables) in its columns, while the $\mathbf{Y}$ matrix contains M observations (responses) in its columns. Many popular models use either or both of the matrix products¹ $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ with examples including principal component analysis (PCA) [4], principal component regression (PCR) [5, 6], ridge regression (RR) [7], ordinary least squares (OLS) via normal equations [8], and various partial least

squares regression (PLS-R) [9, 10], and partial least squares discriminant analysis (PLS-DA) [11–13] algorithms.

Cross-validation [14, 15] assists with model selection for such methods by robustly evaluating performance on unseen data and avoiding overfitting hyperparameter selection. The (sample-wise) P -fold cross-validation scheme splits samples into P nonoverlapping validation partitions, each associated with a training partition comprising the samples not in the validation partition (i.e., the remaining $P - 1$ validation partitions). This necessitates computing matrix products over the submatrices formed by the training partition of each fold of the data set and evaluating on the validation partition.

This is an open access article under the terms of the [Creative Commons Attribution](#) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2025 The Author(s). *Journal of Chemometrics* published by John Wiley & Sons Ltd.

Therefore, a baseline (naive) algorithm increases the running time of model selection by a factor of P . Lindgren et al. [1], with improvements by Lindgren et al. [16], remedy this by computing matrix products once over the entire data set and storing submatrices of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ for each validation partition, which are subtracted from the data set matrix products to obtain the training partition matrix products. This faster algorithm has time complexity independent of P . It is not difficult to adapt the approach by Lindgren et al. [16] to avoid storing the P submatrices (with dimensions $K \times K$ and $K \times M$) as demonstrated, for example, by Gianola and Schön [17], thereby reducing space complexity.

Often, we want to center by subtracting column-wise means and scale by dividing with the column-wise sample standard deviations so that each column in $\mathbf{X}$ and $\mathbf{Y}$ has a mean of 0 and variance of 1 (or slightly smaller with Bessel's correction). For cross-validation, these statistics should be computed over the training partition to avoid the risk of leakage by preprocessing, a common pitfall in machine learning which may overestimate performance and lead to issues with reproducibility [18, 19]. However, this is not the case for the preprocessing done in the fast algorithms by Lindgren et al. [16] or when $\mathbf{X}$ and $\mathbf{Y}$ are simply centered and scaled over the entire data set initially. Generally, such approaches yield training partition matrix products different from the expensive baseline algorithm, which we illustrate at the end of Section 4.1.

In this paper, we develop a fast algorithm (Algorithm 7) where centering and scaling are performed over training partition submatrices, enabling efficient model selection using P -fold cross-validation that is not compromised by information from outside the training partition. Correctness is shown in Proposition 17. Algorithm 7 is asymptotically independent of P , having time complexity $\Theta(NK(K+M))$ (Proposition 19, matching Lindgren et al. [16]) and space complexity $\Theta((K+N)(K+M))$ (Proposition 22, matching Gianola and Schön [17]). Note that the time bound is the same as that of computing $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$, and the space bound is the same as storing $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{X}^T\mathbf{X}$, and $\mathbf{X}^T\mathbf{Y}$. This suggests that our algorithms inexpensively enable P -fold cross-validation with centering and scaling, which is validated empirically in Section 7.

To compute training partition statistics efficiently, we determine how to void the contribution from the validation partition on the statistics over the entire data set. The spirit of this approach is, therefore, the same as the fractionated subparts of Lindgren et al. [16] and akin to what Liland et al. [20] enable for centering of $\mathbf{X}\mathbf{X}^T$ (useful for wide data sets). To our knowledge, the algorithms we develop are the first to achieve fast P -fold cross-validation while properly centering, scaling, or both, using statistics exclusively over the training partition. This finds application in model selection for, for example, PCA, PCR, RR, OLS, and partial least squares (PLS), especially for tall data sets (N greater than K) where computing matrix products may well dominate the time required to fit models.

Section 2 discusses related work. We introduce fast algorithms for cross-validation incrementally in Section 3 (no preprocessing), Section 4 (column-wise centering), and Section 5 (column-wise centering and scaling), showing correctness and

computational complexity. Our results apply generally to any preprocessing combination of centering and scaling, which we discuss in Section 6. Interestingly, whether centering $\mathbf{X}$, $\mathbf{Y}$, or both, each combination results in the same matrix product $\mathbf{X}^T\mathbf{Y}$ (Lemma 9). We present results of benchmarking baseline and fast algorithms for all combinations of preprocessing in Section 7 (implementation by Engström [21] under the permissive Apache 2.0 license) before concluding in Section 8.

2 | Related Work

Fast P -fold cross-validation for methods that require computing $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ has been studied over past decades [1, 16, 17, 22], where efficiency is achieved by computing matrix products only over the validation partition for each fold. This yields an asymptotic running time of $\Theta(NK(K+M))$ matching the cost of computing $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ and so is independent of the choice of P . In these methods, centering and scaling are based on statistics not computed over the particular training partition of the fold, thereby risking leakage from preprocessing. Naively extending these algorithms by computing statistics over the training partition submatrices increases the running time by a factor of P , taking it into baseline algorithm territory.

Combining cross-validation with models relying on either $\mathbf{X}^T\mathbf{X}$, $\mathbf{X}^T\mathbf{Y}$, or both has been studied for PCA [23–29], PCR [28, 30], RR [31], OLS [28], and PLS [26, 28, 30, 32]. These are cases where our results apply. Our algorithms are also readily applicable to speed up the inner loop in repeated double cross-validation [33]. A library for fast cross-validation with centering and scaling for the fast [34] and numerically stable [35] improved kernel PLS algorithms by Dayal and MacGregor [36] that has been made available by Engström et al. [37], which shows practical superiority in terms of execution time over the baseline approach. This paper is the accompanying theoretical treatise of the efficient training partition-wise matrix products implemented by Engström et al. [37].

Cross-validation is similarly useful for models that use the matrix product $\mathbf{X}\mathbf{X}^T$ [20, 38, 39], specifically for wide data sets. Liland et al. [20] give a fast solution that supports training partition centering of $\mathbf{X}\mathbf{X}^T$, though the case of scaling is not handled. When $N < K$, $\mathbf{X}\mathbf{X}^T$ is smaller than $\mathbf{X}^T\mathbf{X}$, and so practitioners would likely prefer computing $\mathbf{X}\mathbf{X}^T$ which our algorithms do not extend to.

Related also is *column-wise* cross-validation, which considers subsets of columns of $\mathbf{X}$ and can be used for feature selection, a fast version of which is given by Stefansson et al. [40]. There are no additional concerns with centering and scaling for column-wise cross-validation, as statistics are computed independently for each column. The column-wise cross-validation approach can be used separately or carried out for each fold in sample-wise P -fold cross-validation, in which case our results directly apply.

Another set of preprocessing methods [41] that operate on each sample individually include standard normal variate (SNV) [42], detrend [42], and convolution with, for example, Savitzky–Golay (SG) filters [43, 44]. If such sample-wise preprocessing is

desired, it must be applied before centering and scaling, and in this case, it is fully compatible with our results. This is unlike multiplicative scatter correction (MSC) [45] since this preprocessing method modifies samples based on a fitting over the training partition, an immediate version of which degrades the time complexity of our fast algorithms.

3 | Cross-Validation Without Preprocessing

We proceed incrementally towards the fast cross-validation algorithm supporting training partition centering and scaling in Section 5 by first considering the case without preprocessing in this section and in Section 4 the case with centering.

An entry (i, j) in a matrix $\mathbf{M}$ is denoted $\mathbf{M}_{ij}$ and is the entry in the i 'th row and j 'th column of $\mathbf{M}$. Let $R = \{1, \dots, N\}$ be the set of indices (rows) and $S \subseteq R$, then we denote by $\mathbf{M}_S$ the submatrix of $\mathbf{M}$ containing each index (row) in S . We omit parentheses and write $\mathbf{M}_S^T$ to mean $(\mathbf{M}_S)^T$, $\mathbf{M}_{S_{ij}}$ to mean $(\mathbf{M}_S)_{ij}$, and $\mathbf{M}_{S_{ij}}^T$ to mean $((\mathbf{M}_S)^T)_{ij}$. To refer to the i 'th row of a matrix, we write $\mathbf{M}_{i*}$, and for the j 'th column, we write $\mathbf{M}_{*j}$. For visual clarity, we may write $\mathbf{M}_i$ instead of $\mathbf{M}_{i*}$ and emphasize that in such cases, we're always indexing rows of the matrix. Data set matrices are $\mathbf{X} \in \mathbb{R}^{N \times K}$ with N rows and K features, and $\mathbf{Y} \in \mathbb{R}^{N \times M}$ with N rows and M observations. We refer to $\mathbf{X}, \mathbf{Y}$, the set of rows R , and dimensions N, K, M as given in this paragraph in the remainder.

Definition 1. training partition, validation partition Let $2 \leq P \leq N$ be the number of (cross-validation) partitions (folds). P -fold cross-validation associates each row $n \in R$ with a *partition* $p \in \{1, \dots, P\}$, indicating which partition n belongs to. A (valid) *partitioning* $\mathcal{P}$ of R is an array of length N such that $\bigcup_{n \in R} \mathcal{P} = \{1, \dots, P\}$ and $\mathcal{P} = p$ means n belongs to p . For $p \in \{1, \dots, P\}$ and *partitioning* $\mathcal{P}$, the validation partition V_p is given by $\{n \in R | \mathcal{P} = p\}$, and the training partition T_p by $R \setminus V_p$.

In the remainder, we will refer to P as the number of partitions (folds). Observe that from Definition 1 follows $T_p \cap V_p = \emptyset$, $T_p \cup V_p = R$, and $\sum_{p=1}^P |V_p| = N$ since all V_p are pairwise disjoint. These properties describe P -fold cross-validation, where the case of $P = N$ is leave-one-out cross-validation. Definition 1 also admits $\mathcal{P} = ((n-1) \bmod P) + 1$ that places every P 'th sample in the same partition as in venetian blinds cross-validation and $\mathcal{P} = \left\lfloor \frac{n-1}{P} \right\rfloor + 1$ that describes P -block-sized contiguous block cross-validation. Observe that we do not assume validation partitions are balanced; that is, they do not need to contain roughly the same amount of samples. Algorithm 1 is used for preprocessing a partitioning and store each V_p .

We consider Algorithm 2 the baseline method for computing matrix products $\mathbf{X}_T^T \mathbf{X}_T$ and $\mathbf{X}_T^T \mathbf{Y}_T$ for each training partition T , with the more efficient alternative being Algorithm 3 using the fractionated subparts insight of Lindgren et al. [16]. To clarify our analysis, we store intermediate computations in variables and allow for the same notation on these variables as we do for matrices. For instance, when $\mathbf{X}_T$ is assigned $\mathbf{X}_T$ and we write $\mathbf{X}_T \mathbf{X}_T^T \leftarrow \mathbf{X}_T^T \mathbf{X}_T$, then $\mathbf{X}_T^T$ is the transposition of $\mathbf{X}_T$ (equal to $\mathbf{X}_T^T$), and $\mathbf{X}_T \mathbf{X}_T^T$ is effectively assigned the value $\mathbf{X}_T^T \mathbf{X}_T$. Below, we show the correctness of the algorithms and analyze their computational complexities.

3.1 | Correctness

Lemma 1. Let $\mathcal{P}$ be a partitioning of R for P partitions and $\mathcal{V}$ the array computed by Algorithm 1. Then for any $n \in R$ and $p \in \{1, \dots, P\}$, $n \in \mathcal{V}$ iff $\mathcal{P} = p$.

Proof. Consider any $p \in \{1, \dots, P\}$. Step 1 ensures that $\mathcal{V}$ is initially the empty set. Steps 3 and 4 extend $\mathcal{V}$ with n iff $\mathcal{P} = p$. From step 2, we consider every $n \in R$, so this holds for all $n \in R$ and $p \in \{1, \dots, P\}$. $\square$

Algorithm 1 Compute Validation Partitions

Input: $\mathcal{P}$ is a valid partitioning of $R = \{1, \dots, N\}$

- 1: Let $\mathcal{V}$ be an array of length P where each element is the empty set
- 2: **for** $n \in R$ **do**
- 3: $p \leftarrow \mathcal{P}[n]$ $\triangleright$ Obtain p which n belongs to
- 4: $\mathcal{V}[p] \leftarrow \mathcal{V}[p] \cup \{n\}$ $\triangleright$ Add n to V_p
- 5: **end for**
- 6: **return** $\mathcal{V}$

Algorithm 2 Baseline Cross-Validation Algorithm

Input: $\mathbf{X} \in \mathbb{R}^{N \times K}$, $\mathbf{Y} \in \mathbb{R}^{N \times M}$, $\mathcal{P}$ is a valid partitioning of $R = \{1, \dots, N\}$

- 1: $R \leftarrow \{1, \dots, N\}$
- 2: Let $\mathcal{V}$ be the result of executing Algorithm 1 with $\mathcal{P}$ and R as input.
- 3: **for** $p \in \{1, \dots, P\}$ **do**
- 4: $V \leftarrow \mathcal{V}[p]$
- 5: $T \leftarrow R \setminus V$
- 6: $\mathbf{X}_T \leftarrow \mathbf{X}_T$, $\mathbf{Y}_T \leftarrow \mathbf{Y}_T$
- 7: $\mathbf{X}_T \mathbf{X}_T^T \leftarrow \mathbf{X}_T^T \mathbf{X}_T$, $\mathbf{X}_T \mathbf{Y}_T^T \leftarrow \mathbf{X}_T^T \mathbf{Y}_T$ $\triangleright$ Obtain $\mathbf{X}_T^T \mathbf{X}_T$ and $\mathbf{X}_T^T \mathbf{Y}_T$
- 8: **end for**

Algorithm 3 Fast Cross-Validation Algorithm

Input: $\mathbf{X} \in \mathbb{R}^{N \times K}$, $\mathbf{Y} \in \mathbb{R}^{N \times M}$, $\mathcal{P}$ is a valid partitioning of $R = \{1, \dots, N\}$

- 1: $R \leftarrow \{1, \dots, N\}$
- 2: Let $\mathcal{V}$ be the result of executing Algorithm 1 with $\mathcal{P}$ and R as input.
- 3: $\mathbf{X}\mathbf{T}\mathbf{X} \leftarrow \mathbf{X}^T\mathbf{X}$, $\mathbf{X}\mathbf{T}\mathbf{Y} \leftarrow \mathbf{X}^T\mathbf{Y}$
- 4: **for** each partition $p \in \{1, \dots, P\}$ **do**
- 5: $V \leftarrow \mathcal{V}[p]$
- 6: $\mathbf{X}_V \leftarrow \mathbf{X}_V$, $\mathbf{Y}_V \leftarrow \mathbf{Y}_V$
- 7: $\mathbf{X}\mathbf{T}\mathbf{X}_V \leftarrow \mathbf{X}_V^T\mathbf{X}_V$, $\mathbf{X}\mathbf{T}\mathbf{Y}_V \leftarrow \mathbf{X}_V^T\mathbf{Y}_V$ ▷ Obtain $\mathbf{X}_V^T\mathbf{X}_V$ and $\mathbf{X}_V^T\mathbf{Y}_V$
- 8: $\mathbf{X}\mathbf{T}\mathbf{X}_T \leftarrow \mathbf{X}\mathbf{T}\mathbf{X} - \mathbf{X}\mathbf{T}\mathbf{X}_V$, $\mathbf{X}\mathbf{T}\mathbf{Y}_T \leftarrow \mathbf{X}\mathbf{T}\mathbf{Y} - \mathbf{X}\mathbf{T}\mathbf{Y}_V$ ▷ Obtain $\mathbf{X}_T^T\mathbf{X}_T$ and $\mathbf{X}_T^T\mathbf{Y}_T$
- 9: **end for**

Proposition 1 shows that the set $\mathcal{V}$ produced by Algorithm 1 is the validation partition V_p according to Definition 1. To show the correctness of Algorithm 3, we prove the fractionated subparts insight by [1], namely, that training partition matrix products can be obtained by subtracting validation partition matrix products from the data set matrix products.

Lemma 2. Given $p \in \{1, \dots, P\}$, let $T = T_p$ be a training partition and $V = V_p$ validation partition, then $\mathbf{X}_T^T\mathbf{X}_T = \mathbf{X}^T\mathbf{X} - \mathbf{X}_V^T\mathbf{X}_V$.

Proof. We show the equivalent statement $\mathbf{X}_T^T\mathbf{X}_T + \mathbf{X}_V^T\mathbf{X}_V = \mathbf{X}^T\mathbf{X}$. Recall that each of $\mathbf{X}^T\mathbf{X}$, $\mathbf{X}_T^T\mathbf{X}_T$ and $\mathbf{X}_V^T\mathbf{X}_V$ are $K \times K$ matrices and consider any entry (i, j) in $\mathbf{X}^T\mathbf{X}$ for $i \in \{1, \dots, K\}$ and $j \in \{1, \dots, K\}$ as given by,

$$(\mathbf{X}^T\mathbf{X})_{ij} = \sum_{n \in R} \mathbf{X}_{in}^T\mathbf{X}_{nj}.$$

Similarly, we have

$$(\mathbf{X}_T^T\mathbf{X}_T)_{ij} = \sum_{n \in T} \mathbf{X}_{in}^T\mathbf{X}_{nj} \text{ and } (\mathbf{X}_V^T\mathbf{X}_V)_{ij} = \sum_{n \in V} \mathbf{X}_{in}^T\mathbf{X}_{nj}.$$

By Definition 1, $T \cap V = \emptyset$ and $T \cup V = R$, so with commutativity of addition, we have

$$\begin{aligned} (\mathbf{X}_T^T\mathbf{X}_T)_{ij} + (\mathbf{X}_V^T\mathbf{X}_V)_{ij} &= \sum_{n \in T} \mathbf{X}_{in}^T\mathbf{X}_{nj} + \sum_{n \in V} \mathbf{X}_{in}^T\mathbf{X}_{nj} \\ &= \sum_{n \in T \cup V} \mathbf{X}_{in}^T\mathbf{X}_{nj} = \sum_{n \in R} \mathbf{X}_{in}^T\mathbf{X}_{nj} \\ &= (\mathbf{X}^T\mathbf{X})_{ij}. \end{aligned}$$

□

Lemma 3. Given $p \in \{1, \dots, P\}$, let $T = T_p$ be a training partition and $V = V_p$ validation partition, then $\mathbf{X}_T^T\mathbf{Y}_T = \mathbf{X}^T\mathbf{Y} - \mathbf{X}_V^T\mathbf{Y}_V$.

Proof. As proof of Lemma 2 with $j \in \{1, \dots, M\}$. □

Proposition 1. Correctness, Algorithm 3 In step 7 of Algorithm 2, $\mathbf{X}\mathbf{T}\mathbf{X}_T = \mathbf{X}_T^T\mathbf{X}_T$ and $\mathbf{X}\mathbf{T}\mathbf{Y}_T = \mathbf{X}_T^T\mathbf{Y}_T$ and are identical to $\mathbf{X}\mathbf{T}\mathbf{X}_T$ and $\mathbf{X}\mathbf{T}\mathbf{Y}_T$ computed in step 8 of Algorithm 3.

Proof. By Lemma 1, both algorithms select validation partition V according to Definition 1. For Algorithm 2, $T = R \setminus V$ by step 5, hence, clearly, $\mathbf{X}\mathbf{T}\mathbf{X}_T = \mathbf{X}_T^T\mathbf{X}_T$ and $\mathbf{X}\mathbf{T}\mathbf{Y}_T = \mathbf{X}_T^T\mathbf{Y}_T$ in step

7. Since step 8 of Algorithm 3 effectively computes $\mathbf{X}^T\mathbf{X} - \mathbf{X}_V^T\mathbf{X}_V$ and $\mathbf{X}^T\mathbf{Y} - \mathbf{X}_V^T\mathbf{Y}_V$, it follows from Lemmas 2 and 3 that $\mathbf{X}\mathbf{T}\mathbf{X}_T$ and $\mathbf{X}\mathbf{T}\mathbf{Y}_T$ computed in step 8 are identical to those computed in step 7 of Algorithm 2. □

3.2 | Computational Complexity

We analyze the asymptotic time and space complexity of Algorithms 2 and 3, showing the latter shaves a $\Theta(P)$ factor off of the running time with no increase in space complexity. To this end, we assume that matrix multiplication is done using the iterative algorithm that requires ABC operations to multiply two matrices with dimensions $A \times B$ and $B \times C$. While there are algorithms with improved bounds (e.g., by Le Gall [46]), the iterative algorithm is used in practice as it enables optimizations (e.g., cache-friendly memory layout, loop unrolling, efficient use of single instruction, multiple data [SIMD]) that significantly improve hardware performance. We also disregard that by the symmetry of $\mathbf{X}^T\mathbf{X}$, we could roughly halve the number of operations needed by computing and mirroring the upper or lower triangular matrix, as this constant does not impact Θ bounds.

Lemma 4. Algorithm 1 requires $\Theta(N)$ operations.

Proof. Storing inputs $\mathcal{P}$ and R require $2N$ operations. Step 1 requires $P = O(N)$ (since $P \leq N$) operations to construct the array $\mathcal{V}$. Indexing into $\mathcal{P}$ (step 3) and $\mathcal{V}$ (step 4) requires $\Theta(1)$ operations. By using an efficient set implementation, n can be added to $\mathcal{V}$ with $\Theta(1)$ operations. We do so N times by step 2; therefore, Algorithm 1 requires $2N + O(N) + \sum_{n \in R} \Theta(1) = \Theta(N)$ operations. □

Lemma 5. Algorithm 1 requires storing $\Theta(N)$ entries.

Proof. Storing inputs $\mathcal{P}$ and R require $2N$ entries. Step 1 requires P entries storing $\mathcal{V}$. Steps 3 and 4 temporarily store $O(1)$ entries to extend $\mathcal{V}$ with one additional entry. $\mathcal{V}$ is extended N times due to step 2. Therefore, the algorithm stores $2N + O(1) + P + N = \Theta(N)$ entries since $P \leq N$. □

Lemma 6. Given a partitioning $\mathcal{P}$ of R , $\sum_{p=1}^P |T_p| = N(P - 1)$.

Proof. By Definition 1, $|T_p| = N - |V_p|$ and $\sum_{p=1}^P |V_p| = N$; therefore,

$$\sum_{p=1}^P |T_p| = \sum_{p=1}^P N - |V_p| = \sum_{p=1}^P N - \sum_{p=1}^P |V_p| = PN - N = N(P-1).$$

□

Proposition 2. Algorithm 2 requires $\Theta(PNK(K+M))$ operations.

Proof. Storing inputs $\mathbf{X}$, $\mathbf{Y}$, and $\mathcal{P}$ requires $NK + NM + N = \Theta(N(K+M))$ operations. In step 1, we construct a set with N elements using N operations. In step 2, we compute $\mathcal{V}$ with $\Theta(N)$ operations by Lemma 4. Consider a partition $p \in \{1, \dots, P\}$ with validation partition V_p and training partition T_p . Step 4 uses $|V_p|$ operations for assignment, and step 5 uses $|V_p| + |T_p| = N$ operations to compute T . In step 6, we select $|T_p|$ rows from both $\mathbf{X}$ and $\mathbf{Y}$ using $|T_p|(K+M)$ operations. Step 7 requires $\Theta(|T_p|K(K+M))$ operations to compute $\mathbf{X}_{T_p}^T \mathbf{X}_{T_p}$ and $\mathbf{X}_{T_p}^T \mathbf{Y}_{T_p}$. Iterating over $\{1, \dots, P\}$ in step 3, it follows from $\sum_{p=1}^P |V_p| = N$ and Lemma 6 that

$$\begin{aligned} & \sum_{p=1}^P \Theta(|V_p|) + 2\Theta(N) + |T_p|(K+M) + \Theta(|T_p|K(K+M)) \\ &= \Theta(N) + \Theta(PN) + N(P-1)(K+M) + \Theta(PNK(K+M)) \\ &= \Theta(PN(K+M)) + \Theta(PNK(K+M)). \end{aligned}$$

Thus, the total number of operations is $\Theta(PNK(K+M))$. □

Proposition 3. Algorithm 3 requires $\Theta(NK(K+M))$ operations.

Proof. Storing inputs $\mathbf{X}$, $\mathbf{Y}$, and $\mathcal{P}$ requires $\Theta(N(K+M))$ operations. Step 1 uses N operations. Step 2 uses $\Theta(N)$ operations by Lemma 4. Step 3 computes $\mathbf{X}^T \mathbf{X}$ and $\mathbf{X}^T \mathbf{Y}$ requiring respectively $\Theta(KNK)$ and $\Theta(KNM)$ operations. So, up to and including step 3 requires a total of $\Theta(NK(K+M))$ operations.

Consider a partition $p \in \{1, \dots, P\}$ with validation partition V_p and training partition T_p . Step 5 uses $\Theta(|V_p|)$ operations for assignment. Step 6 selects $|V_p|$ rows from $\mathbf{X}$, respectively $\mathbf{Y}$, using $|V_p|(K+M)$ operations. Matrix multiplication in step 7 requires $\Theta(|V_p|K(K+M))$ operations. In step 8, we subtract matrices with dimensions $K \times K$ and $K \times M$ using $\Theta(K(K+M))$ operations. So steps 5–8 require $\Theta(|V_p|) + |V_p|(K+M) + \Theta(|V_p|K(K+M)) + \Theta(K(K+M)) = \Theta(|V_p|K(K+M))$ operations. Iterating over $\{1, \dots, P\}$ in step 4, we have by $\sum_{p=1}^P |V_p| = N$ that

$$\begin{aligned} \Theta\left(\sum_{p=1}^P |V_p|K(K+M)\right) &= \Theta\left(K(K+M) \cdot \sum_{p=1}^P |V_p|\right) \\ &= \Theta(NK(K+M)). \end{aligned}$$

It follows that Algorithm 3 requires $\Theta(NK(K+M)) + \Theta(NK(K+M)) = \Theta(NK(K+M))$ operations. □

Proposition 4. Algorithm 2 requires $\Theta(P)$ more operations than Algorithm 3.

Proof. By Propositions 2 and 3, the ratio of operations is $\frac{\Theta(PNK(K+M))}{\Theta(NK(K+M))} = \Theta(P)$. □

Below, we show that the reduction in running time indicated by Proposition 4 does not come at the cost of increasing space usage.

Proposition 5. Algorithm 2 requires storing $\Theta((K+N)(K+M))$ entries.

Proof. Storing inputs $\mathbf{X}$, $\mathbf{Y}$, and $\mathcal{P}$ requires $\Theta(N(K+M))$ entries. Step 1 stores N entries. Step 2 stores $\Theta(N)$ entries by Lemma 5. Steps 4–5 store $|V| + |T| = N$ entries. Step 6 stores $|T|K + |T|M$ entries, while step 7 requires $K^2 + KM$ entries. This totals $\Theta(N(K+M)) + N + \Theta(N) + N + |T|K + |T|M + K^2 + KM$ entries, which since $|T| < N$ is $\Theta(N(K+M)) + \Theta(K^2 + KM)$. Therefore, the number of entries stored at any step of Algorithm 2 is $\Theta((K+N)(K+M))$. □

Proposition 6. Algorithm 3 requires storing of $\Theta((K+N)(K+M))$ entries.

Proof. Storing inputs $\mathbf{X}$, $\mathbf{Y}$, and $\mathcal{P}$ requires $\Theta(N(K+M))$ entries. Step 1 stores N entries. Step 2 stores $\Theta(N)$ entries by Lemma 5. For step 3, $\mathbf{X}^T \mathbf{X}$ and $\mathbf{X}^T \mathbf{Y}$ require $K^2 + KM$ entries. This means $\Theta((K+N)(K+M))$ prior to step 4. Step 5 requires $|V|$ entries. Step 6 requires $|V|K + |V|M$ entries. Steps 7 and 8 both take up $K^2 + KM$ entries. This totals $\Theta((K+N)(K+M)) + |V| + |V|K + |V|M + 2(K^2 + KM)$ entries. Since $|V| < N$, the number of entries stored at any step of Algorithm 3 is $\Theta((K+N)(K+M))$. □

4 | Cross-Validation With Centering

In many cases, column-wise centering of $\mathbf{X}$ and $\mathbf{Y}$ is employed as a preprocessing step to give individual features and observations a mean of zero. This not only simplifies the interpretation of model coefficients but also improves numerical stability, reduces bias from differing feature magnitudes, and may improve the convergence of optimization algorithms. While Algorithm 2 can be immediately extended to support this with no impact on asymptotic complexity, this is not so for Algorithm 3, since directly computing the mean over columns in $\mathbf{X}_T$ and $\mathbf{Y}_T$ reintroduces a dependency on P as evidenced by Lemma 6.

While Lindgren et al. [16] propose a solution for centering that is more efficient in terms of running time, it falls short of capturing training partition centering as defined next. Indeed, their solution computes matrix products that do not match those of Algorithm 4 (baseline cross-validation algorithm with centering), as we explore further in Section 4.1. In contrast, Algorithm 5 performs training partition centering and shares asymptotic complexity with Algorithm 3. It works by computing the mean *once* for the entire data set and subtracting the contribution from samples in V_p as we iterate over each partition p .

Definition 2. Let $S \subseteq R$ denote a set of rows. The mean row vector $\bar{\mathbf{x}}_S$ has width K and is the row vector of column means of

Algorithm 4 Baseline Cross-Validation Algorithm with Centering

Input: $\mathbf{X} \in \mathbb{R}^{N \times K}$, $\mathbf{Y} \in \mathbb{R}^{N \times M}$, $\mathcal{P}$ is a valid partitioning of $R = \{1, \dots, N\}$

- 1: $R \leftarrow \{1, \dots, N\}$
- 2: Let $\mathcal{V}$ be the result of executing Algorithm 1 with $\mathcal{P}$ and R as input
- 3: **for** each partition $p \in \{1, \dots, P\}$ **do**
- 4: $V \leftarrow \mathcal{V}[p]$
- 5: $T \leftarrow R \setminus V$
- 6: $\mathbf{X}_T \leftarrow \mathbf{X}_T$, $\mathbf{Y}_T \leftarrow \mathbf{Y}_T$
- 7: $\mu \mathbf{X}_T \leftarrow \overline{\mathbf{X}_T}$, $\mu \mathbf{Y}_T \leftarrow \overline{\mathbf{Y}_T}$
- 8: $\mathbf{cX}_T \leftarrow \mathbf{X}_T - \mu \mathbf{X}_T$, $\mathbf{cY}_T \leftarrow \mathbf{Y}_T - \mu \mathbf{Y}_T$ ▷ Obtain $\mathbf{X}_T^c$ and $\mathbf{Y}_T^c$
- 9: $\mathbf{cXTX}_T \leftarrow \mathbf{cX}_T^T \mathbf{cX}_T$, $\mathbf{cXTY}_T \leftarrow \mathbf{cX}_T^T \mathbf{cY}_T$ ▷ Obtain $\mathbf{X}_T^c \mathbf{X}_T^c$ and $\mathbf{X}_T^c \mathbf{Y}_T^c$
- 10: **end for**

Algorithm 5 Fast Cross-Validation Algorithm with Centering

Input: $\mathbf{X} \in \mathbb{R}^{N \times K}$, $\mathbf{Y} \in \mathbb{R}^{N \times M}$, $\mathcal{P}$ is a valid partitioning of $R = \{1, \dots, N\}$

- 1: $R \leftarrow \{1, \dots, N\}$
- 2: Let $\mathcal{V}$ be the result of executing Algorithm 1 with $\mathcal{P}$ and R as input
- 3: $\mathbf{XTX} \leftarrow \mathbf{X}^T \mathbf{X}$, $\mathbf{XTY} \leftarrow \mathbf{X}^T \mathbf{Y}$, $\mu \mathbf{x} \leftarrow \bar{\mathbf{x}}$, $\mu \mathbf{y} \leftarrow \bar{\mathbf{y}}$
- 4: **for** each partition $p \in \{1, \dots, P\}$ **do**
- 5: $V \leftarrow \mathcal{V}[p]$
- 6: $\mathbf{X}_V \leftarrow \mathbf{X}_V$, $\mathbf{Y}_V \leftarrow \mathbf{Y}_V$
- 7: $\mathbf{XTX}_V \leftarrow \mathbf{X}_V^T \mathbf{X}_V$, $\mathbf{XTY}_V \leftarrow \mathbf{X}_V^T \mathbf{Y}_V$
- 8: $\mathbf{XTX}_T \leftarrow \mathbf{XTX} - \mathbf{XTX}_V$, $\mathbf{XTY}_T \leftarrow \mathbf{XTY} - \mathbf{XTY}_V$ ▷ Obtain $\mathbf{X}_T^T \mathbf{X}_T$ and $\mathbf{X}_T^T \mathbf{Y}_T$
- 9: $\mu \mathbf{x}_V \leftarrow \overline{\mathbf{x}_V}$, $\mu \mathbf{y}_V \leftarrow \overline{\mathbf{y}_V}$
- 10: $N_V \leftarrow |V|$, $N_T \leftarrow N - N_V$
- 11: $\mu \mathbf{x}_T \leftarrow \frac{N}{N_T} \mu \mathbf{x} - \frac{N_V}{N_T} \mu \mathbf{x}_V$ ▷ Obtain $\overline{\mathbf{x}_T}$
- 12: $\mu \mathbf{y}_T \leftarrow \frac{N}{N_T} \mu \mathbf{y} - \frac{N_V}{N_T} \mu \mathbf{y}_V$ ▷ Obtain $\overline{\mathbf{y}_T}$
- 13: $\mu \mathbf{xTx}_T \leftarrow N_T (\mu \mathbf{x}_T^T \mu \mathbf{x}_T)$, $\mu \mathbf{xTy}_T \leftarrow N_T (\mu \mathbf{x}_T^T \mu \mathbf{y}_T)$ ▷ Obtain $|T| \mathbf{x}_T^T \overline{\mathbf{x}_T}$ and $|T| \mathbf{x}_T^T \overline{\mathbf{y}_T}$
- 14: $\mathbf{cXTX}_T \leftarrow \mathbf{XTX}_T - \mu \mathbf{xTx}_T$ ▷ Obtain $\mathbf{X}_T^c \mathbf{X}_T^c$
- 15: $\mathbf{cXTY}_T \leftarrow \mathbf{XTY}_T - \mu \mathbf{xTy}_T$ ▷ Obtain $\mathbf{X}_T^c \mathbf{Y}_T^c$
- 16: **end for**

$\mathbf{X}_S$, and similarly, $\overline{\mathbf{y}_S}$ of width M is the mean row vector of $\mathbf{Y}_S$. We stack the mean row vectors to obtain matrices $\overline{\mathbf{X}_S}$ with dimension $|S| \times K$ and $\overline{\mathbf{Y}_S}$ with dimension $|S| \times M$, namely,

$$\overline{\mathbf{X}_S} = \begin{bmatrix} \overline{\mathbf{x}_S} \\ \vdots \\ \overline{\mathbf{x}_S} \end{bmatrix} \text{ and } \overline{\mathbf{Y}_S} = \begin{bmatrix} \overline{\mathbf{y}_S} \\ \vdots \\ \overline{\mathbf{y}_S} \end{bmatrix}.$$

For brevity, we adopt the convention that $\overline{\mathbf{x}_S^T}$ and $\overline{\mathbf{x}_S^T}$, respectively, are given by $(\overline{\mathbf{X}_S})^T$ and $(\overline{\mathbf{x}_S})^T$. We also omit parentheses to denote the i 'th element of vector $\overline{\mathbf{x}_S^T}$ by $\overline{\mathbf{x}_S^T}$ and the i 'th element of vector $\overline{\mathbf{y}_S}$ by $\overline{\mathbf{y}_S}$. We write $\bar{\mathbf{x}}$ to mean $\overline{\mathbf{x}_R}$ and $\bar{\mathbf{y}}$ to mean $\overline{\mathbf{y}_R}$ when clear from context.

Definition 3. Let $S \subseteq R$ denote a set of rows. The (column-wise) centering of $\mathbf{X}_S$ is $\mathbf{X}_S^c = \mathbf{X}_S - \overline{\mathbf{X}_S}$, and the (column-wise) centering of $\mathbf{Y}_S$ is $\mathbf{Y}_S^c = \mathbf{Y}_S - \overline{\mathbf{Y}_S}$.

To support column-wise centering of $\mathbf{X}_T$ and $\mathbf{Y}_T$ for a training partition T , we subtract the column-wise means before taking matrix products, namely,

$$\begin{aligned} \mathbf{X}_T^c \mathbf{X}_T^c &= (\mathbf{X}_T^T - \overline{\mathbf{X}_T^T})(\mathbf{X}_T - \overline{\mathbf{X}_T}) \text{ and} \\ \mathbf{X}_T^c \mathbf{Y}_T^c &= (\mathbf{X}_T^T - \overline{\mathbf{X}_T^T})(\mathbf{Y}_T - \overline{\mathbf{Y}_T}). \end{aligned}$$

We refer to these quantities as centered matrix products, and they are the targets of Algorithm 4 and the more efficient Algorithm 5.

4.1 | Correctness

We proceed to show the correctness of Algorithm 5, where we note steps 1–8 are as in Algorithm 3 except also computing mean row vectors of $\mathbf{X}$ and $\mathbf{Y}$ in step 3. To illustrate the purpose of steps 9–15, consider a validation partition $V \subseteq R$ and training partition $T = R \setminus V$. We show in Proposition 7 that the centered matrix product $\mathbf{X}_T^c \mathbf{Y}_T^c$ equals $\mathbf{X}_T^T \mathbf{Y}_T - |T| \mathbf{x}_T^T \overline{\mathbf{y}_T}$. Just as Lemma 2 is used to efficiently compute the left-hand term with running time depending on $|V|$ rather than $|T|$, the same is possible for the right-hand term. Lemma 7 shows that $\overline{\mathbf{x}_T}$ equals $\frac{N}{|T|} \bar{\mathbf{x}} - \frac{|V|}{|T|} \overline{\mathbf{x}_V}$, so we need only compute $\overline{\mathbf{x}_V}$ for each partition (similarly for $\overline{\mathbf{y}_T}$).

Lemma 7. Given a validation partition V and a training partition $T = R \setminus V$, then $\overline{\mathbf{x}_T}$, the mean row vector of $\mathbf{X}_T$, is equal to $\frac{N}{|T|} \bar{\mathbf{x}} - \frac{|V|}{|T|} \overline{\mathbf{x}_V}$.

Proof. Using Definition 2 for $\overline{\mathbf{x}_R}$ and $\overline{\mathbf{x}_V}$, we have

$$\begin{aligned} \frac{N}{|T|} \bar{\mathbf{x}} - \frac{|V|}{|T|} \overline{\mathbf{x}_V} &= \frac{N}{|T|} \frac{1}{N} \sum_{n \in R} \mathbf{x}_n - \frac{|V|}{|T|} \frac{1}{|V|} \sum_{n \in V} \mathbf{x}_n \\ &= \frac{1}{|T|} \sum_{n \in R} \mathbf{x}_n - \frac{1}{|T|} \sum_{n \in V} \mathbf{x}_n, \end{aligned}$$

and since $R = T \cup V$ and $T \cap V = \emptyset$, we can rearrange the sum over R ,

$$\frac{1}{|T|} \sum_{n \in T} \mathbf{x}_n + \frac{1}{|T|} \sum_{n \in V} \mathbf{x}_n - \frac{1}{|T|} \sum_{n \in V} \mathbf{x}_n = \frac{1}{|T|} \sum_{n \in T} \mathbf{x}_n = \bar{\mathbf{x}}_T.$$

Lemma 8. Given a validation partition V and a training partition $T = R \setminus V$, then $\bar{\mathbf{y}}_T$, the mean row vector of $\mathbf{Y}_T$, is equal to $\frac{N}{|T|} \bar{\mathbf{y}}_R - \frac{|V|}{|T|} \bar{\mathbf{y}}_V$.

Proof. As proof of Lemma 7. $\square$

We note that Lemma 9 is interesting beyond the purpose of proving correctness, stating that whether you center either $\mathbf{X}$, $\mathbf{Y}$, or both, when centering is performed, the result is $\mathbf{X}_T^c \mathbf{Y}_T^c$ (i.e., the same centered matrix product), which we explore further in Section 6.

Lemma 9. Let $S \subseteq R$, then the matrix products $\mathbf{X}_S^T \bar{\mathbf{Y}}_S$, $\bar{\mathbf{X}}_S^T \mathbf{Y}_S$ and $\bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S$ are all equal to $|S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S$. That is, $\mathbf{X}_S^T \bar{\mathbf{Y}}_S = \bar{\mathbf{X}}_S^T \mathbf{Y}_S = \bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S = |S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S$.

Proof. Let (i, j) be an entry in $\bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S$. Observe that by Definition 2, each column in $\bar{\mathbf{X}}_S^T$ is equal to the column vector $\bar{\mathbf{x}}_S^T$; thus, for all $n \in S$, we have $\bar{\mathbf{x}}_S^T = \bar{\mathbf{x}}_{S_n}^T$ and so $\bar{\mathbf{x}}_{S_i}^T = \bar{\mathbf{x}}_{S_{in}}^T$. Similarly, for all $n \in S$, each row in $\bar{\mathbf{Y}}_S$ is the mean row vector $\bar{\mathbf{y}}_S = \bar{\mathbf{y}}_{S_n}$ and so $\bar{\mathbf{y}}_S = \bar{\mathbf{y}}_{S_{nj}}$.

(a) By Definition 2 and distributivity of multiplication over addition, it follows that

$$\begin{aligned} (\bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S)_{ij} &= \sum_{n \in S} \bar{\mathbf{x}}_{S_{in}}^T \bar{\mathbf{y}}_{S_{nj}} = \sum_{n \in S} \bar{\mathbf{x}}_{S_i}^T \bar{\mathbf{y}}_S \\ &= \bar{\mathbf{x}}_{S_i}^T \bar{\mathbf{y}}_S \sum_{n \in S} 1 = |S| \bar{\mathbf{x}}_{S_i}^T \bar{\mathbf{y}}_S \\ &= |S| (\bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S)_{ij}, \end{aligned}$$

concluding the proof that $\bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S = |S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S$.

(b) By Definition 2 we have $\bar{\mathbf{y}}_S = \bar{\mathbf{y}}_{S_{nj}} = \frac{1}{|S|} \sum_{n \in S} \mathbf{Y}_{S_{nj}}$. So by distributivity of multiplication over addition, we get

$$\begin{aligned} |S| \bar{\mathbf{x}}_{S_i}^T \bar{\mathbf{y}}_S &= |S| \bar{\mathbf{x}}_{S_i}^T \frac{1}{|S|} \sum_{n \in S} \mathbf{Y}_{S_{nj}} = \bar{\mathbf{x}}_{S_i}^T \sum_{n \in S} \mathbf{Y}_{S_{nj}} \\ &= \sum_{n \in S} \bar{\mathbf{x}}_{S_i}^T \mathbf{Y}_{S_{nj}} = \sum_{n \in S} \bar{\mathbf{x}}_{S_{in}}^T \mathbf{Y}_{S_{nj}} \\ &= (\bar{\mathbf{X}}_S^T \mathbf{Y}_S)_{ij}, \end{aligned}$$

showing that $|S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S = \bar{\mathbf{X}}_S^T \mathbf{Y}_S$.

(c) By Definition 2, we have $\bar{\mathbf{x}}_{S_i}^T = \bar{\mathbf{x}}_{S_{in}}^T = \frac{1}{|S|} \sum_{n \in S} \mathbf{X}_{S_{in}}^T$. Similar to the argument in (b) and by commutativity of scalar multiplication, it follows that

$$\begin{aligned} |S| \bar{\mathbf{x}}_{S_i}^T \bar{\mathbf{y}}_S &= |S| \bar{\mathbf{y}}_S \frac{1}{|S|} \sum_{n \in S} \mathbf{X}_{S_{in}}^T = \bar{\mathbf{y}}_S \sum_{n \in S} \mathbf{X}_{S_{in}}^T \\ &= \sum_{n \in S} \bar{\mathbf{y}}_S \mathbf{X}_{S_{in}}^T = \sum_{n \in S} \mathbf{X}_{S_{in}}^T \bar{\mathbf{y}}_{S_{nj}} \\ &= (\mathbf{X}_S^T \bar{\mathbf{Y}}_S)_{ij}, \end{aligned}$$

hence, $|S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S = \bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S$. Combining (a), (b), and (c), it follows that $\mathbf{X}_S^T \bar{\mathbf{Y}}_S = \bar{\mathbf{X}}_S^T \mathbf{Y}_S = \bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S = |S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S$. $\square$

Lemma 10. Let $S \subseteq R$, then the matrix products $\mathbf{X}_S^T \bar{\mathbf{X}}_S$, $\bar{\mathbf{X}}_S^T \mathbf{X}_S$ and $\bar{\mathbf{X}}_S^T \bar{\mathbf{X}}_S$ are all equal to $|S| \bar{\mathbf{x}}_S^T \bar{\mathbf{x}}_S$. That is, $\mathbf{X}_S^T \bar{\mathbf{X}}_S = \bar{\mathbf{X}}_S^T \mathbf{X}_S = \bar{\mathbf{X}}_S^T \bar{\mathbf{X}}_S = |S| \bar{\mathbf{x}}_S^T \bar{\mathbf{x}}_S$.

Proof. As proof of Lemma 9 considering $\mathbf{X}_S$ in place of $\mathbf{Y}_S$. $\square$

Proposition 7. Let $S \subseteq R$ and $\mathbf{X}_S^c$ and $\mathbf{Y}_S^c$ be centerings per Definition 3. Then $\mathbf{X}_S^c \mathbf{Y}_S^c = \mathbf{X}_S^T \mathbf{Y}_S - |S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S$.

Proof. We expand according to Definition 3, using that transposition is distributive, and from Lemma 9 use that $\mathbf{X}_S^T \bar{\mathbf{Y}}_S = \bar{\mathbf{X}}_S^T \mathbf{Y}_S = \bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S = |S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S$ to derive

$$\begin{aligned} \mathbf{X}_S^c \mathbf{Y}_S^c &= (\mathbf{X}_S - \bar{\mathbf{X}}_S)^T (\mathbf{Y}_S - \bar{\mathbf{Y}}_S) \\ &= \mathbf{X}_S^T \mathbf{Y}_S - \bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S - \bar{\mathbf{X}}_S^T \mathbf{Y}_S + \bar{\mathbf{X}}_S^T \bar{\mathbf{Y}}_S \\ &= \mathbf{X}_S^T \mathbf{Y}_S - |S| \bar{\mathbf{x}}_S^T \bar{\mathbf{y}}_S. \end{aligned}$$

$\square$

Proposition 8. Let $S \subseteq R$ and $\mathbf{X}_S^c$ be a centering per Definition 3. Then $\mathbf{X}_S^c \mathbf{X}_S^c = \mathbf{X}_S^T \mathbf{X}_S - |S| \bar{\mathbf{x}}_S^T \bar{\mathbf{x}}_S$.

Proof. As proof of Proposition 7 using Lemma 10 instead of Lemma 9. $\square$

As mentioned above, Lindgren et al. [16] describe a method for centering validation partition submatrices to support such preprocessing efficiently during cross-validation. The following result uses our notation to express their Equation (20) and is used to show that their method does not satisfy Definition 3 and may lead to leakage by preprocessing.

Lemma 11. The method by Lindgren et al. [16] for obtaining a centered matrix product for training partition T is given by $(\mathbf{X}_T^T \mathbf{X}_T)_{ij} - N \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j (2 - 1/P) + |T| \bar{\mathbf{x}}_i \bar{\mathbf{x}}_{Tj} + |T| \bar{\mathbf{x}}_j \bar{\mathbf{x}}_{Ti}$.

Proof. Let V be a validation partition, $T = R \setminus V$ a training partition, and (i, j) an entry in the matrix product. Equation (20) of Lindgren et al. [16] states the centered matrix product for T is

$$(\mathbf{X}^T \mathbf{X})_{ij} - (\mathbf{X}_V^T \mathbf{X}_V)_{ij} - \bar{\mathbf{x}}_i \sum_{n \in V} \mathbf{x}_{nj} - \bar{\mathbf{x}}_j \sum_{n \in V} \mathbf{x}_{ni} + N/P \cdot \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j.$$

Using Lemma 2 (the fractionated subparts insight due to Lindgren et al. [16]), we can collapse the two left-most terms, and since $V = R \setminus T$, we can express the sums over V using R and T . Then, following the definition of mean row vectors and factoring, we arrive at the conclusion,

$$\begin{aligned} &(\mathbf{X}_T^T \mathbf{X}_T)_{ij} - \bar{\mathbf{x}}_i \sum_{n \in V} \mathbf{x}_{nj} - \bar{\mathbf{x}}_j \sum_{n \in V} \mathbf{x}_{ni} + N/P \cdot \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j \\ &= (\mathbf{X}_T^T \mathbf{X}_T)_{ij} - \bar{\mathbf{x}}_i \left(\sum_{n \in R} \mathbf{x}_{nj} - \sum_{n \in T} \mathbf{x}_{nj} \right) \\ &\quad - \bar{\mathbf{x}}_j \left(\sum_{n \in R} \mathbf{x}_{ni} - \sum_{n \in T} \mathbf{x}_{ni} \right) + N/P \cdot \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j \\ &= (\mathbf{X}_T^T \mathbf{X}_T)_{ij} - \bar{\mathbf{x}}_i N \bar{\mathbf{x}}_j + \bar{\mathbf{x}}_i |T| \bar{\mathbf{x}}_{Tj} - \bar{\mathbf{x}}_j N \bar{\mathbf{x}}_i + \bar{\mathbf{x}}_j |T| \bar{\mathbf{x}}_{Ti} + N/P \cdot \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j \\ &= (\mathbf{X}_T^T \mathbf{X}_T)_{ij} - N \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j - N \bar{\mathbf{x}}_j \bar{\mathbf{x}}_i + N/P \cdot \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j + |T| \bar{\mathbf{x}}_i \bar{\mathbf{x}}_{Tj} + |T| \bar{\mathbf{x}}_j \bar{\mathbf{x}}_{Ti} \\ &= (\mathbf{X}_T^T \mathbf{X}_T)_{ij} - N \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j (2 - 1/P) + |T| \bar{\mathbf{x}}_i \bar{\mathbf{x}}_{Tj} + |T| \bar{\mathbf{x}}_j \bar{\mathbf{x}}_{Ti}. \end{aligned}$$

$\square$

Under our definition of centering, we can apply Proposition 8 which states that an entry (i, j) in the centered matrix product for a training partition T is $(\mathbf{X}_T^T \mathbf{X}_T)_{ij} - |T| \left(\overline{\mathbf{x}_T^T \mathbf{x}_T} \right)_{ij}$. Together with Lemma 11, we therefore have

$$|T| \left(\overline{\mathbf{x}_T^T \mathbf{x}_T} \right)_{ij} = N \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j (2 - 1/P) - |T| \bar{\mathbf{x}}_i \bar{\mathbf{x}}_{Tj} - |T| \bar{\mathbf{x}}_j \bar{\mathbf{x}}_{Ti}.$$

To derive a contradiction, consider the case where columns i, j in the training partition are centered (mean of 0) but that i, j are not centered in the data set as a whole. This means that $(\bar{\mathbf{x}}_T)_i = (\bar{\mathbf{x}}_T)_j = \left(\overline{\mathbf{x}_T^T \mathbf{x}_T} \right)_{ij} = 0$, $\bar{\mathbf{x}}_i \neq 0$, and $\bar{\mathbf{x}}_j \neq 0$. Therefore, we must have $0 = N \bar{\mathbf{x}}_i \bar{\mathbf{x}}_j (2 - 1/P)$. This is a contradiction since $\bar{\mathbf{x}}_i \neq 0$, $\bar{\mathbf{x}}_j \neq 0$, and $N \geq P \geq 2$. Moreover, the nonzero quantity subtracted depends on samples not in the training partition, thus leading to information leakage during cross-validation. In contrast, we now show that Algorithm 5 performs centering without such leakage.

Proposition 9. *Correctness, Algorithm 5* In step 9 of Algorithm 4, $\text{cXTX}_T = \mathbf{X}_T^c \mathbf{X}_T^c$ and $\text{cXTY}_T = \mathbf{X}_T^c \mathbf{Y}_T^c$ and the variables are identical to the quantities cXTX_T and cXTY_T computed in steps 14 and 15 of Algorithm 5.

Proof. By Lemma 1, both algorithms select validation partition V according to Definition 1. For Algorithm 4, $T = R \setminus V$ by step 5, and step 8 computes cX_T as $\mathbf{X}_T^c = \mathbf{X}_T - \bar{\mathbf{X}}_T$ and cY_T as $\mathbf{Y}_T^c = \mathbf{Y}_T - \bar{\mathbf{Y}}_T$ as per Definition 3. Therefore, after step 9, we have that cXTX_T equals $\mathbf{X}_T^c \mathbf{X}_T^c$ and cXTY_T equals $\mathbf{X}_T^c \mathbf{Y}_T^c$.

Considering now Algorithm 5, we have in step 8 that $\text{XTX}_T = \mathbf{X}_T^T \mathbf{X}_T$ and $\text{XTY}_T = \mathbf{X}_T^T \mathbf{Y}_T$ as shown in Proposition 1. That $\mu_{\mathbf{x}_T}$ in step 11 equals $\bar{\mathbf{x}}_T$ follows from Lemma 7, the selected validation partition V , and the quantities computed in steps 9 and 10. Similarly, we have in step 12 that $\mu_{\mathbf{y}_T}$ equals $\bar{\mathbf{y}}_T$ by Lemma 8. Therefore, step 13 computes $\mu_{\mathbf{x}} \mathbf{T} \mathbf{x}_T$ to equal $|T| \overline{\mathbf{x}_T^T \mathbf{x}_T}$ and $\mu_{\mathbf{x}} \mathbf{T} \mathbf{y}_T$ to equal $|T| \overline{\mathbf{x}_T^T \mathbf{y}_T}$. In step 14, it therefore follows from Proposition 8 that cXTX_T equals $\mathbf{X}_T^c \mathbf{X}_T^c$, and in step 15, it follows from Proposition 7 that cXTY_T equals $\mathbf{X}_T^c \mathbf{Y}_T^c$. Hence, cXTX_T and cXTY_T equal the quantities computed in step 9 of Algorithm 4. $\square$

4.2 | Computational Complexity

We now analyze the running time and space complexity of Algorithms 4 and 5.

Proposition 10. *Algorithm 4* requires $\Theta(\text{PNK}(K + M))$ operations.

Proof. Storing the inputs and steps 1–7 of Algorithm 2 are equivalent, in terms of running time, to storing the inputs and steps 1–6 and 9 of Algorithm 4, thus requiring $\Theta(\text{PNK}(K + M))$ operations as shown in the proof of Proposition 2. Computing the means in step 7 is over $|T_p|$ rows and K , respectively, M , columns and so requires $\Theta(|T_p|(K + M))$ operations. Step 8 requires $\Theta(|T_p|(K + M))$ operations for matrix subtractions. For P iterations, steps 7 and 8 require $\Theta\left(\sum_{p=1}^P |T_p|(K + M)\right)$ operations,

which by Lemma 6 is $\Theta(\text{PNK}(K + M))$. Thus, Algorithm 4 requires $\Theta(\text{PNK}(K + M))$ operations. $\square$

Proposition 11. *Algorithm 5* requires $\Theta(\text{NK}(K + M))$ operations.

Proof. In step 3 computing data set means requires $\Theta(\text{NK}) + \Theta(\text{NM}) = \Theta(\text{N}(K + M))$ operations. The remaining computations in storing the inputs and steps 1–8 are equivalent to storing the inputs and steps 1–8 of Algorithm 3, so as in the proof of Proposition 3 require $\Theta(\text{NK}(K + M))$ operations.

Consider a partition $p \in \{1, \dots, P\}$ with validation partition V_p and training partition T_p iterated over in step 4. Computing means in step 9 requires $\Theta(|V_p|(K + M))$ operations, while steps 10–12 require $\Theta(|V_p| + K + M)$ operations to obtain $\bar{\mathbf{x}}_T \in \mathbb{R}^K$ and $\bar{\mathbf{y}}_T \in \mathbb{R}^M$. The matrix multiplications in step 13 require $\Theta(K^2)$ and $\Theta(KM)$ operations, as do the matrix subtractions in steps 14 and 15. Steps 9 through 15 therefore require $\Theta(|V_p|(K + M)) + \Theta(|V_p| + K + M) + \Theta(K^2) + \Theta(KM)$ which is $\Theta(|V_p|(K + M) + K^2 + KM)$. In conducting P iterations and using that $\sum_{p=1}^P |V_p| = N$, this gives

$$\begin{aligned} & \Theta\left(\sum_{p=1}^P |V_p|(K + M) + K^2 + KM\right) \\ &= \Theta\left((K + M) \sum_{p=1}^P |V_p| + \left(K^2 + KM\right) \sum_{p=1}^P 1\right) \\ &= \Theta(N(K + M) + PK^2 + PKM) \end{aligned}$$

total operations. Since $P \leq N$ this is at most $O(N(K + M) + NK^2 + NKM) = O(N(K + M + K^2 + KM)) = O(\text{NK}(K + M))$ operations. The total number of operations of Algorithm 5 is therefore $\Theta(\text{NK}(K + M)) + O(\text{NK}(K + M)) = \Theta(\text{NK}(K + M))$. $\square$

Proposition 12. *Algorithm 4* requires $\Theta(P)$ more operations than Algorithm 5.

Proof. By Propositions 10 and 11, the ratio of operations is $\frac{\Theta(\text{PNK}(K + M))}{\Theta(\text{NK}(K + M))} = \Theta(P)$. $\square$

With Proposition 12, we have shown that Algorithm 5 is asymptotically faster than Algorithm 4, as also demonstrated for cross-validation without preprocessing in Proposition 4. Below, we show that this comes at no cost in terms of space complexity; that is, Algorithms 4 and 5 are of the same space complexity.

Proposition 13. *Algorithm 4* requires storing $\Theta((K + N)(K + M))$ entries.

Proof. For step 7, we store the stacked mean row vectors using $|T|K + |T|M$ entries, which is also required for the centerings in step 8. Storing cXTX_T and cXTY_T in step 9 in Algorithm 4 requires the same amount of entries as storing cXTX_T and cXTY_T in step 7 in Algorithm 2. The remainder of Algorithm 4 is equivalent to Algorithm 2, so as in the proof of Proposition 5 requires storing $\Theta((K + N)(K + M))$ entries. Using $|T| < N$, the total number of entries is $O(N(K + M)) + \Theta((K + N)(K + M)) = \Theta((K + N)(K + M))$. $\square$

Proposition 14. Algorithm 5 requires storing $\Theta((K+N)(K+M))$ entries.

Proof. In step 3, storing $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ requires $\Theta(K+M)$ entries. The remaining storage required for steps 1–8 is equivalent to the storage required by Algorithm 3, so as in the proof of Proposition 6 require $\Theta((N+K)(K+M))$ entries. Step 9 stores $\mu\mathbf{x}_V$ and $\mu\mathbf{y}_V$ requiring $K+M$ entries, the same number of entries needed for $\mu\mathbf{x}_T$ and $\mu\mathbf{y}_T$ in steps 11 and 12. Therefore, steps 9–12 require $\Theta(K+M)$ entries. In step 13, the variables $\mu\mathbf{xTx}_T$ and $\mu\mathbf{xTy}_T$ require K^2 , respectively, KM entries, as do $\mathbf{cXTx}_T$ and $\mathbf{cXTy}_T$ in steps 14 and 15. Therefore, steps 9–15 store $\Theta(K+M) + 2K^2 + 2KM = \Theta(K(K+M))$ entries, meaning the total required is $\Theta((K+N)(K+M)) + \Theta(K(K+M)) = \Theta((K+N)(K+M))$. $\square$

5 | Cross-Validation With Centering and Scaling

We proceed to extend Algorithm 5 to support scaling; that is, we compute $\mathbf{X}_T^{\mathbf{cT}}\mathbf{X}_T^{\mathbf{c}}$ and $\mathbf{X}_T^{\mathbf{cT}}\mathbf{Y}_T^{\mathbf{c}}$ where $\mathbf{X}_T^{\mathbf{c}}$ and $\mathbf{Y}_T^{\mathbf{c}}$ are scaled by the column-wise sample standard deviation computed over the training partition T . Our method computes the sample standard deviation only once for the entire data set, and then, for each partition p , the standard deviation contribution of samples in the validation partition is removed. Adding this important pre-processing extension impacts neither time nor space complexity when compared to Algorithm 5, and to our knowledge, it has not been identified previously.

In what follows, we use Hadamard (element-wise) operators for multiplication ($\odot$), division ($\oslash$), and exponentiation ($\circ$).

Definition 4. Let $S \subseteq R$ denote a set of rows such that $|S| \geq 2$. The (Bessels's corrected) sample standard deviation (row) vector $\widehat{\mathbf{x}}_S$ has width K and is the sample standard deviation of each column in $\mathbf{X}_S$, and $\widehat{\mathbf{y}}_S$ has width M and is the sample standard deviation (row) vector of $\mathbf{Y}_S$, given by

$$\widehat{\mathbf{x}}_S = \left(\frac{1}{|S|-1} \sum_{n \in S} (\mathbf{X}_n - \bar{\mathbf{x}}_S)^{\circ 2} \right)^{\circ \frac{1}{2}} \text{ and } \widehat{\mathbf{y}}_S = \left(\frac{1}{|S|-1} \sum_{n \in S} (\mathbf{Y}_n - \bar{\mathbf{y}}_S)^{\circ 2} \right)^{\circ \frac{1}{2}}.$$

Stacking the sample standard deviation row vectors yields matrices $\widehat{\mathbf{X}}_S$ with dimension $|S| \times K$ and $\widehat{\mathbf{Y}}_S$ with dimension $|S| \times M$ given by

$$\widehat{\mathbf{X}}_S = \begin{bmatrix} \widehat{\mathbf{x}}_S \\ \vdots \\ \widehat{\mathbf{x}}_S \end{bmatrix} \text{ and } \widehat{\mathbf{Y}}_S = \begin{bmatrix} \widehat{\mathbf{y}}_S \\ \vdots \\ \widehat{\mathbf{y}}_S \end{bmatrix}.$$

For brevity, we adopt the convention that $\widehat{\mathbf{X}}_S^{\mathbf{T}}$ and $\widehat{\mathbf{x}}_S^{\mathbf{T}}$ are, respectively, given by $(\widehat{\mathbf{X}}_S)^{\mathbf{T}}$ and $(\widehat{\mathbf{x}}_S)^{\mathbf{T}}$. We also omit parentheses to denote the i 'th element of vector $\widehat{\mathbf{x}}_S^{\mathbf{T}}$ by $\widehat{\mathbf{x}}_{S_i}^{\mathbf{T}}$ and the i 'th element of vector $\widehat{\mathbf{y}}_S$ by $\widehat{\mathbf{y}}_{S_i}$.

Definition 5. Let $S \subseteq R$ denote a set of rows such that $|S| \geq 2$ and let $\mathbf{X}_S^{\mathbf{c}}$ and $\mathbf{Y}_S^{\mathbf{c}}$ be given according to Definition 3. The centering and (sample standard deviation) scaling of $\mathbf{X}_S$ is $\mathbf{X}_S^{\mathbf{cs}} = \mathbf{X}_S^{\mathbf{c}} \oslash \widehat{\mathbf{X}}_S$, and the centering and (sample standard deviation) scaling of $\mathbf{Y}_S$ is $\mathbf{Y}_S^{\mathbf{cs}} = \mathbf{Y}_S^{\mathbf{c}} \oslash \widehat{\mathbf{Y}}_S$.

To support centering and scaling of $\mathbf{X}_{T_p}$ and $\mathbf{Y}_{T_p}$ for a partition p , we subtract the column-wise means and subsequently divide element-wise by the column-wise sample standard deviations before taking matrix products, namely,

$$\begin{aligned} \mathbf{X}_{T_p}^{\mathbf{csT}}\mathbf{X}_{T_p}^{\mathbf{cs}} &= \left(\mathbf{X}_{T_p}^{\mathbf{cT}} \oslash \widehat{\mathbf{X}}_{T_p}^{\mathbf{T}} \right) \left(\mathbf{X}_{T_p}^{\mathbf{c}} \oslash \widehat{\mathbf{X}}_{T_p} \right) \text{ and} \\ \mathbf{X}_{T_p}^{\mathbf{csT}}\mathbf{Y}_{T_p}^{\mathbf{cs}} &= \left(\mathbf{X}_{T_p}^{\mathbf{cT}} \oslash \widehat{\mathbf{X}}_{T_p}^{\mathbf{T}} \right) \left(\mathbf{Y}_{T_p}^{\mathbf{c}} \oslash \widehat{\mathbf{Y}}_{T_p} \right). \end{aligned}$$

We name these quantities centered and scaled matrix products. The condition $|S| \geq 2$ in the definitions above motivates the following requirement.

