

Training Variation of Physically-Informed Deep Learning Models

Ashley Lenau^{1,2}, Dennis Dimiduk^{3,1}, Stephen R. Niezgoda^{1,4*}

¹Department of Materials Science and Engineering, The Ohio State University, Columbus, 43210, Ohio, United States.

²Center for Integrated Nanotechnologies, Sandia National Laboratories, Albuquerque, 87185, New Mexico, United States.

³BlueQuartz Software, Springboro, 45066, Ohio, United States.

⁴Department of Mechanical and Aerospace Engineering, The Ohio State University, Columbus, 43210, Ohio, United States.

*Corresponding author(s). E-mail(s): niezgoda.6@osu.edu;
Contributing authors: atlenau@sandia.gov;
dennis.dimiduk@bluequartz.net;

Abstract

A successful deep learning network is highly dependent not only on the training dataset, but the training algorithm used to condition the network for a given task. The loss function, dataset, and tuning of hyperparameters all play an essential role in training a network, yet there is not much discussion on the reliability or reproducibility of a training algorithm. With the rise in popularity of physics-informed loss functions, this raises the question of how reliable one's loss function is in conditioning a network to enforce a particular boundary condition. Reporting the model variation is needed to assess a loss function's ability to consistently train a network to obey a given boundary condition, and provides a fairer comparison among different methods. In this work, a Pix2Pix network predicting the stress fields of high elastic contrast composites is used as a case study. Several different loss functions enforcing stress equilibrium are implemented, with each displaying different levels of variation in convergence, accuracy, and enforcing stress equilibrium across many training sessions. Suggested practices in reporting model variation are also shared.

Keywords: Physics-informed machine learning, model variation, training reproducibility, microstructure modeling

1 Introduction

Repeatable experiments are the cornerstone of the scientific method. They solidify our understanding and allow scientists to improve theory, apparatus, analysis tools, simulations, etc. A lack of reproducibility in an experiment can be a sign of a flawed understanding of the principles behind the experiment or a flaw in the design of the experiment itself. Materials-based direct numerical simulations, on the other hand, are derived from theory and are typically deterministic - thus essentially 100% repeatable ¹. Error in a computational model is typically a result of model form error, incomplete or incorrect physics in the model, or parameter uncertainty, which is often associated with uncertainty in material properties. Computational uncertainty quantification involves characterizing these uncertainties through the comparison of simulation results with experimental data as part of a Verification and Validation process. The first step to quantifying the uncertainty of a simulation method is understanding the systematic variance and its sources, which will provide information on a method’s reproducibility. In this paper, variance is of particular interest in machine learning (ML) models where different random initializations could lead to different levels of performance for the same method.

In recent years, ML models have seen a rise in popularity in computational materials science. ML and deep learning (DL) models have been used for experimental analysis [1, 2], materials discovery [3, 4], material property prediction [5, 6], and microstructure generation [7, 8], to name a few. Despite these advancements, ML has faced criticism for its “black box” nature in that many question what the model is learning during the training phase [9]. While ML models used in materials science are typically deterministic during execution, meaning that the same inputs will produce the same outputs, modern strategies for ML training utilize stochastic sampling of the training data and parameter initialization, resulting in different network behavior from one training cycle to another. Predictions of ML models are non-deterministic because of the many random variables that go into training and initializing the model, making the variation and repeatability of these models a concern in scientific applications.

The repeatability of training a ML model is an active area of research, and several publications have outlined the many variables that can cause variability in model performance. Ref. [10] implemented several networks with identical training and showed that performance can vary as much as 10.8%. They also show that a network trained 16 times with the same random seed can have drastically different convergence times, as large as a 145.3% relative difference. Ref. [11] showed that different classifiers had different levels of variation in performance for different datasets. Ref. [12] showed that different hyperparameter optimization and learning procedures lead to different variations in performance for different networks. Ref. [13] initialized a network with the same random seed and implemented the network using different versions of TensorFlow, resulting in different accuracy across several training sessions. Even networks trained with a fixed random seed can have variances in biases from different training sessions, as demonstrated by Ref. [14]. The precision used to train the network

¹Of course stochastic simulations are important for analyzing the performance of systems whose behavior depends on the interaction of random processes, typically described by probability models. However, the discussion of stochastic simulation is outside the scope of this work.

(double versus single precision) also affects variability, with Ref. [15] showing that double precision leads to more uniform predictions. Having all these factors causing model performance variations, there is a need to provide code, check for reproducibility through re-implementation [16], and also report a model’s variation. Reporting a network’s improvement compared to another implementation is statistically meaningless unless the model performance variation is also reported, as noted by Pham et al. (2021). For example, Pham and co-authors showed that a network reporting a 0.8% increase in accuracy had a variation in accuracy of 2.9%, meaning that the “improvement” is within the bounds of accuracy variability.

Reporting the random seed initialization used for training (as suggested by [17]’s best practice guide for material scientists) may ensure reproducibility for a single random seed, model architecture, and training strategy, but does not evaluate the consistency of a model architecture or training strategy. This is particularly important when comparing different ML methods for things such as convergence rate or prediction error. Training with multiple random initializations evaluates the reliability and reproducibility of a training strategy. Without this evaluation, the robustness of a training strategy in reference to random seed initialization sensitivity, or slight changes in datasets will remain completely unknown. For example, deep material networks (DMNs) [18] train to homogenize effective material properties for a single composite microstructure, allowing the user to quickly iterate through many different phase properties to achieve a desired effective composite property. While DMNs are fairly cheap to train, they require to be re-trained if the geometry of the composite is changed. For ML models such as DMNs that often need to be re-trained, evaluating the robustness and consistency of a training strategy is extremely important.

In general, there have been limited efforts in studying or reducing a ML model’s training variability. More effort has been focused on defining the epistemic uncertainty of ML model predictions [19–22] or reducing the variability of predictions of a model from a single training run [23]. Other studies have shown that networks with physically-informed losses have lower standard deviations in prediction errors [24–26], with Ref. [26] demonstrating that a curriculum training strategy reduces performance variation of physics-informed neural networks (PINNs). Adding physics-informed losses ensures that the ML model is learning a specified physical boundary condition, adding some interpretability to the training process. Developing ML models for scientific applications require careful and fair comparisons of current training methods and loss functions to previous ones, and we argue that training consistency and reproducibility should be carefully considered in these model comparisons. Yet, recent review articles focused on physics informed ML show a significant lack of discussion on the reproducibility of physics informed ML models [27, 28], with [28] identifying the initialization of PINN models a future research direction. In the context of physics informed ML, very few studies report the training variability in any context. Ref. [29] showed that from a sample of ten different training initializations a physics-informed neural network typically had less success in training the eikonal equation with rectified linear unit (ReLU) activation than with leaky ReLU activation. Ref. [30] evaluated their fractional physics-informed neural network’s sensitivity

to parameter initializations by reporting the standard deviation of the network’s error across ten different trainings.

Measuring a training method’s variability along with its accuracy is not only important in ML model development but in computational materials science as well where physics-informed losses are prominent and often used to comparatively reduce errors, data requirements, convergence, etc. However, as outlined in the previous paragraphs, physically-informed training strategies’ effect on training variability is not well defined in literature, and any comparative improvements by using a physics-informed loss may be in question if the training variability is not reported.

Previous work [31] describes a Pix2Pix network [32] that predicts the stress fields of a high elastic contrast two-phase composite. Several loss functions having physics-based regularization (PBR) were implemented to enforce stress equilibrium and their performance was compared to a network without PBR. Each implementation was trained ten times, and the stress and equilibrium errors were averaged across the different training sessions. We showed that the PBR losses reduced the equilibrium mean squared error (MSE_{equil}) by at least 51%, with two of the PBR losses maintaining similar stress errors compared to the network without PBR. However, when comparing the best-performing models of the different training sessions for each method, the baseline network’s best-performing model outperformed the other PBR models’ stress errors by 4.5%, even though another model with PBR outperformed the baseline on average. This raised the question of how each method varied in performance across different training sessions and if perhaps PBR methods reduce this variation.

In this work, the variability across different training sessions of the three different PBR methods from the previous work is evaluated and compared to the baseline model without PBR. The current study will focus on the consistency of model predictions and errors to assess the performance and reliability of each implementation as a training strategy for stress field prediction. Several PBR methods are implemented to generalize their effect on training a model in comparison to the baseline model. Different feature representation across the different methods and trainings within a method are additionally investigated.

2 Methods

A Pix2Pix generative adversarial network (GAN) [32] was used to predict the normalized stress fields of a high elastic contrast two-phase composite. The network takes in the spatial arrangement of the phases in the composite as input and predicts the corresponding 2D stress fields as output. Several different PBR terms were implemented to penalize generated stress fields’ deviation from stress equilibrium through the evaluation of stress divergence ($\nabla \cdot \sigma = 0$, neglecting external body forces). A complete description of the implemented PBR strategies can be found in [31]. The following are brief summaries for each PBR method.