Definition 6. A partitioning $\mathcal{P}$ of R is *scalable* when $|R \setminus V_p| = |T_p| \geq 2$ for all $p \in \{1, \dots, P\}$.

In Algorithm 6, we make the immediate extension of Algorithm 4 so that $\mathbf{X}_T^{\mathbf{c}}$ and $\mathbf{Y}_T^{\mathbf{c}}$ are scaled by the sample standard deviations. Then we build on Algorithm 5 in Algorithm 7 with the purpose of efficiently computing $\mathbf{X}_T^{\mathbf{csT}}\mathbf{X}_T^{\mathbf{cs}}$ and $\mathbf{X}_T^{\mathbf{csT}}\mathbf{Y}_T^{\mathbf{cs}}$ for each partition p with training partition $T = T_p$. In both Algorithms 6 and 7, we consider *scalable* partitionings and

Algorithm 6 Baseline Cross-Validation Algorithm with Centering and Scaling

Input: $\mathbf{X} \in \mathbb{R}^{N \times K}$, $\mathbf{Y} \in \mathbb{R}^{N \times M}$, $\mathcal{P}$ is a scalable partitioning of $R = \{1, \dots, N\}$

- 1: $R \leftarrow \{1, \dots, N\}$
 - 2: Let $\mathcal{V}$ be the result of executing Algorithm 1 with $\mathcal{P}$ and R as input
 - 3: **for** each partition $p \in \{1, \dots, P\}$ **do**
 - 4: $V \leftarrow \mathcal{V}[p]$
 - 5: $T \leftarrow R \setminus V$
 - 6: $\mathbf{X}_T \leftarrow \mathbf{X}_T$, $\mathbf{Y}_T \leftarrow \mathbf{Y}_T$
 - 7: $\mu\mathbf{X}_T \leftarrow \bar{\mathbf{X}}_T$, $\mu\mathbf{Y}_T \leftarrow \bar{\mathbf{Y}}_T$
 - 8: $\mathbf{cX}_T \leftarrow \mathbf{X}_T - \mu\mathbf{X}_T$, $\mathbf{cY}_T \leftarrow \mathbf{Y}_T - \mu\mathbf{Y}_T$ ▷ Obtain $\mathbf{X}_T^{\mathbf{c}}$ and $\mathbf{Y}_T^{\mathbf{c}}$.
 - 9: $\sigma\mathbf{X}_T \leftarrow \widehat{\mathbf{X}}_T$, $\sigma\mathbf{Y}_T \leftarrow \widehat{\mathbf{Y}}_T$
 - 10: Replace with 1 any entry in $\sigma\mathbf{X}_T$ and $\sigma\mathbf{Y}_T$ that is 0.
 - 11: $\mathbf{csX}_T \leftarrow \mathbf{cX}_T \oslash \sigma\mathbf{X}_T$, $\mathbf{csY}_T \leftarrow \mathbf{cY}_T \oslash \sigma\mathbf{Y}_T$ ▷ Obtain $\mathbf{X}_T^{\mathbf{cs}}$ and $\mathbf{Y}_T^{\mathbf{cs}}$
 - 12: $\mathbf{csXTX}_T \leftarrow \mathbf{csX}_T^{\mathbf{T}}\mathbf{csX}_T$, $\mathbf{csXTY}_T \leftarrow \mathbf{csX}_T^{\mathbf{T}}\mathbf{csY}_T$ ▷ Obtain $\mathbf{X}_T^{\mathbf{csT}}\mathbf{X}_T^{\mathbf{cs}}$ and $\mathbf{X}_T^{\mathbf{csT}}\mathbf{Y}_T^{\mathbf{cs}}$
 - 13: **end for**
-

Algorithm 7 Fast Cross-Validation Algorithm with Centering and Scaling

Input: $\mathbf{X} \in \mathbb{R}^{N \times K}$, $\mathbf{Y} \in \mathbb{R}^{N \times M}$, $\mathcal{P}$ is a scalable partitioning of $R = \{1, \dots, N\}$

```

1:  $R \leftarrow \{1, \dots, N\}$ 
2: Let  $\mathcal{V}$  be the result of executing Algorithm 1 with  $\mathcal{P}$  and  $R$  as input
3:  $\mathbf{X}\mathbf{T}\mathbf{X} \leftarrow \mathbf{X}^T\mathbf{X}$ ,  $\mathbf{X}\mathbf{T}\mathbf{Y} \leftarrow \mathbf{X}^T\mathbf{Y}$ ,  $\mu_X \leftarrow \bar{\mathbf{x}}$ ,  $\mu_Y \leftarrow \bar{\mathbf{y}}$ 
4:  $\Sigma\mathbf{X} \leftarrow \sum_{n \in R} \mathbf{X}_n$ ,  $\Sigma\mathbf{Y} \leftarrow \sum_{n \in R} \mathbf{Y}_n$ ,  $\Sigma sq\mathbf{X} \leftarrow \sum_{n \in R} (\mathbf{X}_n^{\circ 2})$ ,  $\Sigma sq\mathbf{Y} \leftarrow \sum_{n \in R} (\mathbf{Y}_n^{\circ 2})$ 
5: for each partition  $p \in \{1, \dots, P\}$  do
6:    $V \leftarrow \mathcal{V}[p]$ 
7:    $\mathbf{X}_V \leftarrow \mathbf{X}_V$ ,  $\mathbf{Y}_V \leftarrow \mathbf{Y}_V$ 
8:    $\mathbf{X}\mathbf{T}\mathbf{X}_V \leftarrow \mathbf{X}_V^T\mathbf{X}_V$ ,  $\mathbf{X}\mathbf{T}\mathbf{Y}_V \leftarrow \mathbf{X}_V^T\mathbf{Y}_V$ 
9:    $\mathbf{X}\mathbf{T}\mathbf{X}_T \leftarrow \mathbf{X}\mathbf{T}\mathbf{X} - \mathbf{X}\mathbf{T}\mathbf{X}_V$ ,  $\mathbf{X}\mathbf{T}\mathbf{Y}_T \leftarrow \mathbf{X}\mathbf{T}\mathbf{Y} - \mathbf{X}\mathbf{T}\mathbf{Y}_V$  ▷ Obtain  $\mathbf{X}_T^T\mathbf{X}_T$  and  $\mathbf{X}_T^T\mathbf{Y}_T$ 
10:   $\mu_{X_V} \leftarrow \bar{\mathbf{x}}_V$ ,  $\mu_{Y_V} \leftarrow \bar{\mathbf{y}}_V$ 
11:   $N_V \leftarrow |V|$ ,  $N_T \leftarrow N - N_V$ 
12:   $\mu_{X_T} \leftarrow \frac{N}{N_T} \mu_X - \frac{N_V}{N_T} \mu_{X_V}$  ▷ Obtain  $\bar{\mathbf{x}}_T$ 
13:   $\mu_{Y_T} \leftarrow \frac{N}{N_T} \mu_Y - \frac{N_V}{N_T} \mu_{Y_V}$  ▷ Obtain  $\bar{\mathbf{y}}_T$ 
14:   $\mu_{X}\mathbf{T}\mathbf{x}_T \leftarrow N_T (\mu_{X_T}^T \mu_{X_T})$ ,  $\mu_{X}\mathbf{T}\mathbf{y}_T \leftarrow N_T (\mu_{X_T}^T \mu_{Y_T})$  ▷ Obtain  $|\mathbf{T}\mathbf{x}_T^T \bar{\mathbf{x}}_T|$  and  $|\mathbf{T}\mathbf{x}_T^T \bar{\mathbf{y}}_T|$ 
15:   $\mathbf{c}\mathbf{X}\mathbf{T}\mathbf{X}_T \leftarrow \mathbf{X}\mathbf{T}\mathbf{X}_T - \mu_{X}\mathbf{T}\mathbf{x}_T$  ▷ Obtain  $\mathbf{X}_T^{\mathbf{c}T}\mathbf{X}_T^{\mathbf{c}}$ 
16:   $\mathbf{c}\mathbf{X}\mathbf{T}\mathbf{Y}_T \leftarrow \mathbf{X}\mathbf{T}\mathbf{Y}_T - \mu_{X}\mathbf{T}\mathbf{y}_T$  ▷ Obtain  $\mathbf{X}_T^{\mathbf{c}T}\mathbf{Y}_T^{\mathbf{c}}$ 
17:   $\Sigma\mathbf{X}_V \leftarrow \sum_{n \in V} \mathbf{X}_n$ ,  $\Sigma\mathbf{Y}_V \leftarrow \sum_{n \in V} \mathbf{Y}_n$ 
18:   $\Sigma\mathbf{X}_T \leftarrow \Sigma\mathbf{X} - \Sigma\mathbf{X}_V$ ,  $\Sigma\mathbf{Y}_T \leftarrow \Sigma\mathbf{Y} - \Sigma\mathbf{Y}_V$  ▷ Obtain  $\Sigma_{n \in T} \mathbf{X}_n$  and  $\Sigma_{n \in T} \mathbf{Y}_n$ 
19:   $\Sigma sq\mathbf{X}_V \leftarrow \sum_{n \in V} (\mathbf{X}_n^{\circ 2})$ ,  $\Sigma sq\mathbf{Y}_V \leftarrow \sum_{n \in V} (\mathbf{Y}_n^{\circ 2})$ 
20:   $\Sigma sq\mathbf{X}_T \leftarrow \Sigma sq\mathbf{X} - \Sigma sq\mathbf{X}_V$  ▷ Obtain  $\Sigma_{n \in T} (\mathbf{X}_n^{\circ 2})$ 
21:   $\Sigma sq\mathbf{Y}_T \leftarrow \Sigma sq\mathbf{Y} - \Sigma sq\mathbf{Y}_V$  ▷ Obtain  $\Sigma_{n \in T} (\mathbf{Y}_n^{\circ 2})$ 
22:   $\sigma_{X_T} \leftarrow \left( \frac{1}{N_T-1} (N_T \mu_{X_T}^{\circ 2} + \Sigma sq\mathbf{X}_T - 2\mu_{X_T} \odot \Sigma\mathbf{X}_T) \right)^{\frac{1}{2}}$  ▷ Obtain  $\hat{\mathbf{x}}_T$ 
23:   $\sigma_{Y_T} \leftarrow \left( \frac{1}{N_T-1} (N_T \mu_{Y_T}^{\circ 2} + \Sigma sq\mathbf{Y}_T - 2\mu_{Y_T} \odot \Sigma\mathbf{Y}_T) \right)^{\frac{1}{2}}$  ▷ Obtain  $\hat{\mathbf{y}}_T$ 
24:  Replace with 1 any entry in  $\sigma_{X_T}$  and  $\sigma_{Y_T}$  that is 0.
25:   $\sigma_{X}\mathbf{T}\mathbf{x}_T \leftarrow \sigma_{X_T}^T \sigma_{X_T}$ ,  $\sigma_{X}\mathbf{T}\mathbf{y}_T \leftarrow \sigma_{X_T}^T \sigma_{Y_T}$  ▷ Obtain  $\hat{\mathbf{x}}_T^T \hat{\mathbf{x}}_T$  and  $\hat{\mathbf{x}}_T^T \hat{\mathbf{y}}_T$ 
26:   $\mathbf{c}\mathbf{s}\mathbf{X}\mathbf{T}\mathbf{X}_T \leftarrow \mathbf{c}\mathbf{X}\mathbf{T}\mathbf{X}_T \odot \sigma_{X}\mathbf{T}\mathbf{x}_T$  ▷ Obtain  $\mathbf{X}_T^{\mathbf{cs}T}\mathbf{X}_T^{\mathbf{cs}}$ 
27:   $\mathbf{c}\mathbf{s}\mathbf{X}\mathbf{T}\mathbf{Y}_T \leftarrow \mathbf{c}\mathbf{X}\mathbf{T}\mathbf{Y}_T \odot \sigma_{X}\mathbf{T}\mathbf{y}_T$  ▷ Obtain  $\mathbf{X}_T^{\mathbf{cs}T}\mathbf{Y}_T^{\mathbf{cs}}$ 
28: end for

```

employ the standard practice of replacing 0-entries with 1-entries when using the standard deviation vectors and matrices for scaling (to avoid division by zero). Correctness and complexity results follow.

5.1 | Correctness

As in Section 4, enabling fast sample standard deviation-scaling requires us to compute this quantity exclusively over rows in the validation partition. In Lemmas 12 and 13, we show how the training partition sample standard deviations $\hat{\mathbf{x}}_T^T$ and $\hat{\mathbf{y}}_T^T$ are rearrangeable into simpler constituent terms. While these quantities are based on the training partition, Lemma 14 shows they are computable by removing the contribution from validation partition samples from the constituent terms over the entire set of samples, which we need only compute once.

Lemma 12. Let $S \subseteq R$, $|S| \geq 2$, and $\hat{\mathbf{x}}_S$ be the sample standard deviation vector. Then

$$\hat{\mathbf{x}}_S = \left(\frac{1}{|S|-1} \left(|S| \bar{\mathbf{x}}_S^{\circ 2} + \sum_{n \in S} (\mathbf{X}_n^{\circ 2}) - 2\bar{\mathbf{x}}_S \odot \sum_{n \in S} \mathbf{X}_n \right) \right)^{\frac{1}{2}}.$$

Proof. Expanding the Hadamard exponentiation in $\hat{\mathbf{x}}_S$ from Definition 4 yields

$$\begin{aligned} & \left(\frac{1}{|S|-1} \sum_{n \in S} ((\mathbf{X}_n - \bar{\mathbf{x}}_S)^{\circ 2}) \right)^{\frac{1}{2}} \\ &= \left(\frac{1}{|S|-1} \sum_{n \in S} (\mathbf{X}_n^{\circ 2} + \bar{\mathbf{x}}_S^{\circ 2} - 2\bar{\mathbf{x}}_S \odot \mathbf{X}_n) \right)^{\frac{1}{2}}. \end{aligned}$$

By commutativity of addition, the distributivity of Hadamard multiplication over addition, simplifying the summation, and rearranging terms, we get

$$\begin{aligned} & \left(\frac{1}{|S|-1} \left(\sum_{n \in S} (\bar{\mathbf{x}}_S^{\circ 2}) + \sum_{n \in S} (\mathbf{X}_n^{\circ 2}) - \sum_{n \in S} 2\bar{\mathbf{x}}_S \odot \mathbf{X}_n \right) \right)^{\frac{1}{2}} \\ &= \left(\frac{1}{|S|-1} \left(|S| \bar{\mathbf{x}}_S^{\circ 2} + \sum_{n \in S} (\mathbf{X}_n^{\circ 2}) - 2\bar{\mathbf{x}}_S \odot \sum_{n \in S} \mathbf{X}_n \right) \right)^{\frac{1}{2}}. \end{aligned}$$

□

Lemma 13. Let $S \subseteq R$, $|S| \geq 2$, and $\hat{\mathbf{y}}_S$ be the sample standard deviation vector. Then

$$\hat{\mathbf{y}}_S = \left(\frac{1}{|S|-1} \left(|S| \bar{\mathbf{y}}_S^{\circ 2} + \sum_{n \in S} (\mathbf{Y}_n^{\circ 2}) - 2\bar{\mathbf{y}}_S \odot \sum_{n \in S} \mathbf{Y}_n \right) \right)^{\frac{1}{2}}.$$

Proof. As proof of Lemma 12, considering $\widehat{\mathbf{y}}_S$ in place of $\widehat{\mathbf{x}}_S$. $\square$

Lemma 14. Let p be a partition, $T = T_p$ a training partition, and $V = V_p$ a validation partition. Then

$$\begin{aligned}\Sigma_{n \in T} \mathbf{X}_n &= \Sigma_{n \in R} \mathbf{X}_n - \Sigma_{n \in V} \mathbf{X}_n, \\ \Sigma_{n \in T} \mathbf{Y}_n &= \Sigma_{n \in R} \mathbf{Y}_n - \Sigma_{n \in V} \mathbf{Y}_n, \\ \Sigma_{n \in T} (\mathbf{X}_n^{\circ 2}) &= \Sigma_{n \in R} (\mathbf{X}_n^{\circ 2}) - \Sigma_{n \in V} (\mathbf{X}_n^{\circ 2}), \text{ and} \\ \Sigma_{n \in T} (\mathbf{Y}_n^{\circ 2}) &= \Sigma_{n \in R} (\mathbf{Y}_n^{\circ 2}) - \Sigma_{n \in V} (\mathbf{Y}_n^{\circ 2}).\end{aligned}$$

Proof. By Definition 1, we have that $T \cap V = \emptyset$ and $T \cup V = R$. Therefore,

$$\sum_{n \in T} \mathbf{X}_n + \sum_{n \in V} \mathbf{X}_n = \sum_{n \in T \cup V} \mathbf{X}_n = \sum_{n \in R} \mathbf{X}_n;$$

hence, $\Sigma_{n \in T} \mathbf{X}_n = \Sigma_{n \in R} \mathbf{X}_n - \Sigma_{n \in V} \mathbf{X}_n$. The remaining equalities are shown similarly. $\square$

With Propositions 15 and 16, we can efficiently compute $\mathbf{X}_S^{\text{csT}} \mathbf{X}_S^{\text{cs}}$ and $\mathbf{X}_S^{\text{csT}} \mathbf{Y}_S^{\text{cs}}$ given the (already) centered $\mathbf{X}_S^{\text{cT}} \mathbf{X}_S^{\text{c}}$ and $\mathbf{X}_S^{\text{cT}} \mathbf{Y}_S^{\text{c}}$, by using only an outer vector product and Hadamard division over the matrix products.

Proposition 15. Let $S \subseteq R$ and $|S| \geq 2$, then $\mathbf{X}_S^{\text{csT}} \mathbf{Y}_S^{\text{cs}} = (\mathbf{X}_S^{\text{cT}} \mathbf{Y}_S^{\text{c}}) \oslash (\widehat{\mathbf{x}}_S^{\text{T}} \widehat{\mathbf{y}}_S)$.

Proof. Let (i, j) be an entry in $\mathbf{X}_S^{\text{csT}} \mathbf{Y}_S^{\text{cs}}$. By Definition 4, each column in $\widehat{\mathbf{X}}_S^{\text{T}}$ is the column vector $\widehat{\mathbf{x}}_S^{\text{T}}$; thus, for all $n \in S$, we have $\widehat{\mathbf{x}}_S^{\text{T}} = \widehat{\mathbf{x}}_{S_{in}}^{\text{T}}$ and so $\widehat{\mathbf{x}}_{S_i}^{\text{T}} = \widehat{\mathbf{x}}_{S_{in}}^{\text{T}}$. Similarly, for all $n \in S$, each row in $\widehat{\mathbf{Y}}_S$ is the row vector $\widehat{\mathbf{y}}_S = \widehat{\mathbf{y}}_{S_{nj}}$ and so $\widehat{\mathbf{y}}_{S_j} = \widehat{\mathbf{y}}_{S_{nj}}$. Using distributivity of division over summation, we get

$$\begin{aligned}(\mathbf{X}_S^{\text{csT}} \mathbf{Y}_S^{\text{cs}})_{ij} &= \left((\mathbf{X}_S^{\text{cT}} \oslash \widehat{\mathbf{x}}_S^{\text{T}}) (\mathbf{Y}_S^{\text{c}} \oslash \widehat{\mathbf{y}}_S) \right)_{ij} \\ &= \sum_{n \in S} \frac{\mathbf{X}_{S_{in}}^{\text{cT}} \mathbf{Y}_{S_{nj}}^{\text{c}}}{\widehat{\mathbf{x}}_{S_i}^{\text{T}} \widehat{\mathbf{y}}_{S_j}} = \sum_{n \in S} \frac{\mathbf{X}_{S_{in}}^{\text{cT}} \mathbf{Y}_{S_{nj}}^{\text{c}}}{\widehat{\mathbf{x}}_{S_i}^{\text{T}} \widehat{\mathbf{y}}_{S_j}} \\ &= \frac{\sum_{n \in S} \mathbf{X}_{S_{in}}^{\text{cT}} \mathbf{Y}_{S_{nj}}^{\text{c}}}{\widehat{\mathbf{x}}_{S_i}^{\text{T}} \widehat{\mathbf{y}}_{S_j}} = \frac{(\mathbf{X}_S^{\text{cT}} \mathbf{Y}_S^{\text{c}})_{ij}}{(\widehat{\mathbf{x}}_S^{\text{T}} \widehat{\mathbf{y}}_S)_{ij}} \\ &= \left((\mathbf{X}_S^{\text{cT}} \mathbf{Y}_S^{\text{c}}) \oslash (\widehat{\mathbf{x}}_S^{\text{T}} \widehat{\mathbf{y}}_S) \right)_{ij},\end{aligned}$$

showing the equality. $\square$

Proposition 16. Let $S \subseteq R$ and $|S| \geq 2$, then $\mathbf{X}_S^{\text{csT}} \mathbf{X}_S^{\text{cs}} = (\mathbf{X}_S^{\text{cT}} \mathbf{X}_S^{\text{c}}) \oslash (\widehat{\mathbf{x}}_S^{\text{T}} \widehat{\mathbf{x}}_S)$.

Proof. As proof of Proposition 16 considering $\mathbf{X}_S$ in place of $\mathbf{Y}_S$. $\square$

While our targets in this section are $\mathbf{X}_S^{\text{csT}} \mathbf{X}_S^{\text{cs}}$ and $\mathbf{X}_S^{\text{csT}} \mathbf{Y}_S^{\text{cs}}$ (particularly $S = T_p$), we note that these results generalize to the case where $\mathbf{X}_S$ and $\mathbf{Y}_S$ are not centered. Exclusively preprocessing $\mathbf{X}_S$ and $\mathbf{Y}_S$ using sample standard deviation-scaling is therefore possible, as is preprocessing only one of $\mathbf{X}_S$ or $\mathbf{Y}_S$. We revisit these variations in Section 6.

Proposition 17. Correctness of Algorithm 7 In step 12 of Algorithm 6, $\text{csXTX}_T = \mathbf{X}_T^{\text{csT}} \mathbf{X}_T^{\text{cs}}$ and $\text{csXTY}_T = \mathbf{X}_T^{\text{csT}} \mathbf{Y}_T^{\text{cs}}$, and the variables are identical to the quantities XTX_T and XTY_T computed in steps 26 and 27 of Algorithm 7.

Proof. Due to Proposition 1, both algorithms select $T = T_p$ and $V = V_p$ as per Definition 1 (a scalable partitioning is a valid partitioning). Since Algorithm 6 is equivalent to Algorithm 4 up to and including step 8, we have as in Proposition 9 that cX_T equals $\mathbf{X}_T^{\text{c}}$ and cY_T equals $\mathbf{Y}_T^{\text{c}}$. Steps 9–11 find the sample standard deviation matrices (replacing 0-entries with 1-entries) and apply them to the centered matrices so that csX_T is $\mathbf{X}_T^{\text{cs}} = \mathbf{X}_T^{\text{c}} \oslash \widehat{\mathbf{X}}_T$ and csY_T is $\mathbf{Y}_T^{\text{cs}} = \mathbf{Y}_T^{\text{c}} \oslash \widehat{\mathbf{Y}}_T$ as per Definition 5. Therefore, after step 12, we have that csXTX_T equals $\mathbf{X}_T^{\text{csT}} \mathbf{X}_T^{\text{cs}}$ and csXTY_T equals $\mathbf{X}_T^{\text{csT}} \mathbf{Y}_T^{\text{cs}}$.

Consider now Algorithm 7. In steps 15 and 16, we have that cXTX_T equals $\mathbf{X}_T^{\text{cT}} \mathbf{X}_T^{\text{c}}$ and cXTY_T equals $\mathbf{X}_T^{\text{cT}} \mathbf{Y}_T^{\text{c}}$ as shown in Proposition 9. That $\Sigma \mathbf{X}_T$ equals $\Sigma_{n \in T} \mathbf{X}_n$ and $\Sigma \mathbf{Y}_T$ equals $\Sigma_{n \in T} \mathbf{Y}_n$ as computed in step 18 follows from Lemma 14 given the quantities computed in step 17. Also, from Lemma 14 follows that ΣsqX_T equals $\Sigma_{n \in T} (\mathbf{X}_n^{\circ 2})$ and ΣsqY_T equals $\Sigma_{n \in T} (\mathbf{Y}_n^{\circ 2})$ as computed in steps 20 and 21 due to the quantities computed in step 19. By Lemma 12 and the quantities computed in steps 12, 18, and 20, it follows that after step 22, $\sigma \mathbf{x}_T$ equals $\widehat{\mathbf{x}}_T$. Similarly, applying Lemma 13 and the quantities computed in steps 13, 18, and 21, we have that after step 23, $\sigma \mathbf{y}_T$ equals $\widehat{\mathbf{y}}_T$.

Replacing 0-entries of $\sigma \mathbf{x}_T$ and $\sigma \mathbf{y}_T$ in step 24 means that these vectors are the rows in matrices $\sigma \mathbf{X}_T$ and $\sigma \mathbf{Y}_T$ after step in 10 of Algorithm 6 according to Definition 4. Therefore, after step 25, $\sigma \mathbf{x}_T \mathbf{x}_T$ and $\sigma \mathbf{x}_T \mathbf{y}_T$ are equal to $\widehat{\mathbf{x}}_T^{\text{T}} \widehat{\mathbf{x}}_T$ and $\widehat{\mathbf{x}}_T^{\text{T}} \widehat{\mathbf{y}}_T$, respectively. Thus, for step 26, we can apply Proposition 16 to see that csXTX_T equals $\mathbf{X}_T^{\text{csT}} \mathbf{X}_T^{\text{cs}}$, and Proposition 15 for step 27 to see that csXTY_T equals $\mathbf{X}_T^{\text{csT}} \mathbf{Y}_T^{\text{cs}}$. $\square$

5.2 | Computational Complexity

We now analyze the running time and space complexities of Algorithms 6 and 7. Our results show that centering and scaling can be enabled for cross-validation with no asymptotic impact on time and space complexity compared to cross-validation without preprocessing.

Proposition 18. Algorithm 6 requires $\Theta(\text{PNK}(K + M))$ operations.

Proof. Steps 1 through 8 of Algorithm 6 are equivalent to steps 1 through 8 of Algorithm 4. Moreover, step 12 of Algorithm 6 requires the same amount of operations as step 9 of Algorithm 4. Operations required to store the inputs are also equivalent between Algorithm 4 and Algorithm 6. Therefore, as in Proposition 10, these steps require $\Theta(\text{PNK}(K + M))$ operations.

For the cost of the remaining steps 9–11, consider any partition p iterated over in step 3. To compute the sample standard deviations row vectors $\widehat{\mathbf{x}}_{T_p}$ and $\widehat{\mathbf{y}}_{T_p}$ in step 9, we sum over $|T_p|$ rows and K , respectively M , columns. We perform $2K$, respectively $2M$, operations to subtract and square for each row. We take the

Hadamard square root of the resulting vectors and then multiply the scaling factor requiring $2K$, respectively $2M$, operations. Computing $1/(|T_p| - 1)$ requires two operations. Stacking the resulting vectors to construct $\widehat{\mathbf{Y}}_{T_p}$ and $\widehat{\mathbf{Y}}_{T_p}$ requires $|T_p|(K + M)$ operations. In total, step 9 requires $2 + 5|T_p|(K + M)$ operations. Step 10 requires $|T_p|(K + M)$ operations to locate 0-entries in $\widehat{\mathbf{X}}_{T_p}$ and $\widehat{\mathbf{Y}}_{T_p}$ and at most $|T_p|(K + M)$ operations to replace them with 1-entries. Step 11 performs Hadamard division requiring $|T_p|(K + M)$ operations. The total for steps 9–11 is $2 + 7|T_p|(K + M) + O(N + M) = \Theta(|T_p|(K + M))$ operations.

Iterating over P partitions means steps 9–11 require $\Theta\left(\sum_{p=1}^P |T_p|(K + M)\right)$ operations, which by Lemma 6 simplifies to $\Theta(PN(K + M))$ operations. The total number of operations required by Algorithm 6 is therefore $\Theta(PN(K + M)) + \Theta(PNK(K + M)) = \Theta(PNK(K + M))$. $\square$

Proposition 19. Algorithm 7 requires $\Theta(NK(K + M))$ operations.

Proof. In step 4, computing $\sum_{n \in R} \mathbf{X}_n$, $\sum_{n \in R} \mathbf{Y}_n$, $\sum_{n \in R} (\mathbf{X}_n^{\circ 2})$ and $\sum_{n \in R} (\mathbf{Y}_n^{\circ 2})$ require $3N(K + M) = \Theta(N(K + M))$ operations. The remaining computations that store inputs and perform steps 1 through 16 are equivalent to storing inputs and steps 1–15 in Algorithm 5, so as in Proposition 11 require $\Theta(NK(K + M))$ operations. Steps 1–16 in Algorithm 7 therefore require $\Theta(N(K + M)) + \Theta(NK(K + M)) = \Theta(NK(K + M))$ operations.

For the remaining steps, consider any partition p . Step 17 requires $|V_p|(K + M)$ operations. Step 18 requires $K + M$ operations. Step 19 requires $2|V_p|(K + M)$ operations. Steps 20 and 21 require $K + M$ operations. Steps 22 and 23 require $6(K + M) + 2$ operations. Step 24 requires $K + M$ operations to locate 0-entries in $\sigma \mathbf{x}_T$ and $\sigma \mathbf{y}_T$ and at most $K + M$ operations to replace them with 1-entries. The matrix products in step 25 require $\Theta(K(K + M))$ operations. Steps 26 and 27 require $K(K + M)$ operations. The total for steps 17–27 is $3|V_p|(K + M) + 9(K + M) + 2 + O(K + M) + \Theta(K(K + M)) + K(K + M)$ which is $\Theta(|V_p|(K + M) + K(K + M))$ operations. Iterating over P partitions and using that $\sum_{p=1}^P |V_p| = N$, steps 17–27 require

$$\begin{aligned} & \Theta\left(\sum_{p=1}^P |V_p|(K + M) + K(K + M)\right) \\ &= \Theta\left((K + M) \sum_{p=1}^P |V_p| + K\right) \\ &= \Theta((K + M)(N + PK)) \\ &= \Theta(N(K + M)) + \Theta(PK(K + M)) \end{aligned}$$

operations. Since $P \leq N$, this is $O(NK(K + M))$ operations. Thus, Algorithm 7 requires a total of $\Theta(NK(K + M)) + O(NK(K + M)) = \Theta(NK(K + M))$ operations. $\square$

Proposition 20. Algorithm 6 requires $\Theta(P)$ more operations than Algorithm 7.

Proof. By Propositions 18 and 19, the ratio of operations is $\frac{\Theta(PNK(K + M))}{\Theta(NK(K + M))} = \Theta(P)$. $\square$

With Proposition 20, we have shown that Algorithm 7 is asymptotically faster (shaving a factor of $\Theta(P)$) than the baseline Algorithm 6. This corresponds with the results against the baselines without preprocessing or with centering only, as shown in Propositions 4 and 12. Similarly, we can show this incurs no cost in terms of space complexity; that is, Algorithms 6 and 7 are of same space complexity.

Proposition 21. Algorithm 6 requires storing $\Theta((K + N)(K + M))$ entries.

Proof. Step 9 requires $|T|(K + M)$ entries to store $\sigma \mathbf{X}_T$ and $\sigma \mathbf{Y}_T$. Step 10 requires no additional entries to modify $\sigma \mathbf{X}_T$ and $\sigma \mathbf{Y}_T$ in-place. Step 11 requires $|T|(K + M)$ to store $\text{cs} \mathbf{X}_T$ and $\text{cs} \mathbf{Y}_T$. Storing $\text{cs} \mathbf{X}_T \mathbf{X}_T$ and $\text{cs} \mathbf{X}_T \mathbf{Y}_T$ in step 12 in Algorithm 6 requires the same amount of entries as storing $\text{c} \mathbf{X}_T \mathbf{X}_T$ and $\text{c} \mathbf{X}_T \mathbf{Y}_T$ in step 9 in Algorithm 4. The remainder of Algorithm 6 is equivalent to Algorithm 4, so as in the proof of Proposition 13 requires storing $\Theta((K + N)(K + M))$ entries. Summing each contribution and using $|T| < N$, the total number of matrix entries is $2|T|(K + M) + \Theta((K + N)(K + M)) = \Theta((K + N)(K + M))$. $\square$

Proposition 22. Algorithm 7 requires storing $\Theta((K + N)(K + M))$ entries.

Proof. In steps 3 and 4, storing $\sum_{n \in R} \mathbf{X}_n$, $\sum_{n \in R} \mathbf{Y}_n$, $\sum_{n \in R} (\mathbf{X}_n^{\circ 2})$ and $\sum_{n \in R} (\mathbf{Y}_n^{\circ 2})$ requires $2(K + M) = \Theta(K + M)$ entries. The remaining storage required for steps 1–16 is equivalent to the storage required by Algorithm 5, so as in the proof of Proposition 14 require $\Theta((N + K)(K + M))$ entries. Steps 17–23 require $5(K + M)$ entries to store $\Sigma \mathbf{X}_V$, $\Sigma \mathbf{Y}_V$, $\Sigma \mathbf{X}_T$, $\Sigma \mathbf{Y}_T$, $\Sigma \text{sq} \mathbf{X}_V$, $\Sigma \text{sq} \mathbf{Y}_V$, $\Sigma \text{sq} \mathbf{X}_T$, $\Sigma \text{sq} \mathbf{Y}_T$, $\sigma \mathbf{x}_T$, and $\sigma \mathbf{y}_T$. Step 24 requires no additional entries. Steps 25–27 require $2K(K + M)$ entries to store $\sigma \mathbf{x}_T \mathbf{x}_T$, $\sigma \mathbf{x}_T \mathbf{y}_T$, $\text{cs} \mathbf{X}_T \mathbf{X}_T$, and $\text{cs} \mathbf{X}_T \mathbf{Y}_T$. The total number of entries is therefore $\Theta(K + M) + \Theta((N + K)(K + M)) + 5(K + M) + 2K(K + M)$ which is $\Theta((K + N)(K + M))$. $\square$

We remark that for all three of our fast algorithms, we have proved time bounds that match computing $\mathbf{X}^T \mathbf{X}$ and $\mathbf{X}^T \mathbf{Y}$ only and space bounds that match storing $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{X}^T \mathbf{X}$, and $\mathbf{X}^T \mathbf{Y}$. In other words, P -fold cross-validation can be performed in the same time and space (asymptotically) as it takes to compute just one fold, which is supported up to a practical constant by our benchmarks in Section 7.

6 | Preprocessing Combinations

In previous sections, we considered algorithms that constitute three combinations of preprocessing, namely, none, centering both $\mathbf{X}_T$ and $\mathbf{Y}_T$, and centering and scaling both $\mathbf{X}_T$ and $\mathbf{Y}_T$. Our fast methods also work if we want to only center, only scale, or center and scale, either $\mathbf{X}_T$ or $\mathbf{Y}_T$. Note that when both centering and scaling, our propositions rely on centering first.

In the case of $\mathbf{X}_T^T \mathbf{X}_T$, there are four different preprocessing combinations, each resulting in a distinct matrix product. This contrasts the case for $\mathbf{X}_T^T \mathbf{Y}_T$ where the $2^4 = 16$ different preprocessing combinations result in only eight distinct matrix

TABLE 1 | The effects on $\mathbf{X}_T^T \mathbf{Y}_T$ of applying all possible combinations of centering and scaling on $\mathbf{X}_T$ and $\mathbf{Y}_T$.

Scaling	Centering			
	No centering	Center $\mathbf{X}_T$	Center $\mathbf{Y}_T$	Center both
No scaling	$\mathbf{X}_T^T \mathbf{Y}_T$	$\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)$	$\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)$	$\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)$
Scale $\mathbf{X}_T$	$(\mathbf{X}_T^T \mathbf{Y}_T) \oslash (\widehat{\mathbf{x}}_T^T \mathbf{1}_M)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\widehat{\mathbf{x}}_T^T \mathbf{1}_M)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\widehat{\mathbf{x}}_T^T \mathbf{1}_M)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\widehat{\mathbf{x}}_T^T \mathbf{1}_M)$
Scale $\mathbf{Y}_T$	$(\mathbf{X}_T^T \mathbf{Y}_T) \oslash (\mathbf{1}_K^T \widehat{\mathbf{y}}_T)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\mathbf{1}_K^T \widehat{\mathbf{y}}_T)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\mathbf{1}_K^T \widehat{\mathbf{y}}_T)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\mathbf{1}_K^T \widehat{\mathbf{y}}_T)$
Scale both	$(\mathbf{X}_T^T \mathbf{Y}_T) \oslash (\widehat{\mathbf{x}}_T^T \widehat{\mathbf{y}}_T)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\widehat{\mathbf{x}}_T^T \widehat{\mathbf{y}}_T)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\widehat{\mathbf{x}}_T^T \widehat{\mathbf{y}}_T)$	$(\mathbf{X}_T^T \mathbf{Y}_T - T (\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T)) \oslash (\widehat{\mathbf{x}}_T^T \widehat{\mathbf{y}}_T)$

Note: Note that centering $\mathbf{X}_T$, $\mathbf{Y}_T$, or both leads to the same effect on $\mathbf{X}_T^T \mathbf{Y}_T$.

products. This follows from Proposition 9, since as long as either $\mathbf{X}_T$ or $\mathbf{Y}_T$ is centered, the matrix product $\mathbf{X}_T^T \mathbf{Y}_T$ is centered. Conversely, this is not the case for scaling. We show the 8 distinct matrix products $\mathbf{X}_T^T \mathbf{Y}_T$ for all 16 preprocessing combinations in Table 1, where $\mathbf{1}_L$ denotes a row vector of length L with all entries being 1 and where (preprocessed) products are represented using Propositions 9 and 15.

Considering $\mathbf{X}_T^T \mathbf{X}_T$ and $\mathbf{X}_T^T \mathbf{Y}_T$ simultaneously, and if we insist preprocessing of $\mathbf{X}_T$ is the same for both matrix products, there are 16 different preprocessing combinations. In this case, 12 are distinct since centering only $\mathbf{Y}_T$ results in $\mathbf{X}_T^T \mathbf{Y}_T$ being centered while $\mathbf{X}_T^T \mathbf{X}_T$ is not. These insights are relevant to situations where centering is applied asymmetrically in model building.

7 | Benchmarks

In Sections 3–5, we have proven the asymptotic efficiency of Algorithms 3, 5, 7 compared to their baseline alternatives. We now demonstrate their practical efficiency as well. Benchmarks are based on the implementation by Engstrøm [21] (Apache 2.0 license), where the bulk of scientific computations take place using NumPy [47].

Before diving into results for execution time, we note some implementation details by Engstrøm [21] that describe the concrete realization of the pseudo-code. For a training partition T , centering using Algorithms 4 and 6 (baselines) is realized by storing the mean row vectors $\bar{\mathbf{x}}_T$ and $\bar{\mathbf{y}}_T$ rather than the matrices $\bar{\mathbf{X}}_T$ and $\bar{\mathbf{Y}}_T$. Similarly, for scaling in Algorithm 6, we store sample standard deviation vectors $\widehat{\mathbf{x}}_T$ and $\widehat{\mathbf{y}}_T$ rather than $\widehat{\mathbf{X}}_T$ and $\widehat{\mathbf{Y}}_T$. When applying mean centering, the fast algorithms, Algorithm 5 (step 13) and Algorithm 7 (step 14) obtain $|T|\bar{\mathbf{x}}_T^T \bar{\mathbf{y}}_T$ and $|T|\bar{\mathbf{x}}_T^T \bar{\mathbf{x}}_T$ (using Propositions 7 and 8) by computing the outer vector product before multiplying by $N_T = |T|$. This is typically the more numerically stable option. However, it requires $|T|(K + M)$ multiplications, whereas first multiplying by $|T|$ can be done with $\min(K, M)$ multiplications (either choice does not impact time complexity analysis).

Our benchmarks are over data set matrices $\mathbf{X}$ and $\mathbf{Y}$ with $N = 100,000 = 10^5$, $K = 500$, and $M = 10$. Matrices are randomly initialized with 64-bit values and are identical for all runs. We compute matrix products over $P \in \{3, 5, 10, 100, 1000, 10000, 100000\}$

partitions and use a balanced partitioning so that validation partitions are of equal size (except when $P = 3$ where one validation partition contains one sample more than the others). All benchmarks use the same hardware (AMD Ryzen 9 5950X, 3.4GHz), and we restrict NumPy to a single CPU.

Figure 1 compares the execution time of the baseline and the fast algorithms (so without preprocessing, with centering, with centering and scaling) for each value of P . The execution time of baseline algorithms grows linearly with P , supporting our results for their time complexity. For Algorithm 3 and $P \geq 10^4$, as well as Algorithms 5 and 7 for $P \geq 10^3$, the results indicate a dependence on P in terms of execution time, which is not supported by our time complexity results.

We profile the implementation and determine operations whose relative cost increases as P grows, finding those to be computing $\mathbf{X}_T^T \mathbf{X}_T$ (when $P \geq 10^4$), and the outer vector products $\mu \mathbf{x}_T^T \mu \mathbf{x}_T$ and $\sigma \mathbf{x}_T^T \sigma \mathbf{x}_T$ (when $P \geq 10^3$), indicating the relative cost primarily increases for operations involving $\mathbf{X}_T^T \mathbf{X}_T$. Since P directly impacts $|V|$ and $K = 500$, we measure the execution time of NumPy when computing the matrix product $\mathbf{M}^T \mathbf{M}$ where $\mathbf{M} \in \mathbb{R}^{|V| \times 500}$. We normalize execution times by $|V|$ and divide by the smallest normalized execution time we observe. This gives standardized costs c that describe the execution time of matrix multiplication as a function of $|V|$, where c close to 1 means practical constants are stable.

The standardized costs are summarized in Table 2 and illustrate that the overhead of computing $\mathbf{M}^T \mathbf{M}$ grows as $|V|$ decreases and this in a manner consistent with our benchmarks of the fast algorithms. For the fast algorithms when $P = 10^4$ ($P = 10^5$), $\mathbf{X}_T^T \mathbf{X}_T$ requires matrix multiplications that in practice are about 2.5 (17) times more costly than when $P = 10^3$ ($P = 10^4$). With centering and scaling, we compute $2P$ outer vector products² corresponding to $|V| = 1$, which Table 2 shows has a large practical constant for our choice of K . This accounts for the observable dependency on $P \geq 10^4$ in the execution time of Algorithm 3 and dependency on $P \geq 10^3$ for the execution times of Algorithms 5 and 7.

Notwithstanding the practical overhead of computing many relatively small matrix products, our fast algorithms dominate the execution time of baseline algorithms for all values of P and by more than two orders of magnitude when $P \geq 10^3$. For leave-one-out cross-validation with centering and scaling specifically, execution time drops from close to a day to around a minute in this benchmark.

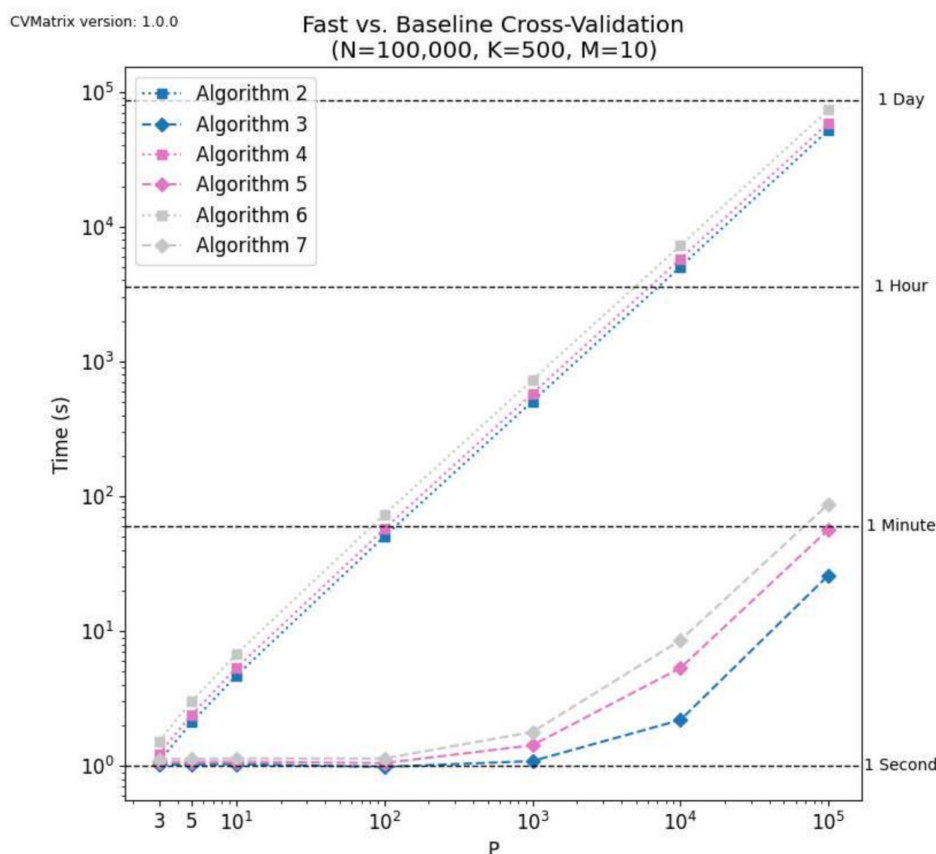


FIGURE 1 | A log-log plot of execution time as a function of P for computing $\mathbf{X}^T\mathbf{X}$, $\mathbf{X}^T\mathbf{Y}$, and their centered and scaled variants, with the baseline and fast algorithms as implemented in *cvmatrix* [21]. Fast algorithms dominate the baselines by a factor of $\Theta(P)$. The execution time dependence on larger P for the fast algorithms is caused by practical constants from computing many relatively small matrix products.

TABLE 2 | Standardized cost c of computing $\mathbf{M}^T\mathbf{M}$ with $\mathbf{M} \in \mathbb{R}^{|V| \times 500}$, where P corresponds to the number of partitions yielding $|V|$ when $N = 100,000$.

P	20	100	200	1000	2000	10,000	20,000	50,000	100,000
$ V $	5000	1000	500	100	50	10	5	2	1
$c \leq$	1	1.01	1.02	1.2	1.4	3	5	11	52

Note: Shaded columns correspond to a configuration of P in Figure 1.

8 | Conclusion

In this paper, we introduced algorithms that significantly accelerate, in theory and practice, partition-based cross-validation for machine learning models. This has application for methods such as PCA, PCR, RR, OLS, and PLS, particularly on tall data sets. Under the fold-based partitioning scheme, we show correctness of our fast algorithms (Propositions 9 and 17), which is a novel contribution since, as we show in Section 4.1, fast alternatives in the literature induce leakage of mean and sample standard deviation statistics between training and validation partitions.

The algorithms we developed work by computing $\mathbf{X}^T\mathbf{X}$, $\mathbf{X}^T\mathbf{Y}$, means, and sample standard deviations only once for the entire data set in the beginning and then, for each cross-validation fold, remove the contribution of the validation partition. This approach eliminates redundant computations existing in the overlap between training partitions and is the same shortcut employed by

Lindgren et al. [16], but now correctly generalized for centering and scaling. A key result is Lemma 9 which ultimately collapses certain combinations of center preprocessing and states that $\mathbf{X}_S^T\mathbf{Y}_S$ is the same as both $\mathbf{X}_S^T\mathbf{Y}_S$ and $\mathbf{X}_S^T\mathbf{Y}_S$ for any $S \subseteq R$, meaning either can be subtracted from $\mathbf{X}_S^T\mathbf{Y}_S$ to perform centering. This insight applies beyond the fold-based partitioning scheme.

Time and space are shown to be independent of the number of folds (Proposition 20) in the most involved case with both centering and scaling and so has the same time complexity as computing $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ (Proposition 19) and space complexity equivalent to storing $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{X}^T\mathbf{X}$, and $\mathbf{X}^T\mathbf{Y}$ (Proposition 22). Our benchmarks of execution time (using implementation by Engström [21], Apache 2.0 license) validate our time complexity results. Therefore, our methods enable practically efficient P -fold cross-validation with proper centering and scaling, potentially decreasing execution time by orders of magnitude, making them a valuable tool for robust model selection.

Peer Review

The peer review history for this article is available at <https://www.webofscience.com/api/gateway/wos/peer-review/10.1002/cem.70008>.

Endnotes

¹ The product $\mathbf{X}^T\mathbf{X}$ is sometimes referred to as the covariance matrix [1], $\mathbf{X}^T\mathbf{Y}$ the cross-covariance matrix [2], and both as kernel matrices [3]. Formally, the sample covariance $\mathbf{X}^T\mathbf{X}$ and sample cross-covariance $\mathbf{X}^T\mathbf{Y}$ matrices are computed over centered $\mathbf{X}$ and $\mathbf{Y}$ and then scaled by $\frac{1}{N-1}$. The Pearson correlation matrix also uses scaling (z-standardizing) of $\mathbf{X}$ and $\mathbf{Y}$. We are concerned with different combinations of centering and scaling and, as such, refer to $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ simply as matrix products.

² Computing $\mathbf{M}^T\mathbf{M}$ is faster, in absolute terms, when $\mathbf{M} \in \mathbb{R}^{2 \times 500}$ than when $\mathbf{M} \in \mathbb{R}^{1 \times 500}$. This insight could be readily applied to roughly halve the practical constants associated with centering and scaling and improve the case of leave-one-out cross-validation. We observe for $\mathbf{M} \in \mathbb{R}^{1 \times 500}$ in our setup that NumPy v1.26.4 no longer uses optimized multiply-accumulate instructions but rather exclusively optimized multiply instructions.

References

1. F. Lindgren, P. Geladi, and S. Wold, "The Kernel Algorithm for PLS," *Journal of Chemometrics* 7, no. 1 (1993): 45–59, <https://doi.org/10.1002/cem.1180070104>.
2. M. Hubert and K. V. Branden, "Robust Methods for Partial Least Squares Regression," *Journal of Chemometrics* 17, no. 10 (2003): 537–549, <https://doi.org/10.1002/cem.822>.
3. S. De Jong and C. J. F. Ter Braak, "Comments on the PLS Kernel Algorithm," *Journal of Chemometrics* 8, no. 2 (1994): 169–174, <https://doi.org/10.1002/cem.1180080208>.
4. H. Hotelling, "Analysis of a Complex of Statistical Variables Into Principal Components," *Journal of Educational Psychology* 24, no. 6 (1933): 417, <https://doi.org/10.1037/h0071325>.
5. H. Hotelling, "The Relations of the Newer Multivariate Statistical Methods to Factor Analysis," *British Journal of Statistical Psychology* 10, no. 2 (1957): 69–79, <https://doi.org/10.1111/j.2044-8317.1957.tb00179.x>.
6. M. G. Kendall, *A Course in Multivariate Analysis* (New York: Hafner Publishing Company, 1957).
7. A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics* 12, no. 1 (1970): 55–67, <https://doi.org/10.1080/00401706.1970.10488634>.
8. J. F. Kenney and E. S. Keeping, "Linear Regression and Correlation," *Mathematics of Statistics* 1 (1962): 252–285.
9. H. Wold, "Estimation of Principal Components and Related Models by Iterative Least Squares," *Multivariate analysis* 1 (1966): 391–420.
10. S. Wold, M. Sjöström, and L. Eriksson, "PLS-Regression: A Basic Tool of Chemometrics," *Chemometrics and Intelligent Laboratory Systems* 58, no. 2 (2001): 109–130, [https://doi.org/10.1016/S0169-7439\(01\)00155-1](https://doi.org/10.1016/S0169-7439(01)00155-1).
11. M. Sjöström, S. Wold, and B. Söderström, *PLS Discriminant Plots* (Elsevier, 1986).
12. L. Ståhle and S. Wold, "Partial Least Squares Analysis With Cross-Validation for the Two-Class Problem: A Monte Carlo Study," *Journal of Chemometrics* 1, no. 3 (1987): 185–196, <https://doi.org/10.1002/cem.1180010306>.
13. M. Barker and W. Rayens, "Partial Least Squares for Discrimination," *Journal of Chemometrics* 17, no. 3 (2003): 166–173, <https://doi.org/10.1002/cem.785>.
14. M. Stone, "Cross-Validatory Choice and Assessment of Statistical Predictions," *Journal of the Royal Statistical Society: Series B (Methodological)* 36, no. 2 (1974): 111–133, <https://doi.org/10.1111/j.2517-6161.1974.tb00994.x>.
15. T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Vol. 2 (Springer, 2009).
16. F. Lindgren, P. Geladi, and S. Wold, "Kernel-Based PLS Regression; Cross-Validation and Applications to Spectral Data," *Journal of Chemometrics* 8, no. 6 (1994): 377–389, <https://doi.org/10.1002/cem.1180080604>.
17. D. Gianola and C.-C. Schön, "Cross-Validation Without Doing Cross-Validation in Genome-Enabled Prediction," *G3: Genes, Genomes, Genetics* 6, no. 10 (2016): 3107–3128, <https://doi.org/10.1534/g3.116.033381>.
18. J.-J. Zhu, M. Yang, and Z. J. Ren, "Machine Learning in Environmental Research: Common Pitfalls and Best Practices," *Environmental Science & Technology* 57, no. 46 (2023): 17671–17689, <https://doi.org/10.1021/acs.est.3c00026>.
19. S. Kapoor and A. Narayanan, "Leakage and the Reproducibility Crisis in Machine-Learning-Based Science," *Patterns* 4, no. 9 (2023): 100804, <https://doi.org/10.1016/j.patter.2023.100804>.
20. K. H. Liland, P. Stefansson, and U. G. Indahl, "Much Faster Cross-Validation in PLSR-Modelling by Avoiding Redundant Calculations," *Journal of Chemometrics* 34, no. 3 (2020): e3201, <https://doi.org/10.1002/cem.3201>.
21. O.-C. G. Engström, "Cvmatrix: A Python Package for Fast Computation of Possibly Centered/Scaled Training Set Kernel Matrices in a Cross-Validation Setting," (2024), <https://pypi.org/project/cvmatrix/>.
22. J. D. Huling and P. Chien, "Fast Penalized Regression and Cross Validation for Tall Data With the OEM Package," *Journal of Statistical Software* 104 (2022): 1–24, <https://doi.org/10.18637/jss.v104.i06>.
23. P. Nomikos and J. F. MacGregor, "Multivariate SPC Charts for Monitoring Batch Processes," *Technometrics* 37, no. 1 (1995): 41–59, <https://doi.org/10.1080/00401706.1995.10485888>.
24. H. T. Eastment and W. J. Krzanowski, "Cross-Validatory Choice of the Number of Components From a Principal Component Analysis," *Technometrics* 24, no. 1 (1982): 73–77, <https://doi.org/10.1080/00401706.1982.10487712>.
25. S. Wold, "Cross-Validatory Estimation of the Number of Components in Factor and Principal Components Models," *Technometrics* 20, no. 4 (1978): 397–405, <https://doi.org/10.1080/00401706.1978.10489693>.
26. S. Wold, K. Esbensen, and P. Geladi, "Principal Component Analysis," *Chemometrics and Intelligent Laboratory Systems* 2, no. 1-3 (1987): 37–52, [https://doi.org/10.1016/0169-7439\(87\)80084-9](https://doi.org/10.1016/0169-7439(87)80084-9).
27. R. Bro, K. Kjeldahl, A. K. Smilde, and H. A. L. Kiers, "Cross-Validation of Component Models: A Critical Look at Current Methods," *Analytical and Bioanalytical Chemistry* 390 (2008): 1241–1251, <https://doi.org/10.1007/s00216-007-1790-1>.
28. L. E. Agelet and C. R. Hurburgh Jr., "A Tutorial on Near Infrared Spectroscopy and Its Calibration," *Critical Reviews in Analytical Chemistry* 40, no. 4 (2010): 246–260, <https://doi.org/10.1080/10408347.2010.515468>.
29. H. Abdi and L. J. Williams, "Principal Component Analysis," *Wiley Interdisciplinary Reviews: Computational Statistics* 2, no. 4 (2010): 433–459, <https://doi.org/10.1002/wics.101>.
30. B. Mevik and H. R. Cederkvist, "Mean Squared Error of Prediction (MSEP) Estimates for Principal Component Regression (PCR) and Partial Least Squares Regression (PLSR)," *Journal of Chemometrics* 18, no. 9 (2004): 422–429, <https://doi.org/10.1002/cem.887>.
31. D. M. Hawkins, S. C. Basak, and D. Mills, "Assessing Model Fit by Cross-Validation," *Journal of Chemical Information and Computer Sciences* 43, no. 2 (2003): 579–586, <https://doi.org/10.1021/ci025626i>.

32. K. M. Sørensen, F. van den Berg, and S. B. Engelsen, "NIR Data Exploration and Regression by Chemometrics—A Primer," *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*, ed. Y. Ozaki, C. Huck, S. Tsuchikawa and S.B. Engelsen (Springer, 2021): 127–189, https://doi.org/10.1007/978-981-15-8648-4_7.
33. P. Filzmoser, B. Liebmann, and K. Varmuza, "Repeated Double Cross Validation," *Journal of Chemometrics* 23, no. 4 (2009): 160–171, <https://doi.org/10.1002/cem.1225>.
34. A. Alin, "Comparison of PLS Algorithms When Number of Objects Is Much Larger Than Number of Variables," *Statistical Papers* 50 (2009): 711–720, <https://doi.org/10.1007/s00362-009-0251-7>.
35. M. Andersson, "A Comparison of Nine PLS1 Algorithms," *Journal of Chemometrics* 23, no. 10 (2009): 518–529, <https://doi.org/10.1002/cem.1248>.
36. B. S. Dayal and J. F. MacGregor, "Improved PLS Algorithms," *Journal of Chemometrics* 11, no. 1 (1997): 73–85, [https://doi.org/10.1002/\(SICI\)1099-128X\(199701\)11:1<73::AID-CEM435>3.0.CO;2-%23](https://doi.org/10.1002/(SICI)1099-128X(199701)11:1<73::AID-CEM435>3.0.CO;2-%23).
37. O.-C. G. Engström, E. S. Dreier, B. M. Jespersen, and K. S. Pedersen, "IKPLS: Improved Kernel Partial Least Squares and Fast Cross-Validation Algorithms for Python With CPU and GPU Implementations Using NumPy and JAX," *Journal of Open Source Software* 9, no. 99 (2024): 6533, <https://doi.org/10.21105/joss.06533>.
38. W. Wu, D. L. Massart, and S. De Jong, "Kernel-PCA Algorithms for Wide Data Part II: Fast Cross-Validation and Application in Classification of NIR Data," *Chemometrics and Intelligent Laboratory Systems* 37, no. 2 (1997): 271–280, [https://doi.org/10.1016/S0169-7439\(97\)00027-0](https://doi.org/10.1016/S0169-7439(97)00027-0).
39. M. A. Van De Wiel, M. M. Van Nee, and A. Rauschenberger, "Fast Cross-Validation for Multi-Penalty High-Dimensional Ridge Regression," *Journal of Computational and Graphical Statistics* 30, no. 4 (2021): 835–847, <https://doi.org/10.1080/10618600.2021.1904962>.
40. P. Stefanesson, U. G. Indahl, K. H. Liland, and I. Burud, "Orders of Magnitude Speed Increase in Partial Least Squares Feature Selection With New Simple Indexing Technique for Very Tall Data Sets," *Journal of Chemometrics* 33, no. 11 (2019): e3141, <https://doi.org/10.1002/cem.3141>.
41. Å. Rinnan, F. Van Den Berg, and S. B. Engelsen, "Gianola and Schön Most Common Pre-Processing Techniques for Near-Infrared Spectra," *TrAC Trends in Analytical Chemistry* 28, no. 10 (2009): 1201–1222, <https://doi.org/10.1016/j.trac.2009.07.007>.
42. R. J. Barnes, M. S. Dhanoa, and S. J. Lister, "Standard Normal Variate Transformation and De-Trending of Near-Infrared Diffuse Reflectance Spectra," *Applied Spectroscopy* 43, no. 5 (1989): 772–777, <https://doi.org/10.1366/0003702894202201>.
43. A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures," *Analytical Chemistry* 36, no. 8 (1964): 1627–1639, <https://doi.org/10.1021/ac60214a047>.
44. J. Steinier, Y. Termonia, and J. Deltour, "Smoothing and Differentiation of Data by Simplified Least Square Procedure," *Analytical Chemistry* 44, no. 11 (1972): 1906–1909, <https://doi.org/10.1021/ac60319a045>.
45. P. Geladi, D. MacDougall, and H. Martens, "Linearization and Scatter-Correction for Near-Infrared Reflectance Spectra of Meat," *Applied Spectroscopy* 39, no. 3 (1985): 491–500, <https://doi.org/10.1366/0003702854248656>.
46. F. Le Gall, "Powers of Tensors and Fast Matrix Multiplication," in *International Symposium on Symbolic and Algebraic Computation* (Association for Computing Machinery, 2014), 296–303.
47. C. R. Harris, K. J. Millman, S. J. van der Walt, et al., "Array Programming With NumPy," *Nature* 585, no. 7825 (2020): 357–362, <https://doi.org/10.1038/s41586-020-2649-2>.

Part II

Near-Infrared Hyperspectral Image Analysis

Chapter 4

Convolutional Neural Networks for Near-Infrared Hyperspectral Image Analysis

MODELING a grain quality parameter with a convolutional neural network (CNN) applied to near-infrared hyperspectral images (HSI) is seemingly straightforward. A CNN, such as the common ResNet (K. He et al. 2016), can readily be applied to grain variety classification and protein content regression. While this works well for grain variety classification, it yields subpar results for protein content regression. In chemometrics, NIR spectra are often preprocessed to remove the physical effects of Rayleigh scattering and thus reveal the underlying chemical information of interest. One common way to do so (Rinnan et al. 2009) is to convolve the spectra with a Savitzky-Golay (SG) filter (Savitzky and Golay 1964; Steinier et al. 1972). Parameterized by their window size, polynomial, and derivative order, SG filters can apply smoothing and take derivatives. With PLS, conducting a grid search cross-validation over many settings is feasible to determine the best one for a given task. Grid search cross-validation, however, quickly becomes computationally infeasible for CNN-based models.

Instead, by choosing only the filter size (analogous to SG window length), the CNN can learn the parameters for an initial spectral convolution layer during its training process. Learning a handful of parameters for a spectral convolution layer facilitates smoothing and the computation of derivatives. Learning the same with only 2D convolutions would require many filters with a very particular set of weights, making it much more difficult in practice. Engstrøm et al. (2023b) show that adding an initial spectral convolution layer greatly increases CNN model performance for regression of protein content on NIR hyperspectral images. Ablation studies show that the spectral convolution layer is much more important than the choice of the CNN base model. In particular, a simple, shallow CNN with an initial spectral convolution layer

will significantly outperform a ResNet-18 without an initial spectral convolution layer. Conversely, adding a spectral convolution filter does not increase grain variety classification performance for CNNs. Interestingly, this pattern repeats for PLS modeling where preprocessing with SG filtering decreases subsequent classification accuracy. This indicates that the physical phenomena in the NIR spectra are generally useful discriminators for grain variety classification but noise in the context of protein content regression. These studies show that insights from chemometrics typically exploited to increase PLS performance can also be used to improve the performance of deep learning models for hyperspectral image analysis of food.

Another insight due to Engstrøm et al. (2023b) is that grain variety classification can be accurately modeled with the spatial and spectral dimensions alone. However, using the entire hyperspectral image enables even more accurate classification. This indicates that it is beneficial to analyze the entire hyperspectral image for problems where both spatial and spectral information are relevant instead of just its spatial or spectral dimensions alone.

There is a mistake in Figure 2 in the following article by Engstrøm et al. (2023b). The 3D convolution kernels do not have size $\{1, 7\} \times 7 \times 7$ but $s \times s \times 7$ with $s \in \{1, 7\}$.