Simple Addition Regularization The weighted addition of the absolute stress divergence values directly to the objective function, analogously to the L1 regularization implemented in the base Pix2Pix network. This method biases the Generator to

synthesize stress fields that are closer to equilibrium. (see Equation 4 in [31])

Sigmoid Regularization The root-mean-square of a divergence field is evaluated to encourage predictions to have similar error convergence as the training data. This method biases the discriminator with an additional weight that captures the probability of whether a divergence field is calculated from a set of “real” (from the training dataset) or synthetic stress fields. (see Equations 5-8 in [31])

$\tan^{-1}$ *Regularization* This method encourages the Generator network to produce stress fields that have similar divergence errors to the training dataset. The $\tan^{-1}$ function is chosen to stabilize gradients of the loss function at large errors while also decreasing to zero in the limit of small deviations in the divergence between generated and training stress fields. (see Equations 9-10 in [31])

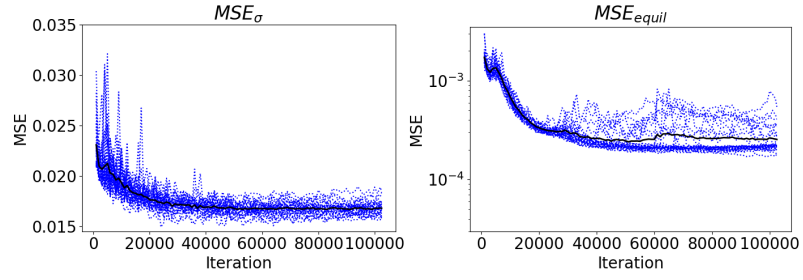
These models were compared to a network that uses the original Pix2Pix objective without any PBR terms, referred to as “baseline” (see Equation 2 in [31]).

The same spinodal decomposition (with varying phase field parameters) dataset, network architecture, regularization methods, and their hyperparameters were used as in our previous work. The various networks were trained on networks trained on identical datasets containing 1,025 image pairs. In this work, each implementation was trained 30 times on the same dataset. Previous work trained each network 10 times and this work trains each network an additional 20 times, for a total of 30 training sessions. A random seed was not fixed for any of the training sessions. A validation dataset was used to evaluate hyperparameters and model selection (described in previous work [31]). Checkpoints were saved during training every 1,000 iterations and were used to evaluate the training statistics after training with a test dataset containing 205 image pairs.

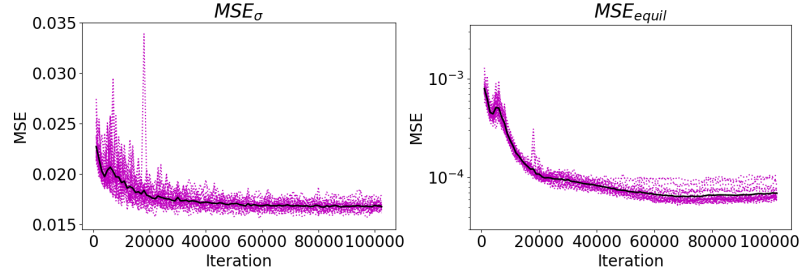
3 Results

Mean squared error (MSE) of the stress fields (MSE_σ) is used to generalize how accurately the model is predicting the values in the stress fields, while the MSE of divergence (MSE_{equil}) summarizes how well the model satisfies stress equilibrium in the predicted stress fields. It should be noted that the MSE_{equil} in predictions is in comparison to the elasto-viscoplastic crystal-plasticity fast Fourier transform (CP-FFT) simulations used to generate the training and validation datasets, not from zero. The CP-FFT simulation converges to stress equilibrium iteratively, with the average root-mean-square of stress divergence for a set of stress fields being $\sim 10^{-3}$ for the dataset.

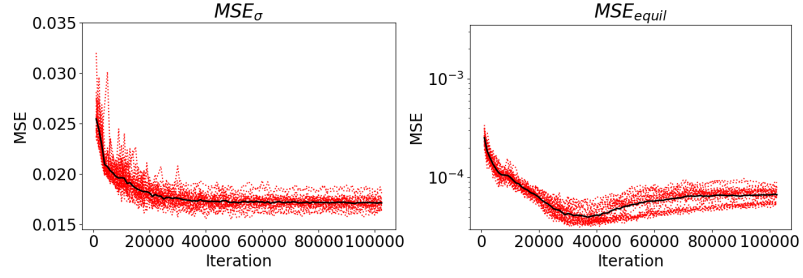
MSE_σ and MSE_{equil} are shown in Figure 1 for each method throughout training for each of the 30 different training sessions. The predicted stress field errors are very similar between each method, which was also observed in the previous work. The $\tan^{-1}$ method has greater MSE_σ values earlier on in training (up to about 0.045), but still converges to similar MSE_σ values as the other methods. The baseline method has visibly more variation in MSE_{equil} than the models having PBR. The $\tan^{-1}$ model



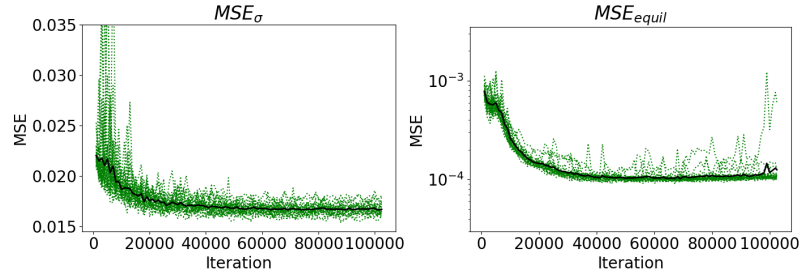
(a) Baseline



(b) Sigmoid



(c) Simple Addition



(d) $\tan^{-1}$

Fig. 1: The MSE_σ (left) and MSE_{equil} (right) throughout training for each of the 30 training sessions. The solid black line indicates the average across the 30 different sessions. (a) No physics-based regularization, (b) simple addition, (c) sigmoid, and (d) $\tan^{-1}$.

has more variation than the other PBR models, but still less than the baseline model. The PBR methods also significantly reduce MSE_{equil} compared to the baseline model.

Table 1 shows the “average performance” of each method. The average performance describes the average errors of the best-performing iterations across the 30 training sessions for a given method. The best-performing iteration is defined as the lowest average MSE_σ from the test dataset and is found for each training session. The average iteration at which this occurs is reported in Table 1 for each method. The deviations in Table 1 show how much the average error and iteration will vary across different training sessions. On average, the sigmoid method will need a little more time to reduce MSE_σ than the baseline method, while the simple addition and $\tan^{-1}$ methods will need less time than the baseline method. The sigmoid and simple addition methods are likely to have slightly higher errors in their stress fields for a given training session, but significantly lower errors in equilibrium than the baseline and $\tan^{-1}$ methods. A similar trade-off between optimizing MSE_σ and MSE_{equil} is seen in the previous work. The average MSE_σ deviates less in the sigmoid and $\tan^{-1}$ methods and all PBR methods have less deviation in MSE_{equil} than the baseline model. The errors from Table 1 are normalized to the baseline model in Table 2 for easier comparison. The simple addition and sigmoid models are likely to have slightly higher stress field errors than the baseline model for a given training session, and the $\tan^{-1}$ method is likely to have the same average stress field errors as the baseline model. All PBR models are likely to lower MSE_{equil} for a given training session, with the simple addition and sigmoid methods by a significant amount.

Table 1: Average performance and variation from 30 training sessions for each method.

	MSE_σ	Iteration	MSE_{equil}
Baseline	0.01614 $\pm$ 0.00050	66,000 $\pm$ 22,000	2.348e-4 $\pm$ 5.94e-5
Sigmoid	0.01630 $\pm$ 0.00036	70,000 $\pm$ 22,000	6.788e-5 $\pm$ 1.42e-5
Simple Addition	0.01663 $\pm$ 0.00056	52,000 $\pm$ 18,000	5.017e-5 $\pm$ 1.26e-5
$\tan^{-1}$	0.01613 $\pm$ 0.00044	59,000 $\pm$ 19,000	1.056e-4 $\pm$ 8.42e-6

Table 2: Average errors shown in Table 1 normalized to the Baseline error.

	MSE_σ	MSE_{equil}
Sigmoid	1.01	0.27
Simple addition	1.03	0.20
$\tan^{-1}$	1.00	0.43

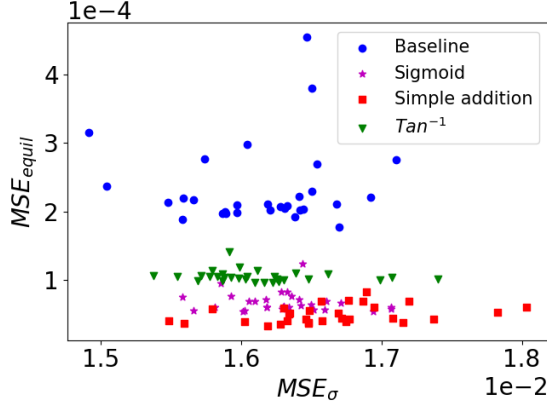


Fig. 2: The average MSE_{σ} and MSE_{equil} for all 30 training sessions for each method.

Figure 2 is a scatter plot of the MSE_{σ} vs. MSE_{equil} from every training session for all methods. This plot further demonstrates the variability in the error metrics for each method. The baseline and simple addition methods have a wider spread of values for MSE_{σ} than the $\tan^{-1}$ and sigmoid methods. All PBR methods are very consistent in the MSE_{equil} across all training sessions.

To better quantify the variability for each method, a bootstrap analysis [33] was performed for MSE_{σ} , MSE_{equil} , and the convergence iteration. Sampling sizes ranged from [2,30] to estimate the model variation for a given number of training sessions within that range. For a given analysis, the average variation for a metric is calculated from 10,000 samplings with the sample size corresponding to the number of training sessions. All samplings are done with replacement and sample from all 30 training sessions. For example, to estimate the average variation in MSE_{σ} for 3 training sessions, 3 MSE_{σ} values are randomly chosen from all 30 training sessions (with replacement, meaning values can be repeated) to make up one sample. A standard deviation of MSE_{σ} can be calculated for that single sample. This is repeated 9,999 more times to get a total of 10,000 standard deviations from the 10,000 samplings and the average standard deviation can be determined for training a network 3 separate times. To estimate the MSE_{σ} variation of 4 training sessions, a single sample is made up of 4 random samplings with replacement from the 30 training sessions, instead of 3.