Improving Deep Learning on Hyperspectral Images of Grain by Incorporating Domain Knowledge from Chemometrics

Ole-Christian Galbo Engstrøm^{1,2,3}, Erik Schou Dreier², Birthe Møller Jespersen³,
and Kim Steenstrup Pedersen^{1,4}

¹: Department of Computer Science (DIKU), University of Copenhagen, Denmark

²: FOSS Analytical A/S, Denmark

³: Department of Food Science (UCPH FOOD), University of Copenhagen, Denmark

⁴: Natural History Museum of Denmark (NHMD), University of Copenhagen, Denmark

ocge@foss.dk, esd@foss.dk, bm@food.ku.dk, kimstp@di.ku.dk

Abstract

We demonstrate how to design and apply domain-specific modifications to convolutional neural networks (CNNs) to improve model performance on hyperspectral images of grain kernels. We use hyperspectral images of grain kernels captured in the near-infrared wavelength range of 900 to 1700 nm as a case for supporting our argumentation. This part of the electromagnetic spectrum contains convoluted signals with chemical and physical information relevant to grain quality. For standard chemometric models, domain knowledge is used to select from a plethora of combinations of preprocessing techniques helpful in extracting relevant chemical and physical features for a given task. By incorporating domain-specific design modifications in the preexisting architectures of ResNet-18 and a simple CNN, we show that model performance can be increased significantly and that applying domain knowledge to CNNs is much more important than complexity is to their performance.

1. Introduction

General deep learning models for image analysis achieve good results across diverse domains [34, 37] including agricultural domains [8, 21, 26, 29, 38]. However, we argue that incorporating domain knowledge into the model design leads to significantly better results than those achievable by general deep learning models. We use grain quality analysis as an example to support our argument.

Grain quality analysis is a multi-parameter problem class consisting of physical and chemical properties [28, 31]. Historically, chemometricians have approached this problem class by combining physical and chemical knowledge with machine learning algorithms to predict quality param-

eters from near-infrared (NIR) spectra of grain [27, 28]. Application of chemometrics and NIR spectroscopy to grain quality analysis is an active field of research and continues as a primary analytical tool for grain quality analysis [9, 13, 28]. While research for grain quality analysis has been conducted within various subfields of computer vision [35, 45], NIR hyperspectral imaging (NIR-HSI) [20] is perhaps the most promising, as it combines spatial features with spectra containing chemical information regarding the biological quality of the grain [32] enabling analysis of both physical and chemical properties [9, 15].

Deep learning algorithms, specifically convolutional neural networks (CNNs), have seen widespread application within HSI for agricultural domains [21, 22, 46]. We know from chemometrics that the choice of spectral preprocessing is essential for downstream model performance [13], yet hard to choose before model validation [39, 42]. Likewise, this paper shows that spectral preprocessing is crucial for CNNs applied to hyperspectral images. However, while exhaustively searching the space of preprocessing techniques is possible for chemometric methods, the computation required for CNNs makes such a search infeasible. Therefore, guided by domain-specific knowledge from chemometrics and the physics of the imaging process, we design an extension to any pre-existing CNN that allows it to learn the optimal spectral preprocessing and, consequently, achieve significantly better performance than its plain counterpart.

To assess the importance of our extensions relative to employing sophisticated CNNs, we design a simple, shallow CNN and compare it to the well-known ResNet-18 from the high-performant ResNet-family [18]. We compare these models' performances to that of Partial Least Squares (PLS) [49], a standard method in chemometrics that we use as a baseline. We explain the models in detail in Sec. 3.

We use two datasets with hyperspectral images of grain

kernels from Engström *et al.* [14] and Dreier *et al.* [12], respectively. The first dataset contains reference values for the protein content in the grain kernels, while the second dataset contains class labels for the grain type of the kernels. We explain the datasets in detail in Sec. 2. For both datasets, it holds that the problems are in a low sample size scenario since obtaining this type of labeled data for model training is particularly costly. Problems like these can particularly benefit from adding constraints and prior knowledge into the model design [16, 30].

To summarise, our contributions are: (1) The design of simple extensions to any pre-existing CNN, facilitating end-to-end training on hyperspectral images of grain with no requirements for the application of advanced preprocessing techniques. (2) Unifying chemometrics and deep learning with an analysis of the CNN extensions’ effects on both chemical and physical prediction tasks using domain knowledge from chemometrics. (3) Exemplifying that incorporating domain knowledge in the CNN design for agricultural image analysis is more critical than increasing model complexity.

In this article, we begin with a presentation of the grain quality analysis datasets in Sec. 2 followed by details of model designs in Sec. 3 and experimental designs in Sec. 4. We analyze and discuss the results in Sec. 5, which we conclude upon in Sec. 6.

2. Datasets

In this paper, we use two NIR-HSI datasets from Engström *et al.* [14] and Dreier *et al.* [12], respectively. Both datasets consist of hyperspectral images with 224 uniformly distributed wavelength channels in the 900 nm - 1700 nm NIR range of the electromagnetic spectrum taken with a *Specim FX17* camera [43]. This spectral range contains chemical information relevant for grain quality assessment [32]. Dataset #1 consists of images of bulk wheat grain kernels with reference values for mean protein contents on a physical sample scale with values between 8.66% and 17.78% [14]. The grain in Dataset #1 is from the FOSS [4] European and World Grain Networks. Dataset #2 consists of images of different types of bulk rye and wheat kernels and contains class labels for the grain variety on an image level where each image has precisely a single grain variety [12]. The dataset includes one rye variety and seven wheat varieties. Thus, the tasks on the datasets are mean protein content regression and grain variety classification, respectively. For Dataset #1, we reuse the data split as provided by [14], who divided it into six splits; five for 5-fold cross-validation (CV) and one for testing. Similarly, for Dataset #2, we reuse the dataset split as provided by [12], who provide a training, validation, and testing split.

The hyperspectral images in both datasets have a spatial size that varies slightly due to the line scan nature

of the camera. A typical size, however, is approximately 800×500 pixels. Both datasets contain images of sparsely and densely packed grain with varying grain density across the images. The authors of [14] and [12] use a similar but slightly different strategy for cropping the hyperspectral images into a collection of crops of 128×128 pixels. Both use the grain density ratio to decide whether a hyperspectral image crop should be retained or discarded in the final dataset. The grain density ratio of a hyperspectral image crop is defined as the number of pixels containing grain to the total amount of pixels. The computation of this ratio is based on a binary semantic segmentation mask produced by thresholding based on Otsu [33] threshold selection provided with both datasets. Where Engström *et al.* [14] use a grain density ratio of 0.1, Dreier *et al.* [12] use one of 0.5. In our work, we opt to use 0.1 for both datasets. Both Engström *et al.* [14] and Dreier *et al.* [12] crop with 50% overlap in both spatial dimensions. However, whereas Engström *et al.* [14] start cropping at the first row and column containing grain, as determined by the segmentation mask, Dreier *et al.* [12] start cropping at the center of the image. In our work, we opt for a third approach, where we start cropping at the top left corner. For Dataset #1, our approach yields 69,630 hyperspectral image crops to be used in the 5-fold CV and 17,783 hyperspectral image crops for testing. For Dataset #2, our approach yields 15,376 hyperspectral image crops for training, 7,967 hyperspectral image crops for validation, and 3,274 hyperspectral image crops for testing. We summarize the datasets in Table 1.

Engström *et al.* [14] and Dreier *et al.* [12] use a different approach for reducing the raw 224 channels. In this work, we opt for the method used by Engström *et al.* [14] consisting of removing the first and last 10 channels, which are noisy due to low camera sensitivity, and binning the remaining 204 channels by averaging neighboring pairs, reducing the number of channels to 102. Engström *et al.* [14] and Dreier *et al.* [12] transform the reflectance images to absorbance images and mask the resulting absorbance, zeroing, any non-grain pixels. We apply the same transformation.

Additionally, for each hyperspectral image crop, a mean grain spectrum is computed for use by chemometric models that take a spectrum as input. We derive the mean grain spectrum by averaging the spectrum of each grain pixel in the hyperspectral image crop, determined by the segmentation mask. Figure 1 shows a masked hyperspectral image crop, a spectrum at a single grain pixel, and a mean grain spectrum over the hyperspectral image crop.

3. Models

As a chemometric baseline model to determine if the comparatively large CNNs utilizing the entire spatio-spectral image yield increased performance over chemo-

Dataset	# image crops	Task
#1	5-fold CV: 69, 630.	Protein content regression. Range: 8.66% – 17.78%.
	Test: 17, 783.	
#2	Train: 15, 376.	Grain variety classification. 1 rye variety and 7 wheat varieties.
	Val: 7, 967.	
	Test: 3, 274.	

Table 1. The number of image crops and the associated task for the two datasets.

metric methods using only a mean spectrum, we implement PLS [49]. On Dataset #1, we apply PLS regression (PLS-R) [50, 51]. On Dataset #2, we apply PLS discriminant analysis (PLS-DA) [6]. PLS-DA is a method that uses a binary encoding of the categorical variables and performs PLS-R from the input to the binary-encoded categorical variables. For multiple target variables, as in multiclass classification, we can choose between two PLS-DA methods [25]. We can train an ensemble of PLS-DA algorithms in a one-versus-all manner for each class or train a single multiclass classifier for all classes using one-hot encoding of the categorical variables, commonly referred to as a PLS2-DA model. Our experiments showed that PLS2-DA was consistently better than an equivalent PLS-DA ensemble. Thus, from now on, we will focus on PLS2-DA regarding chemometric methods for classification on Dataset #2.

We implement two plain CNN-based models to which we apply and assess our extensions. The first plain CNN is a ResNet-18 [18] as used by Engström *et al.* [14] where they swap the ordering of rectified linear unit (ReLU) and batch normalization (BN) [19]. The second is a much simpler 5-layer CNN, which we name SimpleNet, where each convolution layer has the number of filters with the same kernel size as those within the corresponding ResNet block. Following the structure of ResNet-18, each convolution layer uses a ReLU activation followed by a BN layer. Figure 2 shows the architecture of SimpleNet. Both networks have a fully connected linear output layer with one neuron for the protein regression task on Dataset #1 and a fully connected layer with eight neurons and a softmax activation for the grain variety classification task of Dataset #2.

Inspired by Engström *et al.* [14], we experiment with two types of extensions and combinations of these extensions. The first extension consists of adding a 3-dimensional convolution (Conv3D) layer as the first layer in the network. A Conv3D layer facilitates learning spectral smoothing and derivative filters, commonly used in NIR spectroscopy [13, 39]. We denote such a layer by Conv3D_{*n*,(*h*×*w*×*d*)} where *n* is the number of filters and *h*, *w*, and *d* are the kernel’s height, width, and depth, respectively. The second extension, which we name a Downsampler (Ds_{*m*}), consists of

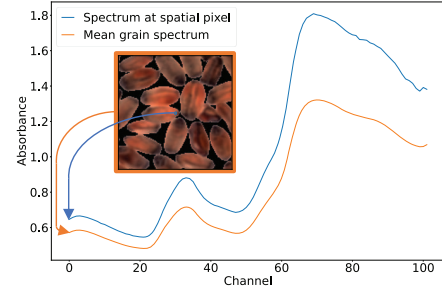


Figure 1. A masked hyperspectral image crop of spatial size 128×128 pixels along with the spectrum at one of the pixels and the mean spectrum of the crop computed over all the grain pixels, both containing 102 wavelength channels.

adding an initial 2-dimensional convolution (Conv2D) layer that applies dimensionality reduction to the spectral dimension with *m* number of filters, each applying a 1×1 kernel for every spectral channel. The number of filters, *m*, in Ds_{*m*} resembles the number of components chosen by PLS. We denote a PLS model using *m* components by *A_m*. While Engström *et al.* [14] experiment with two combinations of these extensions, a Conv3D layer followed by a Ds₃ layer and a Ds₃ without a prior Conv3D layer, we extend this also to include a Conv3D layer without a subsequent Ds layer. Figure 2 shows the combinations of extensions we employ.

Both Engström *et al.* [14] and Dreier *et al.* [12] employ grayscale variants of the plain ResNet-18 with the latter also using a 3-dimensional variant of the plain ResNet-18 where each 2-dimensional convolution and pooling layer is replaced with an equivalent 3-dimensional layer. We follow this practice and employ grayscale and 3-dimensional variants of plain ResNet-18 and plain SimpleNet. The grayscale variant determines if the problems can be solved using a purely spatial approach by averaging the spectral dimension and feeding the resulting grayscale image to the CNN. In contrast, the 3-dimensional variant lends itself naturally to the hyperspectral image, accounting for spatio-spectral features.

3.1. Implementation Details

The CNN and PLS models are implemented using TensorFlow [1], and Keras [10]. While 32 bits of floating point precision is sufficient for the CNNs, the PLS models require 64 bits of floating point precision to converge with increasing values of *A_m*. Unlike CNNs, PLS does not compute a bias coefficient, which is necessary as it assumes a proportionality between the input spectrum and the target variable. To account for this, we can either augment the input spectrum with a constant extra channel or center the target variable. We opt for the latter approach and center the protein content around the mean of each training split in Dataset #1 before training PLS-R. For PLS-DA, the issue is nullified by

applying sigmoid, ensuring that a PLS output of zero yields an equiprobable class prediction when combined with the binary encoding of target variables. For PLS2-DA, using one-hot encoding nullifies the issue altogether.

For the CNNs, we initialize all weights with the Kaiming He Normal Distribution [17] and initialize all biases to zero. Unless explicitly stated otherwise, convolution and pooling layers apply zero-padding. For CNN regressors on Dataset #1, we use the root mean squared error (RMSE) as the loss function. For CNN classifiers on Dataset #2, the loss function is the weighted categorical cross entropy (CCE) with balanced class weights from scikit-learn [36]. In practice, the class weights are close to 1 as Dataset #2 is very balanced. In both cases, we apply L2-regularization (L2) with a regularization parameter of 10^{-3} . For SimpleNet, L2 is applied to all weights. For ResNet-18, we follow the guidelines by Kim *et al.* [23] regarding when to apply L2 to BN weights, termed γ , by Ioffe and Szegedy [19]. Additionally, for all other layers in ResNet-18, we apply L2 to their weights. We use stochastic gradient descent (SGD) optimization with a batch size of 32, a momentum of 0.9, and an initial learning rate of 0.1. We multiply the learning rate by 0.1 if the validation RMSE or CCE plateaus for 10 epochs. When reducing the learning rate, we restore the current best weights and continue the training from that point on. If the validation RMSE or validation CCE plateaus for 50 epochs or reaches a total of 1000 epochs, training is halted and the best weights restored. In practice, all CNNs were halted by the 50 epochs plateau criterion. While training CNNs, we apply data augmentation to the hyperspectral image crops by uniformly randomly flipping them vertically and horizontally.

PLS can be implemented with several different algorithms. We choose to implement PLS using Improved Kernel PLS Algorithm #1 [11] as it is both fast [2] and numerically stable [3]. For PLS-R on Dataset #1, we choose A_m for each of the CV splits as the number of PLS components that yields the lowest validation RMSE. Although previously attempted [44], computing CCE on the output of PLS2-DA is not a good metric of its performance. PLS2-DA outputs a vector that does not necessarily encode a discrete probability distribution. While applying the softmax function to the output of a PLS2-DA model does not change its categorical prediction, when interpreted as an arg max of the prediction vector, it does not encode a probability distribution that allows for meaningful application of CCE. A perfect PLS2-DA model will predict the ground truth before any softmax application. Thus, any subsequent application of softmax would yield a non-zero CCE. Therefore, for PLS2-DA on Dataset #2, we choose A_m based on the highest weighted validation categorical accuracy, which is unaffected by the softmax application.

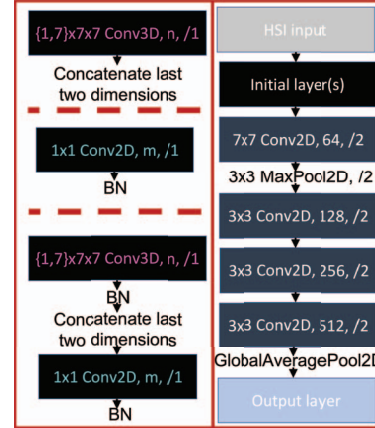


Figure 2. Top left: $\text{Conv3D}_{n,({1,7})x7x7}$. Middle left: DS_m . Bottom left: $\text{Conv3D}_{n,({1,7})x7x7}$ followed by DS_m . Right: The architecture of SimpleNet. The notation is kernel $h \times w$, number of filters, /stride.

4. Experiments

We wish to analyze whether we can use domain-specific knowledge regarding preprocessing of NIR spectra to design end-to-end trained CNNs. Preprocessing of NIR spectra can significantly increase the performance of subsequent linear models such as PLS [39]. However, training CNNs is several orders of magnitude slower than training PLS models, rendering it impossible to test a wide range of spectral preprocessing techniques for CNNs.

Our experiments with preprocessing techniques for CNNs are two-fold. First, we apply only simple preprocessing, such as channel-wise centering and scaling, and allow the CNNs to learn their own more advanced spectral preprocessing by applying different types of Conv3D layers. We compare this approach with optimizing the spectral preprocessing technique on PLS. Optimizing the spectral preprocessing technique on PLS allows us to test a wide range of combinations and apply the best one to the CNNs to assess if the optimal preprocessing technique for PLS is also beneficial to CNNs.

4.1. Spectral Preprocessing Optimization

Preprocessing techniques within NIR spectroscopy generally fall within the two categories of scatter correction and spectral derivatives, aiming to reduce the effects of physical phenomena on the spectrum, such as scattering, thus revealing the chemical information in the spectra [39, 42]. In our experiments, the scatter correction techniques are represented by Standard Normal Variate (SNV) [7], and detrending (Detrend) [7], while a wide range of Savitzky-Golay filters (SG) represents the spectral derivative techniques [40].

SNV standardizes each spectrum by subtracting its mean and dividing by its standard deviation. Detrend subtracts from each spectrum a polynomial that has been fitted to that

spectrum. We use a second-order polynomial as is standard for **Detrend** [39]. When used in unison, it is recommended to apply **SNV** before **Detrend** [7, 39].

SG applies convolution in the spectral dimension with a pre-defined filter that is determined by a window length w , polynomial order p , and derivative order d . The specific filter is denoted $\text{SG}_{w,p,d}$. Simply computing the derivative using finite differences would be infeasible due to noise inflation. **SG** tackles this issue by combining smoothing and spectral derivatives in a single filter by fitting a polynomial within a given window, thus getting a more acceptable signal-to-noise ratio [39]. We do not use padding when convolving with **SG** as this will introduce spectral artifacts [39]. We denote the lack of padding in **SG** and **Conv3D** by prepending a $\dagger$ to the name. When combining scatter correction (**SNV**) with differentiation (**SG**), scatter correction should always be applied first as the scatter correction techniques were designed for correction on raw spectra [39]. We test $\text{SG}_{w,p,d}$ with all unique combinations of $w \in \llbracket 3, 23 \rrbracket$, $p \in \llbracket 0, 3 \rrbracket$, $d \in \llbracket 0, 2 \rrbracket$, using only odd w . For each value of d , two subsequent values of p will yield the same filter coefficients. We remove any redundant filter coefficients and end with a total of 54 uniquely different $\text{SG}_{w,p,d}$ filters.

We also experiment with spectral centering (Center) and scaling (Scale) by computing for each training split the mean spectrum and standard deviation spectrum for channel-wise subtraction and subsequent channel-wise division, respectively. This preprocessing technique is commonly applied to PLS-R [51] and is beneficial to the convergence of neural networks [24, 41, 48]. Indeed, confirming that the literature also applies to our hyperspectral setting, we initially trained CNNs with and without Center $\rightarrow$ Scale and witnessed increased performance when applying Center $\rightarrow$ Scale. Thus, from now on, we focus on CNNs with Center $\rightarrow$ Scale. The mean spectrum and standard deviation spectrum are computed over all grain pixels using a two-pass version of the robust Welford’s algorithm [47]. When we apply the spectral preprocessing techniques in sequence, the ordering is first-to-last, **SNV** $\rightarrow$ **Detrend** $\rightarrow$ $\dagger\text{SG}$ $\rightarrow$ Center $\rightarrow$ Scale. This ordering entails that the mean and standard deviation spectra used in the Center and Scale operations are computed after any previous preprocessing application as they depend on any prior preprocessing applications. When used on hyperspectral images, spectral preprocessing is applied only to the grain pixels, and the background is masked out.

4.2. Comparative Studies

For PLS on both datasets, we experiment with applying **SNV**, **SNV** $\rightarrow$ **Detrend**, **SNV** $\rightarrow$ $\dagger\text{SG}$ and neither scatter correction nor spectral derivatives. For each of these, we experiment with applying Center, Center $\rightarrow$ Scale, and neither Center nor Scale. For the CNNs, we initially exper-

iment with the commonly used Center $\rightarrow$ Scale. Afterward, we extend this to include the preprocessing methods found optimal for PLS on Dataset #1 and Dataset #2, respectively. The best preprocessing technique for CV of PLS-R on Dataset #1 was **SNV** $\rightarrow$ $\dagger\text{SG}_{7,2,2}$ and for PLS2-DA validation on Dataset #2 Center was best. Using the optimal preprocessing techniques, the average optimal number of components for the CV of the best PLS-R model is A_{15} , and for validation of the best PLS2-DA, the optimum is A_{17} . In Figure 3, we show how the different combinations of **SNV**, **Detrend**, and $\dagger\text{SG}$ with their optimal combination of Center and Scale affect the performance of PLS in both datasets.

Based on the results of the preprocessing experiments with PLS, we design modifications of ResNet-18 and SimpleNet to understand if and how these results transfer to CNNs. These modifications include adding Ds_{15} for CNNs on Dataset #1 and Ds_{17} for Dataset #2. Additionally, as $\dagger\text{SG}_{7,2,2}$ is part of the optimal preprocessing for PLS-R on Dataset #1, we include $\dagger\text{Conv3D}_{1,(1 \times 1 \times 7)}$ as it shares the same window length and, as such, can learn the same filter if necessary. Engström *et al.* [14] showed that $\dagger\text{Conv3D}_{1,(1 \times 1 \times 7)}$ can cause instability during training. Therefore, we experiment with adding additional spectral $\dagger\text{Conv3D}$ filters and a spatio-spectral version $\dagger\text{Conv3D}_{3,(1 \times 1 \times 7)}$ and $\dagger\text{Conv3D}_{1,(7 \times 7 \times 7)}$, both having the same spectral window length, to see if this can alleviate the instability during training, which seemingly is the case.

All CNN results, and the best PLS results, laying the foundation for the dataset-specific CNN modifications, are shown in Table 2. Additionally, for each dataset, we compare our best CNN and best PLS model with those of Engström *et al.* [14] and Dreier *et al.* [12], respectively, in Table 3. We show confusion matrices and training and validation loss curves in the supplementary material. Engström *et al.* [14] report better performance for PLS-R than for CNN models. While closing this performance gap, our best CNN and PLS-R models achieve lower RMSE than the best from Engström *et al.* [14]. Furthermore, by constructing an ensemble CNN, we outperform PLS-R and the equivalent ensemble PLS-R. We create the ensemble by taking a uniform average of the predictions from each of the five cross-validated models. On Dataset #2, our best PLS2-DA model significantly outperforms the equivalent from Dreier *et al.* [12]. Our best CNN does not beat the best CNN from Dreier *et al.* [12]. However, they use a grain density of $\geq 50\%$, whereas ours is $\geq 10\%$. When we evaluate our model on test images with at least 50% grain density, we close much of the performance gap, indicating that lower grain density makes the classification task more difficult. The study by Dreier *et al.* [12] support this indication by showing that their classification accuracy diminishes greatly when grain density is $< 50\%$.

Mod. #	Conv3D _{n,(h×w×d)}	Ds _m (CNN) A _m (PLS)	Preprocessing	# param. (×10 ⁶) ResNet-18 / SimpleNet	RMSE (%) ± SEM (%) on Dataset #1 ResNet-18 / SimpleNet	Accuracy on Dataset #2 ResNet-18 / SimpleNet
Dataset #1 and Dataset #2						
1	None	None	Center → Scale	11.5 / 1.87	0.93 ± 0.06 / 0.92 ± 0.06	0.96 / 0.95
2	None	None	Center → Scale → Grayscale	11.2 / 1.56	1.43 ± 0.07 / 1.55 ± 0.05	0.91 / 0.82
3	Fully Conv3D	None	Center → Scale	33.2 / 4.67	0.92 ± 0.04 / 0.86 ± 0.03	0.96 / 0.91
4	Conv3D _{1,(1×1×7)}	None	Center → Scale	11.5 / 1.87	0.74 ± 0.05 / 0.76 ± 0.05	0.90 / 0.95
5	Conv3D _{3,(1×1×7)}	None	Center → Scale	12.1 / 2.51	0.72 ± 0.03 / 0.73 ± 0.04	0.95 / 0.92
6	Conv3D _{1,(7×7×7)}	None	Center → Scale	11.5 / 1.87	0.74 ± 0.04 / 0.81 ± 0.04	0.86 / 0.91
7	Conv3D _{3,(7×7×7)}	None	Center → Scale	12.1 / 2.51	0.68 ± 0.03 / 0.71 ± 0.03	0.86 / 0.86
8	None	Ds ₃	Center → Scale	11.2 / 1.56	1.18 ± 0.08 / 1.25 ± 0.13	0.97 / 0.93
9	Conv3D _{1,(1×1×7)}	Ds ₃	Center → Scale	11.2 / 1.56	0.82 ± 0.13 / 0.71 ± 0.02	0.91 / 0.94
10	Conv3D _{3,(1×1×7)}	Ds ₃	Center → Scale	11.2 / 1.56	0.66 ± 0.01 / 0.71 ± 0.01	0.94 / 0.92
11	Conv3D _{1,(7×7×7)}	Ds ₃	Center → Scale	11.2 / 1.56	0.69 ± 0.01 / 0.71 ± 0.03	0.88 / 0.93
12	Conv3D _{3,(7×7×7)}	Ds ₃	Center → Scale	11.2 / 1.56	0.67 ± 0.02 / 0.71 ± 0.02	0.92 / 0.93
Dataset #1						
PLS-R	-	A ₁₅	SNV → †SG _{7,2,2}	-	0.67 ± 0.01	-
13	None	Ds ₁₅	SNV → †SG _{7,2,2}	11.2 / 1.60	0.67 ± 0.02 / 0.72 ± 0.02	-
14	†Conv3D _{1,(1×1×7)}	Ds ₁₅	SNV	11.2 / 1.60	1.61 ± 0.20 / 1.36 ± 0.28	-
15	†Conv3D _{3,(1×1×7)}	Ds ₁₅	SNV	11.2 / 1.61	0.90 ± 0.23 / 0.71 ± 0.02	-
16	†Conv3D _{1,(7×7×7)}	Ds ₁₅	SNV	11.2 / 1.60	0.89 ± 0.23 / 0.92 ± 0.22	-
Dataset #2						
PLS2-DA	-	A ₁₇	Center	-	-	0.93
17	None	Ds ₁₇	Center	11.24 / 1.61	-	0.94 / 0.93

Table 2. Performance, number of parameters, and preprocessing of all CNN modifications and the best PLS-R and PLS2-DA models. The first 12 modifications are inspired or developed by Engstrøm *et al.* [14] and Dreier *et al.* [12] and are applied to Dataset #1 and Dataset #2. Modifications 13 – 16 are designed based on the best preprocessing methods for PLS-R on Dataset #1 and are used only on Dataset #1. Modification 17 is designed based on the best preprocessing methods for PLS2-DA on Dataset #2 and is used only on Dataset #2. All results shown are computed on the test splits of the respective datasets.

5. Discussion

The spectral dimension is critical to protein content regression, as evident by the high RMSE of the gray-scale

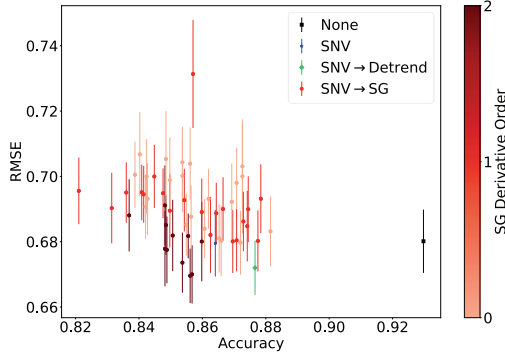


Figure 3. PLS-R and PLS2-DA performance ($\pm$ SEM) on the test sets of Dataset #1 (vertical axis) and Dataset #2 (horizontal axis). The models shown here share Center and Scale applications with the model that achieves the lowest validation RMSE and the highest validation accuracy, respectively. For PLS2-DA, Center has been applied, whereas this has not been applied for PLS-R. The red markers correspond to models that have applied $\text{SNV} \rightarrow \text{SG}$ preprocessing. Here, we choose the red shade based on the derivative order of the SG filter. Similar plots for the window length and polynomial order are provided in the supplementary material. The outlier with a high RMSE is likely due to the combination of a very large SG window length, a low derivative order, and a low polynomial order.

Best model	CNN	PLS
RMSE (%) on Dataset #1		
Ours	0.66 ± 0.01	0.67 ± 0.01
Ensemble (ours)	0.59	0.67
Engstrøm <i>et al.</i> [14]	0.90 ± 0.05	0.75 ± 0.01
Acc. (%) on Dataset #2		
Ours	96.7 (98.9)	93.0 (96.2)
Dreier <i>et al.</i> [12]	(99.8)	(79.9)

Table 3. Comparison of the best CNNs and PLS models with prior work on both datasets using their respective test splits. The numbers in parentheses indicate performances on a grain density of $\geq 50\%$. We use **bold** to highlight the best model on each dataset.

variant, modification #2, shown in Table 2 and supported by the results of Engstrøm *et al.* [14]. To achieve optimal performance, however, the spectral dimension requires delicate treatment. Removing physical phenomena in the spectra by preprocessing them with a combination of SNV , Detrend , and SG increases the downstream model’s ability to perform the chemometric regression analysis of determining protein content. While PLS can obtain decently low RMSE without this preprocessing, as seen in Figure 3, the importance of removing physical phenomena in the spectra is profound for CNNs. Inspecting Table 2 we see a vast decrease in RMSE for modification #13 applying $\text{SNV} \rightarrow \text{SG}_{7,2,2}$ when compared with the plain modification #1 applying the classical Center $\rightarrow$ Scale preprocessing.

Instead of trying different SG filters for CNNs, it is pos-

sible to have them learn their own spectral preprocessing filters using Conv3D . These learned filters converge towards SG for protein content regression. An example of this is shown for the model with the lowest RMSE on the protein regression, ResNet-18 modification #10, in Figure 4. This phenomenon of Conv3D learning to approximate some variation of SG , usually a higher-order polynomial and derivative, is a tendency for every CNN trained on Dataset #1 with an initial Conv3D , spectral and spatio-spectral alike. Plots showing this tendency for our other models are provided in the supplementary material.

As shown, applying $\text{SNV} \rightarrow \text{SG}_{7,2,2}$, found optimal by PLS-R, to the CNNs yields sublime results. However, by applying only SNV and allowing the CNNs to subsequently learn the remaining spectral preprocessing by training $\text{Conv3D}_{1,(1 \times 1 \times 7)}$ as done in modification #14, results worsen significantly. We hypothesize that applying a local centering and scaling of the spectra, e.g., as done by SNV , leads to instability during training. However, this issue seems to be somewhat relieved by increasing the number of spectral filters either directly as done in modification #15 or indirectly by extending Conv3D across the spatial dimensions as done in modification #16. This hypothesis is supported by the studies done by Engstrøm *et al.* [14] whom experience the same instability for their model using $\text{Conv3D}_{1,(1 \times 1 \times 7)}$ and significantly better stability for $\text{Conv3D}_{3,(1 \times 1 \times 7)}$ and $\text{Conv3D}_{1,(7 \times 7 \times 7)}$ alike. They apply Center $\rightarrow$ Scale with mean and standard deviation spectra being computed locally on an image crop basis. While this is not as local as SNV , it is significantly more local than our global strategy.

These effects can lead to a hypothesis that increasing the number of Conv3D filters will lead to even better results. However, inspecting the $\text{Conv3D}_{3,(1 \times 1 \times 7)}$ layer learned by ResNet-18 modifications #10 as shown in Figure 4, two of the filters learn zero-responses and contribute nothing to its downstream responses. Indeed, inspecting the same model’s Ds_3 , shown in Figure 5, where all three filters learn zero responses for the parts of the input signal’s spectral dimension that correspond to the zero-response filters of the prior $\text{Conv3D}_{3,(1 \times 1 \times 7)}$ layer. This tendency is prevalent for all our models employing $\text{Conv3D}_{3,(1 \times 1 \times 7)}$. Similar plots for the remaining models are provided in the supplementary material. Additionally, the fact that modification #2, the Fully Conv3D CNNs, the first layers of which employ $\text{Conv3D}_{64,(7 \times 7 \times 7)}$, do not provide any benefit over the standard 2D CNNs, further supports that applying as many Conv3D filters as possible is not beneficial.

Grain variety classification can be solved reasonably well using only spatial or only spectral information, as evident in Table 2 by the grayscale modification #2 and PLS2-DA. These results are supported by those of Dreier *et al.* [12]. However, simultaneously utilizing spectral and

spatial information enables better performance than utilizing either alone, as evident by the plain CNNs with modification #1. For this task, the treatment of the spectral dimension is also essential, as evident for the PLS2-DA experiments shown in Figure 3. Here removing physical phenomena of the spectral dimension is detrimental to the classification accuracy. Not removing the physical phenomena of the spectra allows the PLS2-DA model to achieve much better accuracy than applying any combination of *SNV*, *Detrend*, or *†SG*. The results correspond with previous studies of grain classification using spectral analysis. They indicate that grain classification relies on the grains' physical characteristics and that spectral preprocessing has a detrimental effect on accuracy [5]. While less critical to the performance than for PLS2-DA, adding a single initial *Conv3D* layer to the CNNs offers no increase in classification accuracy. Inspecting the responses learned by the filters of *Conv3D* for the classification CNNs reveals that they do learn responses approximating those of *†SG* but always of derivative order 0 or 1, performing smoothing and removal of only additive effects, while maintaining the multiplicative effects. These results indicate that even when offered the opportunity as a trainable *Conv3D* layer, the CNNs learn to maintain the physical phenomena of the spectra to a higher degree than filters learned by the protein content regression CNNs. Further, inspecting Figure 3 shows a tendency towards high-order derivatives yielding better protein content regression performance. At the same time, this is not the case for grain variety classification, where the best physical-phenomena-removing preprocessing technique uses a 0 order derivative. Interestingly, when learning multiple *Conv3D*, the classification CNNs share the tendency of having one filter learning a dominant response. Furthermore, inspecting their subsequent *Ds*, it is evident that only the part of the input corresponding to the dominant *Conv3D* filter learns a non-minuscule response. These results indicate that employing multiple spectral filters only stabilizes the model during training but is not needed to learn multiple responses for any single-task prediction. While choosing the optimal value of *A* for PLS is essential to reduce the risk of overfitting, the role of *Ds* in CNNs is less profound. Engström *et al.* [14], witness increased regression performance by employing *Ds*₃ to the plain ResNet-18 while in our studies, we notice an adverse effect for the same modification. However, it is usually beneficial when combined with a prior *Conv3D* layer. For grain variety classification, the effect seems to be minuscule.

The proper choice of spectral preprocessing method is challenging to assess before model validation for chemometric models [39, 42], yet it is required for achieving optimal performance [13]. Our CNN modifications can learn this spectral preprocessing autonomously. In this study, we have devised a blueprint for CNN design that aids in bridg-

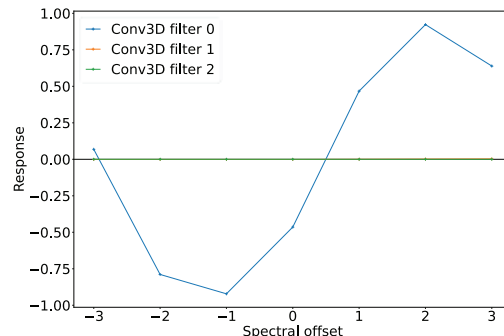


Figure 4. The three filters of *Conv3D*_{3,(1×1×7)} from ResNet-18 modification #10 validated on validation split 1 of Dataset #1. *Conv3D* filter 1 and *Conv3D* filter 2 lie on top of each other at the line Response = 0.

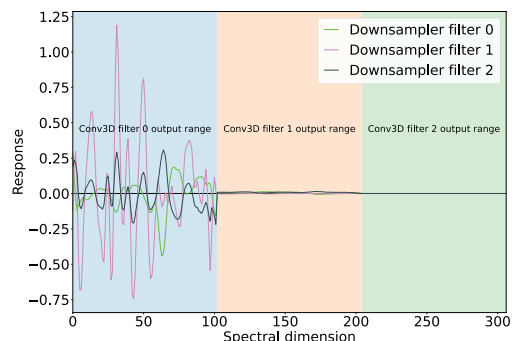


Figure 5. The three filters of *Ds*₃ from ResNet-18 modification #10 validated on validation split 1 of Dataset #1. The background area is colored with respect to the previous *Conv3D* filter responsible for producing this part of the input to the *Ds*₃.

ing the gap between generic deep learning for image analysis and domain-specific problems within chemometrics.

6. Conclusion

In this study, we have shown that by applying knowledge about the relationship between physical and chemical properties to the task at hand, we can design a modification to a standard CNN that significantly improves its predictive performance on hyperspectral images of grain. Adding an initial *Conv3D* layer allows the CNN to learn to remove irrelevant physical effects from the spectrum, thereby performing significantly better than its plain counterpart for regression analysis on the chemical parameter of protein content. However, when physical properties contain information valuable to the task, *Conv3D* filters offer no benefit over plain CNNs. Indeed, the study shows that using the proper modification is much more critical to the CNN's performance than the complexity of the network, as demonstrated by our SimpleNet modifications with a single initial *Conv3D* layer outperforming the plain ResNet-18 on protein content regression.

References

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. **3**
- [2] A. Alin. Comparison of pls algorithms when number of objects is much larger than number of variables. *Statistical papers*, 50(4):711, 2009. **4**
- [3] M. Andersson. A comparison of nine pls1 algorithms. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 23(10):518–529, 2009. **4**
- [4] Foss Analytical A/S. <https://www.fossanalytics.com/>. **2**
- [5] Y. Bao, C. Mi, N. Wu, F. Liu, and Y. He. Rapid classification of wheat grain varieties using hyperspectral imaging and chemometrics. *Applied Sciences*, 9(19):4119, 2019. **8**
- [6] M. Barker and W. Rayens. Partial least squares for discrimination. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(3):166–173, 2003. **3**
- [7] R. J. Barnes, M. S. Dhanoa, and S. J. Lister. Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra. *Applied spectroscopy*, 43(5):772–777, 1989. **4, 5**
- [8] L. Benos, A. C. Tagarakis, G. Dolias, R. Berruto, D. Kateris, and D. Bochtis. Machine learning in agriculture: A comprehensive updated review. *Sensors*, 21(11):3758, 2021. **1**
- [9] N. Caporaso, M. B. Whitworth, and I. D. Fisk. Near-infrared spectroscopy and hyperspectral imaging for non-destructive quality assessment of cereal grains. *Applied spectroscopy reviews*, 53(8):667–687, 2018. **1**
- [10] F. Chollet et al. Keras. <https://keras.io>, 2015. **3**
- [11] B. S. Dayal and J. F. MacGregor. Improved pls algorithms. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 11(1):73–85, 1997. **4**
- [12] E. S. Dreier, K. M. Sørensen, T. Lund-Hansen, B. M. Jespersen, and K. S. Pedersen. Hyperspectral imaging for classification of bulk grain samples with deep convolutional neural networks. *Journal of Near Infrared Spectroscopy*, 30(3):107–121, 2022. **2, 3, 5, 6, 7**
- [13] Z. Du, W. Tian, M. Tilley, D. Wang, G. Zhang, and Y. Li. Quantitative assessment of wheat quality using near-infrared spectroscopy: A comprehensive review. *Comprehensive Reviews in Food Science and Food Safety*, 21(3):2956–3009, 2022. **1, 3, 8**
- [14] O.-C. G. Engstrøm, E. S. Dreier, and K. S. Pedersen. Predicting protein content in grain using hyperspectral deep learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1372–1380, 2021. **2, 3, 5, 6, 7, 8**
- [15] L. Feng, S. Zhu, F. Liu, Y. He, Y. Bao, and C. Zhang. Hyperspectral imaging for seed quality and safety inspection: A review. *Plant methods*, 15(1):1–25, 2019. **1**
- [16] E. Gallup, T. Gallup, and K. Powell. Physics-guided neural networks with engineering domain knowledge for hybrid process modeling. *Computers & Chemical Engineering*, page 108111, 2023. **2**
- [17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. **4**
- [18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. **1, 3**
- [19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. **3, 4**
- [20] D. Ishikawa, M. Ishigaki, and A. A. Gowen. Nir imaging. *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*, pages 517–551, 2021. **1**
- [21] A. Kamilaris and F. X. Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018. **1**
- [22] A. Khan, A. D. Vibhute, S. Mali, and C. H. Patil. A systematic review on hyperspectral imaging technology with a machine and deep learning methodology for agricultural applications. *Ecological Informatics*, page 101678, 2022. **1**
- [23] B. J. Kim, H. Choi, H. Jang, D. G. Lee, W. Jeong, and S. W. Kim. Guidelines for the regularization of gammas in batch normalization for deep residual networks. *arXiv preprint arXiv:2205.07260*, 2022. **4**
- [24] Y. LeCun, I. Kanter, and S. Solla. Second order properties of error surfaces: Learning time and generalization. *Advances in neural information processing systems*, 3, 1990. **5**
- [25] L. C. Lee, C.-Y. Liong, and A. A. Jemain. Partial least squares-discriminant analysis (pls-da) for classification of high-dimensional (hd) data: a review of contemporary practice strategies and knowledge gaps. *Analyst*, 143:3526–3539, 2018. **3**
- [26] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis. Machine learning in agriculture: A review. *Sensors*, 18(8):2674, 2018. **1**
- [27] M. Manley. Near-infrared spectroscopy and hyperspectral imaging: non-destructive analysis of biological materials. *Chemical Society Reviews*, 43(24):8200–8214, 2014. **1**
- [28] M. Manley and P. J. Williams. Applications: food science. *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*, pages 347–359, 2021. **1**
- [29] V. Meshram, K. Patil, V. Meshram, D. Hanchate, and S. D. Ramkteke. Machine learning in agriculture domain: A state-of-art survey. *Artificial Intelligence in the Life Sciences*, 1:100010, 2021. **1**
- [30] N. Muralidhar, M. R. Islam, M. Marwah, A. Karpatne, and N. Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE international conference on big data (big data)*, pages 36–45. IEEE, 2018. **2**
- [31] J. G. Nuttall, G. J. O’leary, J. F. Panozzo, C. K. Walker, K. M. Barlow, and G. J. Fitzgerald. Models of grain qual-

- ity in wheat—a review. *Field crops research*, 202:136–145, 2017. [1](#)
- [32] B. G. Osborne. Near-infrared spectroscopy in food analysis. *Encyclopedia of analytical chemistry: applications, theory and instrumentation*, 2006. [1](#), [2](#)
- [33] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transaction on Systems, Man and Cybernetics*, SMC-9(1):62–66, 1979. [2](#)
- [34] N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh. Deep learning vs. traditional computer vision. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 I*, pages 128–144. Springer, 2020. [1](#)
- [35] D. I. Patrício and R. Rieder. Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review. *Computers and electronics in agriculture*, 153:69–81, 2018. [1](#)
- [36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [4](#)
- [37] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017. [1](#)
- [38] C. Ren, D.-K. Kim, and D. Jeong. A survey of deep learning in agriculture: techniques and their applications. *Journal of Information Processing Systems*, 16(5):1015–1033, 2020. [1](#)
- [39] Á. Rinnan, F. van den Berg, and S. B. Engelsen. Review of the most common pre-processing techniques for near-infrared spectra. *TrAC Trends in Analytical Chemistry*, 28(10):1201–1222, 2009. [1](#), [3](#), [4](#), [5](#), [8](#)
- [40] A. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964. [4](#)
- [41] Nicol N Schraudolph. Centering neural network gradient factors. In *Neural Networks: Tricks of the Trade*, pages 207–226. Springer, 2002. [5](#)
- [42] K. M. Sørensen, F. van den Berg, and S. B. Engelsen. Nir data exploration and regression by chemometrics—a primer. *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*, pages 127–189, 2021. [1](#), [4](#), [8](#)
- [43] Specim. Specim fx17, <https://www.specim.fi/products/specim-fx17/>. [2](#)
- [44] P. J. Trainor, A. P. DeFilippis, and S. N. Rai. Evaluation of classifier performance for multiclass phenotype discrimination in untargeted metabolomics. *Metabolites*, 7(2):30, 2017. [4](#)
- [45] P. Vithu and J. A. Moses. Machine vision system for food grain quality evaluation: A review. *Trends in Food Science & Technology*, 56:13–20, 2016. [1](#)
- [46] C. Wang, B. Liu, L. Liu, Y. Zhu, J. Hou, P. Liu, and X. Li. A review of deep learning used in the hyperspectral image analysis for agriculture. *Artificial Intelligence Review*, 54(7):5205–5253, 2021. [1](#)
- [47] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. [5](#)
- [48] S. Wiesler and H. Ney. A convergence analysis of log-linear training. *Advances in Neural Information Processing Systems*, 24, 2011. [5](#)
- [49] H. Wold. Estimation of principal components and related models by iterative least squares. *Multivariate analysis*, pages 391–420, 1966. [1](#), [3](#)
- [50] S. Wold, C. Albano, W. J. Dunn, K. Esbensen, S. Hellberg, E. Johansson, M. Sjöström, H. Martens, and J. Russwurm. Food research and data analysis. *London: H. Martens and H. Russwurm Jr*, 1983. [3](#)
- [51] S. Wold, M. Sjöström, and L. Eriksson. Pls-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001. [3](#), [5](#)

Chapter 5

Bulk References for Subsample Predictions

IN the previous chapter, Engstrøm et al. (2023b) trained models for grain variety classification and protein content regression by using bulk reference values as ground truths for image crops containing only a small subsample of the bulk. In addition to evaluating the performance of subsample (image crop) protein content predictions, we are interested in evaluating the mean prediction per bulk to compare with the bulk reference values. Interestingly, a bias and scale difference between the predictions and the references arises when evaluating performance on the mean bulk predictions. This means that bulks with a low reference value have their protein content overestimated, and those with a high reference value have their protein content underestimated. This phenomenon arises for both PLS and CNN predictions. A simple bias and scale correction corrects these errors. We also looked into the predictive performance as a function of the grain density (grain-to-background ratio) in image crops for both protein content regression and grain variety classification. We found that a higher density correlates with better performance for both tasks.

This chapter is written as a condensed version of Engstrøm et al. (2023a) with some additional insights that arose after publishing the report, which is reprinted in its entirety in Appendix A.

5.1 Introduction

Engstrøm et al. 2023a analyzed two datasets of NIR hyperspectral images of grain. Dataset #1 is from Engstrøm et al. (2021) and contains bulk samples of wheat grain kernels with an associated mean protein reference value for each bulk. Dataset #2 is from Dreier et al. (2022) and contains bulk samples of eight different grain varieties, seven wheat, and one rye. A bulk sample yielded approximately ten images for both datasets, each containing many individual

grain kernels. In an attempt to generate enough image data to train a CNN, Engstrøm et al. (2023b) inflated the datasets by cropping the images (see, e.g., Figure 1 by Engstrøm et al. 2023a), generating on the order of a thousand image crops per bulk sample. Subsequently, each image crop was assigned the same reference value as the bulk from which it originated. In Dataset #2, the bulks and images are homogeneous, meaning that each contains grain kernels from exactly one of the eight varieties. Therefore, the reference value of the bulk is equally valid as a reference for the subsample. In contrast, the bulk mean protein references in Dataset #1 are unlikely to be accurate for each image crop as some variation within the bulk is to be expected. However, by definition, the bulk reference will be correct for the image crops on average.

Despite this, averaging the image crop predictions for a bulk does not accurately estimate the bulk’s protein content. Engstrøm et al. (2023a) analyze why this happens and propose a simple bias and scale correction for obtaining accurate bulk protein content estimates. Section 5.2 elaborates on this analysis. It will also show that we were wrong in our hypothesis about why there is a discrepancy between the least squares solution for image crop predictions and bulk mean predictions. However, the bias and scale corrections are still valid and necessary. Section 5.3 shows the relationship between grain density and the predictive performance of grain variety classifiers and protein content regressors due to Engstrøm et al. (2023a).

5.2 Bulk Mean and Subsample Prediction Discrepancy

In our work (Engstrøm et al. 2023b), we determined the optimal preprocessing for PLS and the optimal modification of ResNet-18 for protein content regression. Then, we constructed an ensemble of each model type for final evaluation on the test set. Figure 5.1 shows predictions for each image crop (subsample) made by the ensembles. Here, $\mathbf{y}$ is a vector of reference values, and $\hat{\mathbf{y}}$ is a vector of predictions. The subscripts _{ss} and _{test} mean subsample and test set, respectively. For example, $\hat{\mathbf{y}}_{ss,test}$ is the vector of subsample predictions for the test set, and $\mathbf{y}_{ss,test}$ are the associated reference values. sYX is the RMSE obtained after using ordinary least squares (OLS) to adjust the predictions to match the references best. Due to the estimation of bias and scale, sYX has two fewer degrees of freedom than ordinary RMSE. The estimated bias and scale that best adjust $\hat{\mathbf{y}}_{ss,test}$ to $\mathbf{y}_{ss,test}$ are given by $\text{bias}_{ss,test}$ and $\text{scale}_{ss,test}$. Note that because the predictions are on the vertical axis and the references are on the horizontal axis, the least squares solution minimizes the squared horizontal residuals. This corresponds to finding the linear transformation that must be applied to the predicted values to minimize RMSE between them and the reference values. In Figure 5.1, the RMSE quite closely matches the sYX. For the training sets, the lines of best fit almost perfectly

5.2. BULK MEAN AND SUBSAMPLE PREDICTION DISCREPANCY 67

match the identity (see Figure 5 in Appendix A), indicating that the models are well-calibrated, meaning they have found a least squares solution.

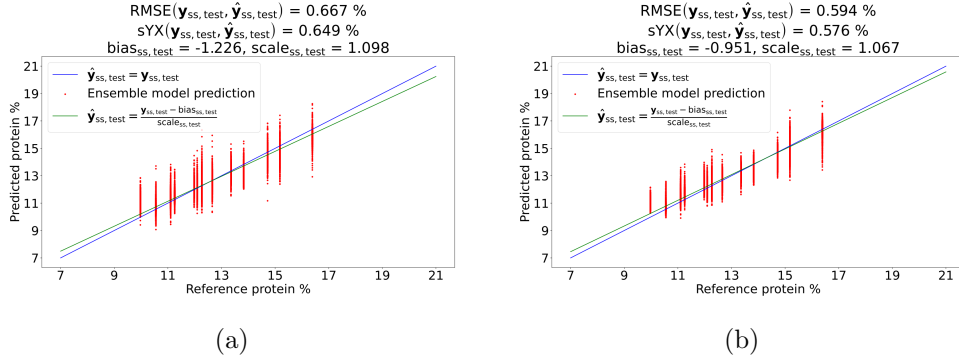


Figure 5.1: Scatter plots of the predicted protein content distribution for each of the 13 bulk samples in the test set. (a) PLS. (b) ResNet-18. This figure is copied from Engstrøm et al. (2023a).

To assess the correctness of the predictions, we can compare the mean prediction for each bulk with its reference. The bulk mean predictions corresponding to the subsample predictions from Figure 5.1 are shown in Figure 5.2. The subscript $_{bm}$ stands for bulk mean. Here, the RMSE and sYX differ significantly, and a clear bias and scale correction are needed to get well-calibrated predictions. The same phenomenon arises for training and validation set predictions (see Figure 4 in Appendix A). It seems always to be the case that the models overestimate the mean protein content for bulks with low reference values and underestimate the mean protein content for bulks with high reference values.

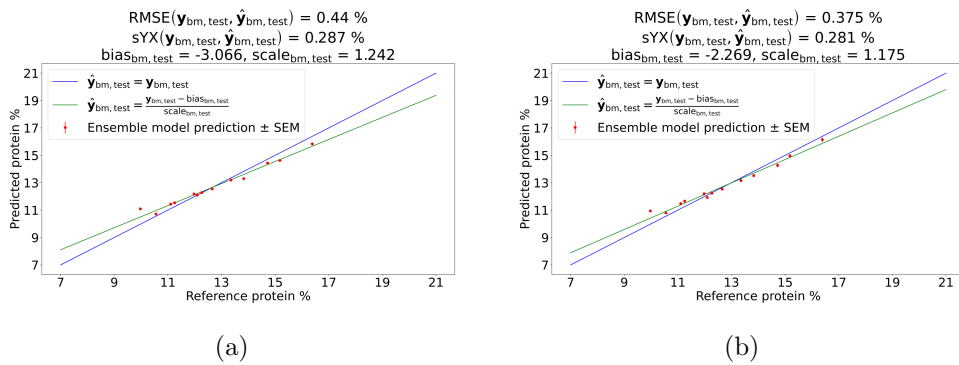


Figure 5.2: Scatter plots of the means of the predicted protein content distribution for each of the 13 bulk samples in the test set. (a) PLS. (b) ResNet-18. This figure is copied from Engstrøm et al. (2023a).

In the technical report by Engstrøm et al. 2023a, we claim that the dis-

crepancy between the well-calibrated predictions for subsamples and the inaccurate bulk mean predictions is due to non-symmetric predicted subsample distributions for each bulk. In particular, we claim that, given the predicted subsample distributions for each bulk are symmetric, the least squares solution for adjusting $\hat{\mathbf{y}}_{\text{ss}}$ to $\mathbf{y}_{\text{ss}}$ is the same as that for adjusting $\hat{\mathbf{y}}_{\text{bm}}$ to $\mathbf{y}_{\text{bm}}$. Indeed, the predicted subsample distributions are asymmetric (Engström et al. 2023a). However, even if the predicted subsample distributions for each bulk are symmetric and well-calibrated, the mean prediction for bulks with low reference values will be overestimated, and the mean prediction for bulks with high reference values will be underestimated. Consider, for example, the simulation shown in Figure 5.3. It shows predictions for two bulk samples, each with four subsamples. The two subsample prediction distributions are symmetric. In Figure 5.3a, the line of best fit for the subsample predictions is the identity. However, the line of best fit for the mean predictions is not the identity. To transform the predictions, such that the identity becomes the line of best fit for the mean predictions, the predicted values must be multiplied by a scale of 1.56 and added with a bias of -7.22 as shown in Figure 5.3b.

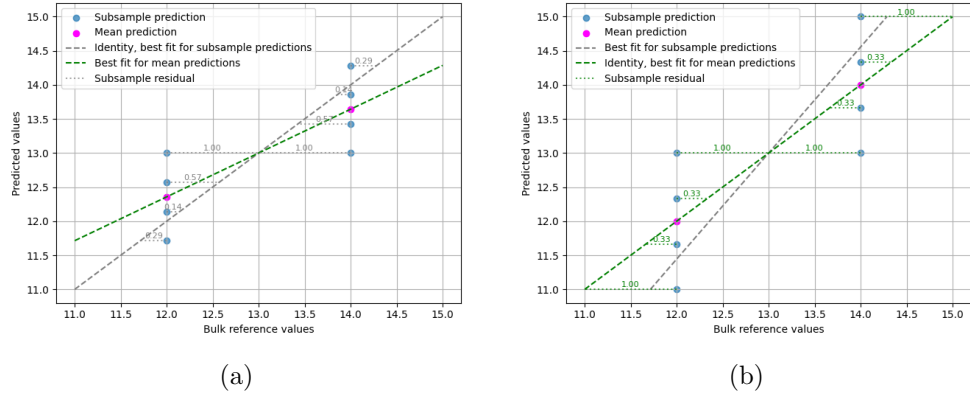


Figure 5.3: Simulation of symmetric predicted subsample distributions for two bulks. The plots show the horizontal residuals from the subsample predictions to the identity. (a) The identity is the line of best fit for the subsample predictions, indicating a well-calibrated model minimizing subsample RMSE. (b) Scale (1.56) and bias (-7.22) corrected predicted values from (a) based on the best fit for the mean predictions. Thus, the identity becomes the line of best fit for the mean predictions, indicating a well-calibrated model minimizing bulk mean RMSE.

When we trained our models to minimize RMSE, we trained to minimize the RMSE for subsamples, corresponding to Figure 5.3a. In practice, however, we want to minimize the RMSE for mean predictions, corresponding to Figure 5.3b. It is evident that these two problems, although deceptively similar, do not share a common solution. The underlying reason for this phenomenon

is still unknown when writing this thesis. Still, it is straightforward to compensate for this discrepancy with a simple bias and scale correction as shown by Engstrøm et al. (2023a). When performing this correction, the bias and scale parameters must be chosen independently from the test set so as not to introduce data leakage. Thus, we can estimate the parameters from the training or validation set. After having conducted this correction in our work (Engstrøm et al. 2023a), we became aware that we could also use the union of the training and validation set, analogous to what Igel and Oehmcke (2023) suggests for bias and scale correction of deep learning regressors.

Figure 5.4 shows the bulk mean predictions, corrected with bias and scale parameters from the training set. Here, we achieve an RMSE of approximately 0.28% for both the PLS and ResNet-18 ensembles. This RMSE matches the standard deviation of the InfratecTM NOVA's (FOSS Analytical A/S 2025) measurement, which was used to obtain the reference values. Thus, this RMSE is the lowest that can be expected from a model that perfectly estimates protein content without considering the InfratecTM NOVA's measurement errors. Based on these results, inflating an NIR-HSI dataset by cropping the images and applying the mean reference to all image crops is a valid strategy for generating enough data to train a CNN. However, care must be taken to compensate for the discrepancy between the least squares solution for subsample predictions and the bulk mean predictions. As a final note in this section, inflating the dataset might be necessary to generate enough data for training CNNs, but it is not necessary for training PLS models. We (Engstrøm et al. 2023a) trained a PLS model directly on the bulk mean spectra and got results matching those in Figure 5.4 (see Figure 7 (c) in Appendix A).

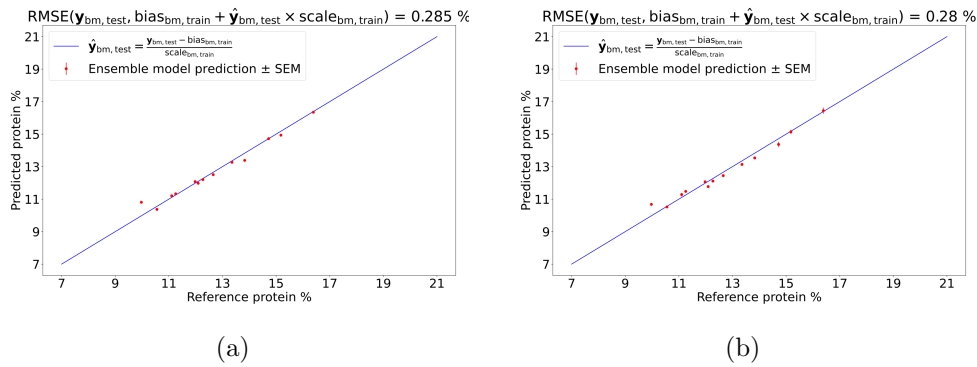


Figure 5.4: Scatter plots of the bias and scale corrected means of the predicted protein content distribution for each of the 13 bulk samples in the test set. (a) PLS. (b) ResNet-18.

5.3 Grain Density and Model Performance

Another relevant insight due to Engstrøm et al. (2023a) is an analysis of the relationship between grain density and model performance for protein content regression and grain variety classification. Recall that both datasets consist of sparsely and densely packed image crops with varying grain densities of at least 0.1 as illustrated in Figure 5.5.

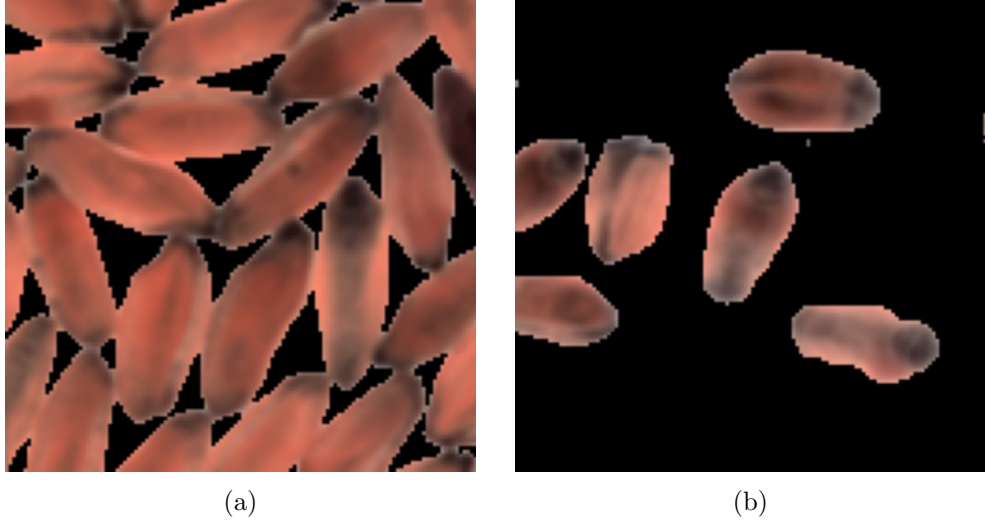


Figure 5.5: Image crops with (a) densely packed grain with a grain density of 0.67 (b) sparsely packed grain with a grain density of 0.13. This figure is copied from Engstrøm et al. (2023a).

For protein content regression on Dataset #1, we expect an image crop to have a true protein content closer to the bulk reference as its grain density increases. We expect this to be reflected in our models by making image crop-wise predictions closer to the bulk reference as the grain density increases. We use the ensembles of best-performing ResNet-18 and PLS models mentioned in Section 5.2 to analyze whether this hypothesis holds. Figure 5.6 shows grain density versus mean RMSE and the standard error on the mean (SEM) for both the ResNet-18 and PLS ensembles. The mean RMSEs are computed across the five constituents of each ensemble to see intra-ensemble agreement, which is high for both PLS and ResNet-18. For both model types, it seems clear that RMSE decreases as grain density increases.

As mentioned previously, for grain variety classification between the eight varieties in Dataset #2, the bulk references are also entirely valid for the image crops due to a bulk consisting of exactly one grain variety. Still, higher-density image crops contain more grain and, thus, potentially, a stronger signal for the models to respond to. Figure 5.7 shows accuracy as a function of grain kernel density for the best CNN (a ResNet-18 modification) and a PLS model

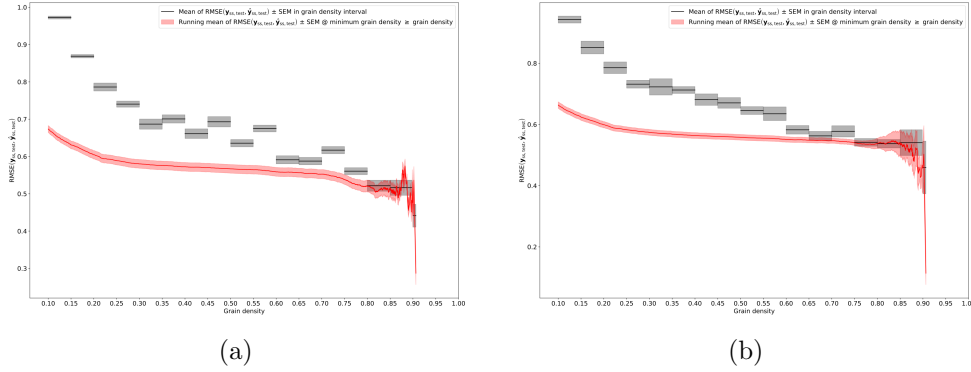


Figure 5.6: Predicted protein RMSE as a function of grain density for (a) PLS and (b) ResNet-18. The gray lines show RMSE for predictions on image crops inside that grain density interval. At any point on the grain density axis, the red curve shows RMSE for predictions on image crops with at least that high grain density. This figure is copied from Engstrøm et al. (2023a).

with optimal preprocessing as determined by Engstrøm et al. (2023b). While the ResNet-18 outperforms the PLS model, both see a clear drop in accuracy when grain density falls below 0.3 and are otherwise relatively stable above 0.3. These results contrast those by Dreier et al. (2022), where accuracy falls drastically when grain density decreases below 0.5. For grain density between 0.1 and 0.2, their accuracy falls below 0.4 for a ResNet-18. The difference, however, is that Engstrøm et al. (2023b) train on image crops with a grain density of at least 0.1, whereas Dreier et al. (2022) use a grain density of at least 0.5 for training and so anything below is out of domain.