The top row of Figure 3 shows results of the bootstrap estimates of the variation in MSE_{σ} , MSE_{equil} , and convergence iteration for 2-30 training sessions. The derivatives are plotted in the bottom row below the corresponding metric. The average standard deviation is converged once the derivative reaches zero. For all metrics, the derivatives for all methods converge to zero at a little over 15 training sessions. This suggests that approximately 15 training sessions are sufficient to measure the variability of a model. All PBR methods significantly reduce the variability in MSE_{equil} . The simple addition method increases the variability in MSE_{σ} compared to the baseline, while the sigmoid and $\tan^{-1}$ methods reduce it. The simple addition and $\tan^{-1}$ methods

reduce the number of iterations needed to reduce MSE_σ compared to the baseline, with the sigmoid model slightly increasing the number of iterations needed.

The average errors from the best, median, and worst performing “best iteration” from the 30 training sessions are shown in Table 3 for each method. The best iteration is defined as the iteration that obtains the lowest average MSE_σ from the test dataset for a given training session. The values shown in Table 3 are the prediction error averages and standard deviations of a single training session’s best iteration across the test dataset (which is different from the standard deviations in Table 1, that measure the deviation in average model performance across different training sessions). Across 30 different training sessions, each method had similar average performances in MSE_σ , with the simple addition method being the most likely to have the largest stress errors, but the best equilibrium errors. The baseline model resulted in the model with the lowest average MSE_σ , but significantly higher MSE_{equil} than the PBR models. The sigmoid and $\tan^{-1}$ models found a better balance in optimizing the stress and equilibrium errors with the sigmoid method prioritizing equilibrium errors and $\tan^{-1}$ the stress field errors. Similar trends were observed in the previous work, although slightly different average performance errors were found. Table 4 lists the percent difference from the average performance in MSE_σ , MSE_{equil} , and convergence across 10 training sessions versus 30 training sessions.

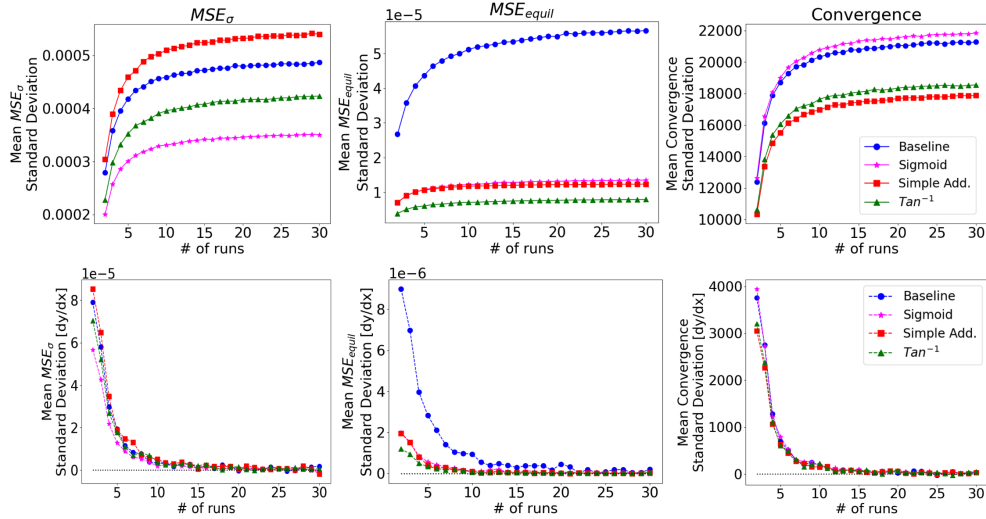


Fig. 3: Top row: Bootstrap analysis to measure the performance variation of average MSE_σ , MSE_{equil} , and convergence iteration as a function of number of training sessions. 10,000 samples were taken for each number of training sessions (sample size). Bottom row: the derivative of the curves plotted in the top row (corresponding column-wise). A line at 0 is plotted in the derivative plots to estimate when the derivative converges to 0. Each row corresponds to the legends on the right for their respective row.

Table 3: The average MSE_σ and MSE_{equil} from the test dataset for the best, median, and worst performance of the 30 training sessions for each method. The standard deviation of error within the test dataset is shown next to the average errors for that single training session. Note that the number of training sessions is an even number which results in two medians. This table shows the results of the better-performing median.

		MSE_σ	Iteration	MSE_{equil}
Baseline	Best	0.01492 $\pm$ 0.0349	24,000	3.153e-4 $\pm$ 5.98e-5
	Median	0.01621 $\pm$ 0.0362	70,000	2.019e-4 $\pm$ 5.83e-5
	Worst	0.01711 $\pm$ 0.0437	51,000	2.761e-4 $\pm$ 3.02e-5
Sigmoid	Best	0.01558 $\pm$ 0.0403	59,000	7.512e-5 $\pm$ 2.19e-5
	Median	0.01630 $\pm$ 0.0400	99,000	6.100e-5 $\pm$ 1.88e-5
	Worst	0.01707 $\pm$ 0.0467	94,000	5.827e-5 $\pm$ 1.89e-5
Simple Addition	Best	0.01548 $\pm$ 0.0311	49,000	4.032e-5 $\pm$ 1.32e-5
	Median	0.01657 $\pm$ 0.0363	69,000	6.949e-5 $\pm$ 2.03e-5
	Worst	0.01803 $\pm$ 0.0429	52,000	6.072e-5 $\pm$ 1.81e-5
Tan^{-1}	Best	0.01538 $\pm$ 0.0372	88,000	1.067e-4 $\pm$ 3.08e-5
	Median	0.01603 $\pm$ 0.0392	63,000	1.021e-4 $\pm$ 2.88e-5
	Worst	0.01740 $\pm$ 0.0430	67,000	1.017e-4 $\pm$ 3.02e-5

Table 4: The difference in average performance for various error metrics and convergence when training each method for an additional 20 training runs.

	# of runs	Baseline	Sigmoid	Simple Addition	Tan^{-1}
MSE_σ	10	0.0162	0.0163	0.0167	0.0160
	30	0.0161	0.0163	0.0166	0.0161
	% difference	-0.617	0.0	-0.599	0.625
MSE_{equil}	10	2.19e-4	6.62e-5	5.36e-5	1.07e-4
	30	2.35e-4	6.79e-5	5.02e-5	1.06e-4
	% difference	7.31	2.57	-6.34	-0.943
Iterations to reduce MSE_σ	10	72,000	73,000	53,000	54,000
	30	66,000	70,000	52,000	59,000
	% difference	-8.3	-4.1	-1.9	9.3

Figure 4 shows the generated stress field in the loading direction (σ_{22}) for each of the models listed in Table 3. This figure shows that for a given training session, the network may capture different features from the dataset. The top-middle image in Figure 4 shows σ_{22} from the test dataset, and a zoomed-in portion of it to the right. This stress field exhibits Gibbs oscillations [34] (the pixelated regions around the phase boundaries and going across the yellow circular phase corresponding to the input image), which is an artifact of the CP-FFT simulation. These features are a result of the high elastic contrast in the composite. The baseline method has gridding artifacts in its worst-performing model. In the median performing model, this gridding

goes away and instead has a slightly pixelated phase boundary, indicating that it is trying to replicate the Gibbs oscillations. The baseline’s best-performing model seems to smooth out the Gibbs oscillations. The worst and median sigmoid models have a slightly more pronounced border at the phase boundary than the best-performing model but seem to otherwise look the same. The simple addition method replicates the Gibbs oscillations in its worst- and median-performing models more prominently than any other method. The simple addition’s median-performing model looks the most similar to the oscillations shown in the target but then smooths them out completely in its best-performing model. The $\tan^{-1}$ method seems to have the opposite trend, where the Gibbs oscillations become more prominent as MSE_σ decreases.

Note that these Gibbs oscillations are more likely to appear later on in training (though not always), which generally doesn’t align with a model’s lowest MSE_σ for a given training session. In addition, some methods were more likely to replicate the Gibbs oscillations than others. For the same input shown in Figure 4, each training session was sampled every 9,000 iterations (from the saved checkpoints) to see the prediction progression. An example of this progression is shown in the Appendix for each method. Each training session was visually inspected to evaluate if the Gibbs oscillations appeared at all throughout the iterations sampled. For example, the best performing simple addition model, the best and worst baseline models, and the best, median, and worst sigmoid models would be considered as not having Gibbs oscillations, while the best $\tan^{-1}$, the median baseline, and the median and worst simple addition models would be considered as having replicated Gibbs oscillations. For the test example listed in Figure 4, the $\tan^{-1}$ and simple addition methods replicated the Gibbs oscillations at some point during a training session 90% of the time, the baseline 67% and sigmoid 57%.