Thus, models can benefit from training on image crops with low grain density for grain variety classification, especially if these are encountered during evaluation. Higher grain density also correlates somewhat with higher classification accuracy, especially around the 0.3 grain density threshold.

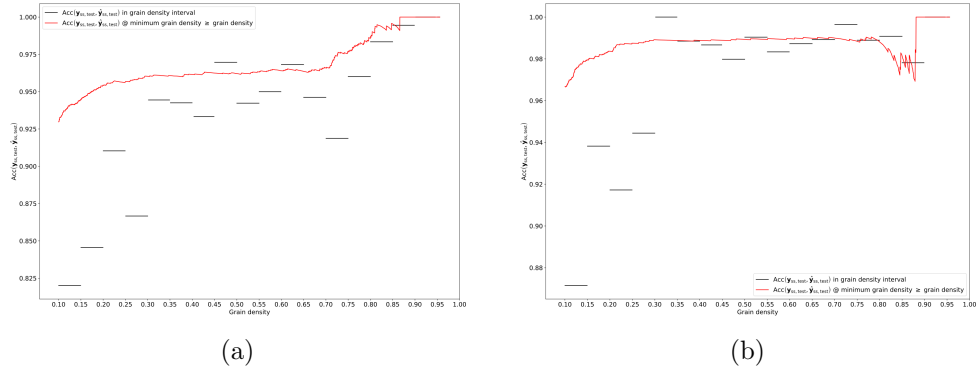


Figure 5.7: Grain variety classification accuracy as a function of grain density for (a) PLS and (b) ResNet-18. The gray lines show accuracy for predictions on image crops inside that grain density interval. At any point on the grain density axis, the red curve shows accuracy for predictions on image crops with at least that high grain density. This figure is copied from Engstrøm et al. (2023a).

Chapter 6

Chemical Maps

THE previous chapters have shown that while hyperspectral images can be used to model chemical parameters, the spectral signals alone are sufficient for accurate analysis. Training a deep learning model such as ResNet-18 on hyperspectral images is extremely computationally expensive compared to training PLS on spectra. Coupled with the high cost of an NIR-HSI platform compared to the price of a spectrometer, it may initially seem difficult to justify using NIR-HSI to model chemical parameters. However, a use case exists that justifies and necessitates an NIR-HSI platform for the modeling of chemical parameters. That is the modeling of the spatial distribution of a chemical parameter throughout a sample. The spatial distribution can be visualized in so-called chemical maps, which are images of pixel-wise predictions generated from an input hyperspectral image.

Chemical maps are commonly generated by training a PLS model on mean spectra and subsequently using it to predict every pixel (spectrum) in the input hyperspectral images. This approach has multiple issues, including non-smooth chemical maps, pixel-wise predictions outside the physically possible range, and a discrepancy between the mean of the predicted distribution and a prediction on the mean spectrum. In the following work (Engstrøm et al. 2025a), we compare PLS-generated chemical maps with those generated by U-Net. U-Net (Ronneberger et al. 2015) is a CNN originally designed for semantic segmentation. In semantic segmentation, each pixel of an input image must be classified as belonging to some class. Chemical map generation is the equivalent regression task. Therefore, based on our previous work with CNNs for chemical parameter regression on hyperspectral images (Engstrøm et al. 2021; Engstrøm et al. 2023b), we augment the U-Net with an initial spectral convolution layer and use it to generate chemical maps. Indeed, with a customized loss function, directly addressing the issues previously mentioned with PLS, we can generate smooth chemical maps with pixel-wise predictions strictly inside the physically possible range with mean values matching the bulk references.

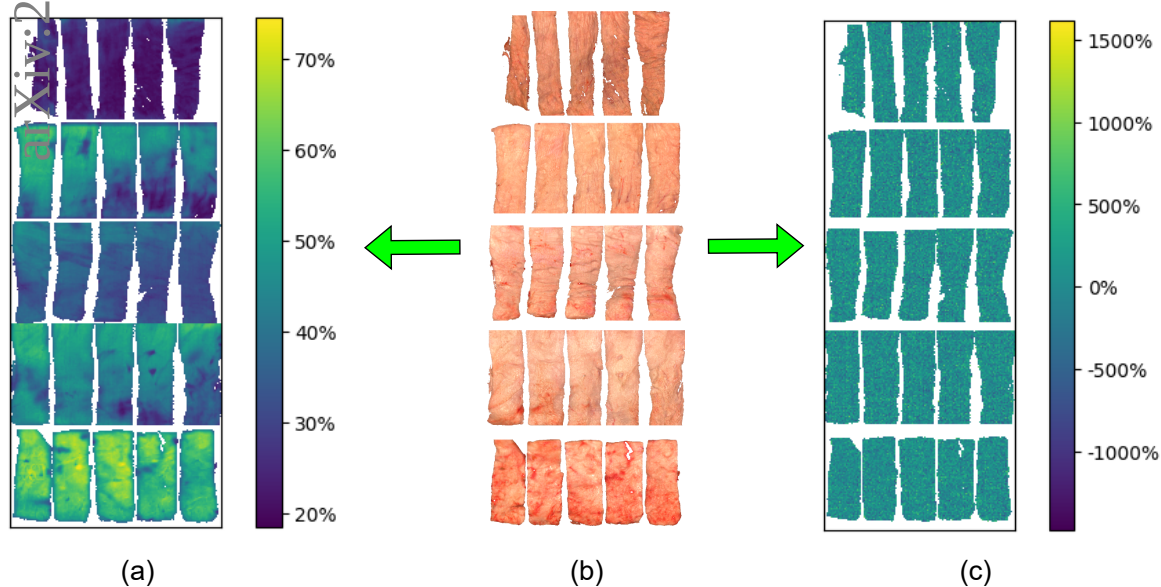
In this work, we use cheap-to-obtain bulk mean references similar to those used in the previous chapters. However, instead of forcing U-Net’s pixel-wise predictions to match the bulk mean reference, we allow for pixel-wise deviations from the mean reference by only comparing the mean of the predicted spatial distribution with the bulk mean reference. While a bulk mean reference is computed across a physical sample corresponding to five images, it seems likely that the discrepancy between the mean bulk reference value and the actual true value for each image is smaller and subject to less variance than in our previous work with protein content regression on grain where each bulk sample corresponded to thousands of image crops. Indeed, the line of best fit for the mean predictions generated by Engstrøm et al. (2025a) exhibits only a slight deviation from the identity.

Graphical Abstract

Transforming Hyperspectral Images Into Chemical Maps: An End-to-End Deep Learning Approach

Ole-Christian Galbo Engstrøm, Michela Albano-Gaglio, Erik Schou Dreier, Yamine Bouzembrak, Maria Font-i-Furnols, Puneet Mishra, Kim Steenstrup Pedersen

Chemical map generation of fat content in pork bellies



Predicted fat distribution on near-infrared hyperspectral images of pork bellies.

(a) Modified U-Net predictions (proposed method).

(b) RGB visualizations of input images.

(c) PLS predictions (standard method).

Highlights

Transforming Hyperspectral Images Into Chemical Maps: An End-to-End Deep Learning Approach

Ole-Christian Galbo Engstrøm, Michela Albano-Gaglio, Erik Schou Dreier, Yamine Bouzembrak, Maria Font-i-Furnols, Puneet Mishra, Kim Steenstrup Pedersen

- Joint use of spatial and spectral dimensions for modeling of hyperspectral images.
- A modified U-Net can accurately generate chemical maps from hyperspectral images.
- Qualitative and quantitative improvement over chemical maps generated by PLS.

Transforming Hyperspectral Images Into Chemical Maps: An End-to-End Deep Learning Approach

Ole-Christian Galbo Engstrøm^{a,b,c}, Michela Albano-Gaglio^d, Erik Schou Dreier^a, Yamine Bouzembrak^e, Maria Font-i-Furnols^d, Puneet Mishra^f, Kim Steenstrup Pedersen^{b,g}

^aFOSS Analytical A/S, Nils Foss Allé 1, Hillerød, 3400, Denmark

^bDepartment of Computer Science (DIKU), University of Copenhagen, Universitetsparken 1, Copenhagen, 2100, Denmark

^cDepartment of Food Science (UCPH FOOD), University of Copenhagen, Rolighedsvej 26, Frederiksberg, 1958, Denmark

^dIRTA - Food Quality and Technology, Finca Camps i Armet, Monells, 17121, Girona, Spain

^eInformation Technology Group, Wageningen University and Research (WUR), Stippeneng 4, Wageningen, 6706 KN, Gelderland, The Netherlands

^fFood and Biobased Research, Wageningen University and Research (WUR), Stippeneng 4, Wageningen, 6708 WE, Gelderland, The Netherlands

^gNatural History Museum of Denmark (NHMD), University of Copenhagen, Universitetsparken 15, Copenhagen, 2100, Copenhagen

Abstract

Current approaches to chemical map generation from hyperspectral images are based on models such as partial least squares (PLS) regression, generating pixel-wise predictions that do not consider spatial context and suffer from a high degree of noise. This study proposes an end-to-end deep learning approach using a modified version of U-Net and a custom loss function to directly obtain chemical maps from hyperspectral images, skipping all intermediate steps required for traditional pixel-wise analysis. We compare the U-Net with the traditional PLS regression on a real dataset of pork belly samples with associated mean fat reference values. The U-Net obtains a test set root mean squared error of between 9% and 13% lower than that of PLS regression on the task of mean fat prediction. At the same time, U-Net generates fine detail chemical maps where 99.91% of the variance is spatially correlated. Conversely, only 2.53% of the variance in the PLS-generated chemical maps is spatially correlated, indicating that each pixel-wise prediction is largely independent of neighboring pixels. Additionally, while the PLS-generated chemical maps contain predictions far beyond the physically possible range of 0-100%, U-Net learns to stay inside this range. Thus, the findings of this study indicate that U-Net is superior to PLS for chemical map generation.

Keywords: Chemical Imaging, U-Net, Spectroscopy, Chemometrics, Near-Infrared

1. Introduction

Hyperspectral imaging (HSI) is an advanced analytical technique that enables the acquisition of spatially distributed spectral information from material surfaces (Mäkelä et al. 33). Unlike traditional spectroscopy, which provides only bulk spectral data, HSI simultaneously captures spectral and spatial information, offering a more comprehensive analysis of heterogeneous samples.

HSI is widely applied across various scientific disciplines and implemented in multiple spectral modalities, including near-infrared (NIR) (Albano-Gaglio et al. 3), Raman (Tao et al. 47), and infrared (IR) (Boseley et al. 14) spectroscopy. The fundamental distinction between HSI and conventional spectroscopy lies in its imaging capability, where each pixel of an image contains a complete spectral profile. This enables the generation of chemical maps representing the spatial distribution of specific chemical components within a sample. However, due to the high-dimensional nature of HSI data, advanced data modeling techniques are required to extract meaningful and interpretable information. The ability to visualize compositional variations in an image format makes HSI a powerful tool in diverse applications, including material science, pharmaceuticals, food quality assessment, and biomedical imaging.

Chemometric modeling plays a crucial role in HSI data processing (Amigo et al. 5, Mäkelä et al. 33), particularly in the generation of chemical maps. Traditionally, chemical maps are generated using supervised or unsupervised methods, where each pixel is analyzed independently. For instance, supervised models are typically trained using mean spectra and subsequently applied to predict chemical properties for individual pixels (Amigo et al. 5). In the

case of supervised modeling, it is essential to note that the minimum sample size required for chemical analysis is often larger than the pixel size in hyperspectral images. Consequently, acquiring pixel-wise reference values for chemical maps is generally infeasible. Instead, reference values are obtained as averaged measurements at a much lower resolution, such as over an entire sample or larger subsets thereof.

Because reference values represent sample-wise means, it is common practice to train models to predict these mean values. A straightforward approach involves averaging the NIR hyperspectral image pixels to generate a mean spectrum, which is then paired with its corresponding mean reference value. These data pairs are typically used to train a Partial Least Squares (PLS) regression model (Wold 50, Wold et al. 51), which is subsequently applied to generate pixel-wise predictions, producing chemical maps (Thumm et al. 48, ElMasry et al. 17, 16, Albano-Gaglio et al. 2). However, this approach has several limitations. Traditional PLS-based chemical maps treat each pixel independently, resulting in predictions with a low degree of spatial correlation. Pixel-wise predictions may fall outside physically meaningful ranges (0-100 %), affecting interpretability. The lack of spatial structure limits the ability to assess variations within samples, thereby constraining downstream optimization.

The significance of spatial and contextual information in HSI data modeling is now widely recognized in the chemometric domain (Jamme and Duponchel 29, Xu and Gowen 52). Recent studies have demonstrated the benefits of integrating spatial information alongside spectral data to enhance model accuracy (Gorretta et al. 24, Gaci et al. 23, Mishra et al. 36). These approaches offer improved pixel-level property prediction; however, they do not fundamentally enhance spatial coherence because the chemical maps are still generated in a pixel-wise manner.

This study aims to bridge the gap between spectral-based models, such as PLS, and the inherently spatio-spectral task of chemical map generation. To this end, a novel method for chemical map generation is proposed. A modification of U-Net (Ronneberger et al. 40), a convolutional neural network (CNN) originally designed for semantic segmentation in medical imaging, is proposed for chemical map generation. In semantic segmentation, each pixel in an input image is to be classified into discrete categories. The equivalent task for chemical map generation involves continuous regression rather than classification. There are several advantages of using U-Net for chemical mapping. Unlike PLS, which processes pixels independently, U-Net jointly considers spatial and spectral information within hyperspectral images. It performs exceptionally well with limited labeled samples (Siddique et al. 42, Luo et al. 31, Azad et al. 9, Archana and Jeevaraj 8). Like PLS models trained on mean spectra, the proposed U-Net requires only mean reference values per sample. A custom multi-objective loss function trains U-Net to generate pixel-wise predictions strictly inside the physically valid range (0-100%) and leverages spatial structure to generate smoothly varying chemical maps, neither of which PLS accomplishes.

This study proposes a new deep learning approach (a modified U-Net) to process HSI data. The study utilizes the NIR-HSI dataset of pork bellies and associated mean fat content reference values from Albano-Gaglio et al. [2] to compare PLS-based and U-Net-based chemical map generation approaches. The remainder of this article is structured as follows: Section 2 presents details on the dataset and modeling approaches. Section 3 and Section 4 analyze and discuss the experimental results. Section 5 provides conclusions on the study and future perspectives on using deep learning for chemical mapping.

2. Materials and methods

2.1. Dataset

This study used the VIS-NIR reflectance HSI dataset of pork bellies introduced by Albano-Gaglio et al. [2]. Bellies were selected from different production systems, feedings, genetics, and sexes to ensure enough variability in their quality parameters to calibrate non-destructive devices and methods, such as HSI systems, to characterize pork bellies. Due to the selection made by Albano-Gaglio et al. [2], the pork bellies belong to pigs spanning five different fat classes, F1-F5, with F1-F3 from common commercial pigs, F4 from Duroc pigs, and F5 from Iberian cross-bred pigs. The dataset consists of hyperspectral images of 182 pork bellies. Fig. 1 illustrates an entire pork belly. To simplify sampling, each pork belly was divided into five vertical sections, 1-5, and three horizontal sections, A-C. Then, Albano-Gaglio et al. [2] physically cut each pork belly to yield five slices, corresponding to the five vertical sections.

The division of 182 pork bellies into five slices each would give 910 slices in total, but as nine slices are missing from the dataset, we end up with 901. The 901 slices represent all 182 pork bellies. Albano-Gaglio et al. [2] imaged

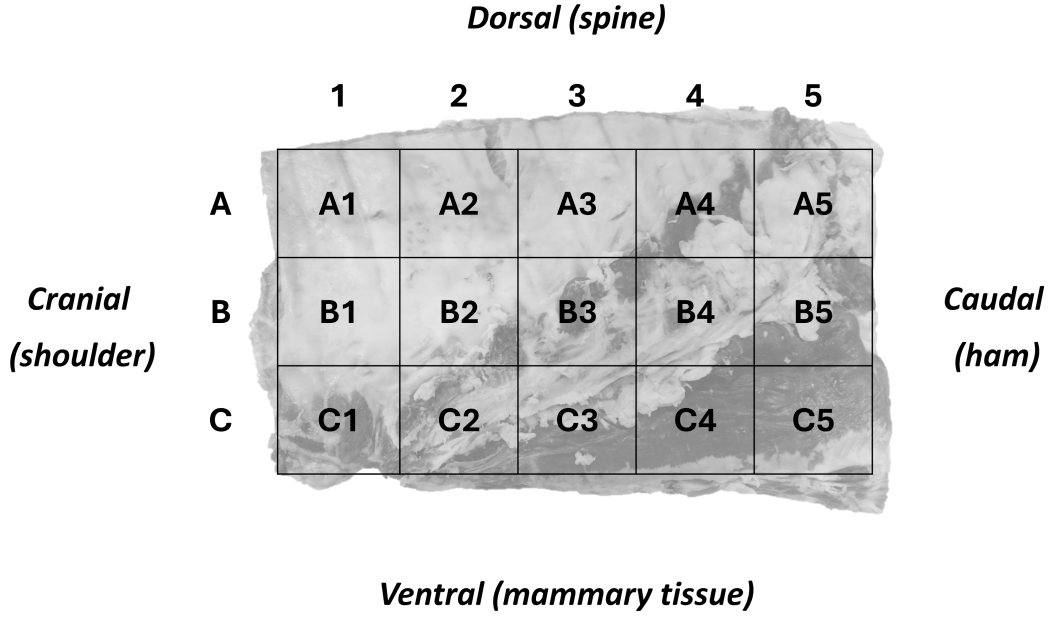


Figure 1: An entire pork belly. Illustration kindly donated by Albano-Gaglio et al. [2]. Each belly was physically cut into five pieces, corresponding to sections 1-5, before being imaged.

each slice using a VIS-NIR platform with 300 uniformly distributed wavelength channels from 386.63 nm to 1015.78 nm. All the images have a width of 900 pixels, while their height varies between 1900 and 2477 pixels. Most images have a height of 2000 pixels, which leads to a mean height of 1982 pixels. Only the latter 124 wavelength channels from 747.92 nm upwards were used for modeling. Using this range serves two purposes. First, it reduces the number of bands, which reduces the computation cost for model training. Second, it allows the models to focus only on the chemical information to predict fat content rather than biasing them with the color information.

Albano-Gaglio et al. [2] associate each image with a segmentation mask computed using PLS discriminant analysis (PLS-DA, Barker and Rayens 10) to distinguish between the slice and the background. The segmentation mask also classifies muscle tissue and other stains as background, retaining only the subcutaneous fat tissue. In this study, the masks were slightly modified by performing a morphological erosion with a disk-shaped structuring element with a radius of one pixel. This erosion ensures that all pixels in the 4-neighborhood of every retained foreground pixel in the hyperspectral image are known to contain only foreground initially. This, in turn, means that the spatial derivatives in both directions of every retained foreground pixel are well-defined, a property required when computing these spatial derivatives in Section 3 to quantify the spatial distribution of predicted chemical maps. An example image and eroded mask is shown in Fig. 2. In practice, due to the small size of the structuring element, the eroded mask is practically identical to the original mask. From now on, unless explicitly stated otherwise, when referring to the image’s mask, the eroded version is meant. With each pork belly, a mean spectrum was computed as the mean of all spectra contained within the masks of the images of vertical slices belonging to that belly. As a chemical parameter, this study focuses on the fat content of the whole minced belly, the reference value of which Albano-Gaglio et al. [2] obtained with wet chemistry.

2.1.1. Dataset split

Albano-Gaglio et al. [2] split the dataset into two subsets, cross-validation (CV) and test, with the DUPLEX algorithm (Snee 44). The DUPLEX algorithm generates two datasets in an iterative process based on Mahalanobis distance (Mahalanobis 32) between the mean spectra of the pork bellies.

In the dataset split provided by Albano-Gaglio et al. [2], the CV set consists of 122 pork bellies, while the test set consists of 60 bellies. Albano-Gaglio et al. [2] randomly split the CV set into five folds (subsets). To make the

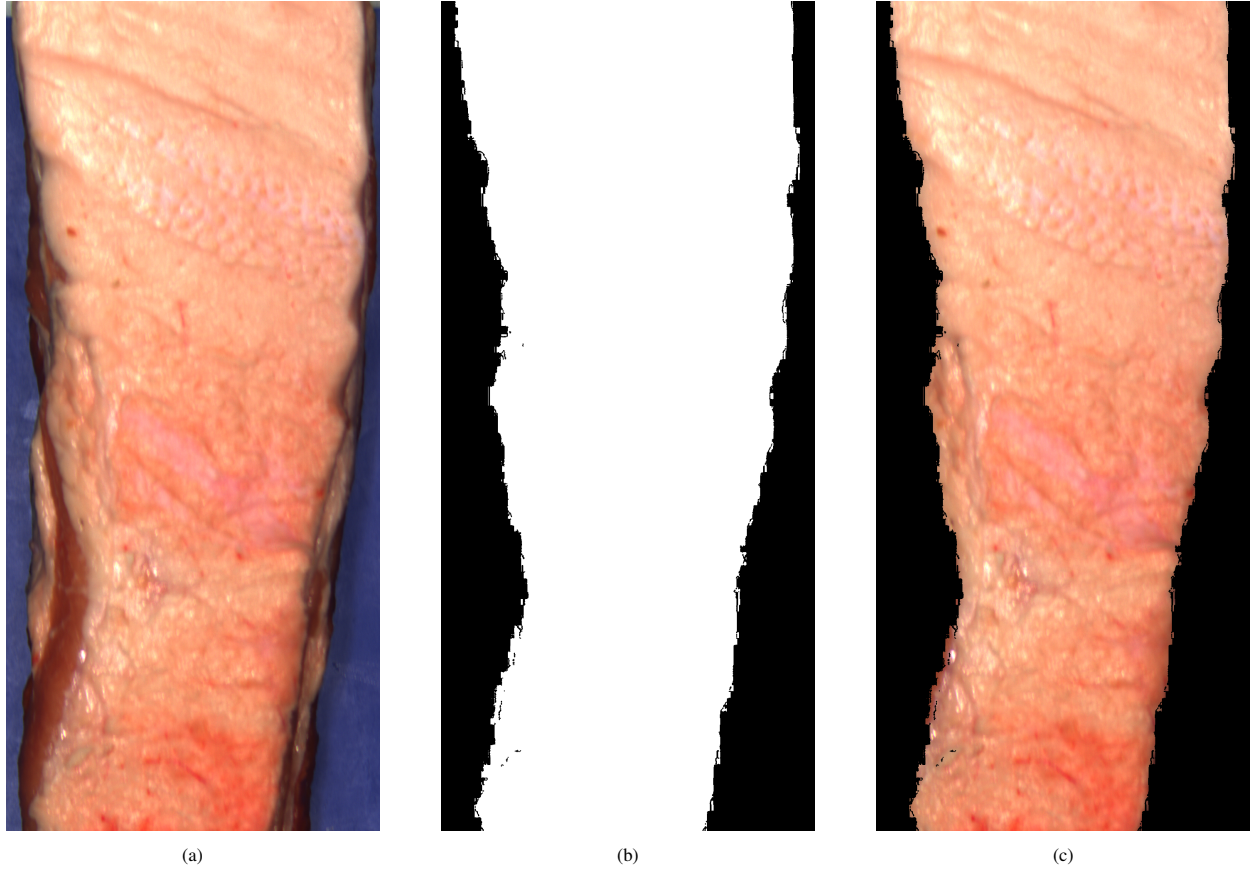


Figure 2: RGB illustrations of an example image and associated eroded mask from the dataset. This image corresponds to one of the five vertical sections of a pork belly, such as illustrated in Fig. 1. (a) Image of pork belly slice. (b) Associated segmentation mask eroded by a disk-shaped structuring element with radius one. (c) Masked image of pork belly slice.

results of this study comparable to those of Albano-Gaglio et al. [2], the same splitting of the CV set into folds from Albano-Gaglio et al. [2] was requested. However, as the randomly generated splits were not stored, this request could not be fulfilled. Instead, a modification of the DUPLEX algorithm to generate not two but six subsets of the dataset was explored. The original test set was kept as one of the subsets to enable a direct comparison between the test set results and those of Albano-Gaglio et al. [2]. The modified DUPLEX algorithm divides the CV set into five folds. Before feeding the mean spectra to DUPLEX, a dimensionality reduction using principal component analysis (PCA, Pearson 38, Hotelling 27) was performed using six components, which was the minimum required to retain at least 99% of the variance of the CV subset. In Table 1, a description of the number of pork bellies and how associated slices are distributed between the dataset splits yielded by DUPLEX is presented. In Fig. 3, distributions of reference fat percentages of these dataset splits are presented. All fat classes, F1-F5, are represented in the CV and test sets.

	Split					Test	Total
	1	2	3	4	5		
No. bellies	25	25	24	24	24	60	182
No. slices	123	125	119	118	118	298	901

Table 1: Number of pork bellies and slices of those bellies for each dataset split. We have reference values for the bellies and images of the slices.

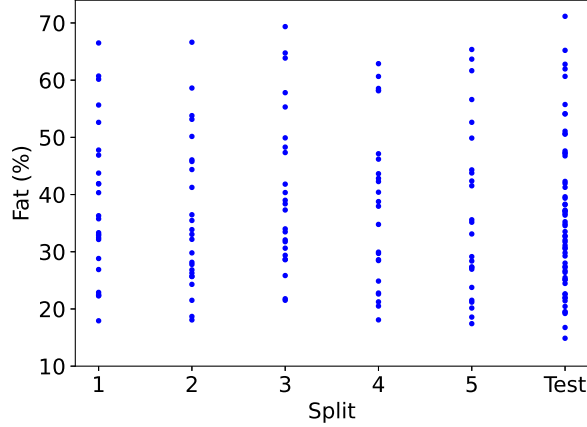


Figure 3: Distribution of reference fat values in the dataset splits.

2.2. Modeling

2.2.1. PLS

A PLS model using the pork bellies’ mean absorbance spectra was calibrated as a baseline. For a pork belly, the mean absorbance spectrum was calculated by first taking the negative logarithm of each pixel in each hyperspectral image of each pork belly slice and then computing the average across all pixels inside the slices’ masks. Subsequently, the spectra were preprocessed by application of Standard Normal Variate (SNV) transformation (Barnes et al. 12) followed by convolution with a Savitzky-Golay filter (Savitzky and Golay 41, Steinier et al. 46) with a window length of seven, a polynomial order of two, and a derivative order of two. These preprocessing steps combat Rayleigh scattering and reveal the chemical information in the spectra (Rinnan et al. 39, Sørensen et al. 45). Finally, a column-wise mean centering of the spectra and the reference fat values was performed using statistics computed only on the training sets.

A 5-fold cross-validation was used to determine the optimal number of components. For each CV fold, the root mean squared error (RMSE) on the validation set was computed for all possible numbers of components. Subsequently, the average over the five folds was calculated to get a mean RMSE for each number of components. This evaluation yielded nine components minimizing the RMSE, so a PLS on the entire CV set using nine components was recalibrated. Thus, the final PLS model was calibrated using all the available data points in the CV set. The PLS calibrations were carried out using the fast (Alin 4) and numerically stable (Andersson 6) Improved Kernel PLS Algorithm 1 (Dayal and MacGregor 15) with the software implemented by Engstrøm et al. [20]. The CV was done with the same software using the fast algorithm by Engstrøm and Jensen [22].

In addition to predicting from the mean spectra for each pork belly, this same PLS model was used to generate chemical maps by pixel-wise prediction as traditionally done in HSI processing (Amigo et al. 5, Mäkelä et al. 33). The same preprocessing was applied to each pixel as for the mean spectra described above.

2.2.2. U-Net

U-Net was modified in two ways: The first modification was replacing the upsampling by transposed convolution, as proposed by Ronneberger et al. [40], with upsampling by bilinear interpolation. Transposed convolutions cause square-like artifacts in the output (Odena et al. 37). Bilinear interpolation avoids these artifacts while reducing the number of parameters in the model, thereby decreasing training time.

The second modification was adding an initial 3D convolution layer with one filter containing one kernel of size (depth, height, width) = (7, 2, 2). Here, the three-dimensional filter was applied across the three spatio-spectral dimensions. An initial 3D convolution layer before a 2D neural network allows learning spectral smoothing and derivatives. Choosing a depth of seven enables the model to learn precisely the Savitzky-Golay filter used to preprocess spectra for PLS. Adding this layer is similar to the approach taken by Engstrøm et al. [21] and Engstrøm et al. [19],

where it was shown to be beneficial for predicting chemical parameters in NIR-HSI images of grain without being detrimental to non-chemical parameters.

Appendix A illustrates the model architecture and the sizes of intermediate tensors. Apart from these modifications, the new design followed the original U-Net implementation. As in the original U-Net implementation, no padding to any convolutions was applied, and only the so-called valid parts of the convolutions were evaluated. The valid parts of convolution are those pixels where the kernel completely overlaps with the input image or intermediate feature map. All convolution layers had their bias initialized to zero and their scale parameters initialized by sampling from a Kaiming He Normal Distribution (He et al. 25). While Ronneberger et al. [40] wrote the original U-Net implementation in Caffe (Vedaldi et al. 49), this study recreated it in PyTorch (Ansel et al. 7) to accommodate the customization needs better and to ensure easy compatibility and integration with other modern works.

For the initial 3D convolution layer, a stride of 2 pixels was applied for the spatial dimensions while keeping a stride of 1 channel for the depth dimension. The stride effectively halves the input size from a height and width of 2360×1272 to 1180×636 and serves to significantly reduce training time and reduce the size of the output. Feeding this input to the U-Net gives an output of size 996×452 . These dimensions align so that U-Net does not drop any rows or columns of its intermediate feature maps in any layers, even with max-pooling layers with a stride of two pixels and non-padded convolution layers. Using these fixed dimensions requires data augmentation as the original dataset contains images with a fixed width of 900 pixels and a varying height.

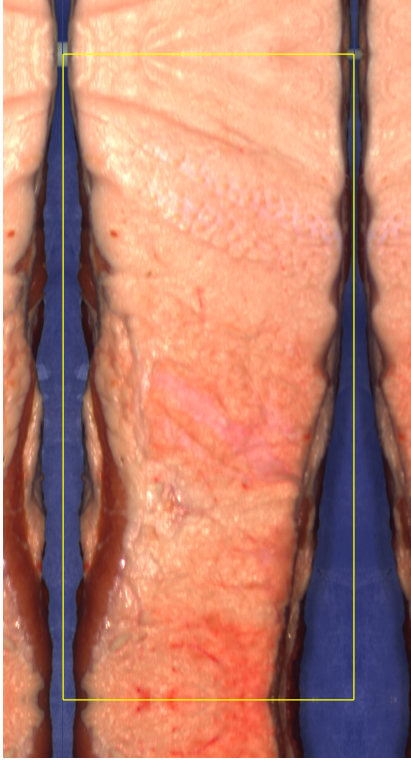
Similarly, the size of the spectral dimension was reduced by a factor of two from 124 channels to 62 channels. However, instead of strided convolutions, the spectral dimension was reduced by averaging neighboring wavelength channels. This reduction simultaneously reduces training time and alleviates the so-called curse of dimensionality, which is prevalent for deep learning on hyperspectral images (Signoroni et al. 43).

In the original design, U-Net produces an output smaller than its input. This shrinking implies that padding must be applied to retain the size of the original input. This study used a two-stage scheme for padding the images before feeding them to the U-Net model. The first stage applies background padding to the height and width until reaching 1992 pixels in height and 904 pixels in width. The background spectrum used for padding is the average spectrum of the image’s left-most and right-most columns, as these contain almost exclusively background for all images. The justification for stage one is that, with this study’s chosen dimensions, the model will produce an output corresponding to the central 1992 pixels in height and 904 pixels in width. Thus, if an image was smaller than this size, it was padded with a background spectrum. Due to using only the valid parts of the convolution, the input image must be further padded to get predictions near the border. In this second stage, an approach similar to the original U-Net design was taken, and, as such, mirror-padding was applied until reaching 2360 pixels in height and 1272 pixels in width. Illustrations of the padding scheme are shown in Fig. 4.

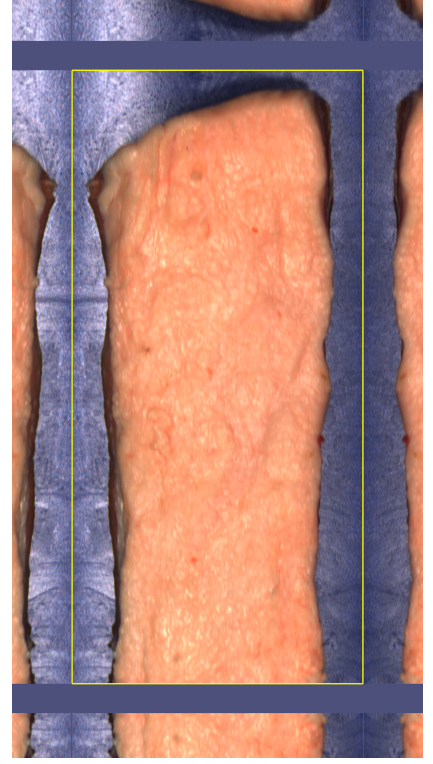
The mask must be modified according to the padding scheme to match the contents of the central 1992×904 pixels of the input, padded hyperspectral image. The original non-eroded mask was used as the starting point for these modifications. Any location padded with the background spectrum was accordingly set to background in the mask. Then, the central 1992×904 pixels of the mask were cropped as these correspond to the pixels for which U-Net makes a prediction. Due to the initial strided 3D convolution, the spatial dimensions of the input to U-Net are halved. Accordingly, the spatial dimensions of the mask must be halved to 996×452 , matching the output of the U-Net. The halving of the mask is computed with bilinear interpolation, after which every pixel in the mask is rounded to zero or one. Finally, the downsampled mask is eroded with a disk-shaped structuring element with a radius of one pixel to allow computation of spatial derivatives in the predictions as explained in Section 2.1. In the context of U-Net, a reference to an image’s mask means this downsampled, eroded version.

During training, images were randomly flipped horizontally and vertically, each with a probability of 0.5. Such flipping was not applied during evaluation.

The task to be solved must be defined to optimize the modified U-Net model. Recall that the data contains images of slices of pork belly and reference values for the mean fat percentage of the whole bellies. This reference value was used for each slice for optimization, an approach similar to Engström et al. [21] and Engström et al. [19]. Therefore, given an input image of a slice, the model aims to predict a fat distribution with a mean equal to the fat reference. Only the pixels inside the mask were considered for a given predicted chemical map. Theoretically, a model can be trained to predict a fat value of zero outside the mask. However, as seen in Fig. 2, the edges of the meat are often classified as background by the mask. Thus, in practice, attempting to learn to predict zero outside the mask with this dataset will likely be detrimental to the overall performance of the model.



(a)



(b)

Figure 4: Example applications of the two-stage padding scheme. The yellow boxes indicate the border between the original images and the padding. (a) Padding of an image with an original height of at least 1992 pixels. Here, two columns of background padding are applied to the left and right, reaching a width of 904 pixels. Then, mirror-padding is applied until a height of 2360 and a width of 1272 pixels. This is the padded version of the image shown in Fig. 2a. (b) Padding of an image with an original height of less than 1992 pixels. Here, background padding is applied until reaching a height of 1992 and a width of 904 pixels, after which mirror-padding is applied until reaching a height of 2360 and a width of 1272 pixels.

Given a batch of hyperspectral images, let $\widehat{\mathbf{Y}} \in \mathbb{R}^{B \times H \times W}$ be the U-Net’s predicted chemical maps with batch size B , height H pixels, and width W pixels. For a given batch, consider also the associated reference fat values $\mathbf{y} \in \mathbb{R}^B$ and segmentation masks $\mathbf{M} \in \{0, 1\}^{B \times H \times W}$. In $\mathbf{M}$, a value of 0 indicates background and a value of 1 indicates foreground in the corresponding pixel of $\widehat{\mathbf{Y}}$. Such pixels are considered outside, respectively, inside the mask. For a batch of images, consider the modified U-Net’s masked prediction, $\widehat{\mathbf{Y}}^{\mathbf{M}}$, defined using element-wise multiplication ($\odot$)

$$\widehat{\mathbf{Y}}^{\mathbf{M}} = \widehat{\mathbf{Y}} \odot \mathbf{M}. \quad (1)$$

Then, the U-Net’s mean fat predictions, $\widehat{\mathbf{y}} \in \mathbb{R}^B$, are defined as its mean predictions inside the masks. Considering $\widehat{\mathbf{y}}$ at batch index $b \in \{1, \dots, B\}$, the mean fat prediction is given by

$$\widehat{\mathbf{y}}_b = \frac{\sum_{h=1}^H \sum_{w=1}^W \widehat{\mathbf{Y}}_{b,h,w}^{\mathbf{M}}}{\sum_{h=1}^H \sum_{w=1}^W \mathbf{M}_{b,h,w}}. \quad (2)$$

Now, the modified U-Net can be optimized by punishing it for predicting an average value that deviates from the reference value. To this end, the mean squared error between the reference value y_b and $\widehat{\mathbf{y}}_b$ was used,

$$MSE(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{1}{B} \sum_{b=1}^B (y_b - \widehat{\mathbf{y}}_b)^2. \quad (3)$$

The true fat contents in each pixel must lie in the range 0 – 100%. However, only optimizing against the mean fat prediction provides no guarantee for a proper range of pixel-wise predictions. Thus, to aid the optimization process, the out-of-bounds loss (OOBL) was defined as the sum of squared deviations from the 0 – 100 range in $\widehat{\mathbf{Y}}^{\mathbf{M}}$

$$OOBL(\widehat{\mathbf{Y}}^{\mathbf{M}}) = \frac{1}{B} \sum_{b=1}^B \sum_{h=1}^H \sum_{w=1}^W \max(-\widehat{\mathbf{Y}}_{b,h,w}^{\mathbf{M}}, 0)^2 + \max(\widehat{\mathbf{Y}}_{b,h,w}^{\mathbf{M}} - 100, 0)^2. \quad (4)$$

Here, the first term in the sum considers pixel-wise predictions below 0, and the latter considers those above 100. No normalization by the sizes of the masks was applied because an out-of-bounds prediction is considered equally nonsensical, regardless of the size of the pork belly slice on which the prediction lies.

In addition to having the correct mean fat value and a pixel-wise range of 0 – 100%, a chemical map is expected to have spatial dependencies among the pixels. In particular, it was assumed that a pixel and its neighbors should have predictions that are not too far apart, which means that the pixel values of the predicted chemical map should be smoothly varying. Therefore, a loss term was added to punish the model for deviations from this assumption. It computes a measure of the predicted chemical map’s total smoothness and is referred to as the smoothness loss (SL). The sum of the squared spatial gradient magnitude of the predicted chemical map was used as a measure of smoothness, and SL is defined as the average over the batch,

$$SL(\widehat{\mathbf{Y}}, \mathbf{M}) = \frac{1}{B} \sum_{b=1}^B \frac{\sum_{h=1}^{H-1} \sum_{w=1}^{W-1} \mathbf{M}_{b,h,w} \left((\widehat{\mathbf{Y}}_{b,h,w} - \widehat{\mathbf{Y}}_{b,h+1,w})^2 + (\widehat{\mathbf{Y}}_{b,h,w} - \widehat{\mathbf{Y}}_{b,h,w+1})^2 \right)}{\sum_{h=1}^{H-1} \sum_{w=1}^{W-1} \mathbf{M}_{b,h,w}}. \quad (5)$$

The squared spatial gradient magnitude is approximated by approximating the spatial first-order derivatives with forward differences between horizontal and vertical neighboring pixel pairs in the predicted chemical map, considering only pairs where both pixels are meat. Consider a pixel, $\widehat{\mathbf{Y}}_{b,h,w}$. If it lies inside the mask, then, due to the previous erosion of the mask, the neighbor pixels $\widehat{\mathbf{Y}}_{b,h+1,w}$ and $\widehat{\mathbf{Y}}_{b,h,w+1}$ are all known to contain meat regardless of whether they lie inside the mask or not. This property is necessary and sufficient for the local derivative of the predicted chemical map at $\widehat{\mathbf{Y}}_{b,h,w}$ to be well-defined over the meat because the derivative is then not affected by the background. Enforcing this constraint is sensible, as the quality of the predicted chemical map is only related to the smoothness inside the chemical map and not to the smoothness between the chemical map and the background. SL is normalized by the mask size to get an average smoothness measure for each predicted chemical map.

L2-regularization was also added to the loss function to combat potential overfitting. The modified U-Net’s weights were organized into a vector, and that vector’s Euclidean (L2) distance¹ was computed. Bias parameters were not regularized. Let θ be the modified U-Net’s weights, indexed by p such that θ_p is the p ’th weight. Then, considering a total of P weights, the L2-regularization loss is given by

$$L2(\theta) = \sum_{p=1}^P \theta_p^2. \quad (6)$$

Finally, given θ , $\widehat{\mathbf{Y}}$, $\mathbf{M}$, and $\mathbf{y}$, Eq. (1) was used to compute $\widehat{\mathbf{Y}}^{\mathbf{M}}$ and Eq. (2) was used to compute $\widehat{\mathbf{y}}$, allowing to express the total loss (TL) as

$$\begin{aligned} TL(\theta, \widehat{\mathbf{Y}}, \mathbf{M}, \mathbf{y}, \widehat{\mathbf{Y}}^{\mathbf{M}}, \widehat{\mathbf{y}}) &= \lambda_{MSE} MSE(\mathbf{y}, \widehat{\mathbf{y}}) \\ &+ \lambda_{OOBL} OOBL(\widehat{\mathbf{Y}}^{\mathbf{M}}) \\ &+ \lambda_{SL} SL(\widehat{\mathbf{Y}}, \mathbf{M}) \\ &+ \lambda_{L2} L2(\theta). \end{aligned} \quad (7)$$

¹The square root was omitted for faster computation. As the square root is a monotonically increasing function, omitting it does not change the model’s optimum.

Here, the lambdas are scalar multipliers used to weigh the four loss terms relative to each other. This study used $\lambda_{MSE} = 1$, $\lambda_{OOBL} = 10^{-3}$, $\lambda_{SL} = 20$, and $\lambda_{L2} = 10^{-3}$.

To train the modified U-Net model using the loss function in Eq. (7), the Adam optimizer (Kingma and Ba 30) was used with an initial learning rate of 10^{-3} , and hyperparameters $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The best model was chosen based on the validation MSE computed by Eq. (3) using the entire validation set as a single batch² The model selection is based on MSE instead of TL, as this study’s primary interest is a solution minimizing MSE while using the other loss terms to guide the properties of such a solution (i.e., smoothly varying and inside the 0 – 100 range). However, as evident in Section 3, MSE dominates the other terms in TL in practice. Therefore, using TL instead of MSE for model selection would likely yield similar results.

For a given fold in the 5-fold cross-validation, during training, the current best weights (the best weights seen so far) and the associated optimizer state at that point in time were tracked. After each epoch, the model was evaluated on the validation set. The best weights were defined as those having yielded the minimum validation MSE so far.

The learning rate was left unchanged for the first 30 epochs, called the burn-in period. After the burn-in epochs, counters for early stopping and learning rate reduction were initiated. After 10 epochs without reaching a new set of best weights, the learning rate was reduced by a factor of 10 but not below 10^{-7} . Immediately before lowering the learning rate, the model was restored to its current best weights, and the optimizer’s state was restored to its state at that time. If, after 30 epochs, no new set of best weights was achieved, training was halted, and the current best weights were restored and returned. If the early stopping was not triggered, the training was halted after 250 epochs. In practice, all training runs were halted by the early stopping criterion, not the 250 epoch limit.

As for PLS, the aim was to have a U-Net trained on all the available data in the CV set. This could be achieved for PLS by averaging the hyperparameter (number of components) over the CV folds and refit on the entire CV set using this hyperparameter setting. For U-Net, however, averaging hyperparameters such as early stopping epochs and epochs for learning rate reduction provide no guarantee that a U-Net trained with these hyperparameters will have its weights end up in a meaningful place. Instead, an ensemble was constructed from all five U-Nets to have a U-Net trained on the entire CV set. Then, the ensemble’s prediction is simply the uniform average of its constituents’ predictions. This ensemble U-Net will simultaneously have been trained and validated on all images in the CV set. Constructing such an ensemble has previously yielded good results for predicting protein content in NIR-HSI images of grain (Engström et al. 19).

3. Results

3.1. U-Net Convergence

Fig. 5 illustrates the total and individual loss terms for CV split 1 throughout the training epochs. It highlights the epochs for which the burn-in period completes (epoch 30), the current best weights (as determined by minimization of validation MSE), and the epochs for learning rate reduction. Graphs for all five CV splits are shown in Appendix B in Fig. B.10. As the training set is continuously visited in batches, it is common to report a mean value of the evaluated metrics computed over the batches of the training set. Conversely, the same metrics are typically reported for the validation set as computed with the final weights of an epoch. However, the model weights are updated after each batch; therefore, such a comparison between the training and validation metrics is based on different weights of the model. Instead, to enable a direct comparison, an evaluation of the final model after each epoch on the entire training and validation set was performed, thus allowing for a direct comparison between training and validation set metrics.

Inspecting Fig. 5, it was clear that there was a downward trend for all loss terms, although it was subject to much noise. The OOBL term quickly converged to zero, indicating that the U-Net had correctly learned to constrain predictions to the 0 – 100% range. Additionally, the MSE term was, by far, the dominant term in the loss function. Indeed, this indicated that the optimization process did not need to compromise the precision of fat content predictions to generate smooth chemical maps that did not contain out-of-bounds predictions. The remaining splits, shown in Fig. B.10, exhibited a similar nature of downward albeit noisy trends for all loss terms.

²The whole validation set could not be stored on the GPU. Therefore, when encountering a sample, the squared errors were accumulated, returning the mean when all samples had been seen. This computation is mathematically equivalent to using the entire validation set as a batch.

The graphs' noisy nature was attributed to the relatively small dataset size compared to the rather large U-Net. Additionally, a batch size of one was used due to memory constraints, which may also lead to noisy gradients.

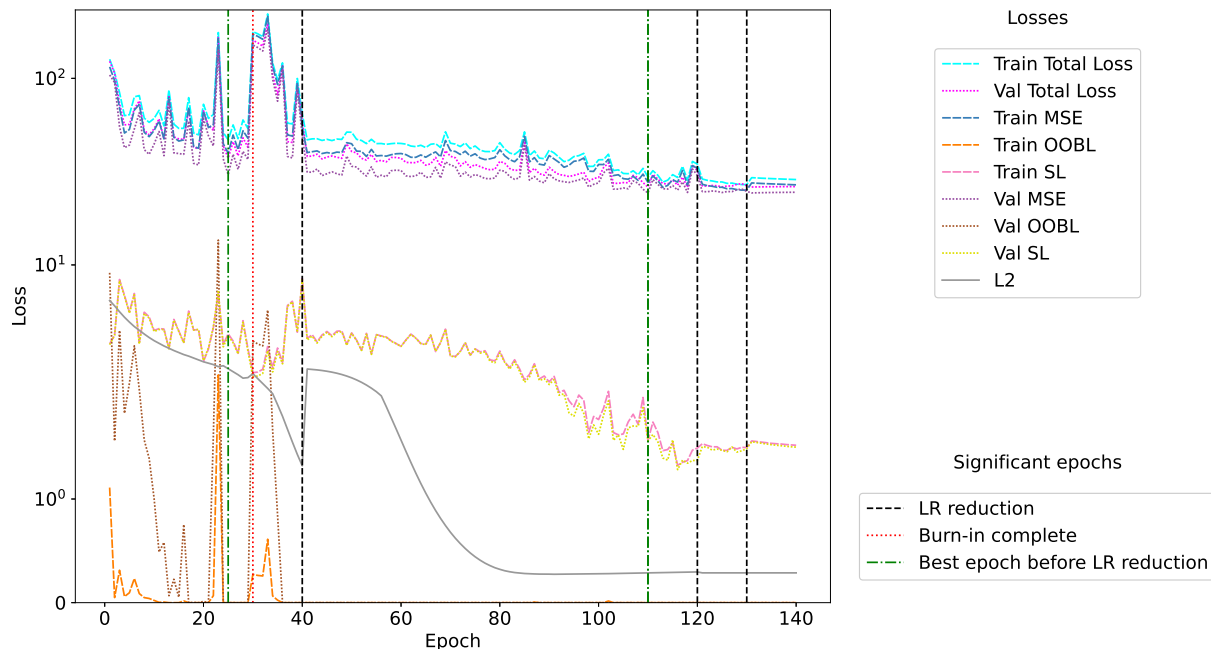


Figure 5: Evolution of total loss and individual loss terms for CV split 1 (after multiplication with their respective weights).

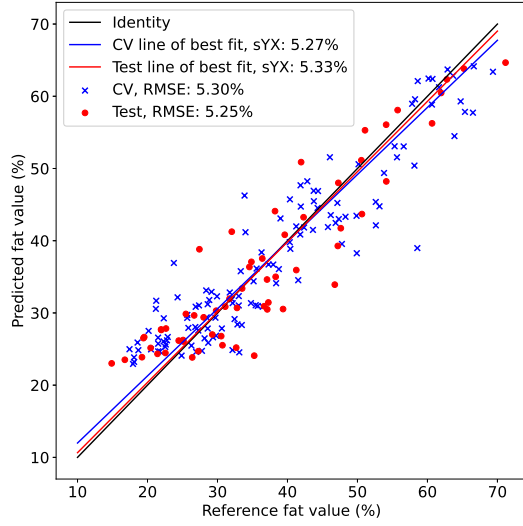
3.2. Pork belly mean fat predictions

The mean fat content was the primary quality parameter for the predictions. The reference values were given for whole pork bellies, not individual slices. Thus, to get a mean pork belly prediction for U-Net, the pixel-wise predictions for all pixels inside the masks of all slices belonging to a given pork belly were averaged. Conversely, the PLS model was trained on mean belly spectra, and thus, its predictions were already on the complete belly level, and no averaging was required. Alternatively, it was possible with PLS to take an approach similar to the U-Net by averaging the pixel-wise predictions of the chemical maps. This averaging yielded a different result for the PLS predictions. In Fig. 6 shows the belly-wise predictions for U-Net, PLS on mean belly spectra, and PLS averaged over the chemical, and Table 2 summarizes their test set RMSEs.

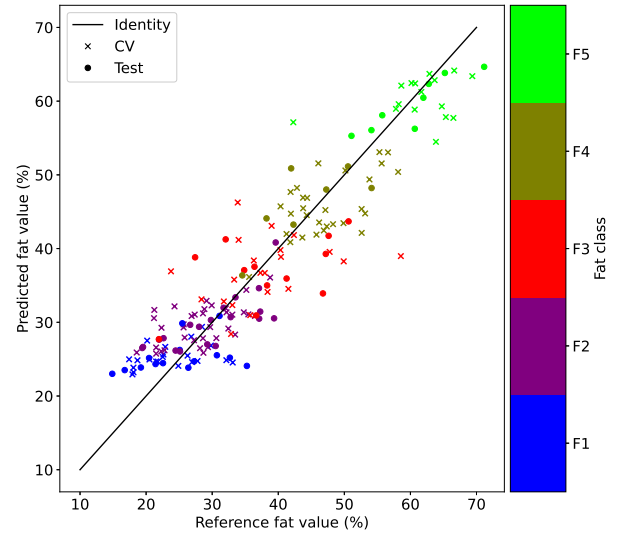
U-Net outperformed PLS on fat classes F2, F4, and F5, while PLS on mean belly spectra showed the best performance on F1, and PLS averaged over chemical maps showed the best performance on F3. Notice also that the U-Net generally had a slightly lower RMSE on the test set than either PLS method. U-Net showed a minimal discrepancy between RMSE on the CV and test sets compared to PLS. In addition to showing RMSE, the line of best fit was computed for each method for both the CV and the test sets. For these lines, s_{YX} was reported, which is the RMSE around the line of best fit but with two fewer degrees of freedom due to the slope and intercept computations. The s_{YX} values are close to the corresponding RMSE values for both U-Net and PLS on mean belly spectra, indicating well-calibrated models. This calibration measure degenerates slightly for PLS when using the average pixel-wise predictions.

3.3. Chemical maps

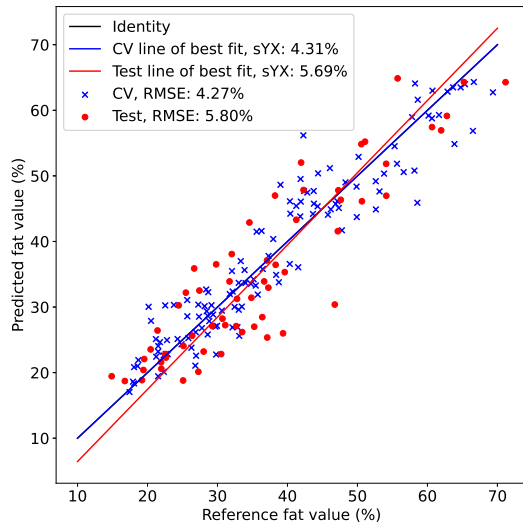
In addition to the mean fat predictions, the chemical maps and distributions of the pixel-wise predictions were of high interest to justify using the NIR-HSI technology. Fig. 7 shows an example of an input image and the predicted chemical maps by U-Net and PLS, respectively. Fig. 8 shows predicted chemical maps for a pork belly from each fat class.



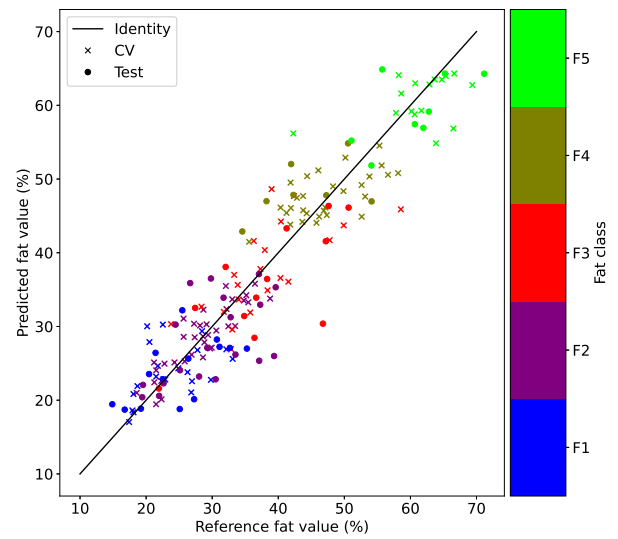
(a)



(b)



(c)



(d)

Figure 6: Pork belly-wise mean fat predictions. For U-Net and pixel-wise PLS, the mean fat prediction for each pork belly was computed as the mean over all pixels of all chemical maps belonging to that pork belly. For PLS on mean spectra, the belly-wise mean prediction was obtained directly. (a) U-Net. (b) U-Net with fat classes highlighted. (c) PLS on mean spectra. (d) PLS on mean spectra with fat classes highlighted.

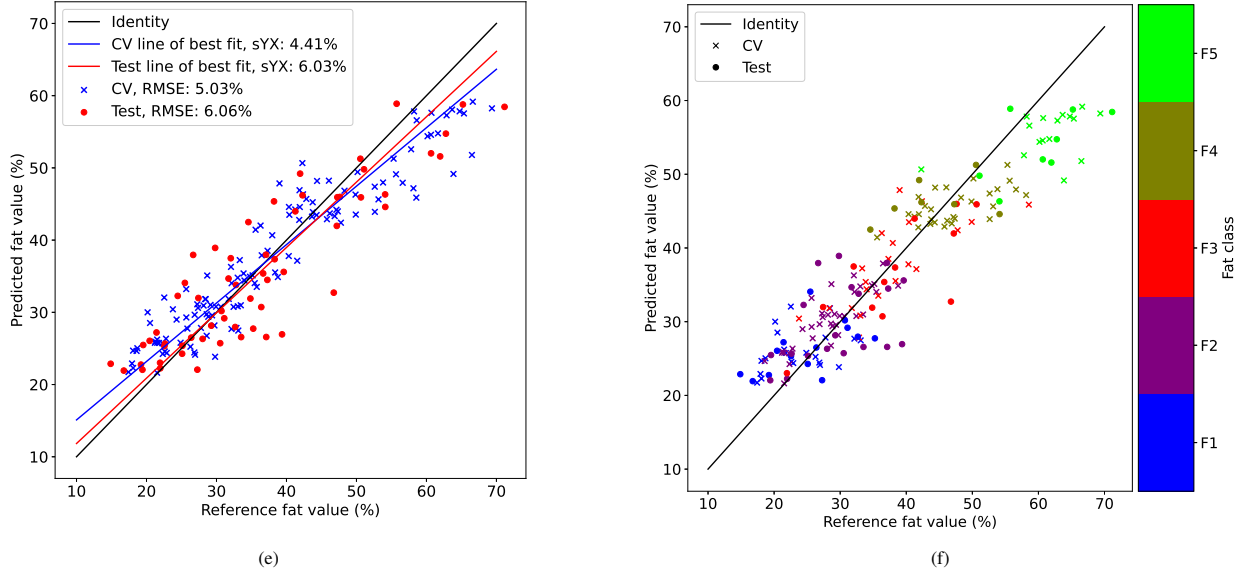


Figure 6: Continued pork belly-wise mean fat predictions. (e) Pixel-wise PLS. (f) Pixel-wise PLS with fat classes highlighted.

Model	Fat class					All
	F1	F2	F3	F4	F5	
U-Net	5.40	4.36	7.29	4.69	3.41	5.25
PLS on mean spectra	4.75	5.97	6.29	7.06	5.03	5.80
Pixel-wise PLS	5.07	6.06	5.41	6.27	8.08	6.06

Table 2: RMSE (in %) of pork belly-wise fat predictions on the test set for U-Net, PLS on pork belly-wise mean spectra, and pixel-wise PLS.

Interestingly, PLS makes predictions that most resemble spatially uncorrelated noise and massive deviations from the 0 – 100% range. However, when averaging the otherwise noisy predictions, PLS achieves a mean fat prediction not too far from the reference value as seen in Fig. 6e. Every PLS chemical map prediction in the dataset, CV and test sets alike, exhibits this phenomenon of noisy chemical maps with somewhat accurate mean values. Conversely, U-Net generated chemical maps that were spatially structured while constraining itself to the 0 – 100% range and simultaneously achieving a mean prediction close to the reference value. Furthermore, while PLS always makes predictions for which the histogram resembles a normal distribution (such as shown in Fig. 7e), U-Net predicts a different distribution for each pork belly slice.

Consider the U-Net generated chemical maps for the pork belly in the second row in Fig. 8a. A systematic line with uniform fat prediction appears at the very top of these chemical maps. This line is a border effect that arises if, and only if, the pork belly slice is close to the top or bottom of the original image and subsequently padded with the background spectrum. For example, this border effect would arise at the bottom but not at the top of the U-Net generated chemical map for the pork belly slice in Fig. 4b. The border effect does not arise on images that are not padded at the top or bottom with the background spectrum. For example, the generated chemical map for the pork belly slice in Fig. 4a would not have the border effect.

3.3.1. Spatial distribution analysis

Here, the spatial distribution of pixel-wise predictions is analyzed. First, a qualitative analysis is presented based on the correlation between fat content and firmness of the bellies, measured by the finger pressure method (Maw et al. 35). Then, a statistical analysis of the prediction distribution was conducted.

For the data used in this study, Albano-Gaglio et al. [3] have measured firmness in all 15 regions of the pork bellies

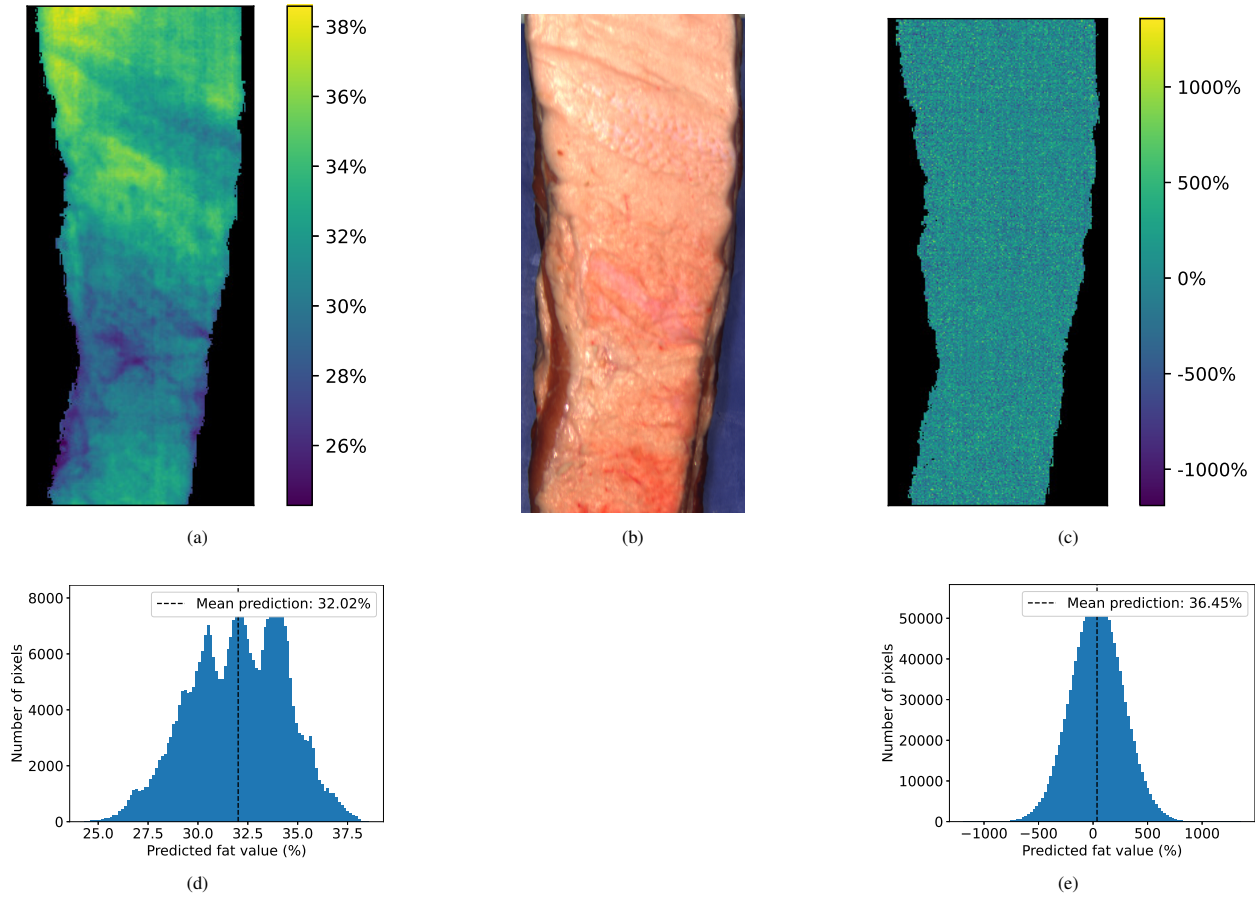


Figure 7: Example chemical map predictions on a slice of pork belly from the test set. (a) Masked U-Net prediction. (b) Input image with a reference mean fat content of 32.79%. For U-Net, the input was the padded version shown in Fig. 4a. (c) Masked PLS prediction. (d) Histogram for the pixel-wise U-Net predictions. (e) Histogram for the pixel-wise PLS predictions.

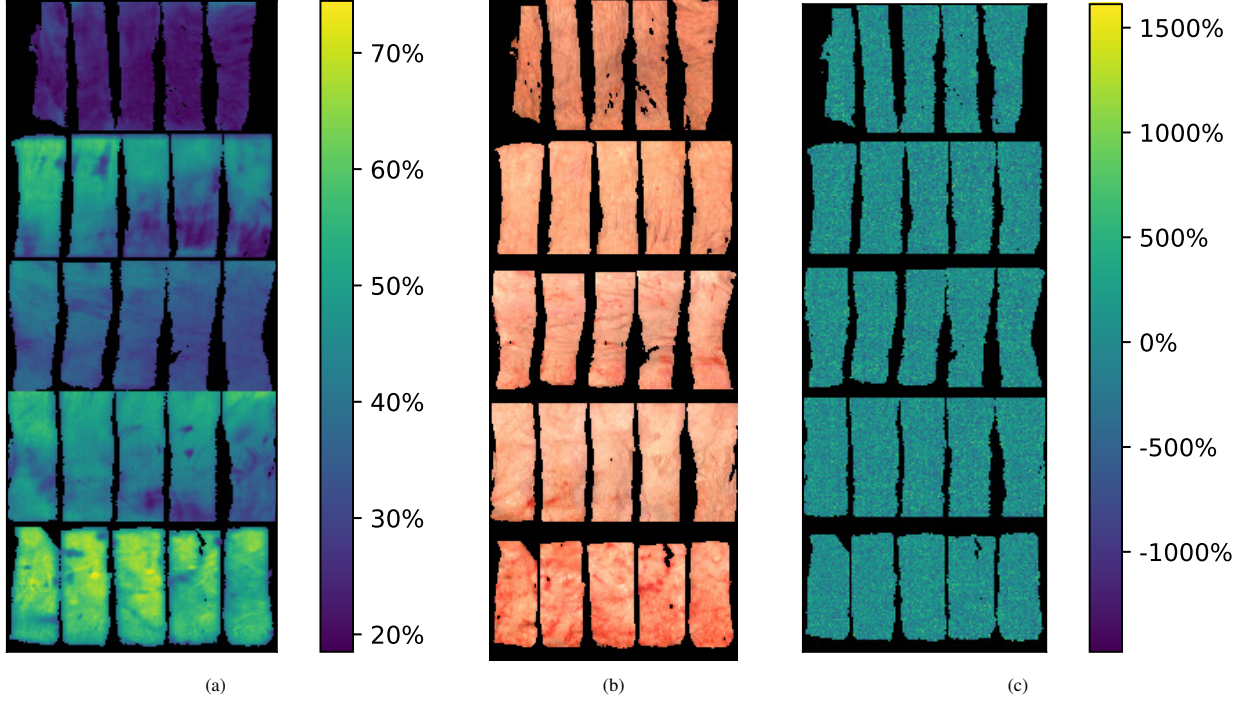


Figure 8: U-Net and PLS generated chemical maps for a test set pork belly from each fat class, F1-F5. Each row represents an entire pork belly. The first row belongs to fat class F1, increasing toward the last row belonging to F5. As in Fig. 1, the five slices of each belly are ordered from left (cranial) to right (caudal), and each slice is oriented with the dorsal side up and the ventral side down. (a) U-Net predictions. (b) RGB illustrations of pork bellies. (c) PLS predictions.

(illustrated in Fig. 1) and reported a negative correlation with fat content. In general, the dorsal cranial part of a pork belly will have a higher fat content than the ventral caudal part. In Fig. 8, this tendency is visible on the chemical maps generated by U-Net. On the other hand, those generated by PLS show no clear structure upon visual inspection.

Fig. 9 plots chemical map predictions on the test set, averaged over each of the 15 regions in the pork bellies, against finger pressure for the same regions. Here, a clear negative correlation exists between predicted fat content and finger pressure. Additionally, U-Net and PLS predict higher fat content for the dorsal cranial sections than for the ventral caudal sections.

Inspired by Herrero-Langreo et al. [26], an analysis of the variance of the spatial distribution of the predicted chemical maps was performed. For this, a predicted chemical map, $\hat{\mathbf{Y}}$, was considered with its mask, $\mathbf{M}$, and its mean prediction, $\hat{y}$ (here, the batch index is omitted for readability). The variance (σ^2) of the part of $\hat{\mathbf{Y}}_{h,w}$ lying inside $\mathbf{M}$ was analyzed. Quantifying how much of the variance was due to spatial correlation was of particular interest. To this end, the so-called nugget effect (C_0) was computed, which is the semi-variogram (γ^2) evaluated as the pair-wise distance goes to zero (Matheron 34).

$$\gamma^2(\hat{\mathbf{Y}}, \mathbf{M}, \delta_h, \delta_w) = \frac{1}{2} \frac{\sum_{h=1}^{H-\delta_h} \sum_{w=1}^{W-\delta_w} \mathbf{M}_{h,w} \left(\left(\hat{\mathbf{Y}}_{h,w} - \hat{\mathbf{Y}}_{h+\delta_h, w+\delta_w} \right)^2 \right)}{\sum_{h=1}^{H-\delta_h} \sum_{w=1}^{W-\delta_w} \mathbf{M}_{h,w}}. \quad (8)$$

Here, between pairs, δ_h and δ_w are the distances in height and width, respectively. As discrete images were used, C_0 was computed as the mean of the semi-variogram evaluated at $(\delta_h, \delta_w) = (0, 1)$ and $(\delta_h, \delta_w) = (1, 0)$

$$C_0(\hat{\mathbf{Y}}, \mathbf{M}) = \frac{\gamma^2(\hat{\mathbf{Y}}, \mathbf{M}, 0, 1) + \gamma^2(\hat{\mathbf{Y}}, \mathbf{M}, 1, 0)}{2}. \quad (9)$$

Under mild assumptions, subtracting C_0 from the total variance, σ^2 , of $\hat{\mathbf{Y}}$ inside $\mathbf{M}$ gives the covariance between

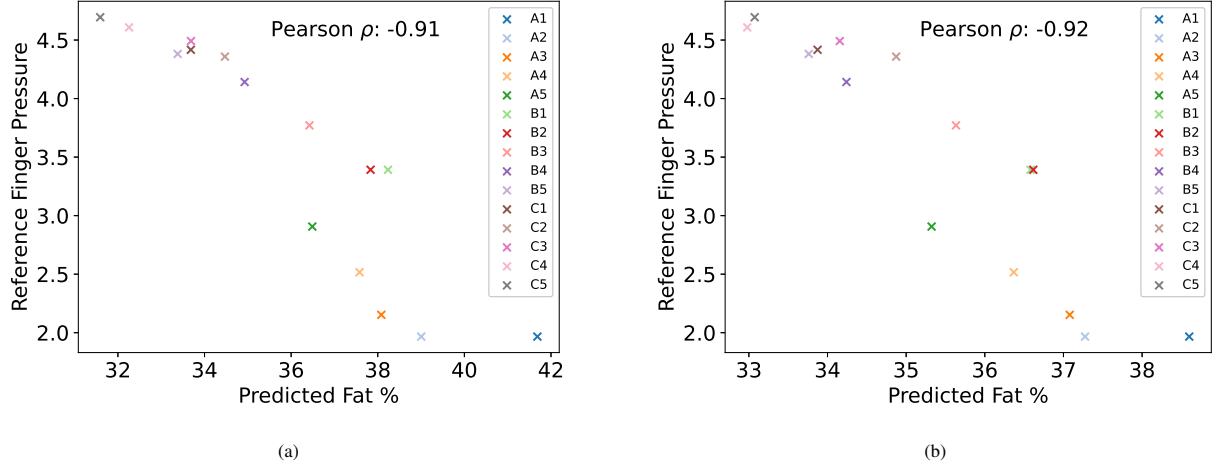


Figure 9: Relationship between mean predicted fat percentages, (a) U-Net. (b) PLS, and mean finger pressure for each of the 15 sections of pork bellies in the test set (see regions in Fig. 1). The Pearson correlation coefficients indicate a high negative linear correlation for both models.

neighboring pixels (Barnes 11). Thus, C_0 is a measure of spatially uncorrelated variance. Consequently, $\frac{C_0}{\sigma^2}$ and $1 - \frac{C_0}{\sigma^2}$ are the ratios of spatially uncorrelated, respectively, correlated variance. Table 3 shows the means of these measures computed over the test set for both PLS and U-Net. Almost all of the variance in U-Net-generated chemical maps was spatially correlated. In contrast, only around three percent of the variance in PLS-generated chemical maps was spatially correlated. The low C_0 value for U-Net predictions can likely be attributed to the following equivalence between C_0 and U-Net’s smoothness loss (SL) term Eq. (5): $C_0(\hat{\mathbf{Y}}, \mathbf{M}) = \frac{1}{4}SL(\hat{\mathbf{Y}}, \mathbf{M})$. As such, U-Net was directly optimized to minimize C_0 , which was not the case for PLS.

Model	Metric			
	σ^2	C_0	$\frac{C_0}{\sigma^2}$	$1 - \frac{C_0}{\sigma^2}$
PLS	52697	51363	0.97	0.03
U-Net	25.17	0.01	0.001	0.999

Table 3: Total variance, σ^2 , and spatially uncorrelated variance, C_0 . As PLS and U-Net predictions have vastly different variances, the ratios of spatially uncorrelated, respectively, correlated variance are better suited for comparison. The values shown are means computed over the test set predictions.

Considering all the results presented in this section, a reader might wonder if a smoothed version of the PLS predictions would show the same underlying structure revealed in the U-Net predictions. Indeed, this was attempted. Smoothing the PLS predictions to a sufficiently high degree keeps the pixel-wise predictions inside the 0-100% range. However, to get a low C_0 , the PLS-generated chemical maps must be smoothed to such a large scale that any fine spatial detail vanishes. Thus, smoothing the PLS predictions does not yield chemical maps equivalent to those generated by U-Net.

4. Discussion

4.1. NIR-HSI depth penetration for fat content analysis

This study’s modeling process was based on the assumption that a correlation exists between the information in the hyperspectral images and the reference fat values. The reference fat values were measured chemically using the entire bellies, while the NIR-HSI camera captures only light reflected on the surface and slightly below the surface due to subsurface scattering. As such, it is unlikely that the models can predict total fat without seeing the entire

bellies in depth. However, as shown in Fig. 9, the chemical maps generated by U-Net and PLS exhibit a high negative correlation with finger pressure. Finger pressure, in turn, has a high negative correlation with fat content (Albano-Gaglio et al. 3), which indicates that the NIR-HSI predictions contain relevant information about the fat distribution in the belly. Thus, it was hypothesized that the models had learned to predict total fat from a measure of subcutaneous fat. It is emphasized, however, that the methods can readily be applied to other image modalities, such as X-ray images and CT scans, which allow image analysis of the entire bellies in depth.

4.2. Bias and scale correction

Consider the U-Net and PLS predictions in Fig. 6. For PLS on mean spectra, the line of best fit for the CV set was guaranteed to be the identity due to the least squares nature of PLS. However, this was not the case for U-Net. Deep learning regressors do not guarantee to find a least squares solution (Igel and Oehmcke 28), especially when training with bulk references and making sample-wise predictions (Engström et al. 18). However, the U-Net’s CV set s_{YX} was very close to its RMSE. Furthermore, bias and scale correction do not change the visual appearance of the chemical maps, nor do they change the values of the ratios in Table 3. The latter points also apply to pixel-wise PLS. For these reasons, no bias or scale corrections were performed.

4.3. Chemical maps

PLS and the modified U-Net had approximately the same performance when evaluated on the mean fat predictions for pork bellies (Fig. 6). However, as evident by the spatial statistics in Table 3, U-Net generated much more meaningful chemical maps than PLS. Additionally, using the PLS model to predict chemical maps gave mean chemical map predictions that were different from those achieved by the same PLS model evaluated on a mean spectrum as evident by the differences between Fig. 6c and Fig. 6e. U-Net, on the other hand, directly generates a chemical map, and, as such, the mean fat prediction of a belly was given by the mean of the pixel-wise predictions on the slices belonging to that belly. In addition to generating spatially aware, locally smooth chemical maps, U-Net almost always stayed inside the 0 – 100% range. This was unlike PLS, where most predictions lay outside this range.

The U-Net loss function does not have a term to accommodate the higher fat concentration in the dorsal cranial section than in the ventral caudal section. Still, this exact behavior emerged when analyzing the distributions of the predicted chemical maps. Thus, the proposed learning framework could model such correlations without explicitly being optimized against them, indicating that the U-Net had learned to generate meaningful chemical maps. Across all five fat classes, U-Net and PLS both predicted higher fat concentrations in the dorsal cranial sections and lower fat concentrations in the ventral caudal sections. U-Net, however, predicted chemical maps with much finer detail than PLS and thus, based on the fat distributions, allowed for a more precise division of pork bellies for downstream processing and value generation.

PLS chemical map predictions were accurate on average and highly correlated with finger pressure (Fig. 9) in the 15 subsections of the pork belly. As such, it may seem that simply applying a smoothing operation to the PLS chemical map predictions would reveal the same spatial structure as those exhibited in U-Net-generated chemical maps. Such smoothing was attempted but to no avail. To reveal spatial structure, the PLS-generated chemical maps had to be smoothened to such a degree that no fine detail remained.

Due to the lack of pixel-wise reference values, it was not possible to directly verify the correctness of predicted chemical maps with this dataset alone. However, the low RMSE shown in Fig. 6b, the spatial structure shown in Fig. 8a, and the spatial correlations shown in Fig. 9a and Table 3 all indicated that the U-Net had learned to generate meaningful chemical maps to the extent of what could be assessed with this dataset.

4.4. U-Net extensions

In the U-Net loss function, Eq. (7), multipliers were used to weigh the different terms relative to each other. The values taken by these multipliers were chosen manually during initial experimentation. The values were chosen such that the numerically dominant term in the loss function would be the mean squared error, while the other parts still made up a small but significant part. The basis for this choice was the main interest in the least squares solution. At the same time, the other terms sought to guide the properties of this solution such that it exhibited local smoothness and contained only values inside the 0 – 100% range.

Instead of the padding scheme used in this study, zero-padding could have been applied to either the input hyperspectral images or the convolution and pooling operations. These zero-padding schemes were attempted during initial experimentation but yielded worse border effects than those arising with the chosen scheme.

While the goal of this study was to serve as a proof of concept, it was likely possible to further decrease the mean squared error of the U-Net by performing more elaborate studies regarding the values of hyperparameters, including the previously mentioned loss-multipliers, learning rate, optimizer, and all other choices that go into training and evaluating a deep learning model. While such performance optimization was not of interest in this study, it is possible to search for more optimal hyperparameters using, e.g., a Tree-Structured Parzen Estimator (TPE) (Bergstra et al. 13) such as implemented by Akiba et al. [1].

In this study, the fat content of pork bellies was analyzed. The dataset includes references for additional parameters (Albano-Gaglio et al. 2). Future developments of this work involve generating chemical maps for multiple parameters using a single model, e.g., a U-Net. A natural extension of the presented loss function would be suitable in this case. First, each parameter should be scaled by its standard deviation to ensure equal contribution to the total loss. Afterward, it is simple to compute an MSE term, Eq. (3), and an SL term, Eq. (5), for each parameter. Interestingly, optimizing the model to regress multiple parameters yields additional possibilities related to the out-of-bounds loss, Eq. (4), that aims to constrain the pixel-wise prediction of each parameter to values between 0 and 100 percent. In addition to constraining each parameter, the sum of parameters should also be constrained in this range. If all parameters can be measured, they should sum to 100 percent. Thus, it is hypothesized that the generalization of the presented work to multiple parameters is not only feasible but perhaps even beneficial compared to the single-parameter case.

Another possible direction for extending this work can be to maintain U-Net's chemical map generation capability while also having it learn to perform segmentation between foreground (pork belly) and background, a task for which it was initially designed (Ronneberger et al. 40).

5. Conclusion

This study has proposed a novel end-to-end deep learning approach based on U-Net for chemical map generation using hyperspectral images of pork bellies with associated mean fat reference values as a case study. By training a modified U-Net with a robust optimization scheme and a multi-faceted customized loss function, using only a mean reference value, the model has learned to generate chemical maps with a high degree of spatial correlation and to make predictions strictly inside the 0-100% range of fat values. These findings contrast the current PLS-based pixel-wise approach to chemical map generation that will generate chemical maps with a low degree of spatial correlation and predictions ranging from -1500-1500%, significantly exceeding the realm of the physically possible. Simultaneously, the U-Net-based approach achieves a lower RMSE than PLS when averaging the chemical maps for comparison with the mean reference values. Thus, this study indicates that the U-Net-based approach enables the generation of chemical maps with meaningful spatial features, emphasizing that the analysis of hyperspectral images can benefit from joint consideration of the spectral and spatial features.

Having shown that deep learning segmentation models such as U-Net can be trained to generate chemical maps from input images directly, further developments in this area are expected to show that they can simultaneously generate chemical maps for multiple parameters and a segmentation mask to distinguish between background and the product of interest in a single forward pass.

6. Funding sources

This work was supported by The Innovation Fund Denmark and FOSS Analytical A/S [grant number 1044-00108B]; MICIU/AEI/10.13039/501100011033/ and FEDER 'Una manera de hacer Europa' [grant number RTI2018-096993-B-I00, 2019-2022]; and the Spanish National Institute of Agricultural Research (INIA) [grant number PRE2019-089669, 2020-2024].

7. Acknowledgments

We want to thank the IRTA technicians Albert Brun, Agustí Quintana, Albert Rossell, Adrià Pacreu, Cristina Canals, and Joel González, for their help in collecting the data used in this project. We also want to thank José M. Martínez for his contribution to the analysis of fatty acids. The CERCA program from the Generalitat de Catalunya is also acknowledged. We also thank Dr. Aneesh Chauhan from WUR for his advice on using complete HSI images for model training.

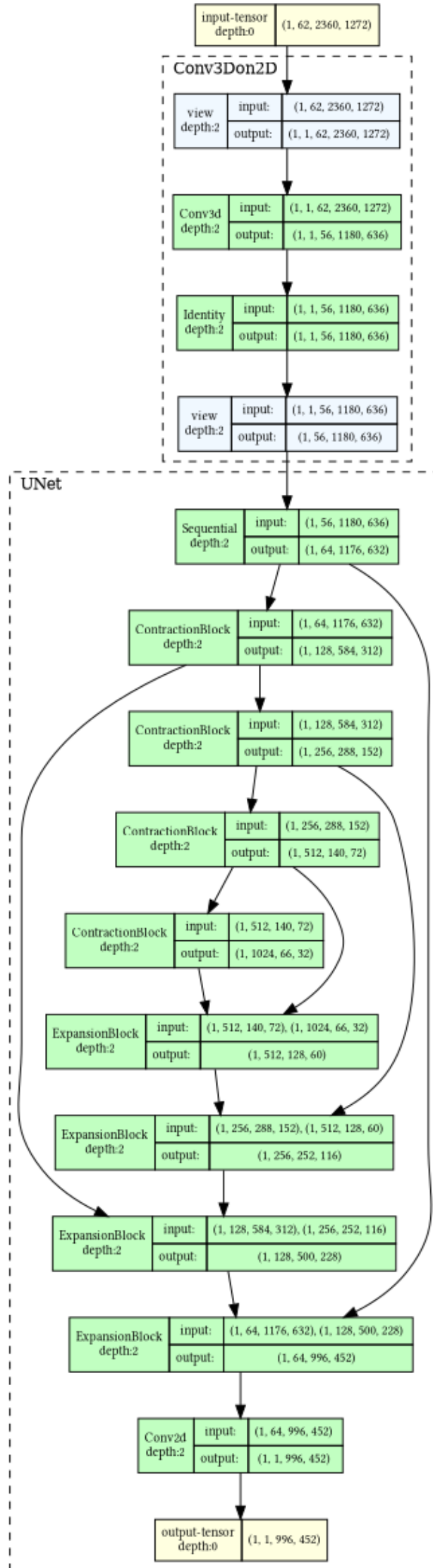
References

- [1] Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 2623–2631. doi:10.1145/3292500.3330701.
- [2] Albano-Gaglio, M., Mishra, P., Erasmus, S.W., Tejeda, J.F., Brun, A., Marcos, B., Zomeño, C., Font-i Furnols, M., 2025. Visible and near-infrared spectral imaging combined with robust regression for predicting firmness, fatness, and compositional properties of fresh pork bellies. *Meat Science* 219, 109645. doi:10.1016/j.meatsci.2024.109645.
- [3] Albano-Gaglio, M., Zomeño, C., Tejeda, J.F., Brun, A., Gispert, M., Marcos, B., Font-i Furnols, M., 2024. Pork belly quality variation and its association with fatness level. *Meat Science* 213, 109482. doi:10.1016/j.meatsci.2024.109482.
- [4] Alin, A., 2009. Comparison of pls algorithms when number of objects is much larger than number of variables. *Statistical Papers* 50, 711–720. doi:10.1007/s00362-009-0251-7.
- [5] Amigo, J.M., Babamoradi, H., Elcoroaristizabal, S., 2015. Hyperspectral image analysis. a tutorial. *Analytica Chimica Acta* 896, 34–51. doi:10.1016/j.aca.2015.09.030.
- [6] Andersson, M., 2009. A comparison of nine pls1 algorithms. *Journal of Chemometrics* 23, 518–529. doi:10.1002/cem.1248.
- [7] Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., et al., 2024. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation, in: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, pp. 929–947. doi:10.1145/3620665.3640366.
- [8] Archana, R., Jeevaraj, P.S.E., 2024. Deep learning models for digital image processing: A review. *Artificial Intelligence Review* 57, 11. doi:10.1007/s10462-023-10631-z.
- [9] Azad, R., Aghdam, E.K., Rauland, A., Jia, Y., Avval, A.H., Bozorgpour, A., Karimijafarbigloo, S., Cohen, J.P., Adeli, E., Merhof, D., 2024. Medical image segmentation review: The success of u-net. *IEEE Transactions on Pattern Analysis and Machine Intelligence* doi:10.1109/TPAMI.2024.3435571.
- [10] Barker, M., Rayens, W., 2003. Partial least squares for discrimination. *Journal of Chemometrics: A Journal of the Chemometrics Society* 17, 166–173. doi:10.1002/cem.785.
- [11] Barnes, R.J., 1991. The variogram sill and the sample variance. *Mathematical geology* 23, 673–678.
- [12] Barnes, R.J., Dhanoa, M.S., Lister, S.J., 1989. Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra. *Applied Spectroscopy* 43, 772–777. doi:10.1366/0003702894202201.
- [13] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems* 24.
- [14] Boseley, R.E., Sylvain, N.J., Peeling, L., Kelly, M.E., Pushie, M.J., 2024. A review of concepts and methods for ftir imaging of biomarker changes in the post-stroke brain. *Biochimica et Biophysica Acta (BBA) - Biomembranes* 1866, 184287. doi:10.1016/j.bbame.2024.184287.
- [15] Dayal, B.S., MacGregor, J.F., 1997. Improved pls algorithms. *Journal of Chemometrics* 11, 73–85. doi:10.1002/(SICI)1099-128X(199701)11:1<73::AID-CEM435>3.0.CO;2-#23.
- [16] ElMasry, G., Mandour, N., Ejeez, Y., Demilly, D., Al-Rejaie, S., Verdier, J., Belin, E., Rousseau, D., 2022. Multichannel imaging for monitoring chemical composition and germination capacity of cowpea (*vigna unguiculata*) seeds during development and maturation. *The Crop Journal* 10, 1399–1411. doi:10.1016/j.cj.2021.04.010.
- [17] ElMasry, G.M., Fulladosa, E., Comaposada, J., Al-Rejaie, S.S., Gou, P., 2021. Selection of representative hyperspectral data and image pretreatment for model development in heterogeneous samples: A case study in sliced dry-cured ham. *Biosystems Engineering* 201, 67–82. doi:10.1016/j.biosystemseng.2020.11.008.
- [18] Engström, O.C.G., Dreier, E.S., Jespersen, B.M., Pedersen, K.S., 2023a. Analyzing near-infrared hyperspectral imaging for protein content regression and grain variety classification using bulk references and varying grain-to-background ratios. doi:10.48550/arXiv.2311.04042, arXiv:2311.04042.
- [19] Engström, O.C.G., Dreier, E.S., Jespersen, B.M., Pedersen, K.S., 2023b. Improving deep learning on hyperspectral images of grain by incorporating domain knowledge from chemometrics, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 485–494. doi:10.1109/ICCVW60793.2023.00055.
- [20] Engström, O.C.G., Dreier, E.S., Jespersen, B.M., Pedersen, K.S., 2024. Ikpls: improved kernel partial least squares and fast cross-validation algorithms for python with cpu and gpu implementations using numpy and jax. *Journal of Open Source Software* 9, 6533. doi:10.21105/joss.06533.
- [21] Engström, O.C.G., Dreier, E.S., Pedersen, K.S., 2021. Predicting protein content in grain using hyperspectral deep learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1372–1380. doi:10.1109/ICCVW54120.2021.00158.
- [22] Engström, O.C.G., Jensen, M.H., 2025. Fast partition-based cross-validation with centering and scaling for $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. *Journal of Chemometrics* 39, e70008. doi:https://doi.org/10.1002/cem.70008.

- [23] Gaci, B., Abdelghafour, F., Ryckewaert, M., Mas-Garcia, S., Louargant, M., Verpont, F., Laloum, Y., Bendoula, R., Chaix, G., Roger, J.M., 2023. A novel approach to combine spatial and spectral information from hyperspectral images. *Chemometrics and Intelligent Laboratory Systems* 240, 104897. doi:10.1016/j.chemolab.2023.104897.
- [24] Gorretta, N., Rabatel, G., Fiorio, C., Lelong, C., Roger, J.M., 2012. An iterative hyperspectral image segmentation method using a cross analysis of spectral and spatial information. *Chemometrics and Intelligent Laboratory Systems* 117, 213–223. doi:10.1016/j.chemolab.2012.05.004.
- [25] He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034. doi:10.1109/ICCV.2015.123.
- [26] Herrero-Langreo, A., Gorretta, N., Tisseyre, B., Gowen, A., Xu, J.L., Chaix, G., Roger, J.M., 2019. Using spatial information for evaluating the quality of prediction maps from hyperspectral images: A geostatistical approach. *Analytica Chimica Acta* 1077, 116–128. doi:10.1016/j.aca.2019.05.067.
- [27] Hotelling, H., 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* 24, 417. doi:10.1037/h0071325.
- [28] Igel, C., Oehmcke, S., 2023. Remember to correct the bias when using deep learning for regression! *KI-Künstliche Intelligenz* 37, 33–40. doi:10.1007/s13218-023-00801-0.
- [29] Jamme, F., Duponchel, L., 2017. Neighbouring pixel data augmentation: A simple way to fuse spectral and spatial information for hyperspectral imaging data analysis. *Journal of Chemometrics* 31, e2882. doi:10.1002/cem.2882.
- [30] Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. doi:10.48550/arXiv.1412.6980, arXiv:1412.6980.
- [31] Luo, Z., Yang, W., Yuan, Y., Gou, R., Li, X., 2024. Semantic segmentation of agricultural images: A survey. *Information Processing in Agriculture* 11, 172–186. doi:10.1016/j.inpa.2023.02.001.
- [32] Mahalanobis, P.C., 1918. On the generalized distance in statistics. *Sankhyā: The Indian Journal of Statistics, Series A (2008-)* 80, S1–S7. URL: <https://www.jstor.org/stable/4872335>.
- [33] Mäkelä, M., Geladi, P., Rissanen, M., Rautkari, L., Dahl, O., 2020. Hyperspectral near infrared image calibration and regression. *Analytica chimica acta* 1105, 56–63. doi:10.1016/j.aca.2020.01.019.
- [34] Matheron, G., 1963. Principles of geostatistics. *Economic geology* 58, 1246–1266. doi:10.2113/gsecongeo.58.8.1246.
- [35] Maw, S.J., Fowler, V.R., Hamilton, M., Petchey, A.M., 2003. Physical characteristics of pig fat and their relation to fatty acid composition. *Meat Science* 63, 185–190. doi:10.1016/S0309-1740(02)00069-4.
- [36] Mishra, P., Albano-Gaglio, M., Font-i Furnols, M., 2024. A short note on deep contextual spatial and spectral information fusion for hyperspectral image processing: Case of pork belly properties prediction. *Journal of Chemometrics* 38, e3552. doi:10.1002/cem.3552.
- [37] Odena, A., Dumoulin, V., Olah, C., 2016. Deconvolution and checkerboard artifacts. *Distill* 1, e3. doi:10.23915/distill.00003.
- [38] Pearson, K., 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2, 559–572. doi:10.1080/14786440109462720.
- [39] Rinnan, Å., van den Berg, F., Engelsen, S.B., 2009. Review of the most common pre-processing techniques for near-infrared spectra. *TrAC Trends in Analytical Chemistry* 28, 1201–1222. doi:10.1016/j.trac.2009.07.007.
- [40] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III* 18, Springer. pp. 234–241. doi:10.1007/978-3-319-24574-4_28.
- [41] Savitzky, A., Golay, M.J.E., 1964. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry* 36, 1627–1639. doi:10.1021/ac60214a047.
- [42] Siddique, N., Paheding, S., Elkin, C.P., Devabhaktuni, V., 2021. U-net and its variants for medical image segmentation: A review of theory and applications. *IEEE access* 9, 82031–82057. doi:10.1109/ACCESS.2021.3086020.
- [43] Signoroni, A., Savardi, M., Baronio, A., Benini, S., 2019. Deep learning meets hyperspectral image analysis: A multidisciplinary review. *Journal of imaging* 5, 52. doi:10.3390/jimaging5050052.
- [44] Snee, R.D., 1977. Validation of regression models: methods and examples. *Technometrics* 19, 415–428. doi:10.1080/00401706.1977.10489581.
- [45] Sørensen, K.M., van den Berg, F., Engelsen, S.B., 2021. Nir data exploration and regression by chemometrics—a primer. *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*, 127–189doi:10.1007/978-981-15-8648-4_7.
- [46] Steinier, J., Termonia, Y., Deltour, J., 1972. Smoothing and differentiation of data by simplified least square procedure. *Analytical Chemistry* 44, 1906–1909. doi:10.1021/ac60319a045.
- [47] Tao, F., Yao, H., Hruska, Z., Rajasekaran, K., Qin, J., Kim, M., Chao, K., 2024. Raman hyperspectral imaging as a potential tool for rapid and nondestructive identification of aflatoxin contamination in corn kernels. *Journal of Food Protection* 87, 100335. doi:10.1016/j.jfp.2024.100335.
- [48] Thumm, A., Riddell, M., Nanayakkara, B., Harrington, J., Meder, R., 2010. Near infrared hyperspectral imaging applied to mapping chemical composition in wood samples. *Journal of Near Infrared Spectroscopy* 18, 507–515. doi:10.1255/jnirs.909.
- [49] Vedaldi, A., Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Darrell, T., 2014. Convolutional architecture for fast feature embedding. *Cornell University*.
- [50] Wold, H., 1966. Estimation of principal components and related models by iterative least squares. *Multivariate analysis*, 391–420.
- [51] Wold, S., Sjöström, M., Eriksson, L., 2001. Pls-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems* 58, 109–130. doi:10.1016/S0169-7439(01)00155-1.
- [52] Xu, J.L., Gowen, A., 2020. Spatial-spectral analysis method using texture features combined with pca for information extraction in hyperspectral images. *Journal of Chemometrics* 34, e3132. doi:10.1002/cem.3132.

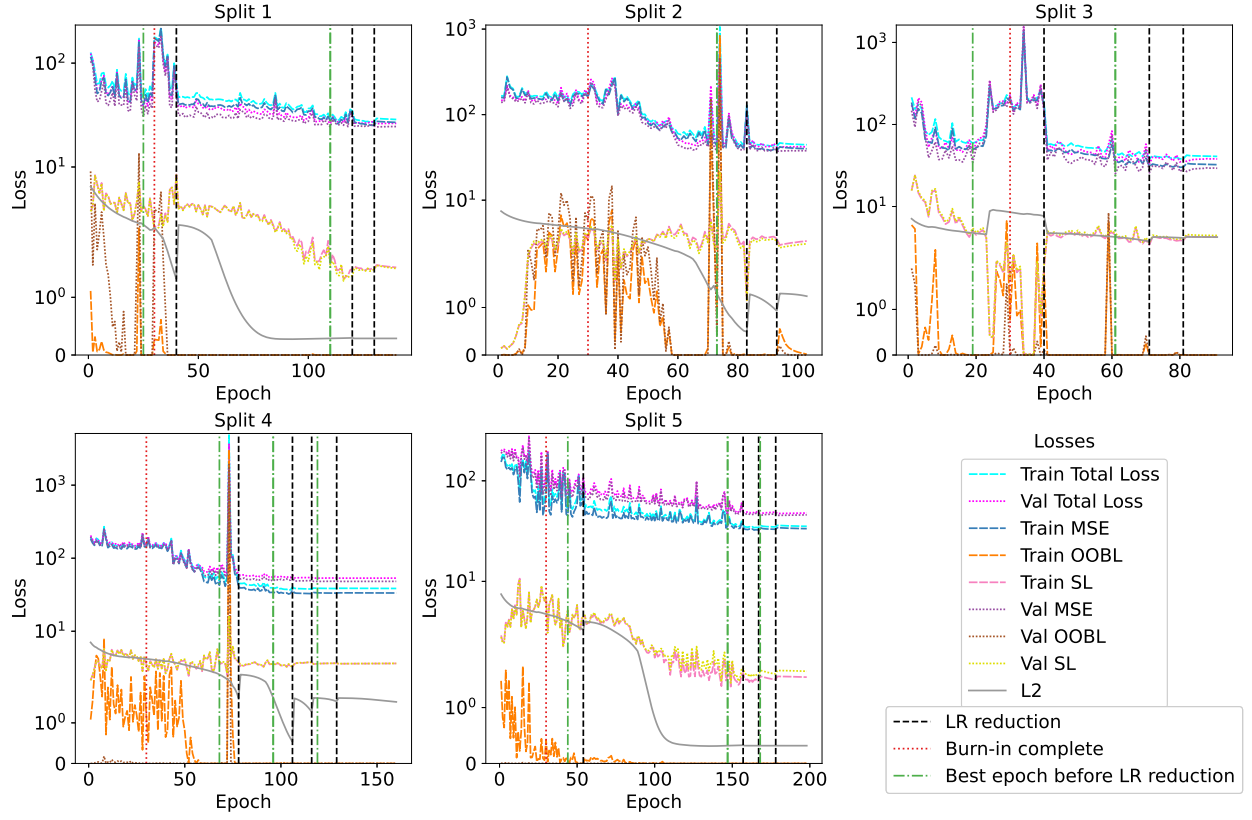
Appendix A. Modified U-Net architecture

This appendix shows architecture of the modified U-Net. It consists of the sequentially linked 3D convolution module and the U-Net module. In addition to showing input and output sizes, it shows the sizes of intermediate tensors.



Appendix B. Loss evolution for U-Net

Fig. B.10 shows the loss evolution of U-Net for all five CV splits.



Chapter 7

Barley Germination

THE previous chapters have shown how joint spatio-spectral analysis of NIR-HSI images can be used for various tasks. Still, we have not encountered a parameter that could not be modeled to a somewhat accurate degree using only spatial or spectral information alone. Aside from modeling a chemical parameter’s spatial distribution, we have yet to see a parameter that requires joint spatio-spectral analysis of NIR-HSI images. Such a parameter may be barley’s germinative capacity, which is the measure of barley’s ability to germinate when it is exposed to moisture. A recent review (Orth et al. 2025) mentions that both spatial and spectral features are relevant to determining whether a barley kernel will germinate. In particular, during germination, a chemical process occurs inside the kernel but may be inhibited if the kernel is sufficiently damaged by, for example, being cleaved. This chapter is written as a technical report of a recent and previously unpublished study into predicting barley’s germinative capacity using the open dataset of RGB and NIR-HSI dataset collected by Engstrøm et al. (2025b). The technical report describing the dataset, from which this chapter will paraphrase and reuse figures freely, is reprinted in Appendix B for convenience. Although this chapter is written by Ole-Christian Galbo Engstrøm, the experiments were carried out in close collaboration with Erik Schou Dreier, Birthe Møller Jespersen, and Kim Steenstrup Pedersen.

7.1 Introduction

Barley is commonly used for malting to produce various beverages. During the malting process, barley must germinate rapidly and uniformly (Gubler et al. 2005). Therefore, to maximize yield, barley’s germinative capacity must be evaluated prior to malting. Current official methods for this evaluation are based on the destructive and manual assessment of a few hundred kernels (European Brewery Convention 1997). This implies that bulks with heterogeneous germinative capacity can not necessarily be divided into subsamples

with uniform germinative capacity.

Furthermore, the relatively small number of kernels used for current methods raises the question of whether the analyzed subsample is representative of the entire bulk. Thus, a rapid and non-destructive assessment method for quantifying barley’s germinative capacity can potentially revolutionize the malting industry. Orth et al. (2025) mention that NIR-HSI may be a viable method. Indeed, Arngren et al. (2011) used NIR-HSI to model the amount of time before single barley kernels germinate after exposure to moisture. They used a dataset with uniformly distributed reference values and divided 60 hours into eight intervals for the classification of germination time. They achieved 41% accuracy on single kernels. While this may seem low and likely not satisfactory for a standard method, it is much better than random guessing.

In the following study, like Arngren et al. (2011), we analyze the germination time of single barley kernels using NIR-HSI images. However, we also include RGB images to compare the predictive performance that can be achieved with a relatively cheap color camera that sees spatial, but no chemical information, a full spatio-spectral NIR-HSI image, and NIR mean spectra. Thus, we attempt to assess whether it is possible to predict barley’s germinative capacity and to what degree of accuracy with all three modalities.

7.2 Dataset

This section briefly describes the dataset and gives sufficient information to understand the subsequent modeling process. For a complete dataset description, including its acquisition, see Appendix B.

We use a time series dataset of RGB and NIR-HSI images of 2242 individual barley kernels. The NIR-HSI images contain 204 uniformly distributed wavelength channels between 938 nm and 1662 nm. The barley kernels are taken from four different malting barley samples: two of the Prospect variety and two of the Laureate variety. Table 1 shows the distribution of kernels per sample. We imaged each barley kernel before exposure to moisture and every 24 hours after exposure to moisture for five consecutive days. Table 2 shows the twelve images associated with a single grain kernel. Consequently, the barley kernels are labeled by the day (1-5) at which a visible sprout emerged or labeled as not germinated if no visible sprout appeared before image acquisition on day 5. The distribution of labels for germinated kernels is shown in Table 3 and visualized together with the distribution of non-germinated kernels in Figure 7.1. Associated with each image is also a segmentation mask based on a combination of Otsu’s method for thresholding (Otsu 1979) and connected components on the thresholded image. The mask is used to extract a mean NIR spectrum from the NIR-HSI image. When extracting the mean NIR spectrum, we first transform every pixel to pseudo-absorbance by taking

the negative of its logarithm.

The dataset is split into six evenly sized partitions (folds) by stratified random sampling to ensure identical distributions of germination days throughout the six partitions to the maximum possible degree. For example, 28 kernels germinated on day one, so four partitions will each contain five of these kernels, and the remaining two partitions will contain four each. Five partitions are used for five-fold cross-validation, and the sixth partition is used for testing. This stratification does not consider the distribution of germination days per barley variety, Section 7.5 discusses this.

Malting barley kernels are typically expected to germinate to a degree close to 100%. Thus, it is clear that something has gone wrong in the germination process of this dataset where less than 20% of the kernels germinate and typically show just a tiny sprout where a long sprout is typically observed (see, for example, Table 3). Engström et al. (2025b) determined the cause for this low germinative capacity to be glue residue. Specifically, the barley kernels were placed inside a 3D-printed grid to facilitate segmentation and to avoid mixing individual kernels during storage. Glue residue was stuck to the underside of these 3D-printed grids. The glue must have mixed with the water used to moist the kernels, and the kernels have subsequently absorbed the contaminated water, which seems to have severely inhibited their germinative capacity.

Barley Variety					
	Prospect_0	Prospect_1	Laureate_0	Laureate_1	Total
No. kernels	624	394	624	600	2242

Table 1: Number of barley kernels for each of the four barley varieties. This table is copied from Engström et al. (2025b).

	Pre-moisture	Day 1	Day 2	Day 3	Day 4	Day 5
RGB						
NIR-HSI						

Table 2: A single barley kernel imaged before exposure to moisture and every 24 hours after exposure to moisture. A sprout appears on Day 2, meaning that the reference value for germination time for this specific kernel is two days. This table is copied from Engström et al. (2025b).

barley variety	Germination Day					
	1	2	3	4	5	Any
Prospect_0	4	29	23	15	12	83
Prospect_1	1	0	1	0	3	5
Laureate_0	13	21	25	15	17	91
Laureate_1	9	43	57	58	35	202
All	28	93	106	88	67	382

Table 3: Number of barley kernels that germinated each day out of those shown in Table 1. No kernels were germinated prior to exposure to moisture. This table is copied from Engström et al. (2025b).

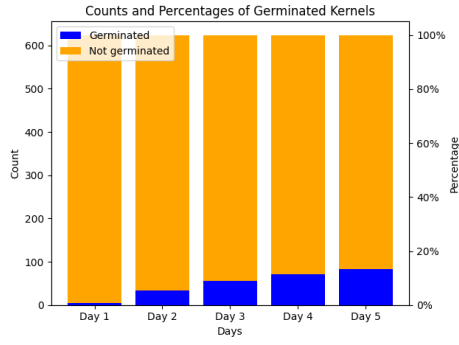
7.3 Modeling

Given the granularity of our labels (Table 3), we have identified three tasks in order of increasing complexity.

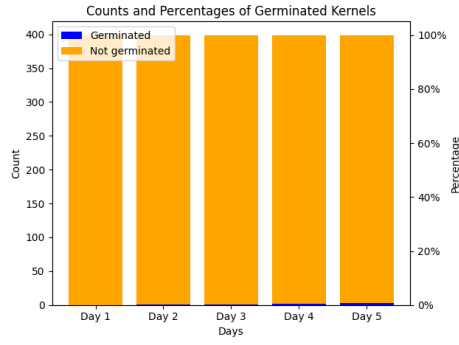
1. Modeling of whether a given kernel is germinated. For this task, we use the images from day five to maximize the number of germinated kernels.
2. Modeling of whether a given kernel will germinate in the future. For this task, we use the images acquired pre-moisture but label the image as germinated if the kernel germinated at any time during the experiment.
3. Modeling of not whether a kernel will germinate but when it will germinate. This is the task of taking the images at pre-moisture and predicting their exact germination day.

Due to the results presented in the next section, we decided to undertake only the first two tasks and not attempt the third task.

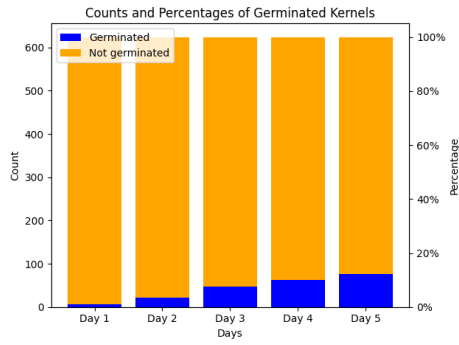
The dataset is heavily unbalanced, so we apply class weights to mitigate this. We compute the class weights for germinated and not germinated independently for each barley variety, the training and validation partitions, and the test partition. We use Pedregosa et al. (2011) to compute the class weights based on a heuristic from King and Zeng (2001). The following is a description of the class weights. Generally, we can compute class weights for a class $c \in \{1 \dots C\}$ where C is the number of unique classes. In our case, $C = 2$, but the following holds in general. Let S be the set of samples with a size of $N = |S|$. Then S can be indexed by $i \in \{1 \dots N\}$ to get sample S_i . Let $\mathbf{1}_c$ be the indicator function for c , evaluating to 1 if its argument belongs to c and otherwise to 0. Then, $N_c = \sum_{i=1}^N \mathbf{1}_c(S_i)$ is the number of samples belonging to c , and the weight for c is defined by



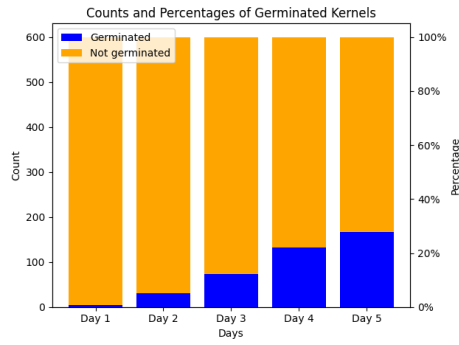
(a) Prospect_0.



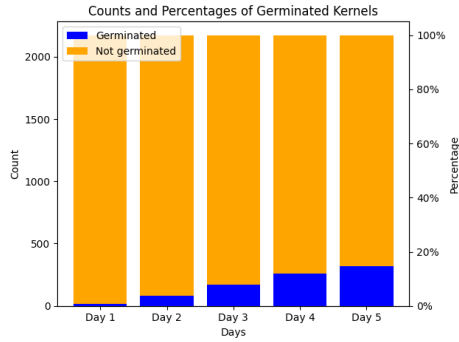
(b) Prospect_1.



(c) Laureate_0



(d) Laureate_1



(e) All varieties.

Figure 7.1: Histograms showing how the number of germinated kernels evolves from day to day. This figure is copied from Engstrøm et al. (2025b).

$$w_c = \frac{N}{CN_c}. \quad (7.1)$$

This implies that the total weight for any given class is constant. That is, $w_c N_c = \frac{N}{C}$ for any c . This is proven by

$$w_c N_c = \frac{N N_c}{C N_c} = \frac{N}{C}. \quad (7.2)$$

This also implies that the total weight across all samples is N , proven by

$$\sum_{c=1}^C w_c N_c = \frac{NC}{C} = N. \quad (7.3)$$

Thus, the average weight across all samples is 1. When used in a deep learning context, this also means that applying these weights does not effectively change the learning rate.

7.3.1 PLS

For PLS-based modeling, we use the weighted variation (Becker and Ismail 2016) implemented by Engström et al. (2024). We model PLS with the mean absorbance spectra and experiment with two combinations of preprocessing. The first combination consists of applying the Standard Normal Variate (SNV, Barnes et al. 1989) transformation followed by convolution with a Savitzky-Golay (SG, Savitzky and Golay 1964; Steinier et al. 1972) filter with a window size of seven, a polynomial order of two, and a derivative order of two. The second combination does not have row-wise preprocessing but a column-wise weighted centering of spectra and target values. These two combinations correspond to the optimal settings for protein content regression and grain variety classification in the work by Engström et al. (2023b).

We train PLS under a five-fold cross-validation scheme and determine the best number of components as the one maximizing weighted accuracy (WAcc) on the validation partitions. Subsequently, we average the five best numbers of components, refit the PLS model on the entire cross-validation dataset with that many components, and use this model for evaluation on the test set. This approach is similar to the one by Engström et al. (2025a).

7.3.2 Image-based Modeling

For image-based modeling, we employ variants of ResNet-18 (K. He et al. 2016) based on the work by Engström et al. (2023b). For RGB images, we use their vanilla variant of ResNet-18. For NIR-HSI images, we employ the ResNet-18 variant with an initial 3D convolution layer with a single filter with a kernel size of (height, width, depth) = (3, 3, 7) and a stride of 1. Subsequently, we employ a downsampler (Engström et al. 2023b) with three filters, effectively reducing the channel size from 204 to 3. The vanilla ResNet-18 then consumes the output of the downsampler. We also experiment with a variant using the 3D convolution layer without the subsequent downsampler for NIR-HSI images. We implement all models and deep learning pipelines using PyTorch (Ansel et al. 2024).

We apply channel-wise centering and scaling to the images based on training partition-wise mean and standard deviation spectra, respectively, RGB-values. We compute the means and standard deviations across all pixels in the training partition for this task.¹ Due to the extreme number of pixels across an entire training partition, the computation of channel-wise means and standard deviations risks suffering from numerical errors. For this reason, we employ a two-pass variant of Welford’s algorithm (Welford 1962) and enhance it by replacing its internal summations with Neumaier summation (Neumaier 1974), which is a compensated summation algorithm improving upon the Kahan-Babuška compensated summation algorithm (Kahan 1965; Babuška 1968). After centering and scaling, we zero-pad the images to be square and have constant side lengths of 336 pixels for RGB images and 112 pixels for NIR-HSI images. These are larger than any side lengths in the un-padded datasets, so no additional resizing or cropping is necessary. During training, we shuffle the training partition uniformly at random and independently for each epoch. We also employ random horizontal and vertical flipping, each with a probability of 0.5.

In the ResNet’s, we initialize all weights by sampling from Kaiming He Normal distributions (K. He et al. 2015) and all biases to zero. The loss function is the summation of weighted binary cross-entropy (WBCE) and L2-regularization (L2) of the weights using a 10^{-6} multiplier for L2.

We train the models under a five-fold cross-validation. We use batches of 32 images and optimize the models with Adam (Kingma and Ba 2017) using hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and an initial learning rate of 10^{-2} . During training, we keep track of the current best weights and associated optimizer state. The current best weights are those achieving the lowest validation partition WBCE. For the first 300 epochs, there is no change in the learning rate. Starting from the 301st epoch, we initialize counters for learning rate reduction and early stopping. After 100 epochs with no improvement in WBCE on the validation set, we restore the current best weights and associated optimizer state. Then, we multiply the learning rate 10^{-1} but cap it at 10^{-7} . The training process halts if there is no improvement in the validation partition WBCE after 400 epochs or if it reaches 10,000 epochs. This implies that a model is trained for at least 700 epochs. In practice, all models were halted by the early stopping criterion.

Similar to our previous works (Engstrøm et al. 2023b; Engstrøm et al. 2025a), we construct an ensemble of the five models yielded by cross-validation. A uniform mean of its constituents’ individual predictions gives the ensemble’s predictions. We evaluate the ensemble on the test set. This gives us a model

¹In practice, we compute the means and standard deviations only across pixels inside the polygons that define the cutouts from the Petri dishes from which the kernel images are extracted. See Figure 4 in Appendix B for an example. We also only apply centering and scaling inside the polygons, leaving the pixels outside of the polygons as 0.

that is simultaneously trained and validated on all cross-validation partitions, like our final PLS model.

7.4 Results

For PLS, the model using only column-wise weighted centering had a few percentage points better WAcc across the validation partitions on both tasks. The best number of components was 11 for task 1 and 8 for task 2. For ResNet-18 on NIR-HSI images, the variant using a Downsampler achieved superior WBCE for task 1, and for task 2, it was tied with the variant without a downsampler. We keep the best PLS model and best ResNet-18 model for evaluation and disregard the others.

Table 4 shows balanced accuracy on the test set for each model for each barley variety. Balanced accuracy is the mean of recall for the two classes (germinated and not germinated). ResNet-18 on NIR-HSI is the best model overall, winning on 6/8 metrics. To avoid cluttering this section, we show confusion matrices only for ResNet-18 on NIR-HSI images. Figure 7.2 and Figure 7.3 show confusion matrices for tasks 1 and 2, respectively.

Task	Model and Modality	barley variety			
		P_0	P_1*	L_0	L_1
1	PLS NIR	56	42	76	56
1	ResNet-18 RGB	71	46	78	76
1	ResNet-18 NIR-HSI	78	48	86	86
2	PLS NIR	56	45	71	54
2	ResNet-18 RGB	48	40	64	57
2	ResNet-18 NIR-HSI	63	49	66	53

Table 4: Test set balanced accuracy (in %) for each barley variety. P and L are short for Prospect and Laureate, respectively. The best score for each barley variety and task is in **bold**. *Prospect_1 only has a single germinated kernel in the test set. Wrongfully predicting this one kernel will give a balanced accuracy of at most 50%.

7.5 Discussion

From the analyses conducted in this study, our dataset seems insufficient for thorough modeling of the germination process. There is a clear problem with too few germinated kernels due to the issues with glue residue mentioned by Engstrøm et al. (2025b). In particular, Prospect_1 having only five kernels germinate in total renders meaningful modeling and evaluation virtually im-

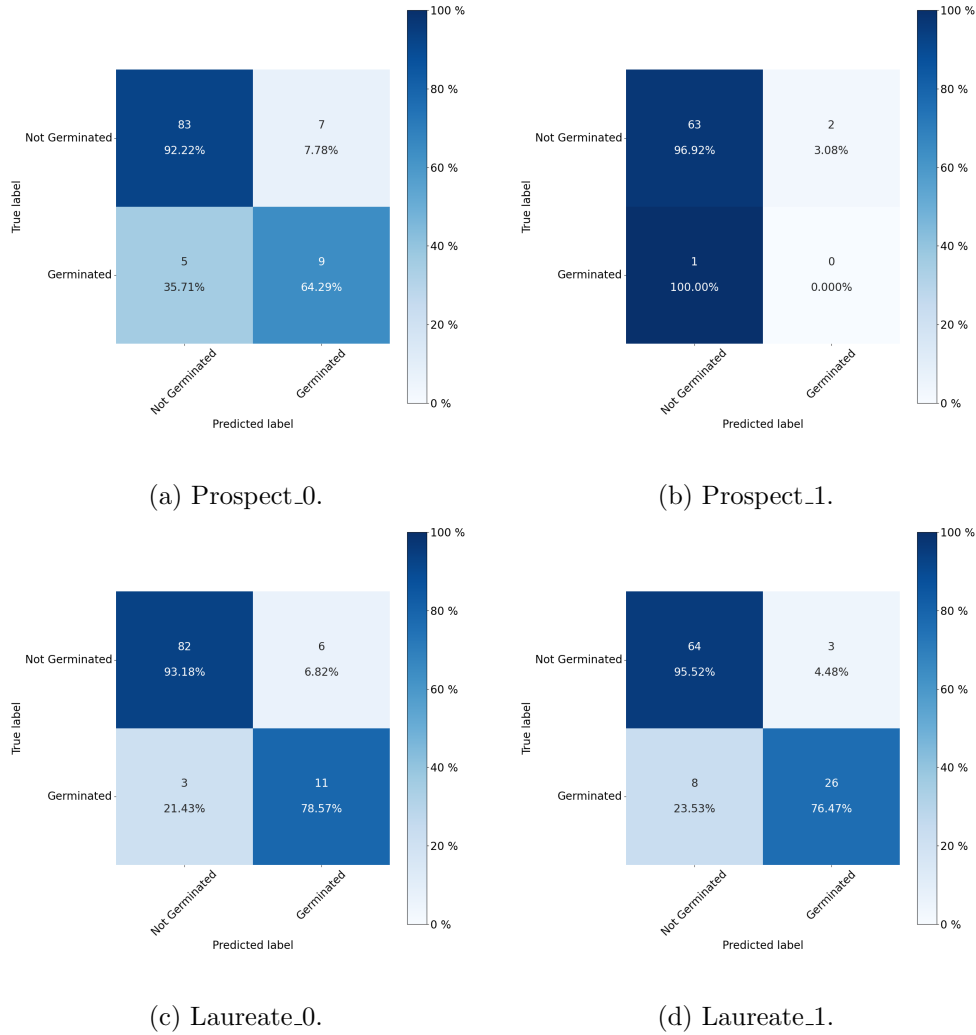


Figure 7.2: Confusion matrices for test set predictions by the ResNet-18 NIR-HSI model on task 1.

possible. The other barley varieties have a higher germinative capacity but have also been severely inhibited. Furthermore, for task 2, we assume equal conditions for the kernels. In particular, for task 2, we cannot model if the glue residue will inhibit some kernels more than others. This is because the images used for task 2 are acquired before the glue is mixed with the kernels. In practice, we do not know how much glue was absorbed by each kernel.

The labeling was done manually by inspecting the RGB images, which have the highest spatial resolution, thus making labeling easier for a human. During image acquisition, germinated kernels were placed to make the sprouts visible to the cameras. Therefore, we do not expect systematic errors due to germinated kernels facing away from the camera.

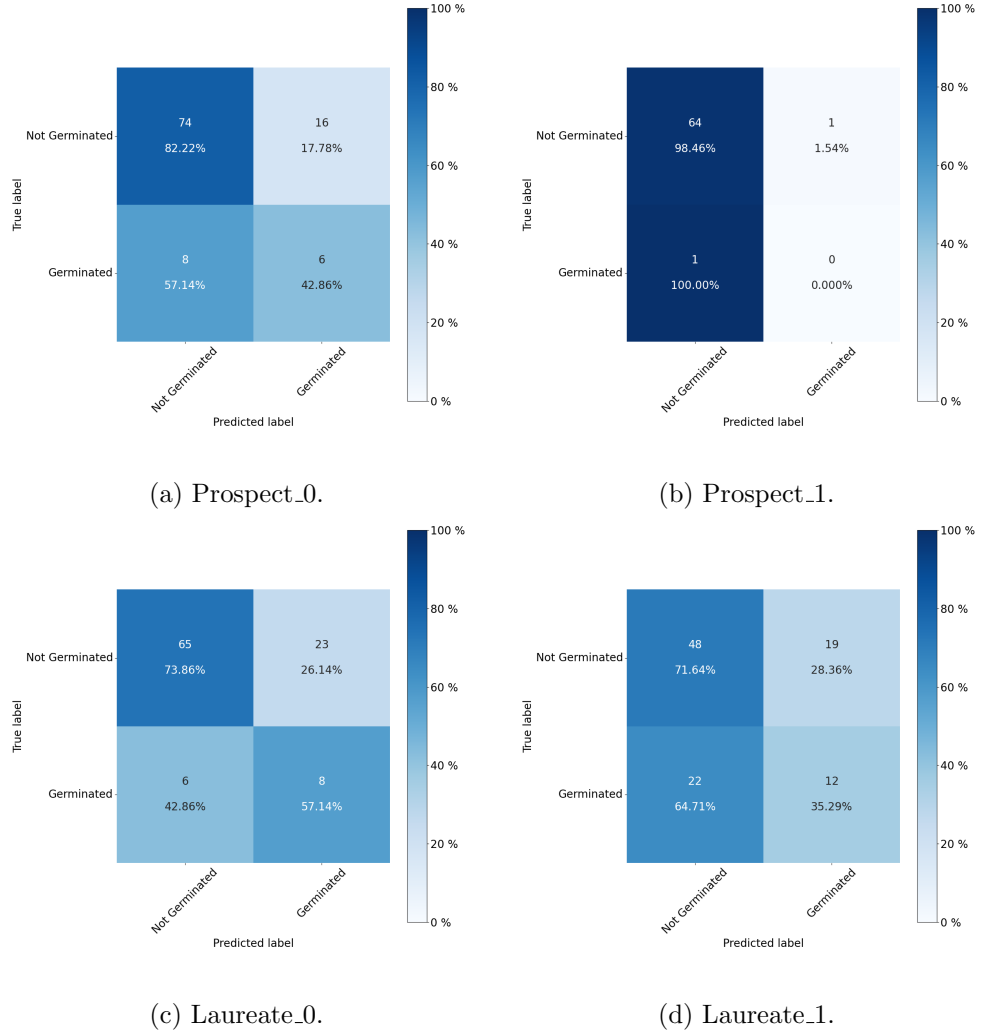


Figure 7.3: Confusion matrices for test set predictions by the ResNet-18 NIR-HSI model on task 2.

Inspecting Table 4, applying PLS to NIR spectra seems to yield about the same performance on tasks 1 and 2. It seems better than random, indicating that spectral information may be relevant for both tasks. By analyzing RGB images with ResNet-18, the detection of germinated kernels is decently accurate. However, considering task 2, the performance of RGB imaging deteriorates significantly and appears more or less random, indicating that spatial and color information alone is insufficient to predict germinative capacity. Still, RGB imaging can undoubtedly be used to detect physical defects such as cleaved kernels. As the dataset consisted of kernels without visible damage, this potential capacity has had no influence on the metrics. Finally, applying ResNet-18 to NIR-HSI images is the most promising strategy. It can detect

germinated kernels to a higher degree than both RGB imaging and PLS on NIR spectra, and it also outperforms them on task 2 on average. Disregarding Prospect_1, it seems that both PLS on NIR spectra and ResNet-18 on NIR-HSI images can predict whether a barley kernel will germinate or not to a degree better than random. Still, further studies on a larger dataset are required to reach firm conclusions.

In reality, barley germination is a continuous process, and so it is likely best modeled as such. However, due to the manual inspection of kernels at discrete intervals, like Arngren et al. (2011), we assign labels based on snapshots of the process at these intervals. Still, a prediction closer to the actual germination day is better than a prediction further away. For example, if a kernel germinates on day 1, predicting day 2 is better than predicting day 3 and should be treated as such. This realization is relevant for task 3, which we decided not to endeavor. For NIR-HSI analysis to be truly useful as a standard analytical method, task 3 must be solved (Orth et al. 2025). If we were to undertake task 3, a continuous metric such as (weighted) mean squared error would have been appropriate for the regression. However, some kernels do not germinate, so assigning a continuous label to them seems challenging. A possible solution to this would be joint classification, based on task 2, and subsequent regression on kernels predicted to germinate by the classifier. By modifying ResNet-18 to have both a classification and a regression head, it can simultaneously make predictions for task 2 and regression predictions for task 3. Such a strategy requires joint optimization, which may be helpful for the overall performance due to cross-learning.

Finally, the stratified sampling in this study was not done per barley variety. So, although the distributions of germinated and non-germinated kernels are identical across partitions, the distributions of germinated and non-germinated kernels for each variety are not necessarily identically distributed. The class weights, however, are computed for each variety and will mitigate this issue. Still, due to the very few germinated kernels, there may simply not be enough variation in the images used for training and validation. Indeed, despite the class weights, the confusion matrices shown in Figure 7.2 and Figure 7.3 seem to be biased towards predicting not germinated for all varieties. Although not shown here, the confusion matrices for PLS and ResNet-18 on RGB images exhibit a similar pattern throughout both tasks on all barley varieties. The exception is PLS for task 1 on both Laureate samples. However, as both Laureate samples germinate more than both Prospect samples, perhaps the PLS model has been biased accordingly despite the weighting due to the low variation.

Further investigations based on the same dataset could resolve some of the issues mentioned in this discussion. However, a larger dataset is needed for conclusive results.

7.6 Conclusion

Initially, we set out to solve three tasks. Namely, predicting whether a barley kernel has germinated, whether it will germinate, and, finally, on which day it will germinate if it will germinate. However, we decided that further modeling would not yield conclusive results due to the performance on the first two tasks and a limited amount of germinated kernels for each day. While the dataset used for this analysis has its clear limitations, we see indications that NIR-HSI is a superior modality to both NIR spectroscopy and RGB imaging for detecting already germinated kernels. Further studies on a larger dataset are required for stronger conclusions, especially regarding the prediction of barley's germinative capacity. Still, we maintain a careful optimism about using joint spatio-spectral analysis of barley's germinative capacity through NIR-HSI images.

Part III

Final Remarks

Chapter 8

Discussion

THIS chapter discusses the hypotheses, studies, and results presented in the previous parts. The discussion is structured in three sections. Section 8.1 is concerned with the PLS and fast cross-validation algorithms presented in Part I. Likewise, Section 8.2 offers an analysis of the results achieved in Part II and relates them to the five research hypotheses enumerated in Chapter 1. Finally, Section 8.3 offers perspectives for future research directions based on this thesis.

8.1 Fast Partial Least Squares

Recall that PLS models a linear relationship between $\mathbf{X} \in \mathbb{R}^{N \times K}$ and $\mathbf{Y} \in \mathbb{R}^{N \times M}$ yielding a regression matrix $\mathbf{B} \in \mathbb{R}^{K \times M}$. Chapter 2 presented IKPLS Algorithms 1 and 2. IKPLS Algorithm 1 is fast if $N \approx K$ and has reasonable runtime if $N < K$. Conversely, if $N > K$, then IKPLS Algorithm 2 is the faster choice, as it computes $\mathbf{X}^T \mathbf{X}$ and uses this instead of the larger $\mathbf{X}$ to derive $\mathbf{B}$. These algorithms are well-suited for NIR-HSI as K is typically in the hundreds, while N may be much larger as in the datasets analyzed in Chapter 4. Conversely, for datasets of NIR spectra measured with spectrometers, it may well be the case that $N \ll K$. In this case, IKPLS Algorithm 2 will be devastatingly slow, and while IKPLS Algorithm 1 will be reasonably fast, an alternative that derives $\mathbf{B}$ through $\mathbf{X} \mathbf{X}^T$ will be much faster. This could, for example, be the PLS algorithms devised by R  nnar et al. (1994), computing both $\mathbf{X} \mathbf{X}^T$ and $\mathbf{Y} \mathbf{Y}^T$, or Liland et al. (2020), computing only $\mathbf{X} \mathbf{X}^T$.

When computing $\mathbf{X} \mathbf{X}^T$ or $\mathbf{Y} \mathbf{Y}^T$, the fast cross-validation algorithms introduced in Chapter 3 can not be used to speed up cross-validation, as it computes $\mathbf{X}^T \mathbf{X}$ and $\mathbf{X}^T \mathbf{Y}$. Instead, in this case, another fast cross-validation algorithm by R  nnar et al. (1995) with improvements by Liland et al. (2020) can be used to speed up the computation of $\mathbf{X} \mathbf{X}^T$ and $\mathbf{Y} \mathbf{Y}^T$. Four unique combinations exist of centering and scaling for $\mathbf{X} \mathbf{X}^T$, namely, centering, scaling, neither, and both. The fast cross-validation algorithm by Liland et al. (2020)

correctly handles centering and no preprocessing but relies on the dataset-wise standard deviation for scaling, thus introducing data leakage from validation partitions to training partitions. This is unlike the algorithms in Chapter 3, which correctly handle all 16 (12 unique) combinations of centering and scaling for $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$.