Figure 5 shows the absolute value of a divergence field for the target and the stress fields shown in Figure 4. The divergence fields calculated from the generated stress fields are scaled to the maximum of the target divergence field. Any yellow pixel in the predicted divergence fields indicates a value greater than or equal to the target’s largest deviation from equilibrium. The baseline method has values that are nearly all greater than the target maximum for its best, median, and worst model. The deviations from equilibrium in the predictions seem to be larger than the example shown in previous work, which is likely a result of the target’s equilibrium field having larger equilibrium errors as well. The $\tan^{-1}$ method has fewer values outside the target range compared to the baseline, but more than the other two PBR models. The $\tan^{-1}$ method’s divergence fields are consistent across best, median, and worst performing models. The divergence fields for the simple addition method deviate more from the target as the performance gets worse. The opposite trend is observed for the sigmoid method, where the divergence deviates more from the target as the performance improves, i.e. as the MSE_σ improves. This may result from a loss competition in the sigmoid method between the stress field error and the equilibrium error. The sigmoid method may improve MSE_σ at the expense of $\text{MSE}_{\text{equil}}$ for some training sessions.

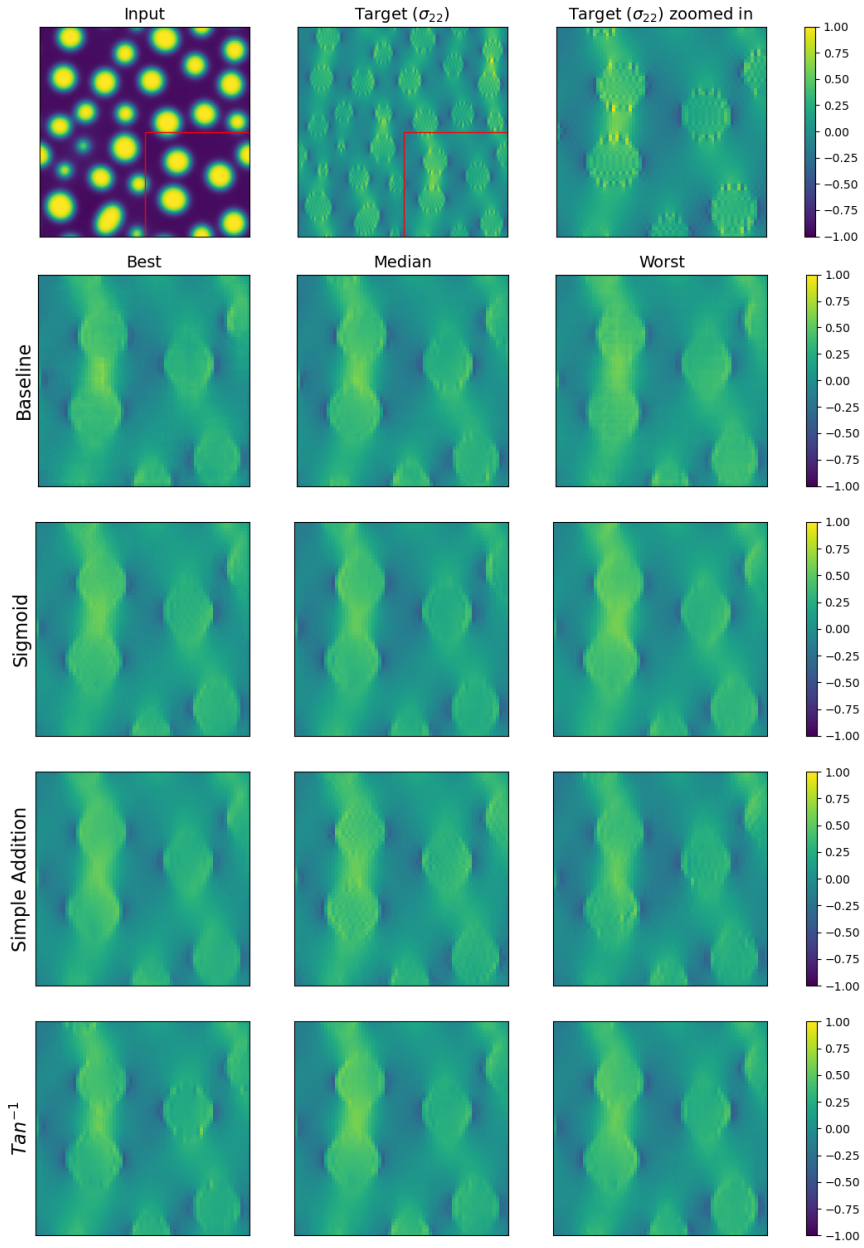


Fig. 4: The σ_{22} stress field (loading direction) for the best, median, and worst performing training session for each method. The stress fields shown are zoomed in on the portion outlined by the red square shown in the top left.