Another point about the fast cross-validation algorithms presented in Chapter 3 is regarding asymptotic optimality. If an algorithm is asymptotically optimal, then it is, at most, a constant factor worse than the best possible algorithm that can theoretically exist. The following is an argument that, in a specific setting, our algorithms (Engstrøm and Jensen 2025) have asymptotically optimal time and space complexities. Recall that the time complexity is $\Theta(NK(K + M))$ and the space complexity is $\Theta((N + K)(K + M))$. Also, recall that P is the number of partitions (folds) in the cross-validation scheme. To recap, the space complexity boils down to storing the input $\mathbf{X}$ and $\mathbf{Y}$ and a single pair of matrix products $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. The space complexity is then based on the assumption that matrix product pairs for all training partitions will use the same space. That is, the space used under one partition will be overwritten under the next partition. Consider the case where the training partition-wise matrix products must be stored, perhaps for later analysis with multiple methods. Then the space complexity increases to $\Theta((N + PK)(K + M))$ while the time complexity is unaffected. Consider the case of leave-one-out cross-validation (LOOCV), the case where $P = N$. Then the space complexity becomes $\Theta((N + NK)(K + M)) = \Theta(NK(K + M))$, matching the time complexity. Time complexity can never decrease below space complexity as it requires at least one unit of time to fill one unit of space. Under the conditions that we must store $\mathbf{X}$, $\mathbf{Y}$, and all $P = N$ pairs of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$, the space complexity matches exactly this. Thus, neither space nor time complexity can be reduced further, and the algorithm is asymptotically optimal under these conditions.

8.2 Near-Infrared Hyperspectral Image Analysis

8.2.1 Joint Spatio-Spectral Modeling

Based on the studies in Chapters 4 and 7, it seems that joint spatio-spectral analysis of the entire NIR-HSI image may outperform spatial or spectral analysis alone for some parameters. In particular, when both spatial and spectral information is relevant for the parameter, as is the case with grain variety classification (Dreier et al. (2022) and Engstrøm et al. (2023b)) and classification of germinated and non-germinated barley kernels (Orth et al. 2025). Conversely, spectral analysis seems to perform comparably with joint spatio-spectral analysis for regression of chemical parameters (Engstrøm et al. 2021; Engstrøm et al. 2023b; Engstrøm et al. 2025a). This is shown for regression of

protein content in wheat in Chapter 4 and for fat content regression in pork bellies in Chapter 6.

It should be noted, however, that these studies compare ResNets for NIR-HSI, RGB, and grayscale images with PLS for NIR spectra. Thus, differences in performance may be due to both the data modality and the models themselves. For example, a more thorough analysis could include a 1D ResNet for NIR spectral analysis. Still, as PLS is the primary tool for analysis of NIR spectra of food (Brereton et al. 2018; Sørensen et al. 2021), and CNNs are the primary tool for image analysis (LeCun et al. 2015), the comparisons can likely be deemed valid.

8.2.2 Learnable Preprocessing

Spectral preprocessing, learnable or manually applied, is highly beneficial when using 2D CNNs for joint spatio-spectral modeling of chemical parameters (Engstrøm et al. 2021; Engstrøm et al. 2023b). Chapter 4 showed how augmenting a 2D CNN with an initial spectral (1D) or spatio-spectral (3D) convolution layer with only a few filters can be used to increase performance on protein content regression in wheat kernels, and we hypothesize that this generalizes to other chemical parameters as well. In particular, the initial layer learns weights much similar to those of a Savitzky-Golay (SG) filter (Savitzky and Golay 1964; Steinier et al. 1972). Thus, CNNs can learn suitable preprocessing directly from the data, contrasting the manual selection of preprocessing for PLS. However, the number of convolution filters and the kernel size must still be chosen manually. Additionally, each filter will output a transformed representation of the input NIR-HSI image. This implies that if n filters are used, then the output of this initial layer will be n times as large as the input NIR-HSI image, which may already be quite large. Therefore, using multiple filters is associated with a significant computational overhead, both in terms of memory and runtime.

Alternatively, as shown in Chapter 4, a spectral preprocessing may be chosen by grid search cross-validation of a PLS model and subsequently applied before feeding the input data to a 2D CNN to yield results comparable to using the initial 1D or 3D convolution layer. This approach decreases the computational overhead of training the CNN, especially since the PLS cross-validation can be carried out with the algorithm presented in Chapter 3. Still, this assumes a correlation between the optimal preprocessing for PLS and CNNs and requires that the optimal preprocessing is included in the grid search. Therefore, it may be more reasonable to have the CNN learn directly from the input data instead of relying on PLS to find a suitable preprocessing. When learning from the data, the studies by Engstrøm et al. (2021) and Engstrøm et al. (2023b) showed that applying a single 1D spectral filter may lead to unstable training. Therefore, a general suggestion is to use a single initial spatio-spectral filter, which offers better stability and does not multiply the

size of the input NIR-HSI image in its output. Indeed, this is the approach taken in the subsequent studies in Chapter 6 and Chapter 7.

In addition to the initial 1D and 3D convolutions, we experimented with so-called downsamplers in Chapter 4. Downsamplers reduce the spectral dimension to one by computing a dot product between a learned vector and every spectrum. By stacking the outputs from multiple downsamplers, we can control the size of the spectral dimension in the output of this layer, not unlike choosing the number of components for a PLS model. Such downsampling has a regularizing effect and may be beneficial in some cases, as shown in 4 and 7. Still, this layer also requires manually choosing the number of filters. Another possible approach is to use many downsamplers and then apply L1 regularization to the output like in sparse autoencoders (Bank et al. 2023). This way, the spectral regularization can be coupled with the loss function for joint optimization of spectral regularization and objective. Either approach, however, requires manual tuning of hyperparameters.

8.2.3 Bulk References

Chapters 4 and 5 studied datasets of NIR-HSI images of bulk grain samples with reference values for grain variety and mean protein content, respectively. To generate enough data to train CNNs, we cropped the images of bulks and the bulks' reference values to their respective image crops. This data inflation strategy is entirely valid for the grain variety dataset as each NIR-HSI image contained exactly one grain type. On the other hand, wheat's protein content is heterogeneously distributed, so the bulk reference values may not reflect the true protein content in the image crops. On average, however, by definition, the bulk reference value will match the true protein content in the image crops. Therefore, we hypothesized that training an accurate regression model with these pairs of image crops and bulk mean reference values would be possible. However, as shown in Chapter 5, the average protein prediction for all image crops belonging to a bulk is not a good estimate of the bulk's mean protein reference value.¹ This is due to a discrepancy between the least squares solution for regression from image crop-wise predictions, respectively, *average* image crop-wise predictions to bulk references. Therefore, bias and scale corrections are needed to accurately estimate bulk mean protein content when pairing subsamples with bulk references. In fact, deep learning models should always have their bias, and sometimes scale, corrected for any task (Igel and Oehmcke 2023).

¹In the analysis by Engström et al. (2023a), all image crops are weighted equally when taking their average predicted protein value. However, they do have different grain density (grain-to-background ratio). Thus, using a grain-density-weighted average would have been more accurate when considering the mean prediction. Still, as the grain was randomly distributed on the images and each bulk yielded more than 1,000 image crops, any errors due to the non-weighted average are assumed to cancel out.

When correcting for the bias and scale, the mean predictions align well with the bulk mean reference values. However, a similar performance can be achieved much faster by applying PLS to the mean spectra of the bulks and their reference values (Engstrøm et al. 2023a). Thus, applying PLS to the mean spectrum is the most sensible choice if the mean of a chemical parameter is of interest and only bulk mean references are available. Additionally, a mean spectrum can be acquired with a spectrometer, even accurately for heterogeneous samples (Tanzilli et al. 2024), avoiding the need for an expensive NIR-HSI platform.

8.2.4 Chemical Maps

However, a NIR-HSI image is required if a chemical parameter’s spatial distribution is to be modeled, that is, chemical map generation. Based on the paragraphs above, it may seem that training a PLS model on a mean spectrum with an associated bulk mean reference can readily be used to generate chemical maps through pixel-wise predictions. As mentioned in Section 1.2, this approach is commonly taken. Chapter 6 studied chemical map generation of fat in pork bellies and showed that pixel-wise predictions may lead to unsatisfactory chemical maps with predictions of physically impossible fat values vastly outside the 0 – 100% range (Engstrøm et al. 2025a). Additionally, there is a discrepancy between the fat predictions on mean spectra and the fat predictions obtained by averaging the chemical maps (this is also noticed by, for example, Barbin et al. 2013). Instead, by augmenting a 2D U-Net with a spectral convolution filter as studied in Chapter 4, chemical maps could be generated directly from the input NIR-HSI image and optimized against a bulk mean reference. This approach handled all the issues mentioned for pixel-wise PLS and showed that joint spatio-spectral analysis can outperform spectral analysis of the spatial distribution of a chemical parameter.

8.2.5 Barley Germination

NIR-HSI images are beneficial for analyzing parameters where both spatial and spectral information are relevant. However, we had yet to find a food quality parameter that could not be somewhat accurately modeled with spectral or spatial information alone, apart from modeling the spatial distribution of a chemical parameter. Modeling barley’s germinative capacity seemed a promising candidate as barley germination relies on both chemical and physical, visible parameters (Orth et al. 2025). Chapter 7 studied this. Unfortunately, the dataset suffered from a low degree of germination (Engstrøm et al. 2025b), making modeling and evaluation unreliable. A larger dataset with more germinated kernels is required to determine whether barley’s germinative capacity can be accurately modeled with NIR-HSI. On the other hand, classification of germinated and non-germinated kernels was possible

with both NIR spectra, RGB images, and NIR-HSI images, and the latter modality gave the highest balanced accuracy. These results corroborate earlier studies where joint spatio-spectral modeling outperformed spectral and spatial modeling alone of parameters where spatial and spectral information are relevant.

8.3 Future Work

In Chapter 2, we introduced the IKPLS implementation that was later augmented to incorporate sample weighting (Becker and Ismail 2016). The fast cross-validation algorithm presented in Chapter 3 does not work with sample weights. Thus, augmenting the algorithm to work with weights is a natural extension. Indeed, such augmentation seems feasible while maintaining proper centering and scaling and without increasing either space or time complexity. In the same spirit, attempts to augment the fast cross-validation algorithm by Liland et al. (2020) with sample weights and standard deviation scaling could be made, both jointly and separately, to decrease the computational cost of cross-validation on wide datasets.

The results achieved for chemical map generation in Chapter 6 yield several interesting directions for future research. As mentioned by Engstrøm et al. (2025a), using U-Net for simultaneous modeling of chemical maps for multiple parameters may enhance results due to cross-learning and the possibility of adding terms in the loss function regarding the interaction between multiple chemical parameters. Coupling this with semantic segmentation to distinguish food samples from the background in NIR-HSI images may yield a complete end-to-end chemical map generation model.

Additionally, the work by Engstrøm et al. (2025a) showed that a CNN can be trained with bulk mean references and relatively few images. This indicates a possibility for training on large NIR-HSI images of entire bulk grain samples, thus circumventing the need to generate image crops, apply mean references to these, and subsequently compensate for the biases as otherwise done in Chapters 4 and 5. In particular, it seems possible that a single model can simultaneously perform instance segmentation to identify individual kernels and subsequently regress the protein content in these, optimized such that the model's mean protein prediction across the entire image matches the bulk reference value. As it is feasible to obtain reference protein values for a few grain kernels, these can be used to evaluate the accuracy of the predictions on a kernel level, not just on a bulk level. With this framework, it may also be possible to train the model on large datasets of individual grains with bulk references and later fine-tune it on a small dataset with kernel-wise reference values.

As previously mentioned, the results in Chapter 7 warrant further research to yield a conclusive result regarding whether NIR-HSI enables accurate mod-

eling of barley’s germinative capacity. The insights from Engstrøm et al. (2025a) regarding using bulk mean references to model the distribution of parameters may also apply to this task, albeit with slight modification. Indeed, if the use of bulk mean references allows accurate modeling, the dataset acquisition process used by Engstrøm et al. (2025b) simplified, reducing both manual labor and the complexity of the image acquisition pipeline. In particular, the following setup is imagined.

1. Acquire images of several bulks of barley kernels once before exposure to moisture.
2. Expose each bulk grain sample to moisture to initialize the germination process.
3. For each bulk, count the number of germinated kernels at discrete intervals, for example, once a day.

Then, for each bulk image, we train a model for simultaneous instance segmentation (localization of individual kernels) and prediction of germinative capacity (how many days until the kernel will sprout) for each kernel. Then, due to step 3, we have the entire discrete distribution of germinative capacities. Therefore, the model can be optimized to minimize the Kullback-Leibler divergence (Kullback and Leibler 1951) between the true distribution and the predicted distribution. Finally, as with the future research idea regarding protein distribution in grain, the model can be fine-tuned and evaluated on a smaller dataset of barley kernels with individual reference values. Unlike the approach taken by Engstrøm et al. (2025b), this approach requires only a single imaging session per grain kernel and does not require the tracking or labeling of each kernel.

In summary, there seems to be ample opportunity for modeling parameters in the future using both bulk and individual reference values.

Chapter 9

Conclusion

THE first part of this thesis has presented two practical open-sourced tools for Python that speed up classical chemometrics. The first tool contains several fast PLS implementations that are particularly useful for datasets with many samples. The second tool implements a set of exceptionally fast algorithms that can speed up cross-validation for PLS and many other traditional machine learning and chemometric models.

The second part of this thesis has investigated food analysis through applications of NIR-HSI images for PLS- and CNN-based modeling. Importantly, studies showed how extending any preexisting 2D CNN with an initial, dedicated spectral convolution layer significantly enhances its ability to predict chemical parameters in NIR-HSI images by learning spectral preprocessing. Furthermore, compared to spatial and spectral analysis, joint spatio-spectral analysis of entire NIR-HSI images improves the modeling of parameters where both chemical and visual information are relevant.

Additionally, while analyzing spectral information alone is sufficient and recommended for accurate modeling of chemical parameters, the joint spatio-spectral analysis yields significantly better results than pixel-wise spectral analysis when modeling the spatial distribution of chemical parameters to generate chemical maps. Indeed, this thesis presented a novel approach to chemical map generation, requiring only a bulk mean reference value to estimate the chemical map while considering spatial dependencies. Thus, this study showed that NIR-HSI images can be used to model the spatial distribution of chemical parameters throughout the images.

Studies also investigated pairing bulk mean reference values with subsamples for dataset inflation. Generally, this approach is feasible and may yield accurate predictions of the bulk mean references. However, a bias and scale correction must be applied post-training when this approach is applied to regression analysis on heterogeneous distributions.

The last study in this thesis investigated barley’s germinative capacity as a candidate for a food quality parameter requiring NIR-HSI for accurate

modeling. However, it was inconclusive, primarily due to a fault in the dataset acquisition that led to a limited amount of germinated kernels. Therefore, it remains unknown whether a parameter exists that can not be analyzed decently accurately with spatial or spectral information but can be with joint spatio-spectral analysis.

This thesis concludes that joint spatio-spectral analysis of NIR-HSI images is beneficial and should be considered over spectral or visual analysis alone in two cases. First, it improves the modeling of parameters where both chemical and visual information are relevant. Second, it improves the modeling of spatial distributions of chemical parameters compared to pixel-wise spectral analysis.

Finally, based on the results achieved, this thesis offered perspectives on future research of NIR-HSI applications for food analysis. In particular, it may be possible to accurately model both discrete and continuous distributions of quality parameters in heterogeneous bulk samples of grain using cheap-to-obtain bulk mean reference values.

Bibliography

- Acquarelli, J., T. van Laarhoven, J. Gerretzen, T. N. Tran, L. M. C. Buydens, and E. Marchiori (2017). “Convolutional neural networks for vibrational spectroscopic data analysis”. In: *Analytica chimica acta* 954, pp. 22–31. DOI: [10.1016/j.aca.2016.12.010](https://doi.org/10.1016/j.aca.2016.12.010).
- Adesokan, M., B. Otegbayo, E. O. Alamu, M. A. Olutoyin, and B. Maziya-Dixon (2024). “Evaluating the dry matter content of raw yams using hyperspectral imaging spectroscopy and machine learning”. In: *Journal of Food Composition and Analysis* 135, p. 106692. ISSN: 0889-1575. DOI: [10.1016/j.jfca.2024.106692](https://doi.org/10.1016/j.jfca.2024.106692).
- Albano-Gaglio, M., P. Mishra, S. W. Erasmus, J. F. Tejada, A. Brun, B. Marcos, C. Zomeño, and M. Font-i-Furnols (2025). “Visible and near-infrared spectral imaging combined with robust regression for predicting firmness, fatness, and compositional properties of fresh pork bellies”. In: *Meat Science* 219, p. 109645. DOI: [10.1016/j.meatsci.2024.109645](https://doi.org/10.1016/j.meatsci.2024.109645).
- Alin, A. (2009). “Comparison of pls algorithms when number of objects is much larger than number of variables”. In: *Statistical Papers* 50, pp. 711–720. DOI: [10.1007/s00362-009-0251-7](https://doi.org/10.1007/s00362-009-0251-7).
- Amigo, J. M., I. Martí, and A. Gowen (2013). “Chapter 9 - Hyperspectral Imaging and Chemometrics: A Perfect Combination for the Analysis of Food Structure, Composition and Quality”. In: *Chemometrics in Food Chemistry*. Ed. by F. Marini. Vol. 28. Data Handling in Science and Technology. Elsevier, pp. 343–370. DOI: [10.1016/B978-0-444-59528-7.00009-0](https://doi.org/10.1016/B978-0-444-59528-7.00009-0).
- Andersson, M. (2009). “A comparison of nine pls1 algorithms”. In: *Journal of Chemometrics* 23.10, pp. 518–529. DOI: [10.1002/cem.1248](https://doi.org/10.1002/cem.1248).
- Ansel, J., E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, et al. (2024). “Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation”. In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 929–947. DOI: [10.1145/3620665.3640366](https://doi.org/10.1145/3620665.3640366).
- Arngren, M., P. W. Hansen, B. Eriksen, J. Larsen, and R. Larsen (2011). “Analysis of pregerminated barley using hyperspectral image analysis”.

- In: *Journal of agricultural and food chemistry* 59.21, pp. 11385–11394. DOI: [10.1021/jf202122y](https://doi.org/10.1021/jf202122y).
- Babuška, I. (1968). “Numerical stability in mathematical analysis.” In: *IFIP Congress (1)*. Vol. 68, pp. 11–23.
- Bank, D., N. Koenigstein, and R. Giryes (2023). “Autoencoders”. In: *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pp. 353–374. DOI: [10.1007/978-3-031-24628-9_16](https://doi.org/10.1007/978-3-031-24628-9_16).
- Barbin, D. F., G. ElMasry, D.-W. Sun, and P. Allen (2013). “Non-destructive determination of chemical composition in intact and minced pork using near-infrared hyperspectral imaging”. In: *Food Chemistry* 138.2, pp. 1162–1171. ISSN: 0308-8146. DOI: [10.1016/j.foodchem.2012.11.120](https://doi.org/10.1016/j.foodchem.2012.11.120).
- Barker, M. and W. Rayens (2003). “Partial least squares for discrimination”. In: *Journal of Chemometrics* 17.3, pp. 166–173. DOI: [10.1002/cem.785](https://doi.org/10.1002/cem.785).
- Barnes, R. J., M. S. Dhanoa, and S. J. Lister (1989). “Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra”. In: *Applied Spectroscopy* 43.5, pp. 772–777. DOI: [10.1366/0003702894202201](https://doi.org/10.1366/0003702894202201).
- Becker, J.-M. and I. R. Ismail (2016). “Accounting for sampling weights in PLS path modeling: Simulations and empirical examples”. In: *European Management Journal* 34.6, pp. 606–617. DOI: [10.1016/j.emj.2016.06.009](https://doi.org/10.1016/j.emj.2016.06.009).
- Björck, Å. and U. G. Indahl (2017). “Fast and stable partial least squares modelling: A benchmark study with theoretical comments”. In: *Journal of Chemometrics* 31.8, e2898. DOI: [10.1002/cem.2898](https://doi.org/10.1002/cem.2898).
- Brereton, R. G., J. Jansen, J. Lopes, F. Marini, A. Pomerantsev, O. Rodionova, J. M. Roger, B. Walczak, and R. Tauler (2018). “Chemometrics in analytical chemistry—part II: modeling, validation, and applications”. In: *Analytical and bioanalytical chemistry* 410, pp. 6691–6704. DOI: [10.1007/s00216-018-1283-4](https://doi.org/10.1007/s00216-018-1283-4).
- Brereton, R. G. and G. R. Lloyd (2014). “Partial least squares discriminant analysis: taking the magic away”. In: *Journal of Chemometrics* 28.4, pp. 213–225. DOI: [10.1002/cem.2609](https://doi.org/10.1002/cem.2609).
- Burger, J. and P. Geladi (2006). “Hyperspectral NIR imaging for calibration and prediction: a comparison between image and spectrometer data for studying organic and biological samples”. In: *Analyst* 131 (10), pp. 1152–1160. DOI: [10.1039/B605386F](https://doi.org/10.1039/B605386F).
- Cai, M., X. Li, J. Liang, M. Liao, and Y. Han (2024). “An effective deep learning fusion method for predicting the TVB-N and TVC contents of chicken breasts using dual hyperspectral imaging systems”. In: *Food Chemistry* 456, p. 139847. ISSN: 0308-8146. DOI: [10.1016/j.foodchem.2024.139847](https://doi.org/10.1016/j.foodchem.2024.139847).
- Cruz-Tirado, J. P., J. M. Amigo, and D. F. Barbin (2023). “Determination of protein content in single black fly soldier (*Hermetia illucens* L.) larvae by near infrared hyperspectral imaging (NIR-HSI) and chemometrics”. In: *Food Control* 143, p. 109266. ISSN: 0956-7135. DOI: [10.1016/j.foodcont.2022.109266](https://doi.org/10.1016/j.foodcont.2022.109266).

- Cui, C. and T. Fearn (2018). “Modern practical convolutional neural networks for multivariate regression: Applications to NIR calibration”. In: *Chemometrics and Intelligent Laboratory Systems* 182, pp. 9–20. DOI: [10.1016/j.chemolab.2018.07.008](https://doi.org/10.1016/j.chemolab.2018.07.008).
- Dayal, B. S. and J. F. MacGregor (1997). “Improved pls algorithms”. In: *Journal of Chemometrics* 11.1, pp. 73–85. DOI: [10.1002/\(SICI\)1099-128X\(199701\)11:1%3C73::AID-CEM435%3E3.0.CO;2-%23](https://doi.org/10.1002/(SICI)1099-128X(199701)11:1%3C73::AID-CEM435%3E3.0.CO;2-%23).
- Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. (2020). “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929*. DOI: [10.48550/arXiv.2010.11929](https://doi.org/10.48550/arXiv.2010.11929).
- Dreier, E. S., K. M. Sørensen, T. Lund-Hansen, B. M. Jespersen, and K. S. Pedersen (2022). “Hyperspectral imaging for classification of bulk grain samples with deep convolutional neural networks”. In: *Journal of Near Infrared Spectroscopy* 30.3, pp. 107–121. DOI: [10.1177/09670335221078356](https://doi.org/10.1177/09670335221078356).
- Ebrahimi, S., R. Pourdarbani, S. Sabzi, M. H. Rohban, and J. I. Arribas (2023). “From harvest to market: Non-destructive bruise detection in kiwifruit using convolutional neural networks and hyperspectral imaging”. In: *Horticulturae* 9.8, p. 936. DOI: [10.3390/horticulturae9080936](https://doi.org/10.3390/horticulturae9080936).
- ElMasry, G., D.-W. Sun, and P. Allen (2013). “Chemical-free assessment and mapping of major constituents in beef using hyperspectral imaging”. In: *Journal of Food Engineering* 117.2, pp. 235–246. ISSN: 0260-8774. DOI: [10.1016/j.jfoodeng.2013.02.016](https://doi.org/10.1016/j.jfoodeng.2013.02.016).
- Engstrøm, O.-C. G., M. Albano-Gaglio, E. S. Dreier, Y. Bouzembrak, M. Fonti-Furnols, P. Mishra, and K. S. Pedersen (2025a). *Transforming Hyperspectral Images Into Chemical Maps: An End-to-End Deep Learning Approach*. DOI: [10.48550/arXiv.2504.14131](https://doi.org/10.48550/arXiv.2504.14131). arXiv: [2504.14131 \[cs.CV\]](https://arxiv.org/abs/2504.14131).
- Engstrøm, O.-C. G., E. S. Dreier, B. M. Jespersen, and K. S. Pedersen (2023a). *Analyzing near-infrared hyperspectral imaging for protein content regression and grain variety classification using bulk references and varying grain-to-background ratios*. DOI: [10.48550/arXiv.2311.04042](https://doi.org/10.48550/arXiv.2311.04042). arXiv: [2311.04042 \[cs.CV\]](https://arxiv.org/abs/2311.04042).
- (2023b). “Improving deep learning on hyperspectral images of grain by incorporating domain knowledge from chemometrics.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. © 2023 IEEE. Reprinted, with permission, from Ole-Christian Galbo Engstrøm, Erik Schou Dreier, Birthe Møller Jespersen, and Kim Steenstrup Pedersen, Improving Deep Learning on Hyperspectral Images of Grain by Incorporating Domain Knowledge from Chemometrics, 2023., pp. 485–494. DOI: [10.1109/ICCVW60793.2023.00055](https://doi.org/10.1109/ICCVW60793.2023.00055).
- (2024). “Ikpls: improved kernel partial least squares and fast cross-validation algorithms for python with cpu and gpu implementations using numpy and

- jax". In: *Journal of Open Source Software* 9.99, p. 6533. DOI: [10.21105/joss.06533](https://doi.org/10.21105/joss.06533).
- Engstrøm, O.-C. G., E. S. Dreier, B. M. Jespersen, and K. S. Pedersen (2025b). *A Time Series Dataset of NIR Spectra and RGB and NIR-HSI Images of the Barley Germination Process*. DOI: [10.48550/arXiv.2504.16658](https://doi.org/10.48550/arXiv.2504.16658). arXiv: [2504.16658](https://arxiv.org/abs/2504.16658) [cs.CV].
- Engstrøm, O.-C. G., E. S. Dreier, and K. S. Pedersen (2021). "Predicting protein content in grain using hyperspectral deep learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1372–1380. DOI: [10.1109/ICCVW54120.2021.00158](https://doi.org/10.1109/ICCVW54120.2021.00158).
- Engstrøm, O.-C. G. and M. H. Jensen (2025). "Fast partition-based cross-validation with centering and scaling for $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ ". In: *Journal of Chemometrics* 39.3, e70008. DOI: [10.1002/cem.70008](https://doi.org/10.1002/cem.70008).
- European Brewery Convention (1997). *Germinative Energy of Barley*. Technical report. EBC. URL: <https://brewup.eu/ebc-analytica/category/chemical-physical/barley> (visited on 05/26/2025).
- Florián-Huamán, J., J. P. Cruz-Tirado, D. F. Barbin, and R. Siche (2022). "Detection of nutshells in cumin powder using NIR hyperspectral imaging and chemometrics tools". In: *Journal of Food Composition and Analysis* 108, p. 104407. ISSN: 0889-1575. DOI: [10.1016/j.jfca.2022.104407](https://doi.org/10.1016/j.jfca.2022.104407).
- FOSS Analytical A/S (2025). *Infratec™ NOVA*. URL: <https://www.fossanalytics.com/en/products/infratec-nova> (visited on 05/20/2025).
- Gowen, A., C. P. O'Donnell, P.J. Cullen, G. Downey, and J. M. Frias (2007). "Hyperspectral imaging – an emerging process analytical tool for food quality and safety control". In: *Trends in Food Science & Technology* 18.12, pp. 590–598. ISSN: 0924-2244. DOI: [10.1016/j.tifs.2007.06.001](https://doi.org/10.1016/j.tifs.2007.06.001).
- Gubler, F., A. A. Millar, and J. V. Jacobsen (2005). "Dormancy release, ABA and pre-harvest sprouting". In: *Current Opinion in Plant Biology* 8.2, pp. 183–187. ISSN: 1369-5266. DOI: [10.1016/j.pbi.2005.01.011](https://doi.org/10.1016/j.pbi.2005.01.011).
- Hastie, T., R. Tibshirani, and J. H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer. DOI: [10.1198/jasa.2004.s339](https://doi.org/10.1198/jasa.2004.s339).
- He, H.-J., Y. Wang, X. Ou, H. Ma, H. Liu, and J. Yan (2023). "Rapid determination of chemical compositions in chicken flesh by mining hyperspectral data". In: *Journal of Food Composition and Analysis* 116, p. 105069. ISSN: 0889-1575. DOI: [10.1016/j.jfca.2022.105069](https://doi.org/10.1016/j.jfca.2022.105069).
- He, K., X. Zhang, S. Ren, and J. Sun (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034. DOI: [10.1109/ICCV.2015.123](https://doi.org/10.1109/ICCV.2015.123).
- (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).

- Helin, R., U. G. Indahl, O. Tomic, and K. H. Liland (2022). “On the possible benefits of deep learning for spectral preprocessing”. In: *Journal of Chemometrics* 36.2, e3374. DOI: [10.1002/cem.3374](https://doi.org/10.1002/cem.3374).
- Hotelling, H. (1933). “Analysis of a complex of statistical variables into principal components”. In: *Journal of Educational Psychology* 24.6, p. 417. DOI: [10.1037/h0071325](https://doi.org/10.1037/h0071325).
- (1957). “The relations of the newer multivariate statistical methods to factor analysis”. In: *British Journal of Statistical Psychology* 10.2, pp. 69–79. DOI: [10.1111/j.2044-8317.1957.tb00179.x](https://doi.org/10.1111/j.2044-8317.1957.tb00179.x).
- Igel, C. and S. Oehmcke (2023). “Remember to correct the bias when using deep learning for regression!” In: *KI-Künstliche Intelligenz* 37.1, pp. 33–40. DOI: [10.1007/s13218-023-00801-0](https://doi.org/10.1007/s13218-023-00801-0).
- Ishikawa, D., M. Ishigaki, and A. Gowen (2021). “NIR Imaging”. In: *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*. Ed. by Y. Ozaki, C. Huck, S. Tsuchikawa, and S. B. Engelsen. Singapore: Springer Singapore, pp. 517–551. ISBN: 978-981-15-8648-4. DOI: [10.1007/978-981-15-8648-4_22](https://doi.org/10.1007/978-981-15-8648-4_22).
- Kahan, W. (1965). “Pracniques: further remarks on reducing truncation errors”. In: *Communications of the ACM* 8.1, p. 40. DOI: [10.1145/363707.363723](https://doi.org/10.1145/363707.363723).
- Kamruzzaman, M., G. ElMasry, D.-W. Sun, and P. Allen (2012). “Non-destructive prediction and visualization of chemical composition in lamb meat using NIR hyperspectral imaging and multivariate regression”. In: *Innovative Food Science & Emerging Technologies* 16, pp. 218–226. ISSN: 1466-8564. DOI: [10.1016/j.ifset.2012.06.003](https://doi.org/10.1016/j.ifset.2012.06.003).
- Kamruzzaman, M., Y. Makino, and S. Oshita (2016). “Rapid and non-destructive detection of chicken adulteration in minced beef using visible near-infrared hyperspectral imaging and machine learning”. In: *Journal of Food Engineering* 170, pp. 8–15. ISSN: 0260-8774. DOI: [10.1016/j.jfoodeng.2015.08.023](https://doi.org/10.1016/j.jfoodeng.2015.08.023).
- Kendall, M. G. (1957). *A course in multivariate analysis*. Hafner Publishing Company New York.
- King, G. and L. Zeng (2001). “Logistic regression in rare events data”. In: *Political analysis* 9.2, pp. 137–163. DOI: [10.1093/oxfordjournals.pan.a004868](https://doi.org/10.1093/oxfordjournals.pan.a004868).
- Kingma, D. P. and J. Ba (2017). *Adam: A method for stochastic optimization*. DOI: [10.48550/arXiv.1412.6980](https://doi.org/10.48550/arXiv.1412.6980). arXiv: [1412.6980](https://arxiv.org/abs/1412.6980) [cs.LG].
- Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems* 25.
- Kullback, S. and R. A. Leibler (1951). “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1, pp. 79–86. DOI: [10.1214/aoms/1177729694](https://doi.org/10.1214/aoms/1177729694).

- LeCun, Y., Y. Bengio, and G. Hinton (2015). “Deep learning”. In: *nature* 521.7553, pp. 436–444. DOI: [10.1038/nature14539](https://doi.org/10.1038/nature14539).
- LeCun, Y., B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel (1989). “Handwritten digit recognition with a back-propagation network”. In: *Advances in neural information processing systems* 2.
- LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (2002). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.
- Liland, K. H., P. Stefansson, and U. G. Indahl (2020). “Much faster cross-validation in pls-r modelling by avoiding redundant calculations”. In: *Journal of Chemometrics* 34.3, e3201. DOI: [10.1002/cem.3201](https://doi.org/10.1002/cem.3201).
- Ma, C., Z. Ren, Z. Zhang, J. Du, C. Jin, and X. Yin (2021). “Development of simplified models for nondestructive testing of rice (with husk) protein content using hyperspectral imaging technology”. In: *Vibrational Spectroscopy* 114, p. 103230. ISSN: 0924-2031. DOI: [10.1016/j.vibspec.2021.103230](https://doi.org/10.1016/j.vibspec.2021.103230).
- Manley, M. and P. J. Williams (2021). “Applications: Food Science”. In: *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*. Ed. by Y. Ozaki, C. Huck, S. Tsuchikawa, and S. B. Engelsen. Singapore: Springer Singapore, pp. 347–359. ISBN: 978-981-15-8648-4. DOI: [10.1007/978-981-15-8648-4_15](https://doi.org/10.1007/978-981-15-8648-4_15).
- Medina-García, M., J. M. Amigo, M. A. Martínez-Domingo, E. M. Valero, and A. M. Jiménez-Carvelo (2025). “Strategies for analysing hyperspectral imaging data for food quality and safety issues – A critical review of the last 5 years”. In: *Microchemical Journal*, p. 113994. ISSN: 0026-265X. DOI: [10.1016/j.microc.2025.113994](https://doi.org/10.1016/j.microc.2025.113994).
- Mishra, P., D. Passos, F. Marini, J. Xu, J. M. Amigo, A. Gowen, J. J. Jansen, A. Biancolillo, J. M. Roger, D. N. Rutledge, and A. Nordon (2022). “Deep learning for near-infrared spectral data modelling: Hypes and benefits”. In: *TrAC Trends in Analytical Chemistry* 157, p. 116804. ISSN: 0165-9936. DOI: [10.1016/j.trac.2022.116804](https://doi.org/10.1016/j.trac.2022.116804).
- National Institute of Standards and Technology (2013a). *WEIGHTSD – Weighted Standard Deviation*. Technical report. NIST. URL: <https://www.itl.nist.gov/div898/software/dataplot/refman2/ch2/weightsd.pdf> (visited on 04/30/2025).
- (2013b). *WEIGHTVAR – Weighted Variance*. Technical report. NIST. URL: <https://www.itl.nist.gov/div898/software/dataplot/refman2/ch2/weightvar.pdf> (visited on 04/30/2025).
- Neumaier, A. (1974). “Rundlingsfehleranalyse einiger Verfahren zur Summation endlicher Summen”. In: *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)* 54, pp. 39–51. DOI: [10.1515/9783112550045-008](https://doi.org/10.1515/9783112550045-008).
- Orth, S. H., G. P. Fox, M. Manley, and S. Hayward (2025). “Current Detection Methods and Future Trends for Barley Pre-Harvest Sprouting, is NIR the Preferred Method of the Future?” In: *Journal of the American Society of Brewing Chemists*, pp. 1–12. DOI: [10.1080/03610470.2025.2489331](https://doi.org/10.1080/03610470.2025.2489331).

- Orth, S. H., F. Marini, G. P. Fox, M. Manley, and S. Hayward (2023). “Multi-block spectral imaging for identification of pre-harvest sprouting in *Hordeum vulgare*”. In: *Microchemical Journal* 191, p. 108742. ISSN: 0026-265X. DOI: [10.1016/j.microc.2023.108742](https://doi.org/10.1016/j.microc.2023.108742).
- Osborne, B. G. (2006). “Near-Infrared Spectroscopy in Food Analysis”. In: *Encyclopedia of Analytical Chemistry*. John Wiley & Sons, Ltd. ISBN: 9780470027318. DOI: [10.1002/9780470027318.a1018](https://doi.org/10.1002/9780470027318.a1018).
- Otsu, N. (1979). “A threshold selection method from gray-level histograms”. In: *IEEE Transaction on Systems, Man and Cybernetics* SMC-9.1, pp. 62–66. DOI: [10.1109/TSMC.1979.4310076](https://doi.org/10.1109/TSMC.1979.4310076).
- Pearson, K. (1901). “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572. DOI: [10.1080/14786440109462720](https://doi.org/10.1080/14786440109462720).
- Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.
- Polder, G. and A. Gowen (2020). “The hype in spectral imaging”. In: *Journal of Spectral Imaging* 9. DOI: [10.1255/jsi.2020.a4](https://doi.org/10.1255/jsi.2020.a4).
- Pourdarbani, R., S. Sabzi, R. Zohrabi, G. García-Mateos, R. Fernandez-Beltran, J. M. M. Martínez, and M. H. Rohban (Oct. 2023). “Comparison of 2D and 3D convolutional neural networks in hyperspectral image analysis of fruits applied to orange bruise detection”. In: *Journal of Food Science* 88. DOI: [10.1111/1750-3841.16801](https://doi.org/10.1111/1750-3841.16801).
- Rännar, S., P. Geladi, F. Lindgren, and S. Wold (1995). “A PLS kernel algorithm for data sets with many variables and few objects. Part II: Cross-validation, missing data and examples”. In: *Journal of Chemometrics* 9.6, pp. 459–470. DOI: [10.1002/cem.1180090604](https://doi.org/10.1002/cem.1180090604).
- Rännar, S., F. Lindgren, P. Geladi, and S. Wold (1994). “A PLS kernel algorithm for data sets with many variables and fewer objects. Part 1: Theory and algorithm”. In: *Journal of Chemometrics* 8.2, pp. 111–125. DOI: [10.1002/cem.1180080204](https://doi.org/10.1002/cem.1180080204).
- Rinnan, Å., F. van den Berg, and S. B. Engelsen (2009). “Review of the most common pre-processing techniques for near-infrared spectra”. In: *TrAC Trends in Analytical Chemistry* 28.10, pp. 1201–1222. DOI: [10.1016/j.trac.2009.07.007](https://doi.org/10.1016/j.trac.2009.07.007).
- Ronneberger, O., P. Fischer, and T. Brox (2015). “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III* 18. Springer, pp. 234–241. DOI: [10.1007/978-3-319-24574-4_28](https://doi.org/10.1007/978-3-319-24574-4_28).
- Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpthy, A. Khosla, M. Bernstein, et al. (2015). “Imagenet large scale vi-

- sual recognition challenge". In: *International Journal of Computer Vision* 115, pp. 211–252.
- Savitzky, A. and M. J. E. Golay (1964). "Smoothing and differentiation of data by simplified least squares procedures." In: *Analytical Chemistry* 36.8, pp. 1627–1639. DOI: [10.1021/ac60214a047](https://doi.org/10.1021/ac60214a047).
- Shi, T., Y. Gao, J. Song, M. Ao, X. Hu, W. Yang, W. Chen, Y. Liu, and H. Feng (2024). "Using VIS-NIR hyperspectral imaging and deep learning for non-destructive high-throughput quantification and visualization of nutrients in wheat grains". In: *Food Chemistry* 461, p. 140651. ISSN: 0308-8146. DOI: [10.1016/j.foodchem.2024.140651](https://doi.org/10.1016/j.foodchem.2024.140651).
- Sjöström, M., S. Wold, and B. Söderström (1986). *Pls discriminant plots*. Elsevier, pp. 461–470.
- Sørensen, K. M., F. van den Berg, and S. B. Engelsen (2021). "Nir data exploration and regression by chemometrics—a primer". In: *Near-Infrared Spectroscopy: Theory, Spectral Analysis, Instrumentation, and Applications*, pp. 127–189. DOI: [10.1007/978-981-15-8648-4_7](https://doi.org/10.1007/978-981-15-8648-4_7).
- Stähle, L. and S. Wold (1987). "Partial least squares analysis with cross-validation for the two-class problem: a Monte Carlo study". In: *Journal of Chemometrics* 1.3, pp. 185–196. DOI: [10.1002/cem.1180010306](https://doi.org/10.1002/cem.1180010306).
- Steinier, J., Y. Termonia, and J. Deltour (1972). "Smoothing and differentiation of data by simplified least square procedure". In: *Analytical Chemistry* 44.11, pp. 1906–1909. DOI: [10.1021/ac60319a045](https://doi.org/10.1021/ac60319a045).
- Stone, M. (1974). "Cross-validatory choice and assessment of statistical predictions". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 36.2, pp. 111–133. DOI: [10.1111/j.2517-6161.1974.tb00994.x](https://doi.org/10.1111/j.2517-6161.1974.tb00994.x).
- Su, Z., C. Zhang, T. Yan, J. Zhu, Y. Zeng, X. Lu, P. Gao, L. Feng, L. He, and L. Fan (2021). "Application of hyperspectral imaging for maturity and soluble solids content determination of strawberry with deep learning approaches". In: *Frontiers in Plant Science* 12, p. 736334. DOI: [10.3389/fpls.2021.736334](https://doi.org/10.3389/fpls.2021.736334).
- Tanzilli, D., M. Cocchi, J. M. Amigo, A. D'Alessandro, and L. Strani (2024). "Does hyperspectral always matter? A critical assessment of near infrared versus hyperspectral near infrared in the study of heterogeneous samples". In: *Current Research in Food Science* 9, p. 100813. DOI: [10.1016/j.crfs.2024.100813](https://doi.org/10.1016/j.crfs.2024.100813).
- Welford, B. P. (1962). "Note on a method for calculating corrected sums of squares and products". In: *Technometrics* 4.3, pp. 419–420. DOI: [10.1080/00401706.1962.10490022](https://doi.org/10.1080/00401706.1962.10490022).
- Wieme, J., K. Mollazade, I. Malounas, M. Zude-Sasse, M. Zhao, A. Gowen, D. Argyropoulos, S. Fountas, and J. Van Beek (2022). "Application of hyperspectral imaging systems and artificial intelligence for quality assessment of fruit, vegetables and mushrooms: A review". In: *Biosystems Engineering* 222, pp. 156–176. ISSN: 1537-5110. DOI: [10.1016/j.biosystemseng.2022.07.013](https://doi.org/10.1016/j.biosystemseng.2022.07.013).

- Wold, H. (1966). “Estimation of principal components and related models by iterative least squares”. In: *Multivariate analysis*, pp. 391–420.
- Wold, S. (1995). “Chemometrics; what do we mean with it, and what do we want from it?” In: *Chemometrics and intelligent laboratory systems* 30.1, pp. 109–115. DOI: [10.1016/0169-7439\(95\)00042-9](https://doi.org/10.1016/0169-7439(95)00042-9).
- Wold, S., M. Sjöström, and L. Eriksson (2001). “Pls-regression: a basic tool of chemometrics”. In: *Chemometrics and Intelligent Laboratory Systems* 58.2, pp. 109–130. DOI: [10.1016/S0169-7439\(01\)00155-1](https://doi.org/10.1016/S0169-7439(01)00155-1).
- Xi, Q., Q. Chen, W. Ahmad, J. Pan, S. Zhao, Y. Xia, Q. Ouyang, and Q. Chen (2025). “Quantitative analysis and visualization of chemical compositions during shrimp flesh deterioration using hyperspectral imaging: A comparative study of machine learning and deep learning models”. In: *Food Chemistry* 481, p. 143997. ISSN: 0308-8146. DOI: [10.1016/j.foodchem.2025.143997](https://doi.org/10.1016/j.foodchem.2025.143997).
- Zhu, H., A. Gowen, H. Feng, K. Yu, and J.-L. Xu (2020). “Deep spectral-spatial features of near infrared hyperspectral images for pixel-wise classification of food products”. In: *Sensors* 20.18, p. 5322. DOI: [10.3390/s20185322](https://doi.org/10.3390/s20185322).

Part IV

Appendices

Appendix A

Technical Report: Bulk References

The technical report by Engstrøm et al. (2023a) follows. Two typos should be addressed: In Figure 7, subfigure e is the validation set, and f is the test set.

TECHNICAL REPORT**Near-Infrared Hyperspectral Imaging for Wheat Protein Regression and Grain Variety Classification**

Analyzing Near-Infrared Hyperspectral Imaging for Protein Content Regression and Grain Variety Classification Using Bulk References and Varying Grain-to-Background Ratios

Ole-Christian Galbo Engstrøm^{1,2,3} | Erik Schou Dreier² |
Birthe Møller Jespersen³ | Kim Steenstrup Pedersen^{1,4}

¹Department of Computer Science (DIKU), University of Copenhagen, Denmark.

²FOSS Analytical A/S, Denmark.

³Department of Food Science (UCPH FOOD), University of Copenhagen, Denmark.

⁴Natural History Museum of Denmark (NHMD), University of Copenhagen, Denmark.

Correspondence

Ole-Christian Galbo Engstrøm, Department of Computer Science, University of Copenhagen Faculty of Science, Universitetsparken 1, 2100 Copenhagen, Denmark.
Email: ocge@foss.dk

Present address

R&D, Foss Analytical A/S, Nils Foss Allé 1, 3400 Hillerød, Denmark.

Funding information

FOSS Analytical A/S; Innovation Fund Denmark, Grant Number: 1044-00108B.

Based on previous work, we assess the use of NIR-HSI images for calibrating models on two datasets, focusing on protein content regression and grain variety classification. Limited reference data for protein content is expanded by subsampling and associating it with the bulk sample. However, this method introduces significant biases due to skewed leptokurtic prediction distributions, affecting both PLS-R and deep CNN models. We propose adjustments to mitigate these biases, improving mean protein reference predictions. Additionally, we investigate the impact of grain-to-background ratios on both tasks. Higher ratios yield more accurate predictions, but including lower-ratio images in calibration enhances model robustness for such scenarios.

Keywords — bulk references, near-infrared hyperspectral imaging, deep learning, protein regression, grain classification.

Abbreviations: Acc, Accuracy; CNN, Convolutional Neural Network; CV, Cross-Validation; HSI, Hyperspectral Imaging; NIR, Near-Infrared; OLS-R, Ordinary Least Squares Regression; PLS-DA, Partial Least Squares Discriminant Analysis; PLS-R, Partial Least Squares Regression; RGB, Red, Green, and Blue; RMSE, Root Mean Squared Error; SEM, Standard Error of The Mean; SG, Savitzky-Golay; SNV, Standard Normal Variate; sYX, Standard Error of Estimate.

1 | INTRODUCTION

Hyperspectral imaging (HSI) is an imaging technique that extracts spatial and spectral features where each pixel contains an entire wavelength spectrum. The hyperspectral image can be analyzed with deep learning models for image analysis where traditionally convolutional neural networks (CNNs) excel [21, 23]. Alternatively, from relevant spatial locations, spectral information can be extracted from the hyperspectral image for subsequent analysis with chemometric methods such as partial least-squares [29], which can be implemented for either regression (PLS-R) [30, 31] or classification through discriminant analysis (PLS-DA) [3].

This work concerns near-infrared hyperspectral imaging (NIR-HSI) for protein content regression analysis of wheat grain kernels and grain variety classification for wheat and rye grain kernels. Protein content regression is a typical chemometric problem that lends itself naturally to PLS-R due to the linear relationship between spectral absorbance in the near-infrared (NIR) wavelengths and protein content [19]. However, deep learning for image analysis performs well within agricultural domains [5, 16, 17, 18, 24]. Grain variety classification can be solved with spectral features using PLS-DA or PLS2-DA or spatial features alone using CNNs, but simultaneously utilizing spectral and spatial features will increase performance [12, 11]. Recently, CNNs utilizing the entire spatio-spectral data cube have been applied to grain quality analysis [11, 12, 13] with Engstrøm *et al.* [13] designing domain-specific CNNs for protein content prediction on NIR-HSI images of bulk wheat kernel samples and Dreier *et al.* [11] designing CNNs for grain variety classification. Engstrøm *et al.* [12] improve upon both results and compare them with those achievable by PLS-R, PLS-DA, and PLS2-DA, respectively.

Engstrøm *et al.* [12] show that a domain-specific CNN can outperform a PLS-R model - even when the latter is subject to an extensive search for the best preprocessing method. However, linear models such as PLS-R have a significant advantage over deep learning models such as CNNs because they can be calibrated on significantly less data. In comparison, deep learning methods require substantial amounts of data to converge meaningfully. Obtaining reference values for the images can be a costly and time-consuming process. It is particularly costly to obtain accurate reference values for the protein contents of individual grain kernels as the protein content may vary by several percentage points between each kernel - even when they originate from the same ear of the plant. Data augmentation techniques can be applied to inflate the training data, thus mitigating the need to analyze the protein content of every grain kernel. For protein content regression, one such approach is to measure the mean protein content of an entire bulk sample and then use the mean protein value as the reference for each bulk subsample originating from that bulk sample. Unless the bulk sample consists of grain kernels with identical protein values, the mean protein value of the bulk sample is unlikely to be equal to the mean protein value of the bulk subsamples, yielding inaccurate reference values for the training data. However, the inaccuracies will be two-sided; on average, the mean reference value will be correct per definition. Engstrøm *et al.* [12, 13] have shown that this approach allows for meaningful convergence of both CNNs and PLS-R.

Dreier *et al.* [11] show that models for grain variety classification trained on images with a high grain density, that is the grain-to-background ratio, (≥ 0.5) fail to accurately predict on images with a lower grain density (< 0.5) than that which they were calibrated on. Consequently, it is interesting to see how grain variety classification models perform on a lower grain density when calibrated on these. Engstrøm *et al.* [13] use a grain density of 0.1 for models calibrated to regress protein content and models calibrated to classify grain variety alike.

In this work, we delve deep into the results achieved by Engstrøm *et al.* [12]. We use statistical approaches to investigate the implications of using the bulk mean protein value as a reference for subsamples when calibrating regression models, revealing significant biases in the predictions of both CNNs and PLS-R models due to the shapes of the predicted protein distributions. We also offer two approaches for the mitigation of these biases - one approach

for PLS-R only and one approach for both CNNs and PLS-R. Additionally, we analyze the effects of grain density for both protein content regression using the mean bulk reference and grain variety classification. We show that for both tasks, a higher grain density is preferable, but in the case of grain variety classification, calibrating the model with a mixture of low and high grain density samples will allow it to predict much more accurately the low grain density samples while achieving accuracy on high grain density samples comparable with previous work.

This report presents the dataset and models in Section 2 and a nomenclature in Section 3. In Section 4, we analyze the implications of using the bulk sample mean protein content as a reference for predictions on subsamples of the bulk. Here, we also show how to correct the subsequent bulk mean protein content predictions linearly, yielding significantly better bulk mean protein content predictions. In Section 5, our analysis of the distributions of predicted protein content in the bulk subsamples reveals why using the bulk sample mean as a reference for its subsamples leads to biased results. We analyze the effects of the grain density for both protein content regression and grain variety classification in Section 6. All results are concluded in Section 7.

2 | DATASETS AND MODELS

We use the two datasets of hyperspectral images of bulks of grain kernels captured with a Specim FX17 near-infrared hyperspectral line scan camera [26] introduced in [13] and [11], respectively. Figure 1 shows the imaging setup used by Engstrøm *et al.* [13]. Dreier *et al.* [11] use a similar setup. The two datasets contain images of densely and sparsely packed bulk grain samples. An example of a densely and a sparsely packed image from Engstrøm *et al.* [13] is shown in Figure 2. We preprocess the images according to the strategy devised by Engstrøm *et al.* [12]. This preprocessing includes cropping the images in sizes of 128×128 pixels with an overlap of 64 pixels in both spatial dimensions, spectral binning to 102 wavelength channels, and discarding any image crops where less than 10% of the pixels contain grain. Additionally, a mean grain spectrum is associated with each image crop, which is used for calibrating PLS. The mean grain spectrum is computed over the pixels that contain grain as determined by a binary segmentation mask using an Otsu threshold [20]. Initially, the hyperspectral image contains 224 uniformly distributed wavelength channels. However, due to a reduced sensitivity in the camera's sensors near the edges of its spectrum [26], the first and last 10 wavelength channels are discarded. Additionally, the spectral channels are averaged in neighboring pairs as the camera's average spectral resolution is 8 nm, with its sensors sampling at every 3.5 nm [26]. Thus, after preprocessing, the spectral dimension contains 102 uniformly distributed wavelength channels in the 938 – 1662 nm part of the NIR spectrum, which contains molecular vibration bands relevant to grain quality analysis [19], in particular to the protein content that is of interest to this study [6, 15, 27]. Figure 3 shows a hyperspectral image crop with its associated mean grain spectrum and the grain spectrum at an arbitrary grain pixel.

The first dataset, Dataset #1, is of wheat grain kernels, where the associated task is to regress the protein content in the grain kernels. The dataset has associated bulk mean protein reference values measured with a FOSS Infratec™Nova [2]. This dataset consists of 63 distinct bulk samples of wheat grain kernels, of which 50 are used for five-fold cross-validation and 13 for testing. The subsequent cropping yields 69,630 image crops for five-fold cross-validation and 17,783 for testing. The second dataset, Dataset #2, comprises seven wheat varieties and a single rye variety. Each image contains exactly a single variety. The task of this dataset is to classify the grain variety on a given image crop. Table 1 and Table 2 summarize the datasets.

While Engstrøm *et al.* [12] experiment with a plethora of preprocessing techniques for PLS-R and PLS2-DA and model designs for CNNs, we focus our attention on the best performing combination of preprocessing and model for each of PLS-R, PLS2-DA, and CNN. For PLS-R on spectral data from Dataset #1, the best preprocessing technique

Dataset split	# bulk subsamples in total (and per split)	# bulk samples in total (and per split)
5-fold CV	69,630 (13,734 / 13,676 / 14,672 / 13,618 / 13,930)	50 (10 / 10 / 10 / 10 / 10)
Test	17,783	13

TABLE 1 Dataset #1. The number of bulk subsamples (image crops and their mean grain spectra) and bulk samples in total and for each dataset split. There is only one test split.

Dataset split	# bulk subsamples
Training	15,376
Validation	7,967
Test	3,274

TABLE 2 The number of bulk subsamples (image crops and their mean grain spectra) and bulk samples for each dataset split. Each split contains all eight grain varieties. The distribution of grain varieties is approximately uniform in each dataset split.

was found to be applying Standard Normal Variate (SNV) [4] followed by the application of a Savitzky-Golay (SG) filter [25] with a window-size of 7, a polynomial order of 2 and a derivative order of 2. Additionally, for PLS-R, the protein reference values are centered around the mean protein reference value of the training splits. The best preprocessing technique for PLS2-DA on spectral data from Dataset #2 was applying only a spectral centering (Center).

For each dataset, the best-performing CNN is a modification of ResNet-18 [14]. For Dataset #1, the modification involves adding two initial layers, with the first one serving to learn a spectral preprocessing technique much similar to that of an SG filter and the second layer reducing the spectral dimensionality by downsampling it through a learnable weighted combination of the wavelength channels output by the first layer. For Dataset #2, the modification involves adding only one initial layer, namely the layer that learns to downsample the spectral dimensionality. Engstrøm et al. [12] explain the architectures in detail. They named the models ResNet-18 Modification #10 for Dataset #1 and ResNet-18 Modification #8 for Dataset #2. Moving forward, in this report, we will refer to these models by Modified ResNet-18 Regressor and Modified ResNet-18 Classifier, respectively.

Engstrøm et al. [12] use five-fold cross-validation for calibrating their models on Dataset #1. They show that Modified ResNet-18 Regressor performance can be improved on the test set by constructing an ensemble model that outputs an arithmetic mean of all five cross-validated models. Constructing an ensemble for the PLS-R yields neither better nor worse results on the test set than using an individual PLS-R model. For consistency, we focus our work with Dataset #1 on the mean model predictions that are the outputs of the ensemble Modified ResNet-18 Regressor and the ensemble PLS-R while still providing SEMs on the ensembles' predictions to show the degree of disagreement present in the ensembles' constituents.

3 | NOMENCLATURE

The remaining parts of this report extensively use mathematical symbols and terms. This section contains a description of them.

The following notation is used for mathematical symbols:

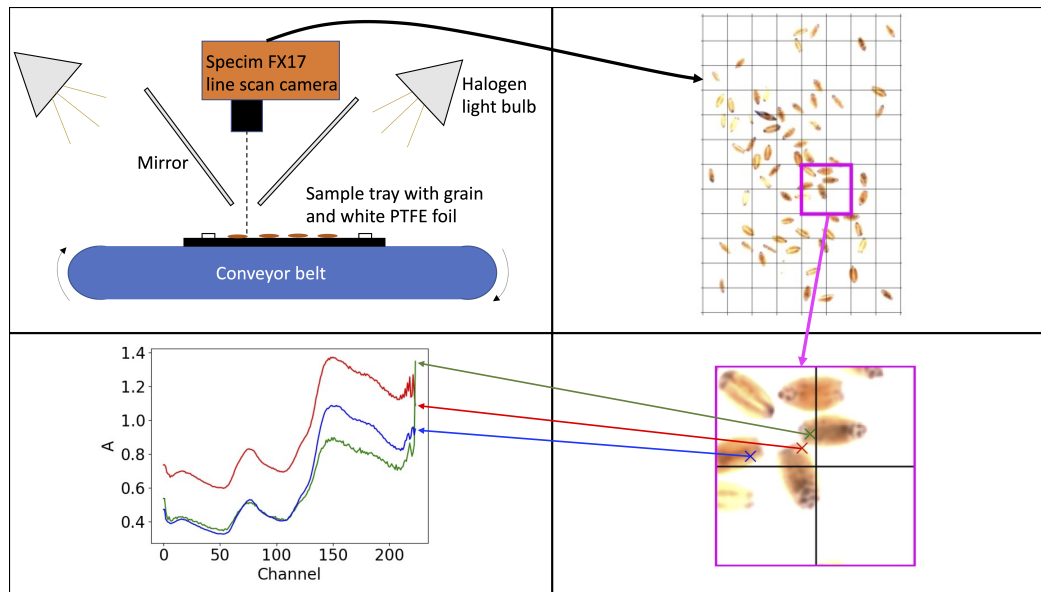


FIGURE 1 The hyperspectral imaging setup. This figure shows, clockwise from top-left, the imaging setup, a hyperspectral image of sparsely packed grain from one camera acquisition, a 128×128 pixels hyperspectral image crop, and the pseudo absorbance spectra of three pixels containing grain. The image is taken from Engstrøm *et al.* [13].

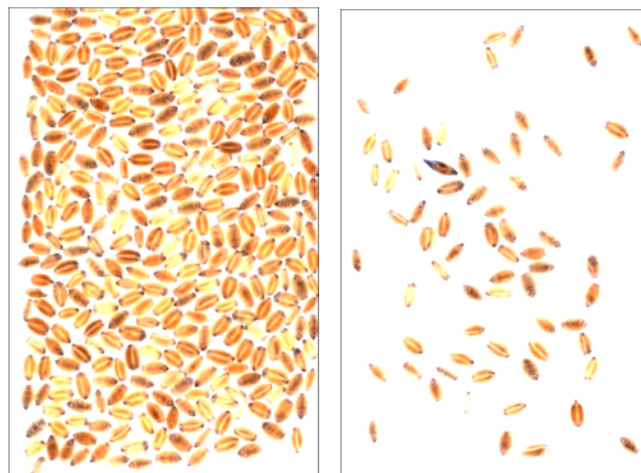


FIGURE 2 This figure shows examples of densely (left) and sparsely (right) packed grain images. Each of the 63 bulk samples is distributed along 8 sparsely packed images and approximately 10 – 14 densely packed images [13]. The image is taken from Engstrøm *et al.* [13].

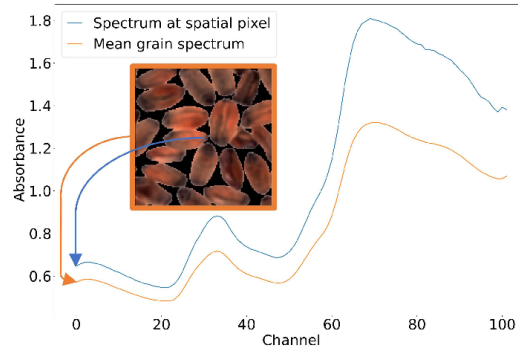


FIGURE 3 A masked hyperspectral image crop of spatial size 128×128 pixels along with the spectrum at one of the pixels and the mean spectrum of the crop computed over all the grain pixels, both containing 102 wavelength channels. Taken from Engstrøm et al. [12]. The discrepancy between grain colors in this Figure and those in Figure 1 and Figure 2 are due to differences in the calculations that transform hyperspectral images to pseudo-RGB and are only for visualization purposes.

i	Subscript integer index counter.
train	Subscript used to reference training set data.
val	Subscript used to reference validation set data.
test	Subscript used to reference test set data.
ss	Subscript used to indicate bulk subsample.
bm	Subscript used to indicate bulk subsample means. I.e. averaging over all ss for each individual bulk sample.
bulk	Subscript used to indicate bulk. Used only in context with PLS-R calibrated on mean spectra from bulk samples.
n	Number of samples. Will either be number of bulk samples or number of bulk subsamples dependent on context.
ϵ	Error terms vector of shape $n \times 1$.
$\mathbf{y}$	Reference value column vector of shape $n \times 1$.
$\hat{\mathbf{y}}$	Prediction value column vector of shape $n \times 1$.
$\mathbf{Y}$	Augmented reference value matrix of shape $n \times 2$. The first column is ones. The second column is $\mathbf{y}$.
$\hat{\mathbf{Y}}$	Augmented prediction value matrix of shape $n \times 2$. The first column is ones. The second column is $\hat{\mathbf{y}}$.

The following describes "ensemble model prediction," a term often used in our figures. Recall that, for Dataset #1, we concern ourselves with ensemble models. The "ensemble model prediction" is a uniform average over the ensembles' constituents. Recall that the ensembles are trained with a 5-fold CV. Thus, when used in training sets, the mean model prediction is an average prediction over the four constituents with the data point(s) in question in their training split. For the same reason, "ensemble model prediction," when used in the context of validation splits, is the prediction of only the single constituent with the particular data point(s) in its validation split. When used in the context of the test split, "ensemble model prediction" is a uniform average over all five of the ensembles' as the test split is the same for all constituents.

For the reasons mentioned above, the contents of any prediction vectors and matrices associated with the training or test splits of Dataset #1 are ensemble model predictions unless explicitly stated otherwise.

4 | BULK MEAN AS REFERENCE FOR SUBSAMPLE PREDICTIONS

While the RMSE Equation (1) is a good measure of how well a regression model performs, it is a summary statistic that does not give any information about potential biases in the model. In particular, we are interested in whether calibrating our models on the bulk subsamples allows them to accurately predict the bulk mean reference value - i.e. if a model optimized to minimize $\text{RMSE}(\mathbf{y}, \hat{\mathbf{y}})$ is simultaneously a good model for minimizing $\text{RMSE}(\mathbf{y}_{\text{bm}}, \hat{\mathbf{y}}_{\text{bm}})$. Recall that we are only sure about the values of $\mathbf{y}_{\text{bm}}$ as these are the ones measured with the FOSS Infratec™Nova [2] but that we distribute $\mathbf{y}_{\text{bm},i}$ to all subsamples originating from bulk sample i to inflate the amount of data available for calibration, resulting in $\mathbf{y}$.

$$\text{RMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2} \quad (1)$$

Ideally, the models would have been calibrated such that the predictions lie on the straight line $\mathbf{y}_{\text{bm}} = \hat{\mathbf{y}}_{\text{bm}}$. In practice, however, this may be difficult to achieve due to various sources of errors. Instead, aggregating the error terms in ϵ , the relationship can be written as Equation (2).

$$\mathbf{y}_{\text{bm}} = \hat{\mathbf{y}}_{\text{bm}} + \epsilon \quad (2)$$

Using RMSE as a loss function, as done by Engstrøm et al. [12], the task is to find $\hat{\mathbf{y}}_{\text{bm}}$ such that $\sum_{i=0}^{n-1} \epsilon_i^2$ is minimized - this is a variant of linear regression known as least squares regression. To assess how well our models perform least squares regression, we compare their $\text{RMSE}(\mathbf{y}_{\text{bm}}, \hat{\mathbf{y}}_{\text{bm}})$ with their $\text{sYX}(\mathbf{y}_{\text{bm}}, \hat{\mathbf{y}}_{\text{bm}})$, defined in Equation (3). sYX uses OLS-R (Equation (4)), a closed form solution to the least squares regression problem, to compute a bias_{bm} and a scale_{bm} parameter that minimize $\sum_{i=0}^{n-1} \epsilon_i^2$ in Equation (5). As we estimate bias_{bm} and scale_{bm} , we lose two degrees of freedom in the subsequent RMSE computation, hence the $n - 2$ term in Equation (3).

$$\text{sYX}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{n-2} \sum_{i=0}^{n-1} (\mathbf{y}_i - (\hat{\mathbf{y}}_i \times \text{scale} + \text{bias}))^2} \quad (3)$$

$$\text{OLS-R}(\hat{\mathbf{Y}}, \mathbf{y}) = \begin{bmatrix} \text{bias} \\ \text{scale} \end{bmatrix} = (\hat{\mathbf{Y}}^T \hat{\mathbf{Y}})^{-1} \hat{\mathbf{Y}}^T \mathbf{y} \quad (4)$$

$$\mathbf{y}_{\text{bm}} = \text{bias} + \text{scale} \times \hat{\mathbf{y}}_{\text{bm}} + \epsilon \quad (5)$$

In Figure 4 we plot $\hat{y}_{bm}$ against y_{bm} , show $RMSE(y_{bm}, \hat{y}_{bm})$, $sYX(y_{bm}, \hat{y}_{bm})$, and plot the regression line found by OLS-R($\hat{y}_{bm}, y_{bm}$). Note that $bias_{bm}$ and $scale_{bm}$ tell us how to adjust $\hat{y}_{bm}$ so that it lies on a line that best matches y_{bm} . However, for visualization purposes, it is clearer to show the equivalent, inverse relationship - i.e., how to adjust y_{bm} so that its line best matches $\hat{y}_{bm}$ which is what we have plotted.

By inspecting Figure 4, it is evident that all plots show an sYX that is much lower than the corresponding $RMSE$. Likewise, all plots exhibit a $bias_{bm}$ of less than 0 and a $scale_{bm}$ of greater than 1. These values imply that bulk samples with a low protein reference value are systematically overpredicted and that bulk samples with a high protein reference value are systematically underpredicted. This phenomenon is persistent across both models, and all dataset splits. Interestingly, in Figure 5, where we plot $\hat{y}_{ss}$ against y_{ss} , this phenomenon is hard to notice as $RMSE(y_{ss}, \hat{y}_{ss})$ and $sYX(y_{ss}, \hat{y}_{ss})$ are very close to each other.

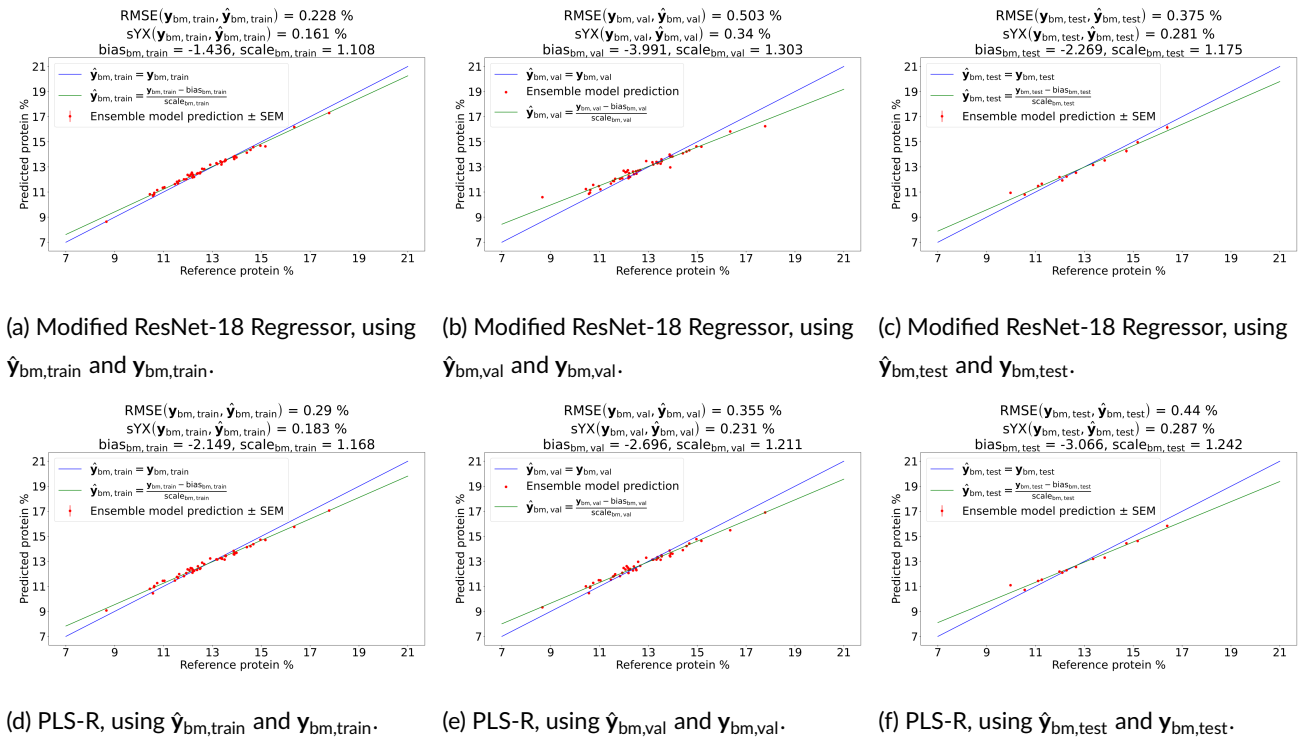


FIGURE 4 Modified ResNet-18 Regressor and PLS-R errorbar plots of $\hat{y}_{bm}$ and y_{bm} .

These results indicate that simply taking the bulk sample-wise average over the subsample predictions, $\hat{y}_{ss}$, to achieve $\hat{y}_{bm}$ does not lead to a good estimate of the bulk sample reference, y_{bm} . Instead, a linear correction must be applied to achieve the best relationship. However, while $bias_{test}$ and $scale_{test}$ tell us how to best correct our predictions, we can not use these parameters as that would make the test predictions biased. Instead, we are free to choose the bias and scale parameters from either the training or validation set. In Figure 6 we show how the test set predictions look when correcting the test set predictions using either $bias_{bm,training}$ and $scale_{bm,training}$ or $bias_{bm,val}$ and $scale_{bm,val}$. Interestingly, the best choice of parameters differs for Modified ResNet-18 Regressor and PLS-R. For Modified ResNet-18 Regressor, doing a correction based on the training set parameters is optimal, while basing the correction on the validation set parameters is optimal for PLS-R. For both models, however, using either of the two sets of parameters for a linear correction yields better results than not doing a linear correction at all as evident by comparing the RMSEs in Figure 4 with those on the test sets in Figure 6.

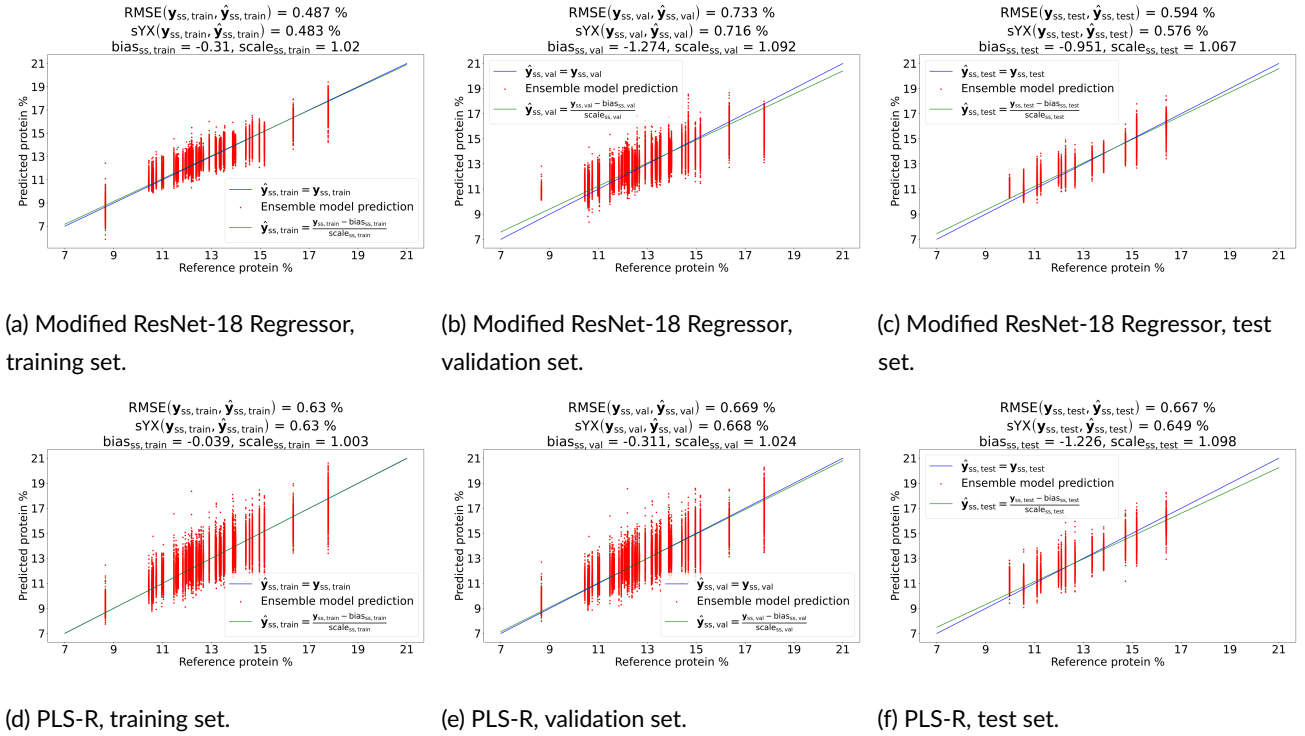
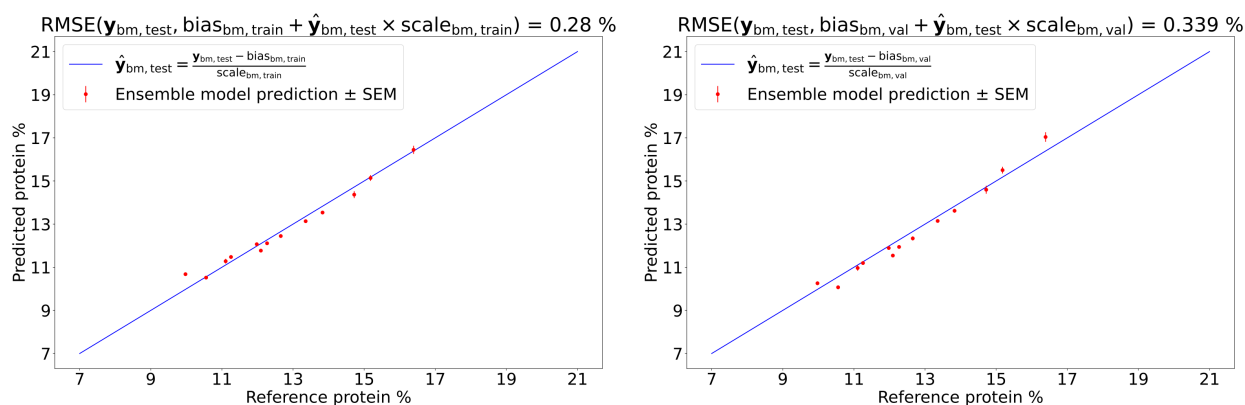


FIGURE 5 Modified ResNet-18 Regressor and PLS-R prediction plots of $\hat{y}_{ss}$ and y_{ss} .

4.1 | Calibrating PLS-R directly on bulk samples

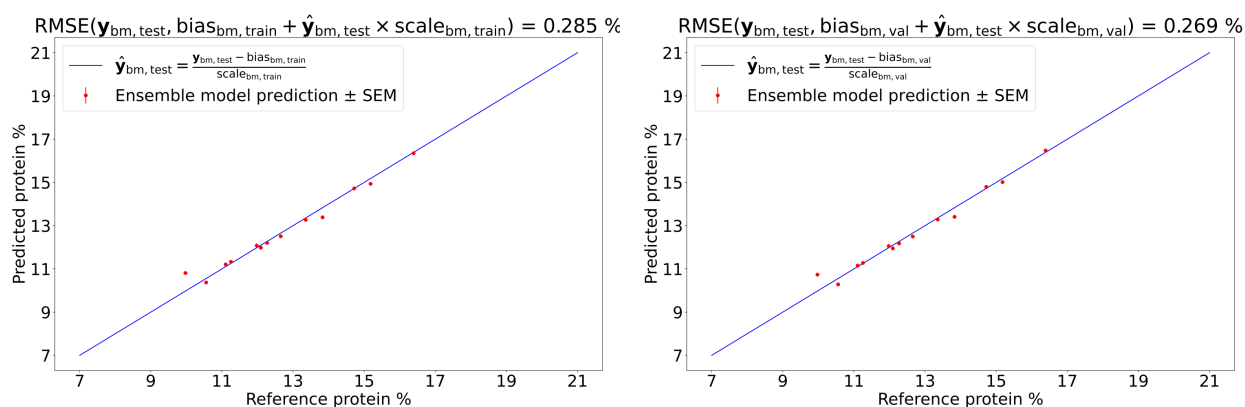
While calibrating Modified ResNet-18 Regressor, a function with 11.2 million parameters, requires a lot of data, PLS-R can be calibrated with much less data. Here, we want to calibrate PLS-R on the 50 mean spectra from the bulk samples in the CV dataset splits and assess its performance on the 13 mean spectra in the test dataset split. Following the precedence set with the previous models, we construct an ensemble of five PLS-R models to predict $\hat{y}_{bulk}$ and minimize $RMSE(y_{bulk}, \hat{y}_{bulk})$. We will call this ensemble model PLS-R_{bulk}. Note that, by definition, $y_{bulk} = y_{bm}$. We use the same preprocessing for PLS-R_{bulk} that we used for PLS-R.

In Figure 7, we show errorbar plots for PLS-R_{bulk} on all three dataset splits. We show the errorbar plots both for $\hat{y}_{bulk}$ and for $\hat{y}_{bm}$. As can be seen on the top row in Figure 7, the $RMSE(y_{bulk}, \hat{y}_{bulk})$ and $sYX(y_{bulk}, \hat{y}_{bulk})$ are very similar, indicating that the model has solved the task for which it was calibrated. The bottom row also reveals values for $bias_{bm}$ and $scale_{bm}$ much closer to 0 and 1, respectively, than those in the bottom row of Figure 4, indicating that calibrating a model to minimize $RMSE(y_{bulk}, \hat{y}_{bulk})$ will enable it to minimize $RMSE(y_{bm}, \hat{y}_{bm})$ much better than a model trained to minimize $RMSE(y_{ss}, \hat{y}_{ss})$. When we compare $RMSE(y_{bulk}, \hat{y}_{bulk})$ with those linearly corrected test set predictions of Modified ResNet-18 Regressor and PLS-R in Figure 6, they are approximately identical, indicating that either calibration approach is equally viable for predicting the mean protein content. For the sake of completeness, we show errorbar plots and corresponding RMSE and sYX of the bulk subsample predictions using PLS-R_{bulk} in Figure 8. The results are much worse than those of Modified ResNet-18 Regressor and PLS-R in Figure 5. The corresponding bias and scale parameters are also quite extreme, indicating that calibrating a model to minimize $RMSE(y_{bulk}, \hat{y}_{bulk})$ will not simultaneously let it minimize $RMSE(y_{ss}, \hat{y}_{ss})$.



(a) Modified ResNet-18 Regressor, test set using bias and scale from training set mean predictions.

(b) Modified ResNet-18 Regressor, test set using bias and scale from validation set mean predictions.



(c) PLS-R, test set using bias and scale from training set mean predictions.

(d) PLS-R, test set using bias and scale from validation set mean predictions.

FIGURE 6 Test set errorbar plots of linearly corrected predictions based on bias and scale parameters from the training and validation splits, respectively.

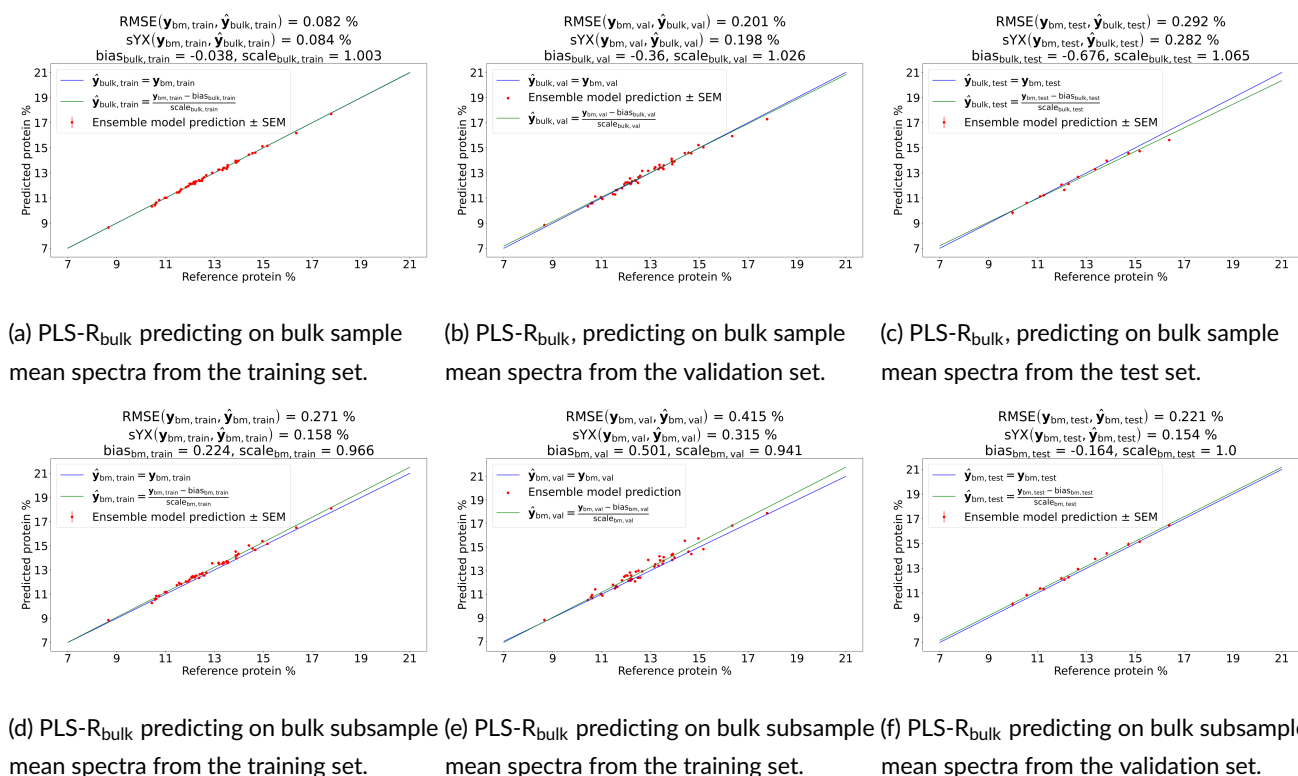


FIGURE 7 Errorbar plots for PLS- R_{bulk} mean predictions. While the test set RMSE of PLS- R_{bulk} given by $\text{RMSE}(\mathbf{y}_{\text{bm},\text{test}}, \hat{\mathbf{y}}_{\text{bm},\text{test}})$ is lower than all other test set RMSEs, we hypothesize that this is insignificant, as the corresponding training and validation RMSEs, shown in the bottom row of this figure, are not equally better than their counterparts from the other models as evident by comparing them to those shown in Figure 4.

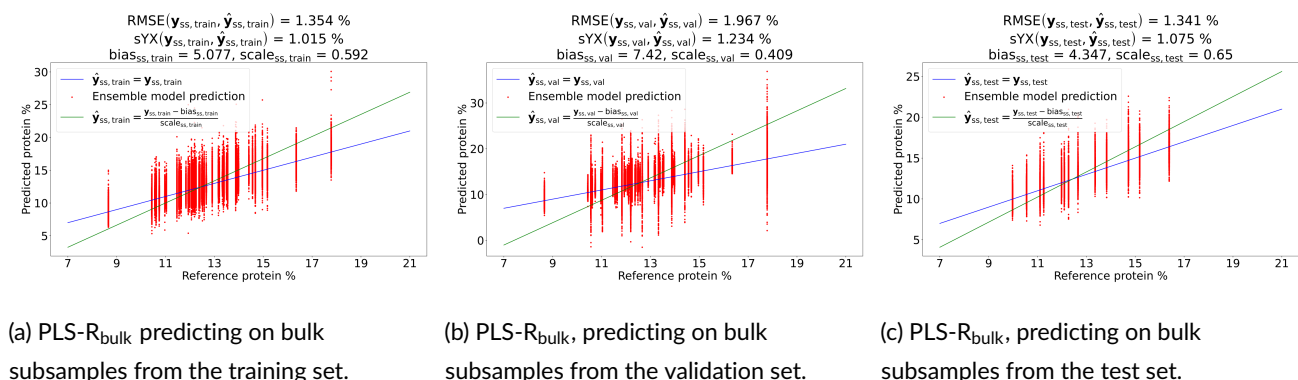


FIGURE 8 Errorbar plots for PLS- R_{bulk} bulk subsample predictions.

5 | STATISTICAL ANALYSIS OF PREDICTION DISTRIBUTIONS

The phenomena causing the discrepancy between minimizing $\text{RMSE}(\mathbf{y}_{ss}, \hat{\mathbf{y}}_{ss})$ or $\text{RMSE}(\mathbf{y}_{bm}, \hat{\mathbf{y}}_{bm})$ and $\text{RMSE}(\mathbf{y}_{bulk}, \hat{\mathbf{y}}_{bulk})$ can be found by inspecting the distributions of $\hat{\mathbf{y}}_{ss}$, grouped by bulk sample. If the distribution of $\hat{\mathbf{y}}_{ss}$ is symmetric, then $\text{OLS-R}(\hat{\mathbf{y}}_{bm}, \mathbf{y}_{bm}) = \text{OLS-R}(\hat{\mathbf{y}}_{ss}, \mathbf{y}_{ss})$. While this is not the case, we wish to analyze whether the discrepancy is statistically significant or if it can be attributed to randomness. Two common symmetric distributions are the uniform distribution and the normal distribution. It is clear from inspecting the plots that the distribution of $\hat{\mathbf{y}}_{ss}$ is not uniform for any of the models on any of the dataset splits. Instead, we focus our attention on analyzing if, with high confidence, $\hat{\mathbf{y}}_{ss}$ is not normally distributed.

While inspections of the skewnesses and kurtoses are suitable for visual interpretation, they can hardly prove that each bulk sample's prediction distribution is not normal. Instead, statistical tests based on skewness and excess kurtosis are suggested by The American Statistical Association to detect deviations from normality [9]. The skewness test [7] and kurtosis test [1] are both constructed as two-sided Z -tests where the Z -scores will be approximately normally distributed under the null hypothesis that the underlying prediction distribution is also normally distributed. The related p -value tells us the probability of observing a Z -score at least as extreme as the one calculated if the underlying prediction distribution is normally distributed. Additionally, D'Agostino's K^2 -test, an omnibus test combining the tests for skewness and kurtosis [8, 10], summarizes the previously mentioned Z -tests to compute a K^2 -score that is the sum of the squared Z -scores and which will be χ^2 -distributed with two degrees of freedom if the two Z -scores are normally distributed.

Neither of the previously mentioned tests provides evidence for normality - i.e., they can not confirm the null hypothesis. Instead, they can provide evidence for deviation from normality. However, if the evidence for deviation from normality is strong enough, it is safe to assume that the predictions are not normally distributed. All statistical moments and tests are computed using software from scikit-learn [22].

Initially, we focus on asymmetry, which can be measured by skewness. If our prediction distributions are symmetric, they will have a skewness of 0. We note that the reverse relationship does not necessarily hold - i.e., 0 skewness does not imply a symmetric distribution. However, if a distribution has non-zero skewness, it is not symmetric. We plot the skewnesses, Z -scores, and related p -values for Modified ResNet-18 Regressor, PLS-R, and PLS-R_{bulk} in Figure 9, Figure 10, and Figure 11, respectively. These tests imply that, beyond any reasonable doubt, most of the prediction distributions are too skewed to be normally distributed and are, in fact, asymmetric.

We would also like to inspect the excess kurtosis, which is 0 for a normal distribution, is bounded below by -2 , and is unbounded above. Kurtosis measures a distribution's tendency to produce outliers and not, as otherwise often stated, its peakedness [28]. Negative excess kurtosis implies a platykurtic distribution that produces fewer or less extreme outliers than a normal distribution. Likewise, a positive excess kurtosis implies a leptokurtic distribution that produces more or more extreme outliers than a normal distribution. In turn, a distribution with an excess kurtosis of 0 is called mesokurtic. We show the excess kurtoses and associated kurtosis tests for Modified ResNet-18 Regressor, PLS-R, and PLS-R_{bulk} in Figure 12, Figure 13, and Figure 14, respectively. These tests imply that, beyond any reasonable doubt, most of the prediction distributions are leptokurtic and produce too many or extreme outliers than would be expected from a normal distribution.

Finally, we show the omnibus tests for normality for all three models in Figure 15. Here, it is evident that most of the prediction distributions for all the models are, in fact, not normal. The non-zero skewnesses combined with the high excess kurtoses imply that the distributions contain more or more extreme outliers than the normal distribution and that these outliers tend to show up on one side of the means of the distributions. Recall that RMSE is a measure that weights outliers heavily due to the square term in the deviation from the mean. In contrast, the arithmetic

mean only weights outliers linearly. Thus, a model that is calibrated to minimize $\text{RMSE}(\mathbf{y}_{ss}, \hat{\mathbf{y}}_{ss})$, where outliers are weighted heavily, will yield a different set of model parameters than those achieved by minimizing $\text{RMSE}(\mathbf{y}_{bm}, \hat{\mathbf{y}}_{bm})$, where outliers are weighted less heavily.

While our analysis reveals the skewed and leptokurtic distributions that give rise to the discrepancy between the two minimization problems, the underlying reasons for the shapes of these distributions remain unknown. We note that, especially for Modified ResNet-18 Regressor, the distributions seem to be skewed towards a protein % that is slightly higher than the CV reference mean value, and the skewness gets more extreme as the bulk sample mean reference value deviates further from the CV reference mean value. Additionally, for Modified ResNet-18 Regressor, the training set kurtoses are lower for bulk samples that have a reference protein % close to the mean CV reference protein. These effects could indicate that the Modified ResNet-18 Regressor has learned to recognize grain from the different bulk samples in the training set and map them close to the reference value, thus circumventing the features that explain protein content, severely overfitting, and rendering ineffective the strategy that assigns the bulk sample mean reference values to bulk subsamples.

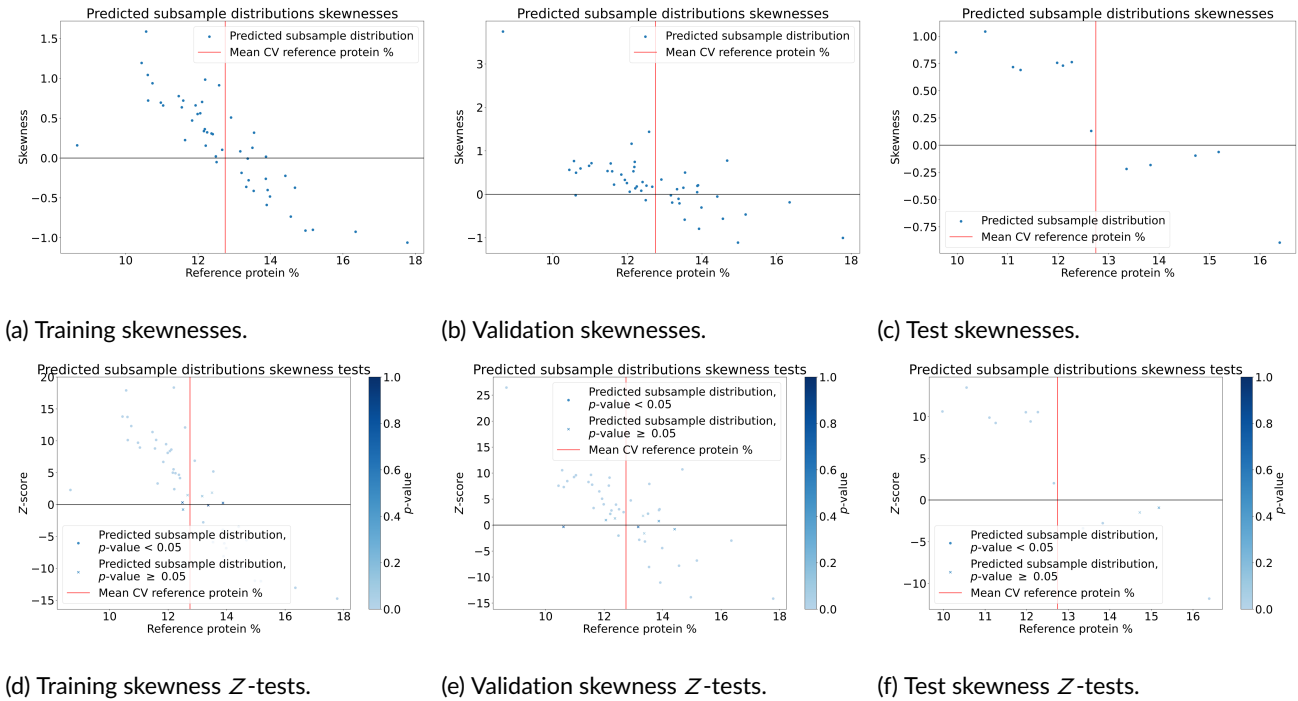


FIGURE 9 Modified ResNet-18 Regressor subsample prediction distribution skewnesses and skewness tests for each bulk sample as a function of the reference protein of the bulk sample.

6 | THE INFLUENCE OF GRAIN DENSITY ON MODEL PERFORMANCES

In this section, we investigate the effect of grain density on protein content regression on Dataset #1 and grain variety classification on Dataset #2. Recall that the grain density of an image crop is the ratio of its amount of pixels containing grain to its total amount of pixels. We do not analyze the predictions of the ensemble models on Dataset #1. Instead, we analyze the ensembles' constituents individually. In particular, we are interested in measuring the average performance of each constituent as a function of grain density. This allows us to compute a mean $\text{RMSE}(\mathbf{y}_{ss, \text{test}}, \hat{\mathbf{y}}_{ss, \text{test}})$

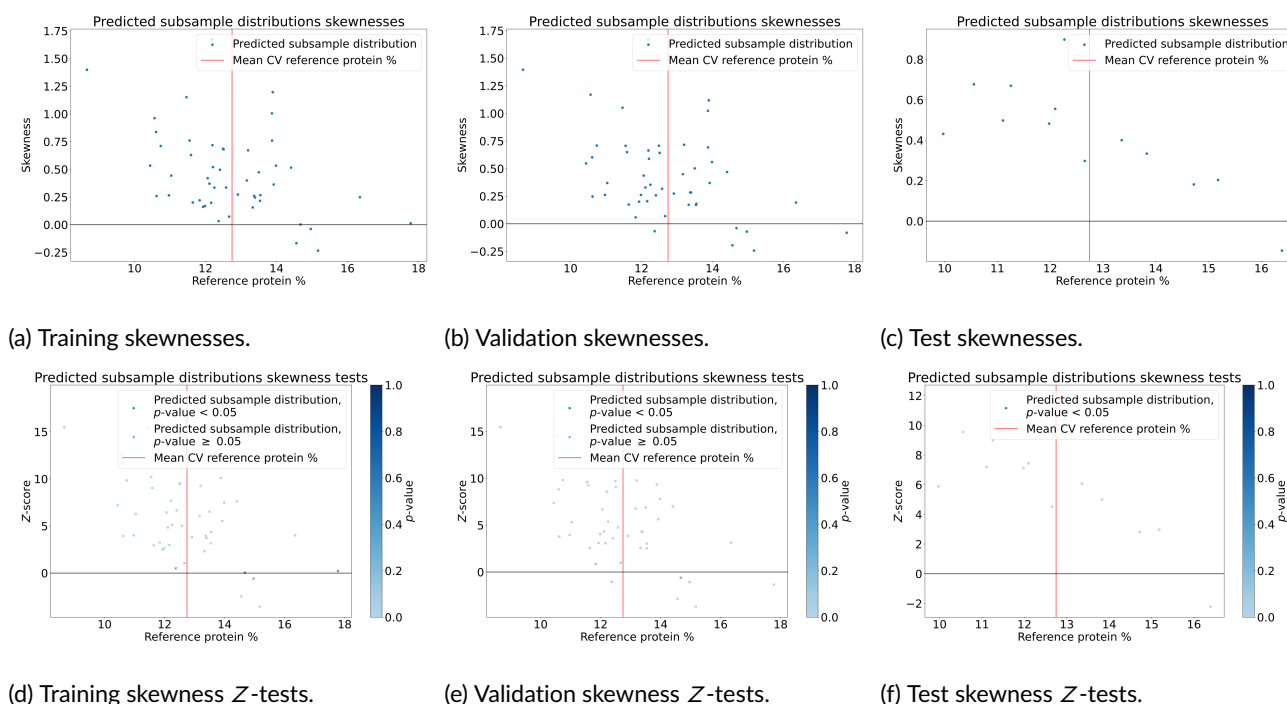


FIGURE 10 PLS-R subsample prediction distribution skewnesses and skewness tests for each bulk sample as a function of the reference protein of the bulk sample.

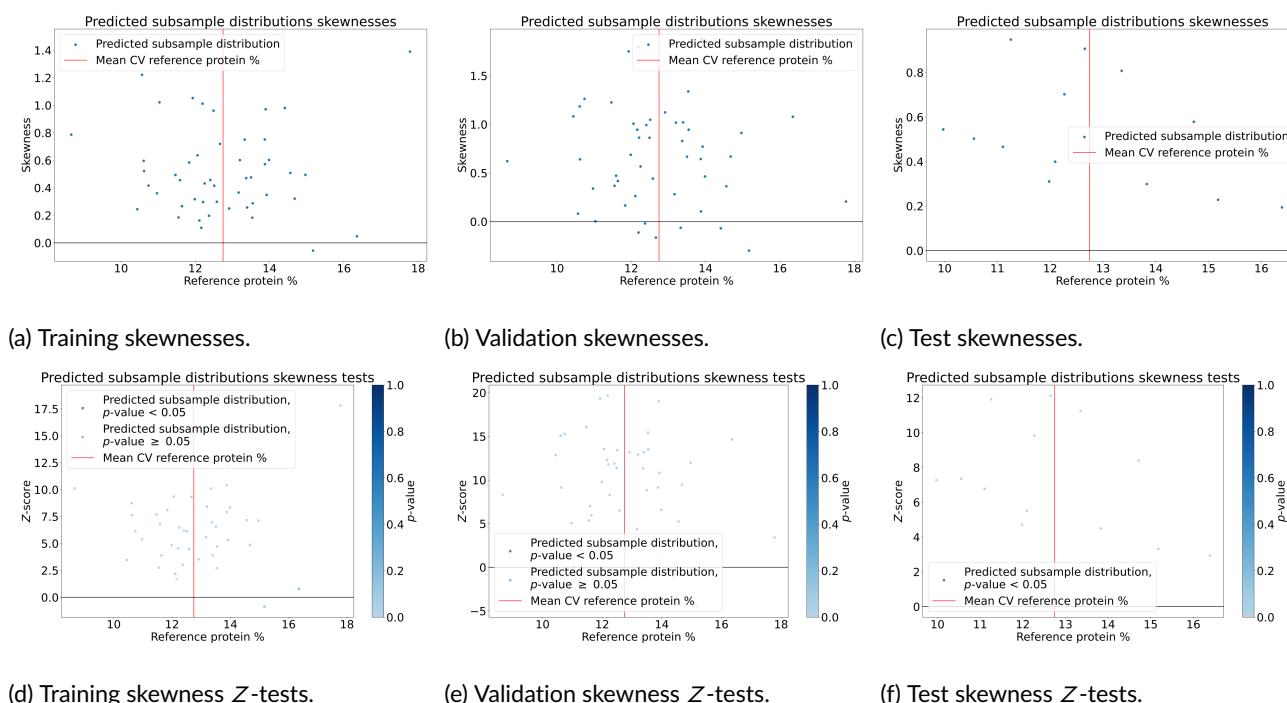


FIGURE 11 PLS-R_{bulk} subsample prediction distribution skewnesses and skewness tests for each bulk sample as a function of the reference protein of the bulk sample.

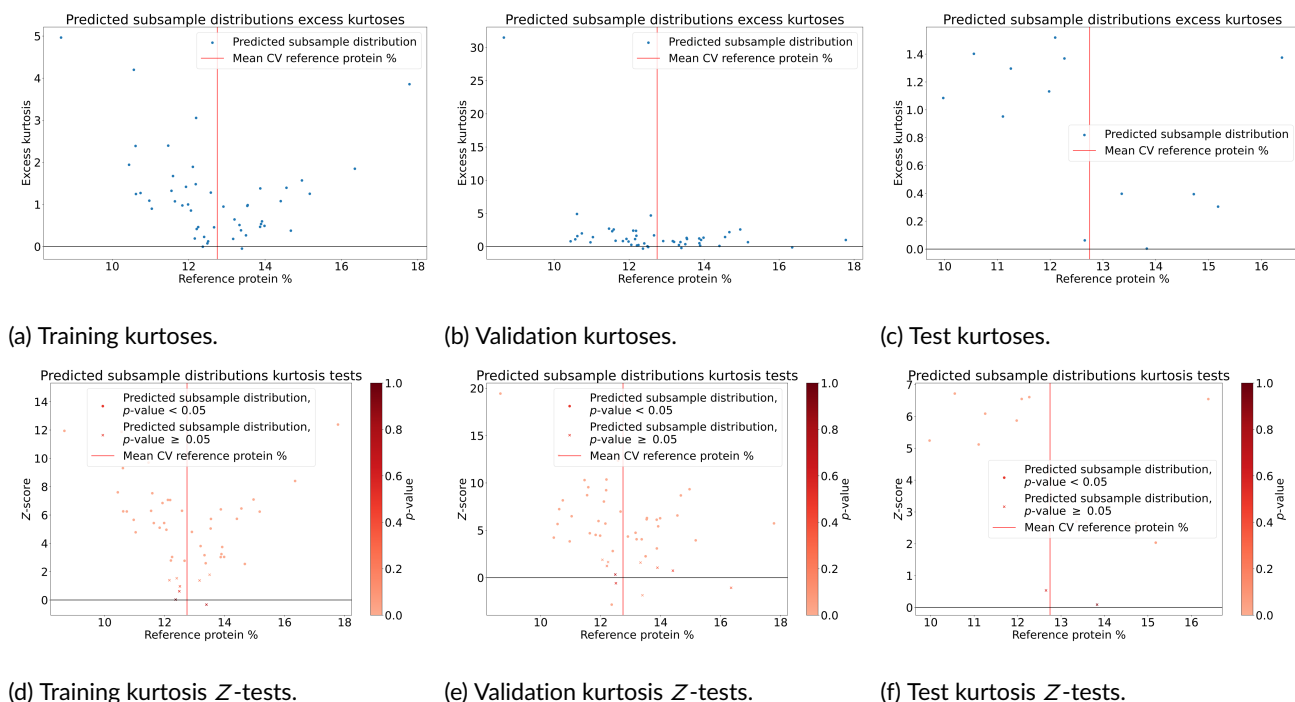


FIGURE 12 Modified ResNet-18 Regressor subsample prediction distribution kurtoses and kurtosis tests for each bulk sample as a function of the reference protein of the bulk sample.

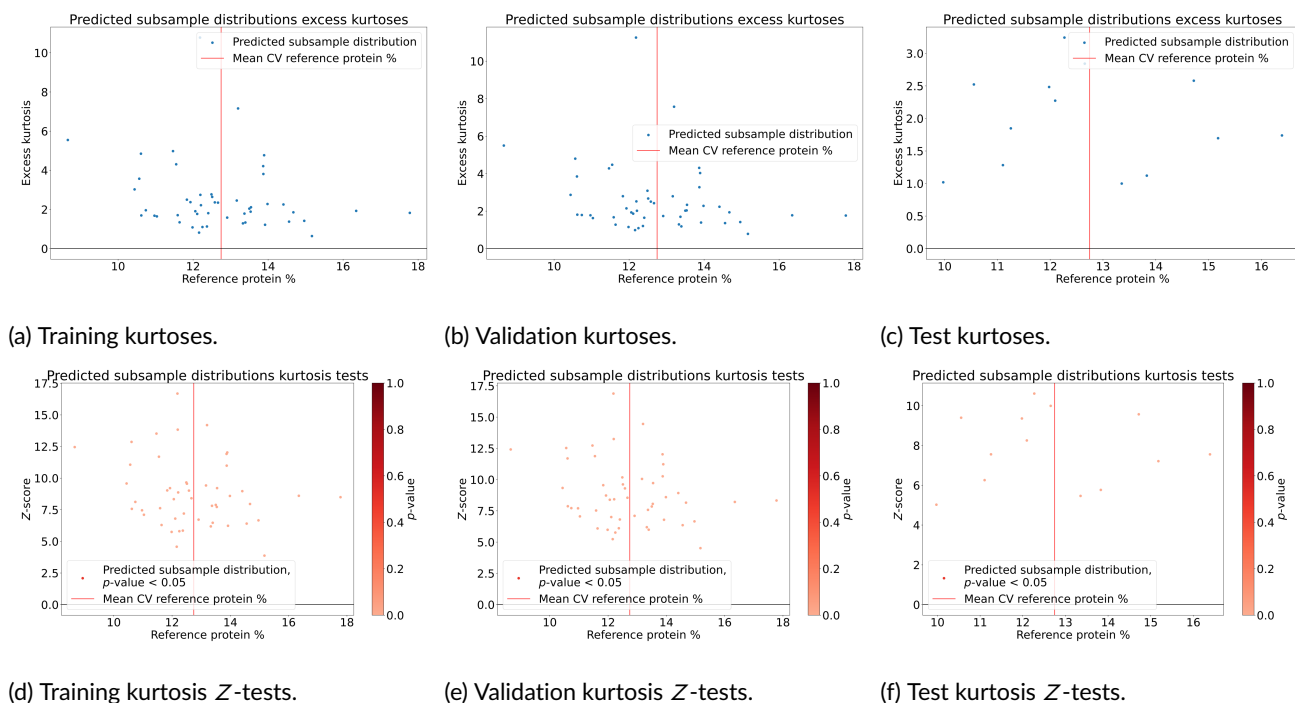


FIGURE 13 PLS-R subsample prediction distribution kurtoses and kurtosis tests for each bulk sample as a function of the reference protein of the bulk sample.

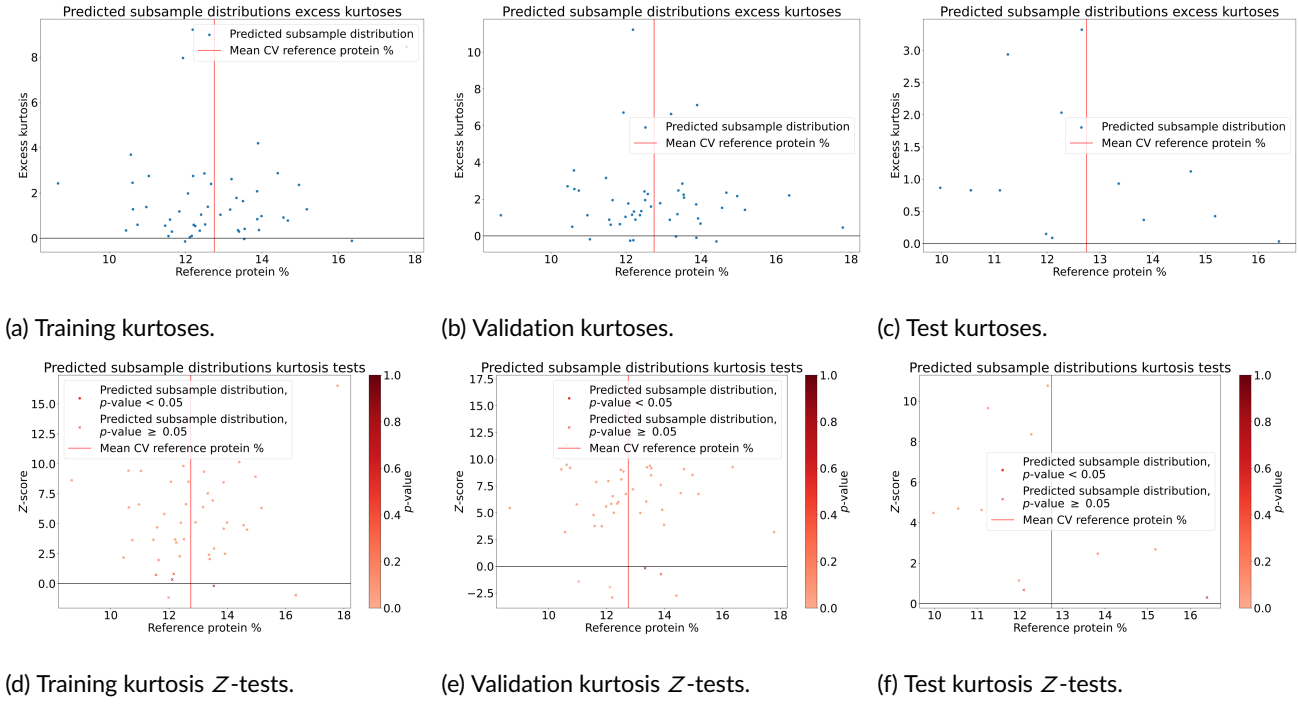


FIGURE 14 PLS- R_{bulk} subsample prediction distribution kurtoses and kurtosis tests for each bulk sample as a function of the reference protein of the bulk sample.

and an associated SEM. Engstrøm et al. [12] use a grain density of at least 0.1 for all their model calibrations and evaluations on both datasets.

In Figure 16 we show the $\text{RMSE}(\mathbf{y}_{\text{ss,test}}, \hat{\mathbf{y}}_{\text{ss,test}})$ for grain density intervals of 0.05 and for any given value of minimum grain density. Here, it is clear that for all models, $\text{RMSE}(\mathbf{y}_{\text{ss,test}}, \hat{\mathbf{y}}_{\text{ss,test}})$ deteriorates as grain density decreases. This is unsurprising as the reference value, computed as a mean over a bulk sample, is expected to be more accurate as grain density increases, and, likewise, any reasonably well-calibrated model will reflect this relationship in its predictions.

On Dataset #2, Dreier et al. [11] use a grain density of at least 0.5 for all their calibrations but also do a test set evaluation for grain density < 0.5 and show that accuracy rapidly diminishes as grain density decreases. Additionally, on Dataset #2, there are no ensemble models as the models were trained with a single training and validation dataset split. In Figure 17 we show the $\text{Acc}(\mathbf{y}_{\text{ss,test}}, \hat{\mathbf{y}}_{\text{ss,test}})$, defined in Equation (6), for Modified ResNet-18 Classifier and PLS2-DA. Recall that for Dataset #2, unlike for Dataset #1, the reference value is equally accurate for a low and a high grain density as only a single grain variety is present in any given subsample. While Modified ResNet-18 Classifier is better than PLS2-DA, both models have a relatively stable performance until grain density decreases below 0.3. At the interval with the lowest grain density of 0.1 to 0.15, Modified ResNet-18 Classifier reaches its minimum accuracy of ~ 0.87 , and PLS2-DA reaches its minimum of ~ 0.82 . Interestingly, however, both test accuracies are much higher than the $\text{Acc}(\mathbf{y}_{\text{ss,val}}, \hat{\mathbf{y}}_{\text{ss,val}})$ of ~ 0.5 achieved by Dreier et al. [11] on the validation split with their models that were trained on a grain density ≥ 0.5 . These results reveal that while easier with a high grain density, getting accurate predictions on a low grain density subsample is feasible as long as the model has been calibrated on low grain density subsamples. Our results also show that both the spatio-spectral CNN and the purely spectral PLS2-DA benefit from a larger grain density.

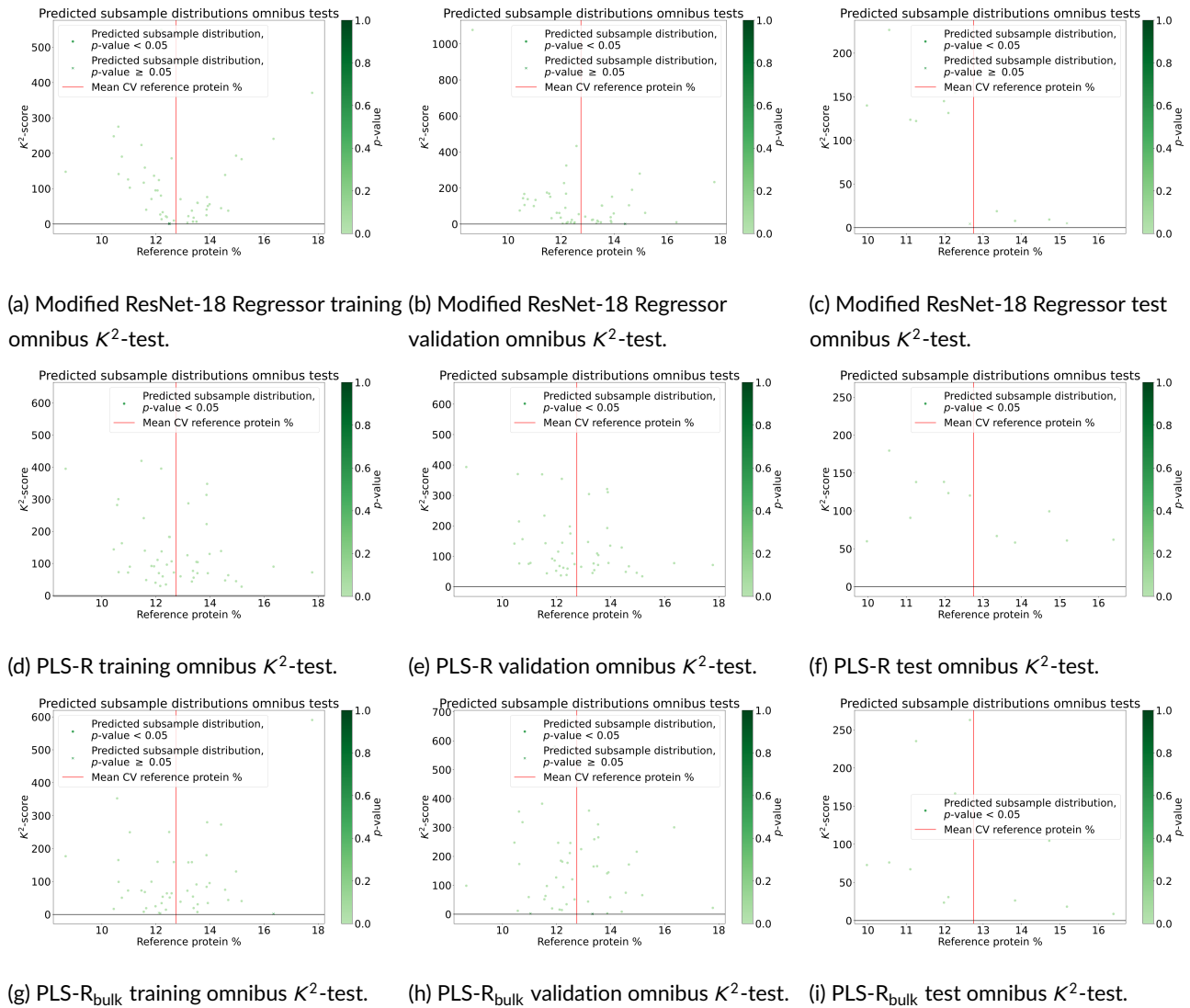
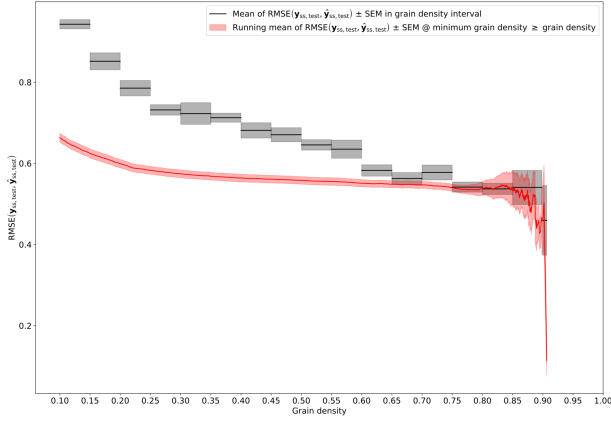
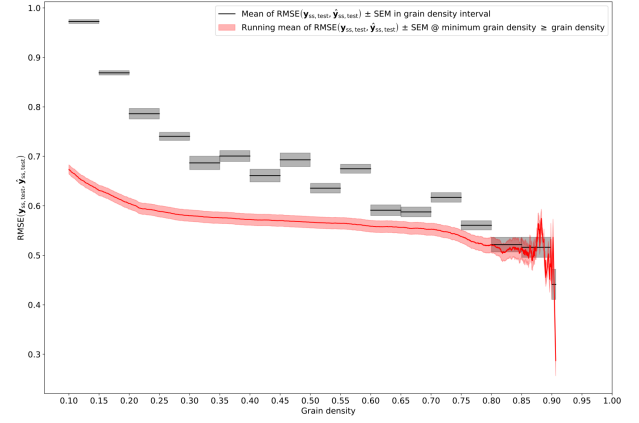


FIGURE 15 Modified ResNet-18 Regressor, PLS-R, and PLS-R_{bulk} subsample prediction distribution omnibus tests.

$$\text{Acc}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=0}^{n-1} \mathbb{1}(\mathbf{y}_i = \hat{\mathbf{y}}_i) \quad (6)$$



(a) Modified ResNet-18 Regressor.



(b) PLS-R.

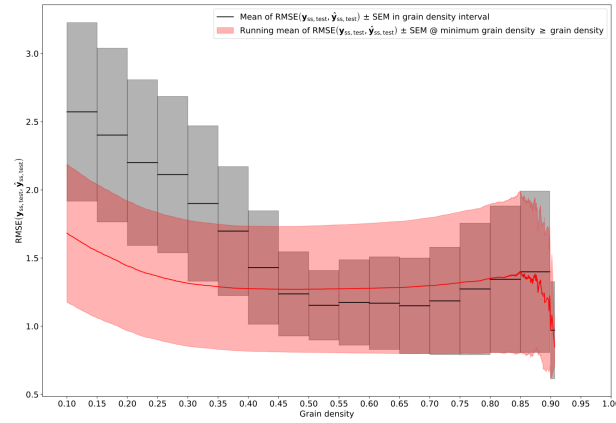
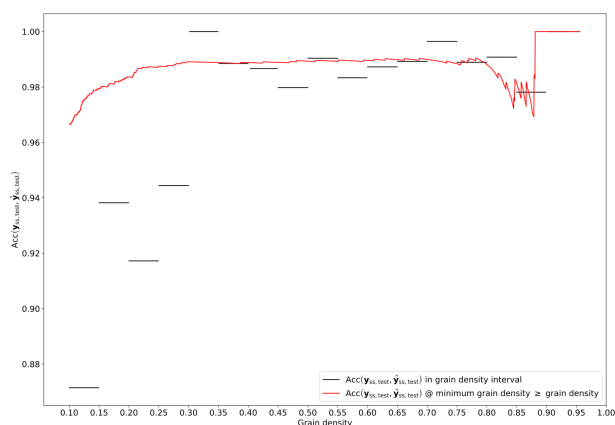
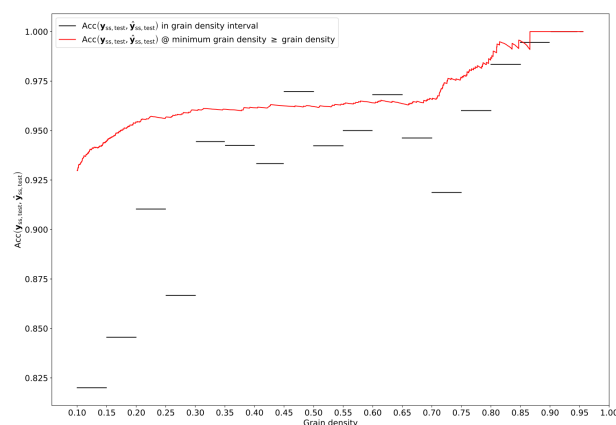
(c) PLS-R_{bulk}

FIGURE 16 $\text{RMSE}(\mathbf{y}_{\text{ss,test}}, \hat{\mathbf{y}}_{\text{ss,test}})$ as a function of the grain density. The red graphs should be interpreted right-to-left, i.e., for each of the three regression model types at any grain density, they represent the mean RMSE and associated SEM achieved by the five ensemble model constituents by predicting test set subsamples with grain density at least that high.



(a) Modified ResNet-18 Classifier.



(b) PLS2-DA.

FIGURE 17 $\text{Acc}(\mathbf{y}_{ss,\text{test}}, \hat{\mathbf{y}}_{ss,\text{test}})$ as a function of the grain density. The red graphs should be interpreted right-to-left, i.e., for both of the classification model types at any grain density, they represent the mean RMSE and associated SEM achieved by the five ensemble model constituents by predicting test set subsamples with grain density at least that high.

7 | CONCLUSION

This report shows that calibrating CNNs and PLS models to predict protein content in grain bulk subsamples using mean references from grain bulk samples can lead to biased average predictions. We have investigated the models' prediction distributions of the protein content in the grain bulk subsamples and found that they are both leptokurtic and skewed, implying that they produce more or more extreme outliers than a normal distribution and that these outliers typically occur on one side of the mean value. The underlying reason for the existence of the skewed leptokurtic distribution remains unknown. However, their existence implies that minimizing the outlier-sensitive RMSE on grain bulk subsamples leads to incorrect mean predictions on the grain bulk samples. We have devised two strategies to mitigate this issue. The first strategy works for both CNNs and PLS models and consists of a linear correction of the mean predictions on the grain bulk samples. The second strategy works for PLS only and consists of calibrating it directly on the bulk samples instead of the bulk subsamples. Either strategy will lead to approximately equally good performance, and both strategies lead to much more accurate mean protein content regression than just calibrating on the bulk subsamples using the mean reference values.

Additionally, this report shows that calibrating models using a grain density of at least 10 percent is feasible for both protein regression and grain type classification using both CNNs and PLS. We have shown that performance increases with grain density for both protein content regression and grain type classification. However, compared with previous work, we have shown that for grain variety classification, including low grain density samples in the model calibration leads to more accurate predictions on low grain density samples. Simultaneously, on high grain density samples, our classifiers are competitive with previous work that calibrated models only on high grain density samples.

References

- [1] F. J. Anscombe and W. J. Glynn. Distribution of the kurtosis statistic b_2 for normal samples. *Biometrika*, 70(1):227–234, 1983.

- [2] Foss Analytical A/S. Infratec™nova, <https://www.fossanalytics.com/en/products/infratec-nova>.
- [3] M. Barker and W. Rayens. Partial least squares for discrimination. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(3):166–173, 2003.
- [4] R. J. Barnes, M. S. Dhanoa, and S. J. Lister. Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra. *Applied spectroscopy*, 43(5):772–777, 1989.
- [5] L. Benos, A. C. Tagarakis, G. Dolias, R. Berruto, D. Kateris, and D. Bochtis. Machine learning in agriculture: A comprehensive updated review. *Sensors*, 21(11):3758, 2021.
- [6] N. Caporaso, M. B. Whitworth, and I. D. Fisk. Protein content prediction in single wheat kernels using hyperspectral imaging. *Food chemistry*, 240:32–42, 2018.
- [7] R. B. D’Agostino. Transformation to normality of the null distribution of g_1 . *Biometrika*, pages 679–681, 1970.
- [8] R. B. D’Agostino. An omnibus test of normality for moderate and large sample sizes. *Biometrika*, 58(34):1–348, 1971.
- [9] R. B. D’Agostino, A. Belanger, and R. B. D’Agostino Jr. A suggestion for using powerful and informative tests of normality. *The American Statistician*, 44(4):316–321, 1990.
- [10] R. B. D’Agostino and E. S. Pearson. Tests for departure from normality. empirical results for the distributions of b_2 and $\sqrt{b}$. *Biometrika*, 60(3):613–622, 1973.
- [11] E. S. Dreier, K. M. Sørensen, T. Lund-Hansen, B. M. Jespersen, and K. S. Pedersen. Hyperspectral imaging for classification of bulk grain samples with deep convolutional neural networks. *Journal of Near Infrared Spectroscopy*, 30(3):107–121, 2022.
- [12] O.-C. G. Engstrøm, E. S. Dreier, B. M. Jespersen, and K. S. Pedersen. Improving deep learning on hyperspectral images of grain by incorporating domain knowledge from chemometrics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 485–494, 2023.
- [13] O.-C. G. Engstrøm, E. S. Dreier, and K. S. Pedersen. Predicting protein content in grain using hyperspectral deep learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1372–1380, 2021.
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [15] ICR Holford, AD Doyle, and CC Leckie. Nitrogen response characteristics of wheat protein in relation to yield responses and their interactions with phosphorus. *Australian Journal of Agricultural Research*, 43(5):969–986, 1992.
- [16] A. Kamilaris and F. X. Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.
- [17] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis. Machine learning in agriculture: A review. *Sensors*, 18(8):2674, 2018.
- [18] V. Meshram, K. Patil, V. Meshram, D. Hanchate, and S. D. Ramkteke. Machine learning in agriculture domain: A state-of-art survey. *Artificial Intelligence in the Life Sciences*, 1:100010, 2021.
- [19] B. G. Osborne. Near-infrared spectroscopy in food analysis. *Encyclopedia of analytical chemistry: applications, theory and instrumentation*, 2006.
- [20] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transaction on Systems, Man and Cybernetics*, SMC-9(1):62–66, 1979.

- [21] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh. Deep learning vs. traditional computer vision. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1* 1, pages 128–144. Springer, 2020.
- [22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [23] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.
- [24] C. Ren, D.-K. Kim, and D. Jeong. A survey of deep learning in agriculture: techniques and their applications. *Journal of Information Processing Systems*, 16(5):1015–1033, 2020.
- [25] A. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.
- [26] Specim. Specim fx17, <https://www.specim.fi/products/specim-fx17/>.
- [27] WM Strong. Nitrogen requirements of irrigated wheat on the darling downs. *Australian Journal of Experimental Agriculture*, 21(111):424–431, 1981.
- [28] P. H. Westfall. Kurtosis as peakedness, 1905–2014. rip. *The American Statistician*, 68(3):191–195, 2014.
- [29] H. Wold. Estimation of principal components and related models by iterative least squares. *Multivariate analysis*, pages 391–420, 1966.
- [30] S. Wold, C. Albano, W. J. Dunn, K. Esbensen, S. Hellberg, E. Johansson, M. Sjöström, H. Martens, and J. Russwurm. Food research and data analysis. *London: H. Martens and H. Russwurm Jr*, 1983.
- [31] S. Wold, M. Sjöström, and L. Eriksson. Pls-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001.

Appendix B

Technical Report: Barley Germination Dataset

The technical report by Engstrøm et al. (2025b) follows.

A Time Series Dataset of NIR Spectra and RGB and NIR-HSI Images of the Barley Germination Process

Ole-Christian Galbo Engstrøm^{1,2,3}

Erik Schou Dreier¹

Birthe Møller Jespersen⁴

Kim Steenstrup Pedersen^{2,5}

OCGE@FOSS.DK

ESD@FOSS.DK

BPMJ@UCL.DK

KIMSTP@DI.KU.DK

¹ FOSS Analytical A/S, Denmark

² Department of Computer Science (DIKU), University of Copenhagen, Denmark

³ Department of Food Science (UCPH FOOD), University of Copenhagen, Denmark

⁴ University College Lillebælt (UCL), Denmark

⁵ Natural History Museum of Denmark (NHMD), University of Copenhagen, Denmark

1 Introduction

We provide an open-source dataset of RGB and NIR-HSI (near-infrared hyperspectral imaging) images with associated segmentation masks and NIR spectra of 2242 individual malting barley kernels. We imaged every kernel pre-exposure to moisture and every 24 hours after exposure to moisture for five consecutive days. Every barley kernel was labeled as germinated or not germinated during each image acquisition. The barley kernels were imaged with black filter paper as the background, facilitating straight-forward intensity threshold-based segmentation, e.g., by Otsu’s method (Otsu 1979). This dataset facilitates time series analysis of germination time for barley kernels using either RGB image analysis, NIR spectral analysis, NIR-HSI analysis, or a combination hereof.

This report contains an overview of the dataset in Section 2, a description of the image and spectrum acquisition in Section 3, and a description of the barley kernel germination process in Section 4. These sections contain sufficient information for modeling purposes for understanding the conception of the dataset. We kindly thank our sponsors in Section 5. In Appendix A, we give a complete and detailed description of the image acquisition process and the methods used to identify and track individual kernels throughout several imaging sessions.

2 Dataset overview

The dataset consists of images of 90 Petri dishes, most with 25 barley kernels. A polymer grid is placed inside each Petri dish so that each grid cell is occupied by exactly one barley kernel. During acquisition, the Petri dishes were imaged simultaneously by an RGB camera and a NIR-HSI camera, yielding RGB and NIR-HSI images of the same barley kernels under the same conditions. Each Petri dish was imaged before exposure to moisture and after exposure to moisture every 24 hours for the next five days. Table 1 shows an example of tracking a single barley kernel throughout

the six imaging sessions. Figure 1 shows an example of an RGB and NIR-HSI pair of images of the same Petri dish taken simultaneously.

This dataset consists of four different samples of malting barley: two of the variety Prospect and two of the variety Laureate. Table 2 shows the number of kernels from each variety. Table 3 and Figure 2 show the number of kernels classified as germinated after each day.

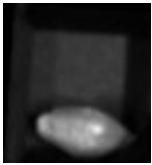
	Pre moisture	Day 1	Day 2	Day 3	Day 4	Day 5
RGB						
NIR-HSI						

Table 1: A single barley kernel imaged before exposure to moisture and every 24 hours after exposure to moisture. A sprout appears on Day 2, meaning that the reference value for germination time for this specific kernel is two days.

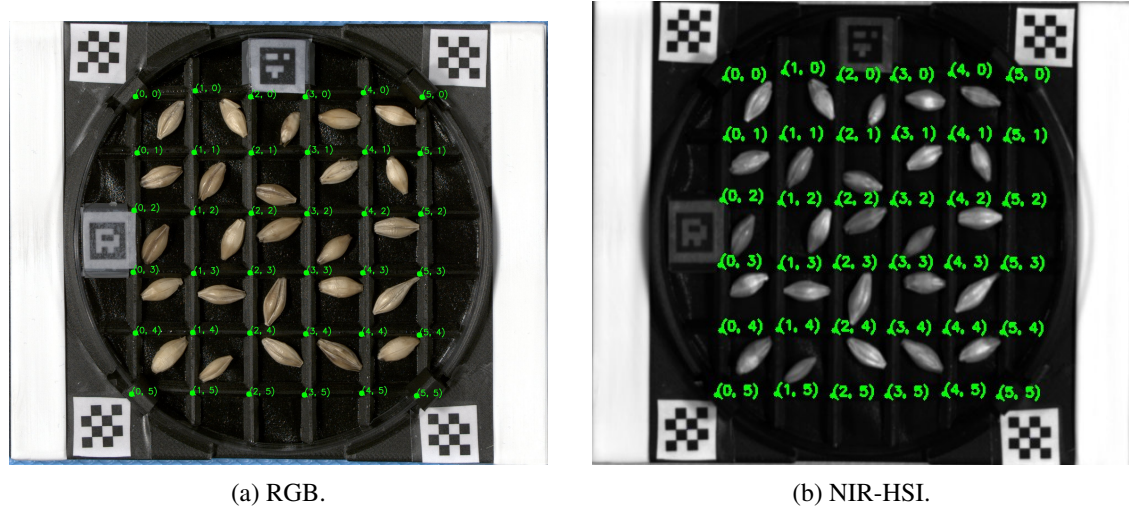


Figure 1: A Petri dish imaged at day 1. The barley kernel inside the top left square is the one shown in Table 1.

	Barley Variety				Total
	Prospect_0	Prospect_1	Laureate_0	Laureate_1	
No. kernels	624	394	624	600	2242
No. Petri dishes	25	16	25	24	90

Table 2: Number of barley kernels and Petri dishes for each of the four barley varieties.

Barley Variety	Germination Day					
	1	2	3	4	5	Any
Prospect_0	4	29	23	15	12	83
Prospect_1	1	0	1	0	3	5
Laureate_0	13	21	25	15	17	91
Laureate_1	9	43	57	58	35	202
All	28	93	106	88	67	382

Table 3: Number of barley kernels that germinated each day out of those shown in Table 2. No kernels were germinated prior to exposure to moisture.

2.1 Dataset availability

We release the dataset and license it under CC BY-NC 4.0. It is available for download at [The University of Copenhagen’s Electronic Research Data Archive](#).

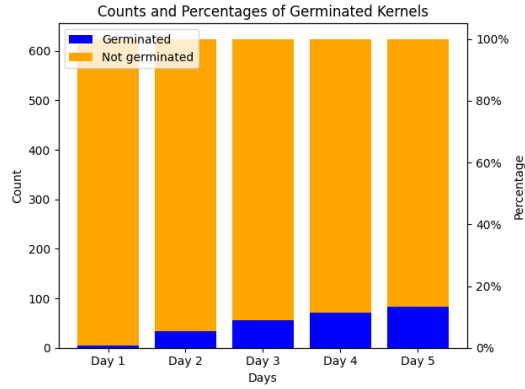
We provide the germination labels and images of extracted kernels, as shown in Table 1. We also provide segmentation masks for all images to distinguish the barley kernels from the background. We also provide an NIR absorbance spectrum computed as an average of the pseudo absorbance inside the segmentation mask for each NIR-HSI image. Should dataset users be interested in applying their grain kernel detection or segmentation methods, we also provide the dataset of entire Petri dish images and the coordinates of the 36 grid corners, as shown in Figure 1.

Initially, we had envisioned 100 Petri dishes with 25 barley kernels each, totaling 2500 kernels. However, due to mold infections and human error, we lost ten Petri dishes and a few individual kernels from the remaining 90 Petri dishes. The retained dataset contains 2242 kernels. Every barley kernel in the retained dataset is imaged all six times with both cameras.

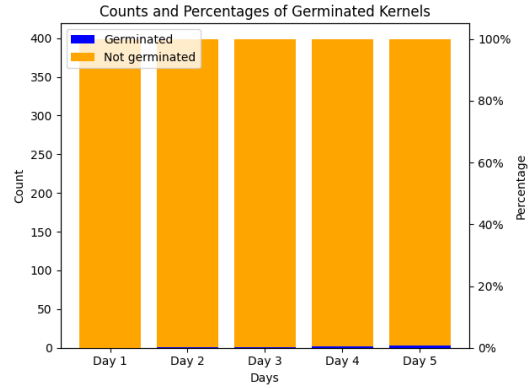
3 Image and spectrum acquisition

3.1 Imaging setup

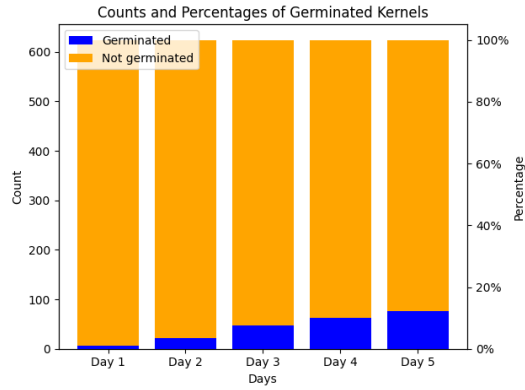
Figure 3 illustrates the imaging setup, which consists of a conveyor band moving a Petri dish with grain kernels underneath two line scan cameras, an RGB camera, and a NIR-HSI camera. On both ends of the setup are three 20-watt halogen light bulbs, ensuring bright illumination throughout the visual and NIR spectrum. During image acquisition, a Petri dish is placed on a plate carrying four quadratic chessboards for pixel size standardization and two windings of white PTFE (polytetrafluoroethylene) foil serving as white references. Dark references were acquired for the NIR-HSI camera



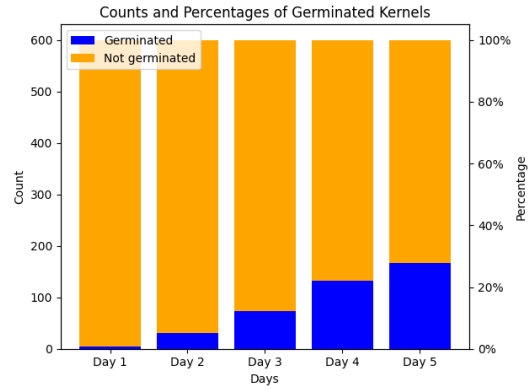
(a) Prospect_0.



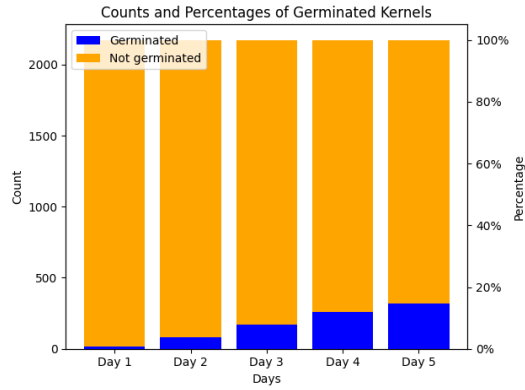
(b) Prospect_1.



(c) Laureate_0



(d) Laureate_1



(e) All varieties.

Figure 2: Histograms showing how the number of germinated kernels evolves from day to day.

by taking images with the shutter closed. For the RGB camera, no dark reference was necessary. See Appendix A for further details on the image standardization process.

The RGB camera is a Basler Ace 2 Pro, while the NIR-HSI camera is a Specim FX17, with 224 uniformly distributed wavelength channels in the 900-1700nm spectrum. The RGB camera uses the standard 8 bits for each color channel, and the images are therefore stored as uint8 arrays. However, The NIR-HSI camera uses 12 bits for each wavelength channel, requiring 1.5 bytes. We store the NIR-HSI images in the mono12p format to facilitate optimal storage. This format packs two 12-bit values sequentially in three bytes (24 bits). Along with the dataset, we provide functions to convert both ways between mono12p and uint16, allowing easy analysis of uint16 images while reducing storage requirements by saving them as mono12p images. The functions are written using the Python library NumPy (Harris et al. 2020).

3.2 Image acquisition and single kernel tracking

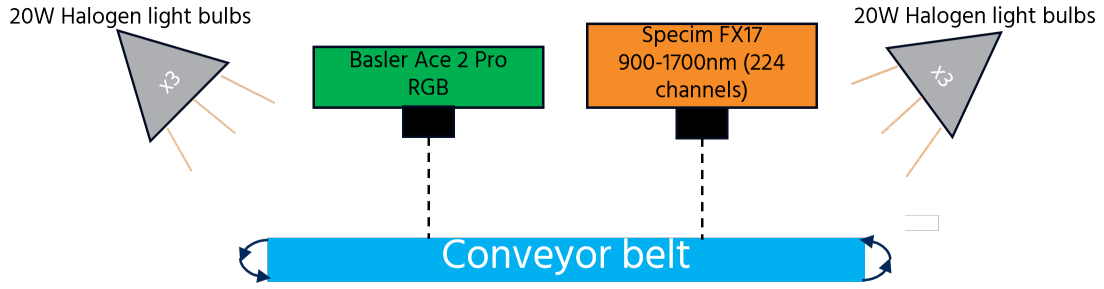
Each Petri dish was imaged simultaneously with an RGB and an HSI camera during image acquisition. The polymer grid inside the Petri dish enables straightforward extraction of individual kernels from entire Petri dish images. To extract a single barley kernel, we rely on the polygon defined by the coordinates of the four surrounding grid points. To extract a rectangular image, we compute the circumscribed rectangle of the polygon, making sure to mask out (set to zero) any part of the rectangle outside the polygon. This process yields single kernel images, as shown in Table 1.

The ArUco codes are fixed on the grids, which are, in turn, fixed on the Petri dishes. Thus, based on the location of the ArUco codes’ eight corners (four per code), we can use random sample consensus (RANSAC, Fischler and Bolles 1981) to estimate an affine transformation matrix from a previous image of the same Petri dish to the current image. Based on the affine transformation matrix, we can track each grid cell and each barley kernel across multiple imaging sessions and between the RGB and NIR-HSI images. Appendix A contains a more detailed description of the single kernel tracking and grid detection.

3.3 Segmentation mask and spectrum acquisition

For each barley kernel, RGB and HSI alike, we compute a segmentation mask by applying Otsu’s method (Otsu 1979) on a grayscale version of the image, computed as a uniform average over all the spectral channels. After applying Otsu’s method, we compute all connected components of the foreground and retain only the largest set of connected components. Chaining Otsu’s method with connected components attempts to remove any reflections in the moist filter paper that might otherwise be classified as foreground by Otsu’s method.

We use the segmentation mask on the NIR-HSI images to compute a mean pseudo-absorbance spectrum for each kernel image. This computation takes the negative logarithm of every pixel inside the mask in the original reflectance image and then averages the result. Sometimes, the hyperspectral camera measured zero reflectance in one or more wavelength channels of a pixel. We deemed such zero measurements to be sensor faults. Since the logarithm of zero is undefined, we instead infer the reflectance value by linear interpolation between neighboring wavelength channels in the same pixel. This procedure alleviated all cases of measured zero reflectance in this dataset. Additionally, as the hyperspectral camera has reduced sensitivity in the first and last then spectral channels, we remove these from the mean spectra to facilitate easier downstream analysis for chemometricians.



(a) An illustration of the imaging setup. The plate carrying the Petri dishes, the white reference, and the chessboards moves along the conveyor belt at a constant velocity. The conveyor belt is illuminated from each side by three 20W halogen light bulbs. The RGB and HSI cameras both operate in line-scan mode.



(b) An example raw output from the RGB camera.

(c) An example raw output from the HSI camera, converted to grayscale for visualization purposes.

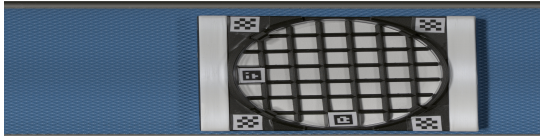
Figure 3: Illustration of the imaging setup and example raw camera outputs.

4 Germination

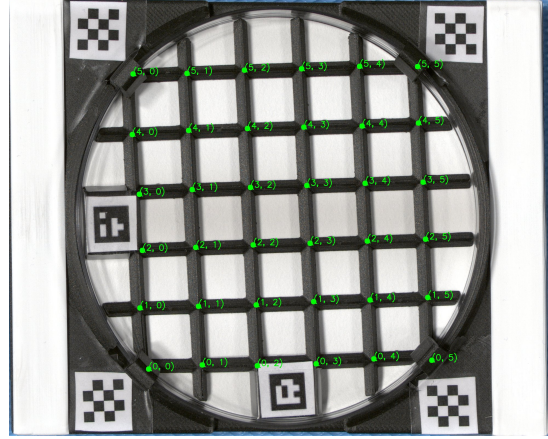
To provoke barley germination, we used The European Brewery Convention (EBC) method 3.6.2 (Anonymous 2019). The following is a description of the method. Two layers of black filter paper were placed at the bottom of each Petri dish, atop which we placed the grid and barley kernels. After the initial image acquisition (pre-moisture), we distributed 4ml tap water evenly across the filter paper, thus allowing the barley kernels to absorb the moisture. Between imaging sessions (Day 1 to Day 5), each Petri dish was contained inside a zip-lock plastic bag to contain the moisture and diminish vaporization. The plastic bags were carefully placed inside a box in a dark room at room temperature. After image acquisition, a human expert labeled each barley kernel as germinated or not germinated based on the RGB images.

Readers and users familiar with barley germination may wonder why so few kernels germinate (visualized in Figure 2). After collecting the dataset, we investigated, revealing the reason. Underneath the 3D printed polymer grid used in Petri dishes (shown in Figure 1) were glue remains from a glue stick used in the 3D printing process. The glue severely inhibits the barley kernels' ability to germinate.

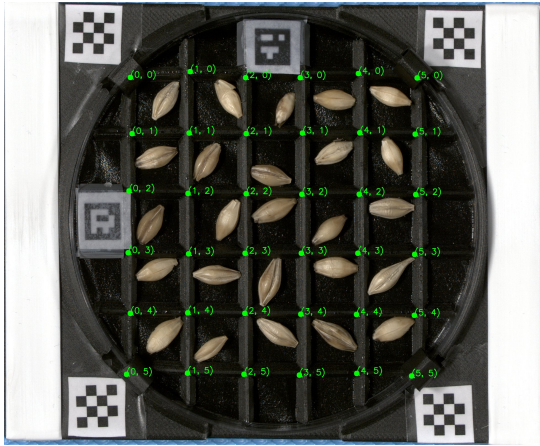
We investigated if there was a difference in germination degree between black versus white filter paper, imaging (placing the Petri dishes under the bright illumination from the halogens shown in Figure 3) versus no imaging and using a grid versus not using a grid. This investigation showed a significant difference between using and not using the grid. We subsequently tried to isolate whether it was due to the grid's material or the underlying glue remains. The investigation clearly showed that the glue remains were the inhibiting factor.



(a) Raw RGB image with white filter paper for easier localization of black grid lines.



(b) Figure 4a after extracting the plate, applying white balance correction, size correction, and the grid finding algorithm.



(c) The Petri dish from Figure 4a with black instead of white filter paper for higher contrast with the barley kernels. An affine transformation matrix from Figure 4b to this image is computed based on the 8 corners of the ArUco codes. This affine transformation matrix is then used to move the grid from Figure 4b to this image. This image is acquired 1 day after exposure to moisture.



(d) A grayscale visualization of the HSI image corresponding to the RGB image in Figure 4c. The placement of the grid is derived based on an affine transformation between the centers of the 4 chessboards in Figure 4c and those in this image.



(e) A cutout of the grid cell in Figure 4c defined by a polygon between the grid points $\{(0, 0), (1, 0), (1, 1), (0, 1)\}$.



(f) A cutout of the grid cell in Figure 4d defined by a polygon between the grid points $\{(0, 0), (1, 0), (1, 1), (0, 1)\}$.

Figure 4: An overview of the image processing pipeline using Petri dish 26 as an example. Notice how the grid corners are uniquely defined by the detection in the RGB image in Figure 4b. Localization of the same grid in a subsequent RGB or HSI image can be derived based on corresponding affine transformation matrices based on the ArUco codes' corners and the chessboards' centers, respectively.

Still, even with the glue inhibiting the germination process, we envision that the dataset can still be used for modeling the germination process. In particular, the dataset can be used directly to classify whether or not a barley kernel is germinated. However, when training models to predict the evolution of the germination process, the bias of this dataset should be considered if transferring to applications outside this dataset. Predicting whether a given barley kernel will germinate in the future depends on the subsequent treatment that the barley kernel receives. In this dataset, the barley kernel received treatment with water contaminated with glue remains, which is not representative of the typical process without glue remains. Thus, a given barley kernel may have germinated given the proper treatment, although it would not have germinated with the treatment we gave.

5 Funding

This dataset has been acquired as part of an industrial Ph.D. project receiving funding from FOSS Analytical A/S and The Innovation Fund Denmark. Grant number 1044-00108B.

6 Acknowledgments

We would like to thank our amazing colleagues at FOSS Analytical A/S for their help in constructing the imaging setup and writing its control software, designing and constructing the plate carrying the Petri dishes and grids used to separate individual kernels, and acquiring the images.

References

- Anonymous. European brewevery convention, 2019.
- R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. doi:[10.1145/361237.361242](https://doi.org/10.1145/361237.361242).
- M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi:[10.1145/358669.358692](https://doi.org/10.1145/358669.358692).
- C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett,

A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, September 2020. doi:[10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2).

N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transaction on Systems, Man and Cybernetics*, SMC-9(1):62–66, 1979. doi:[10.1109/TSMC.1979.4310076](https://doi.org/10.1109/TSMC.1979.4310076).

Appendix A. Image processing

In this section, we describe how to identify individual grid cells and track them across different images of the same Petri dish. We use Petri dish 26 as an example, but the process is the same for every Petri dish.

The overall strategy is to initially detect each of the unique grids using the RGB camera once. Then, for subsequent RGB images, we localize the grid based on an affine transformation from the initially detected grid and the current RGB image. This affine transformation is based on the eight corners of the ArUco codes. Any affine transformation of the grid can be computed based on the ArUco codes as they are fixed on the grid.

We can not count on the detection of the ArUco codes to locate the grid on an HSI image. This is because the paper on which the ArUco codes are printed absorbs water and thus becomes very dark in the NIR wavelengths, diminishing the contrast between the otherwise white paper and the black ArUco code. Instead, we opt for an affine transformation based on detecting the plate-fixed chessboards on the HSI image and its corresponding RGB image - the latter for which we derived the grid coordinates described above. As the Petri dish does not move relative to the plate between acquiring a pair of RGB and HSI images, we can use this affine transformation matrix to localize the grid coordinates on an HSI image, given the grid coordinates on its corresponding RGB image.

Using affine transformations instead of executing the entire grid-detection pipeline for every image acquisition gives robustness and consistency in localizing grid coordinates across multiple images of the same physical Petri dish. Additionally, this strategy allows switching between filter papers (backgrounds) to maximize contrast between the foreground. Therefore, we use a white filter paper for the initial grid detection to maximize contrast with the black grid. Then, we use black filter paper for subsequent images to maximize contrast with the brown barley kernels and, importantly, the white sprouts.

In Section A.1, we show how to initially detect the grid coordinates from an RGB image with a white filter paper background. Then, in Section A.2 and Section A.3, we show how to localize the same grid in subsequent RGB and HSI images, respectively, with grain and black filter paper background. This strategy allows us to track individual grid cells (and therefore barley kernels) across different images of the same Petri dish.

A.1 Initial grid detection

The entire process of grid detection is outlined in Figure 5. The raw RGB camera output (Figure 5a) is grayscaled (Figure 5b) to allow for an Otsu thresholding (Otsu 1979) so we can identify the two largest connected components (Figure 5d), corresponding to the location of the two white references (Figure 5e). For each color channel, we can then compute an average intensity for each row of the white reference (Figure 5f). As the camera is used in line-scan mode, we can use this result to

estimate how much light is hitting each point on the plate. We use the third quantile of the intensity of each row to get a robust value against shadows and small dust particles that may have stuck to the white references. We also use the location of the white references as a bounding box to extract the plate (Figure 5g). We can then use the white-reference intensities to adjust the intensity for each row to mitigate the effects of non-uniform illumination (Figure 5h). The intensity correction is based on Equation (1). Here i, j, c are the row (height), column (width), and channel (depth) indices, respectively. W and D are white and dark references, respectively, and both are 2-dimensional containing, representing the measured white and dark intensities for each channel, c , at each row, i . For the RGB images $D_{i,c} = 0$ for all i, c as the RGB camera has no dark current.¹

$$I_{i,j,c}^{\text{corrected}} = \frac{I_{i,j,c}^{\text{raw}} - D_{i,c}}{W_{i,c} - D_{i,c}} \quad (1)$$

We can then detect the chessboard squares, and because they are squares in the physical world, we can use their sizes to estimate the factor by which to squish the horizontal dimension of the image. Because the horizontal dimension is the largest, and the camera works in line-scan by acquiring one vertical line at a time, there is no loss of information. We downsize the image using bilinear interpolation (Figure 5i). The process described thus far finds the plate, applies the white standardization, and size correction. This provides a strong starting point for the subsequent analysis to identify the grid cells.

First, we detect the two ArUco codes. Each is identified by its unique id and the four corners. These will be used for subsequent analysis. Then, we perform smoothing followed by edge detection, the result of which is the input to a Hough Circle Transform (Duda and Hart 1972). After thresholding the circle size, if multiple circles remain, we choose the one with its center closest to the center of the image (Figure 5j). The goal is to segment the Petri dish for further analysis (Figure 5k). After segmenting the Petri dish, we perform smoothing and subsequent edge detection, then masking the ArUco codes. We feed the result to a Hough Line Transform (Duda and Hart 1972) to detect the borders of the grid lines Figure 5l. We average the detected borders to get an estimate of the location of the grid lines (Figure 5m). We then compute the intersection between all the grid lines, retaining only intersections inside the image’s borders (Figure 5n).

Now, we have detected points that correspond to grid intersections. Further analysis will refine this estimate to find a unique orientation of the grid. Notice how points in the far left of Figure 5n do not correspond to grid intersections. However, the subsequent analysis is robust enough to consider this.

Consider now Figure 5o. For each ArUco code, we compute its center by averaging the coordinates of its corners. Then, from the center, we compute a vector (red) to the center of the Hough Circle found in Figure 5j. We then compute vectors from the ArUco codes center to all (green) intersections. We retain the closest two that form an angle of less than 90 degrees with the corresponding (red) vector to the center.

For each ArUco code, consider now the pair of points found before. These two points each lie on precisely two lines, and the two points share exactly one of the lines. We now choose an arbitrary point from the pair (the final grid is independent of the point choice). Now, choose the line *not shared by the other point*. In Figure 5p, we paint magenta (yellow) the line chosen from the point pair of ArUco code with the lowest (highest) id.

1. We blocked the RGB camera’s lens with a black cap and let it capture several million pixels while having the halogen illumination system active. Every single pixel in the resulting image had a value of $\{R,G,B\} = \{0, 0, 0\}$.

Next, consider Figure 5q. On the magenta (yellow) line, we find all (green) intersection points that lie on this line. We then use the previously computed vectors from the center of the ArUco code with the lowest (highest) id to these points. We retain those that form an angle of less than 90 degrees with the (red) vector from the ArUco code to the center. We sort the vectors by length and retain only the six closest vectors, shown in yellow (magenta).

Now, for each point on the yellow (magenta) vectors, identify the other line on which it lies. That is, not the magenta (yellow) line already detected. This other line is then marked yellow (magenta), and it is the line for which the y-coordinate (x-coordinate) of the grid is constant. The y-coordinate (x-coordinate) value is determined by the length of the vector associated with the point used to detect it. This is all visualized in Figure 5r.

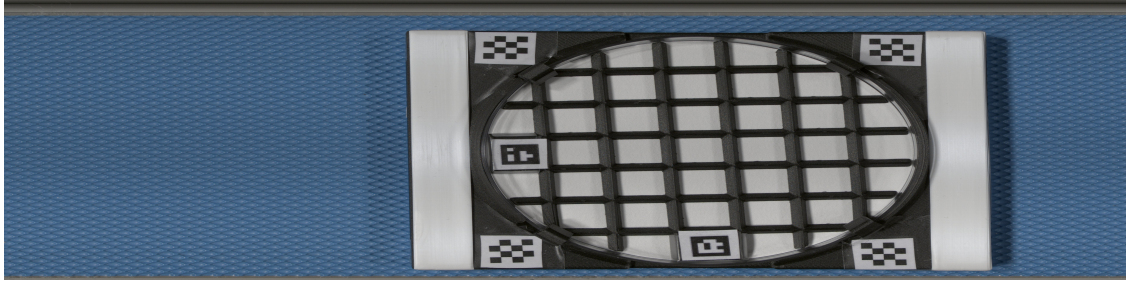
Now, we can compute the intersection points between all magenta and yellow lines. Each point can then be given a unique (x, y) coordinate based on the magenta and yellow line on which it lies. To robustify the grid detection, we move the points to a local minimum in intensity space as a final step. This step adjusts the grid points to lie in the middle of a physical grid intersection. We do not adjust points with coordinates $(x, y) \in \{(0, 0), (0, 5), (5, 0), (5, 5), (3, 0), (2, 0), (0, 3), (0, 2)\}$. This is because these points lie near the ArUco codes or the borders of the Petri dish, and, as such, a local intensity minimum is not a good estimate of the physical grid intersection. The final grid detection is shown in Figure 5s.

A.2 RGB grid localization

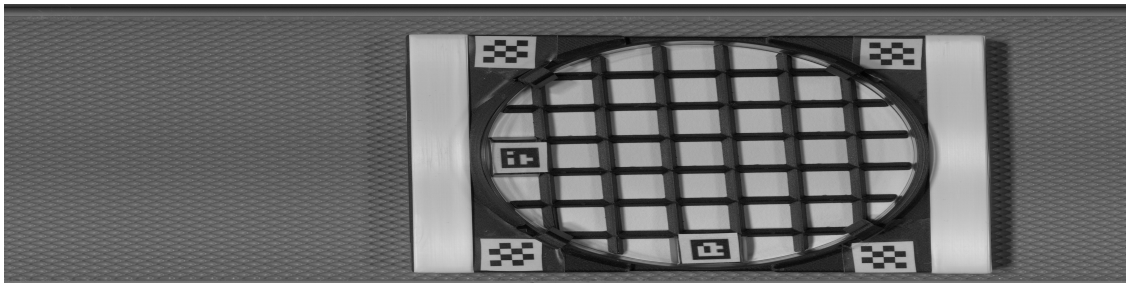
In this section, we explain how to use the result from Section A.1 to derive the location of the grid of Petri dish 26 in a subsequent RGB image of that same Petri dish. Here, we use the image acquired on day 1 as an example. The entire process is outlined in Figure 6. As previously, we extract the plate from the image, perform white-correction with the white references, and use the chessboards to correct size. This takes us from the raw RGB camera output in Figure 6a to the white- and size-corrected plate image in Figure 6i. We then detect the ArUco codes' 8 corners on Figure 6i and use the robust RANSAC algorithm (Fischler and Bolles 1981) to estimate an affine transformation between those and the corresponding ArUco code corners in Figure 5i. Given this transformation, we can directly transform the grid from Figure 5s to our current RGB image (Figure 6j). We can then use the coordinates to extract single grid cells and thus single barley kernels (e.g., Figure 6k).

A.3 HSI grid localization

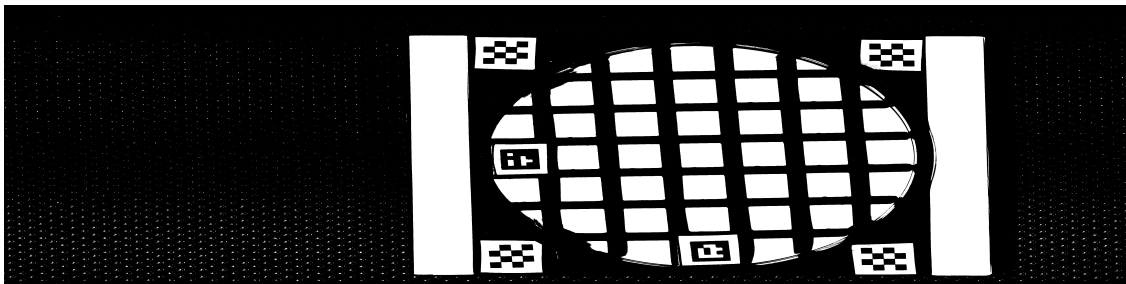
In this section, we explain how to use the result from Section A.2 to derive the location of the grid of Petri dish 26 in the corresponding HSI image of Petri dish 26 acquired on day 1. The entire process is outlined in Figure 8. As with the RGB images, we use a combination of Otsu thresholding (Otsu 1979) and connected components to derive the white references, which are then used to detect the plate and white correction. However, unlike the RGB camera, the HSI camera has a non-zero dark signal that must be corrected. To measure the dark signal, we take an image with the camera's shutter closed and compute the row-wise median of that dark image. Then, to ensure alignment with the white reference signal, we retain only the rows of the dark signal corresponding to the rows of the HSI image that also contain at least one pixel of white reference. The intensity (white and dark) correction is then computed based on Equation (1) using the values shown in Figure 7e and Figure 7f.



(a) The raw image output by the RGB camera with white filter paper for easier localization of black grid lines.



(b) Grayscale version of Figure 5a achieved by averaging the color channels.



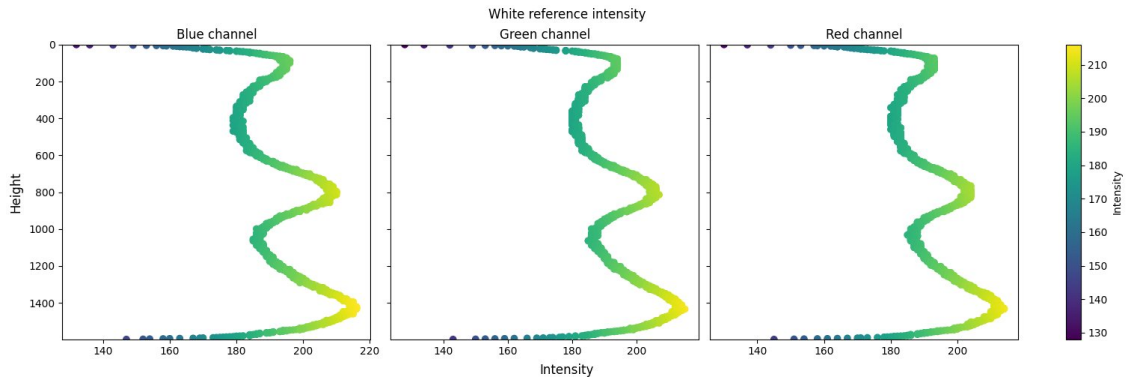
(c) Binary version of Figure 5b computed using Otsu's method (Otsu 1979).



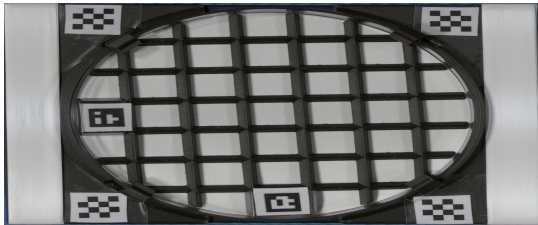
(d) The two largest connected components from Figure 5c. They correspond to the location of the two white references.



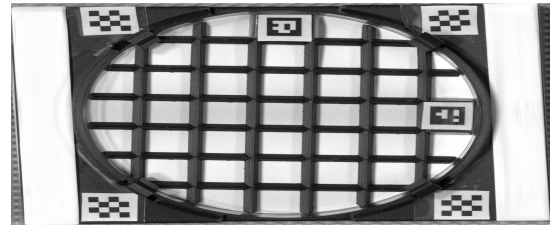
(e) Applying Figure 5d as a binary mask to Figure 5a to extract the white references.



(f) The row-wise third quantile of intensity of the white references from Figure 5e. Any pixel in Figure 5e that is not part of any white reference, as determined by Figure 5d, is ignored in this computation.



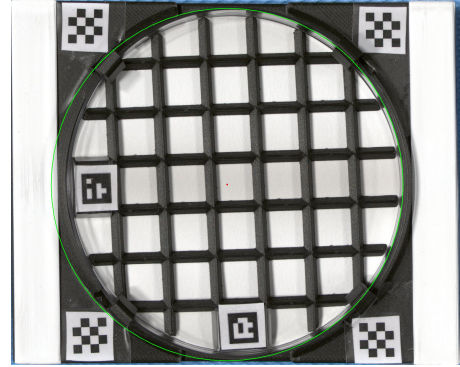
(g) Figure 4a cut off at the borders of the white references as determined by Figure 5d.



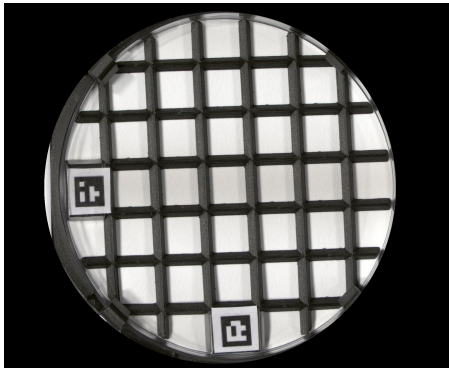
(h) The image resulting from white-correction of Figure 5g by Equation (1) using the values from Figure 5e as shown in Figure 5f. This image is grayscale to enable detection of the chessboards.



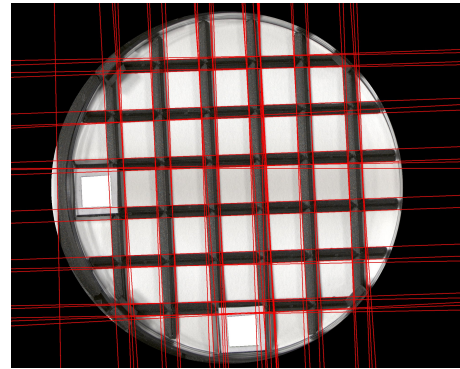
(i) The image resulting from size-correction of the colorized version of Figure 5h using bilinear interpolation with a factor based on the ratio between the height and width of the chessboard squares.



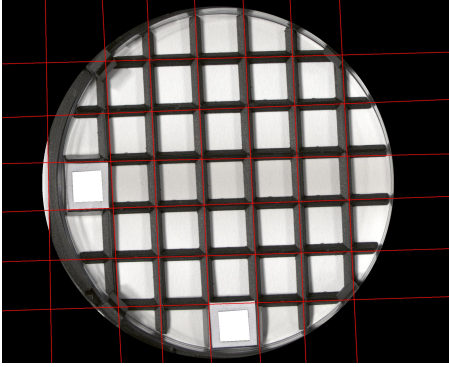
(j) Approximating the location of the Petri dish in Figure 5i with a Circle Hough Transform (Duda and Hart 1972). The circle is shown with its border (green) and its center (red). Smoothing and edge detection (neither shown) are performed prior the Circle Hough Transform.



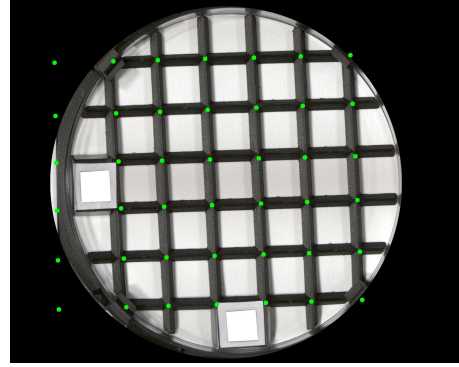
(k) The image resulting from masking Figure 5i with the circle found in Figure 5j.



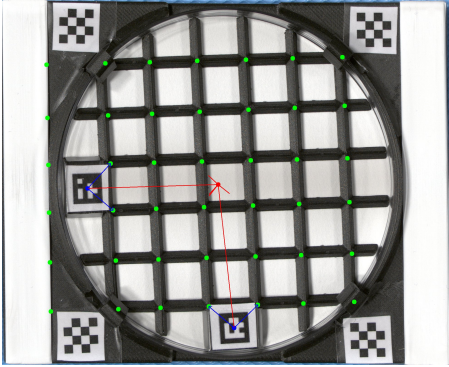
(l) Masking the ArUco codes and performing a Hough Line Transform. Smoothing and edge detection (neither shown) are performed prior to the Hough Line Transform.



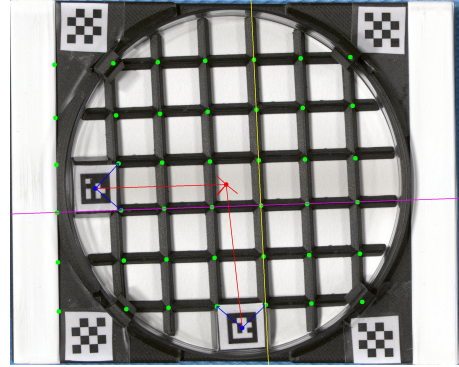
(m) The result of averaging similar Hough lines from Figure 5l.



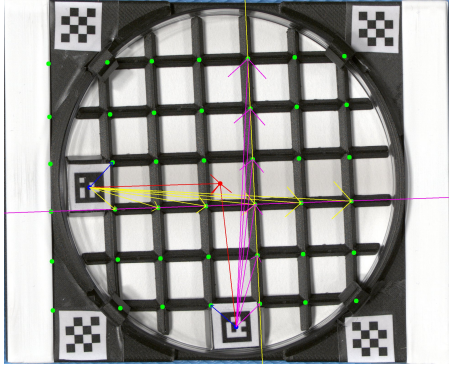
(n) Find the intersections between all the lines in Figure 5m. Discard any intersections that lie outside the image.



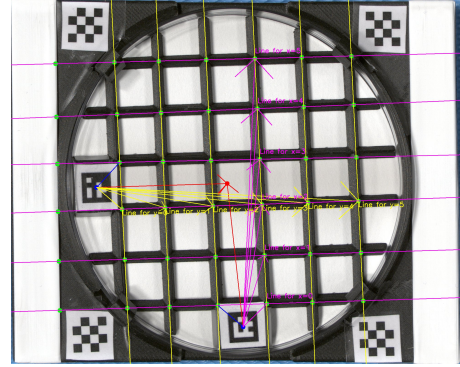
(o) From the center of each ArUco marker, find the vector (shown in red) to the center of the circle found in Figure 5j. Then, for each ArUco code's center, find the two closest points that form an angle of less than 90 degrees with the corresponding red vectors (shown by blue vectors).



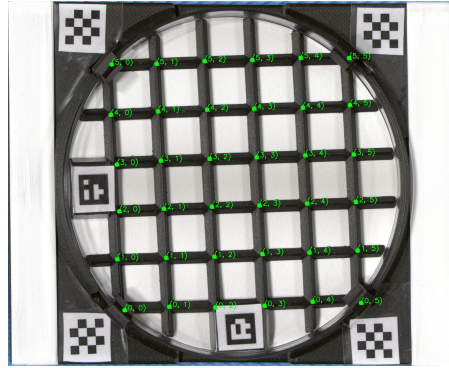
(p) Each of the two pairs of points found in Figure 5o lie on a line found in Figure 5m. Choose an arbitrary point from each of the two pairs and choose the line *not shared by the other point from the same pair*. The magenta (yellow) line goes along the point chosen from the pair closest to the ArUco code with the lowest (highest) id.



(q) Find the intersections from Figure 5n that lie on the magenta (yellow) line found in Figure 5p. Compute the vector from the center of the ArUco code with the highest (lowest) id to the intersections and discard any where the angle between the vector and the corresponding red vector (found in Figure 5o) is more than 90 degrees. The retained vectors are shown in yellow (magenta).

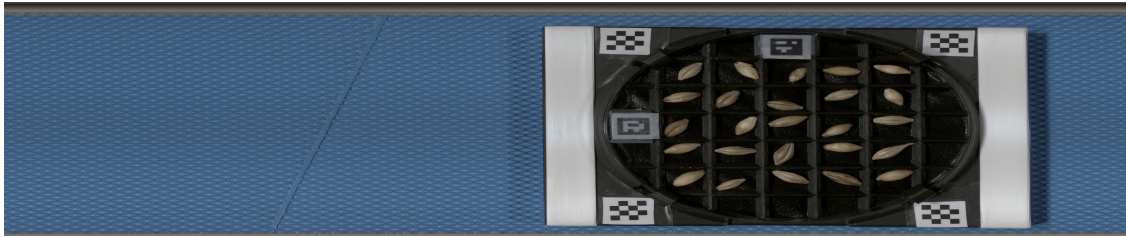


(r) Enumerate the retained yellow (magenta) vectors from Figure 5q by their lengths. Each yellow (magenta) vector is the intersection between the magenta (yellow) line and a yellow (magenta) line. This intersecting yellow (magenta) line is where $y \in \{0 \dots 5\}$ ($x \in \{0 \dots 5\}$) as determined by its enumeration.

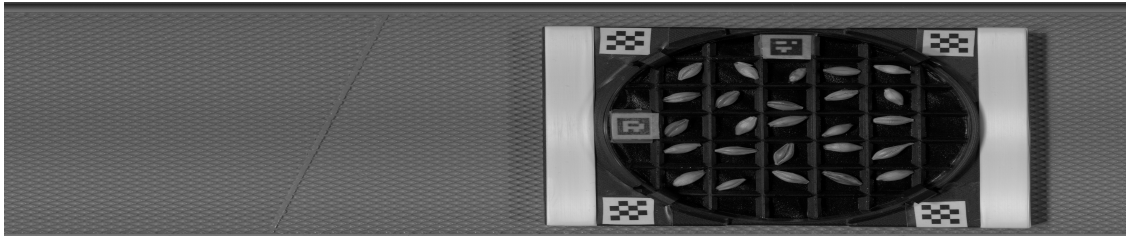


(s) Identify the intersection points between the yellow and magenta lines from Figure 5r by their (x, y) coordinates. Then, adjust the location of each point to lie in a local minimum in intensity space (after grayscaleing and smoothing, neither shown). The points in the corners and adjacent to the ArUco codes are not adjusted.

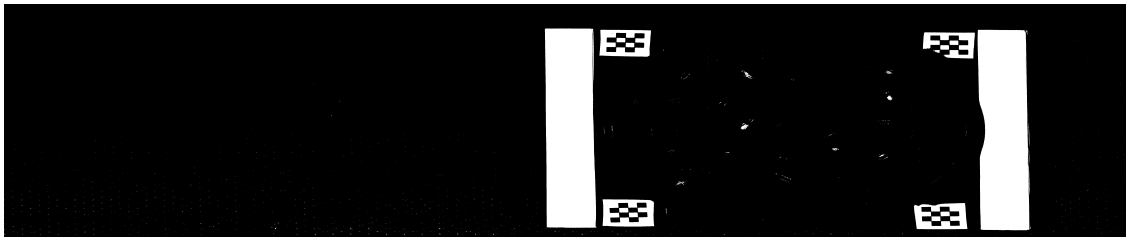
Figure 5: An overview of the image processing pipeline using Petri dish 26 as an example. Notice how the grid coordinates are uniquely defined by the detection in the RGB image in Figure 4b. Localization of the same grid in a subsequent RGB or HSI image can be derived based on corresponding affine transformation matrices based on the ArUco codes' corners and the chessboards' centers, respectively.



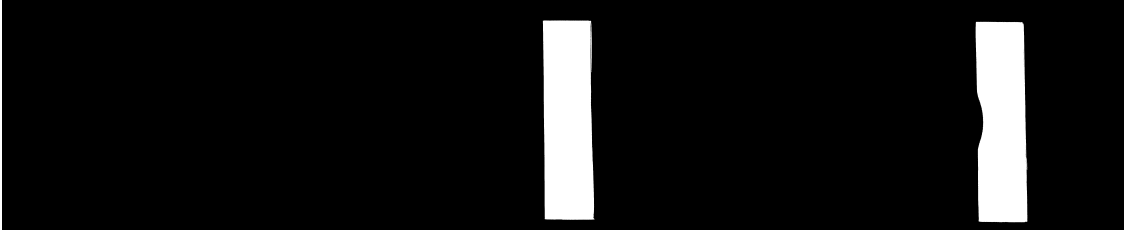
(a) The raw image output by the RGB camera with the black filter paper and barley kernels.



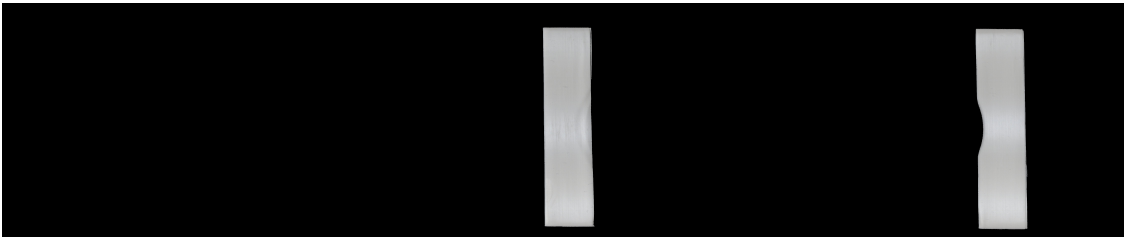
(b) Grayscale version of Figure 6a achieved by averaging the color channels.



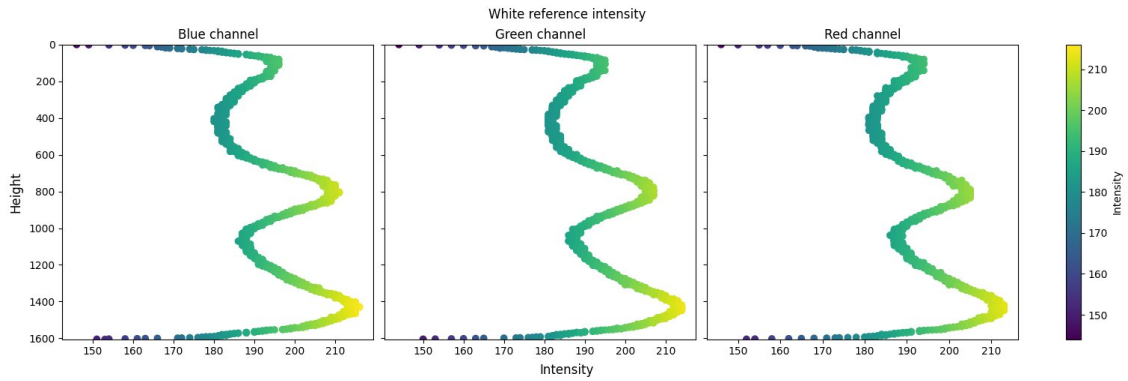
(c) Binary version of Figure 6b computed using Otsu's method (Otsu 1979).



(d) The two largest connected components from Figure 6c. They correspond to the location of the two white references.



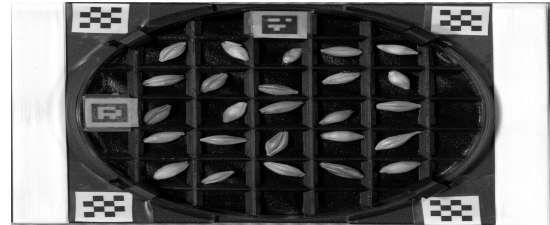
(e) Applying Figure 6d as a binary mask to Figure 6a to extract the white references.



(f) The row-wise third quantile of intensity of the white references from Figure 6e. Any pixel in Figure 6e that is not part of any white reference, as determined by Figure 6d, is ignored in this computation. The 8-bit color channels give an intensity range between 0 and 255, both ends inclusive.



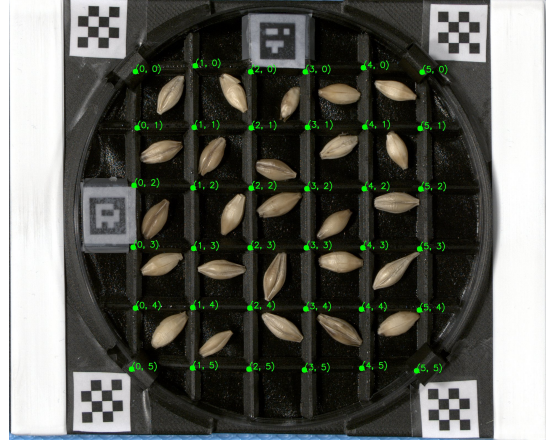
(g) Figure 6a cut off at the borders of the white references as determined by Figure 6d.



(h) The image resulting from white-correction of Figure 6g by Equation (1) using the values from Figure 6e as shown in Figure 6f. This image is grayscale to enable detection of the chessboards.



(i) The image resulting from size-correction of the colorized version of Figure 6h using bilinear interpolation with a factor based on the ratio between the height and width of the chessboard squares.



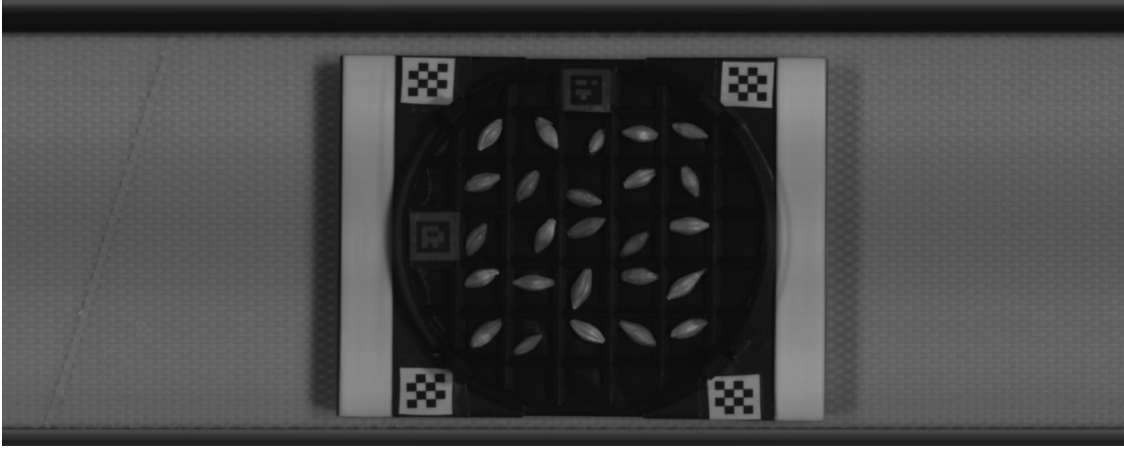
(j) The Petri dish from Figure 6a. Using the RANSAC algorithm (Fischler and Bolles 1981), an affine transformation matrix from Figure 5s to Figure 6i is computed based on the 8 corners of the ArUco codes. This affine transformation matrix is then used to move the grid from Figure 5s to Figure 6i without having to redetect the grid.



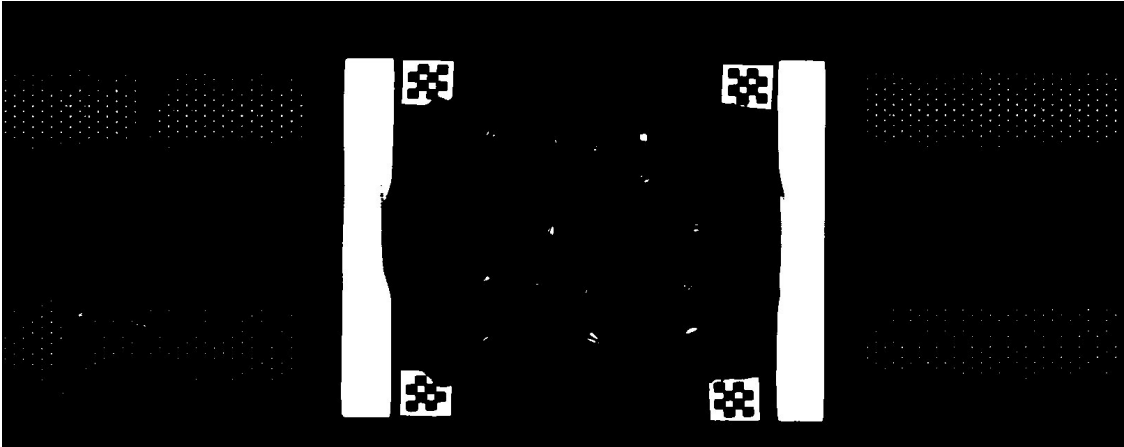
(k) A cutout of the grid cell in Figure 6j defined by a polygon between the grid points $\{(0, 0), (1, 0), (1, 1), (0, 1)\}$.

Figure 6: An overview of the image processing pipeline using Petri dish 26 as an example. This image is taken 1 day after exposure to moisture. The stage of the pipeline shown here is that of localization of the grid shown in Figure 6j. Due to the previous detection of the same grid in Figure 5s, localization of the grid in this image only requires detection of the ArUco codes.

Subsequent size correction using bilinear interpolation with a factor based on the size of the chessboard squares is then applied to standardize the pixel size. Although less profound than with the RGB images, the width of the chessboards is larger than their height. This, coupled with the fact that the HSI camera acquires one column of the image at a time (line scan), ensures no loss of information when downsizing the image. The result is shown in Figure 7i. Using the RANSAC algorithm (Fischler and Bolles 1981), we then use the centers of the four chessboards in Figure 7i and those in Figure 6i to transform the grid from Figure 6j to Figure 7j. This, in turn, allows us to cut out individual grid cells (and thus barley kernels) for subsequent analysis. For example, the same physical barley kernel is imaged at the same time in RGB (Figure 6k) and HSI (Figure 6k), enabling direct comparison between the two modalities.



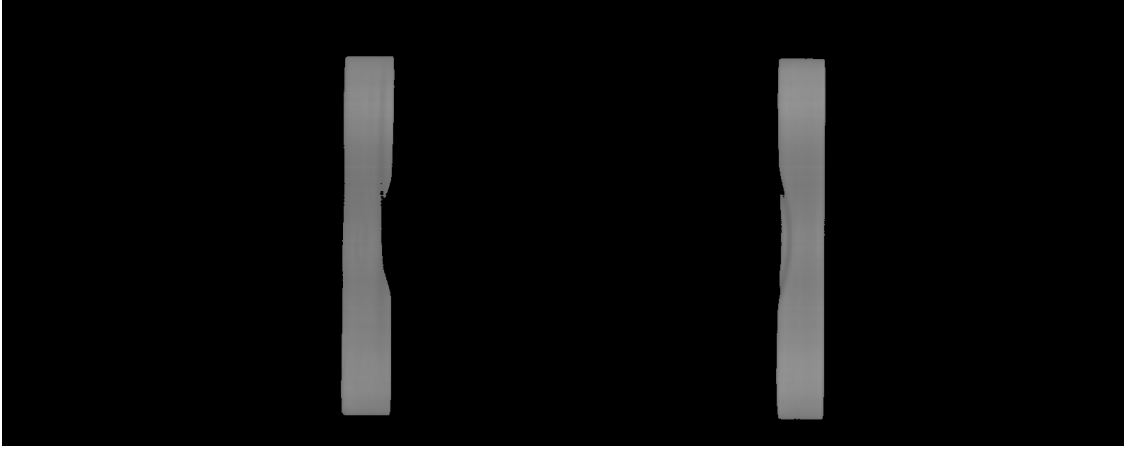
(a) Raw output from the HSI camera, converted to grayscale for visualization purposes.



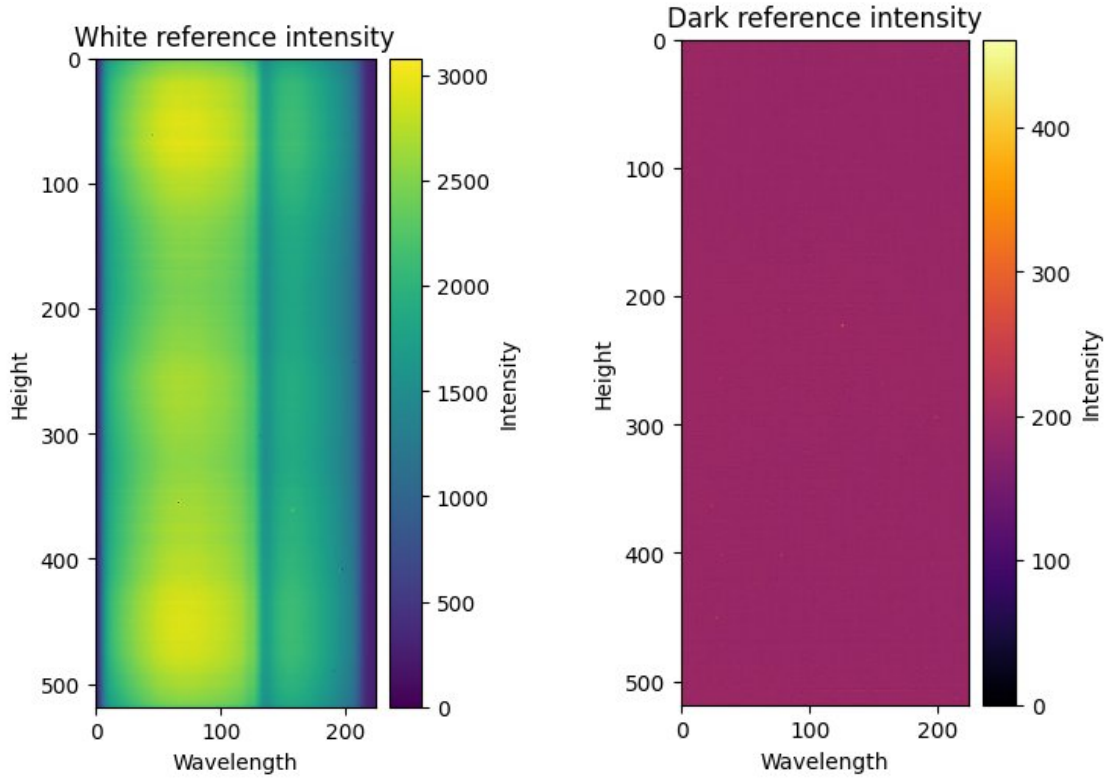
(b) A binary version of Figure 7a computed using Otsu's method (Otsu 1979).



(c) The two largest connected components from Figure 7b. They correspond to the location of the two white references.

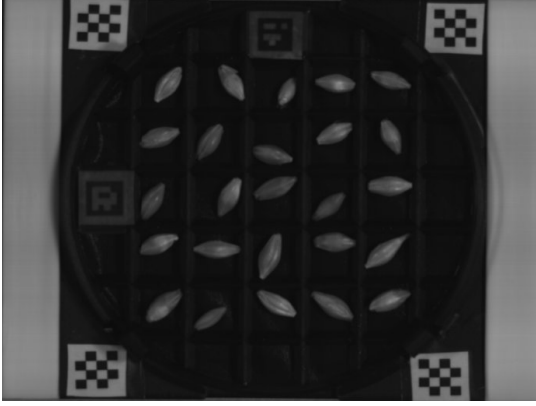


(d) Applying Figure 7c as a binary mask to the HSI version of Figure 7a to extract the white references.

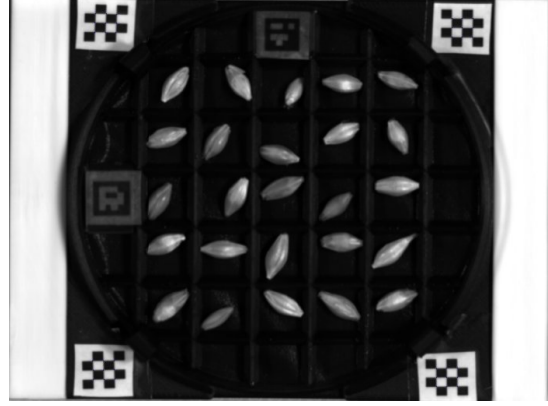


(e) The row-wise third quantile of intensity of the white references from Figure 7d. Any pixel in Figure 7d that is not part of any white reference, as determined by Figure 7c, is ignored in this computation. The 12-bit spectral channels give an intensity range between 0 and 4095, both ends inclusive.

(f) The row-wise median of intensity of a dark reference image taken with the HSI cameras' shutter closed. We retain only rows of the dark image corresponding to the rows where the white reference is located as determined by Figure 7c. This ensures consistency between the white and dark correction. The 12-bit spectral channels give an intensity range between 0 and 4095, both ends inclusive.



(g) Figure 7a cut off at the borders of the white references as determined by Figure 6d.



(h) The image resulting from white- and dark-correction of the HSI version of Figure 7a using the values from Figure 7e and Figure 7f. This image is grayscale to enable detection of the chessboards.



(i) Grayscale version of the image resulting from size-correction of the HSI version of Figure 7h using bilinear interpolation with a factor based on the chessboard squares.



(j) The Petri dish from Figure 7a. Using the RANSAC algorithm (Fischler and Bolles 1981), an affine transformation matrix from Figure 6j to Figure 7i is computed based on the four chessboard centers. This affine transformation matrix is then used to move the grid from Figure 6j to Figure 7i without having to redetect the grid.



(a) A grayscale cutout of the grid cell in Figure 7j defined by a polygon between the grid points $\{(0, 0), (1, 0), (1, 1), (0, 1)\}$.

Figure 8: An overview of the image processing pipeline using Petri dish 26 as an example. This image is taken 1 day after exposure to moisture. The stage of the pipeline shown here is that of localization of the grid shown in Figure 7j. Due to the previous detection of the same grid in Figure 5s and subsequently in Figure 6j, localization of the grid in this image only requires detection of the chessboards.