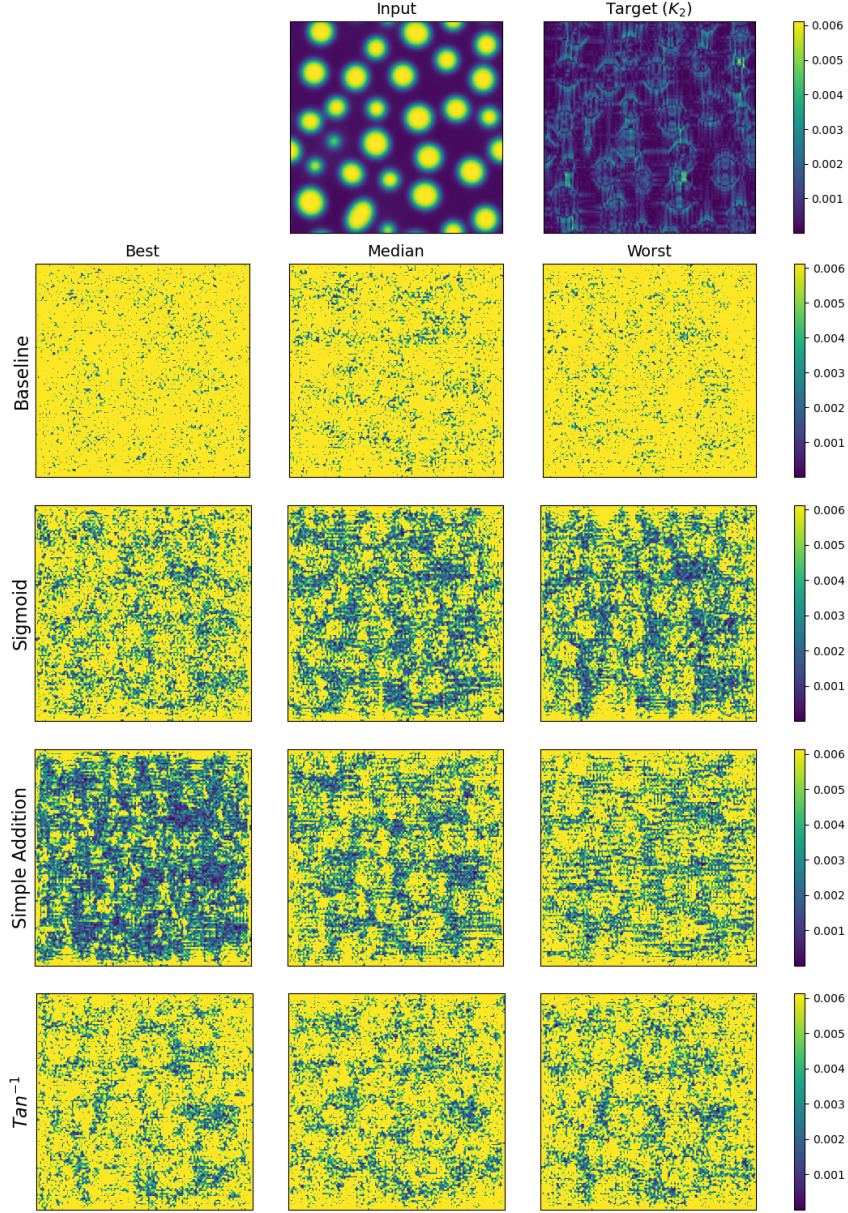


Fig. 5: The absolute value of a divergence field (K_2 from Equation 3 in Ref. [31] is shown here) for the best, median, and worst performing training session for each method. The divergence fields are scaled to the target divergence fields' minimum and maximum values. Yellow pixels indicate a value greater than or equal to the target's largest deviation from equilibrium.

4 Discussion

The goal for developing our network models was to create a more reproducible training strategy that consistently reduces the stress field error (MSE_σ) and stress equilibrium error ($\text{MSE}_{\text{equil}}$) across different training runs. Thus, we discuss the behavior of the developed networks relative to these goals.

The variation in stress field error, equilibrium error, and convergence across 30 separate training sessions was studied for models with and without PBR. As shown in our previous work, all PBR methods decreased the equilibrium errors for the predicted stress fields when compared to the baseline method. This work shows that it additionally reduces the *variation* of equilibrium error across different training sessions. Note that the simple addition and baseline methods have the greatest variation in MSE_σ across training sessions, as shown by the standard deviation values in Table 1 and Figures 2 & 3, but likely for very different reasons. The simple addition method may have a larger MSE_σ variation since it seems to prioritize $\text{MSE}_{\text{equil}}$ which may be over-constraining the network. The PBR weight may be too strong such that the network sometimes minimizes $\text{MSE}_{\text{equil}}$ at the expense of MSE_σ , resulting in a network that produces stress fields that lower $\text{MSE}_{\text{equil}}$ without considering MSE_σ . At other times it may learn that reducing MSE_σ also reduces $\text{MSE}_{\text{equil}}$, which may explain the larger MSE_σ variability. Conversely, the baseline model has nothing to enforce equilibrium within its predictions, resulting in the greatest equilibrium error variation of the evaluated models.

A key takeaway from this work is that having only a few training sessions is not sufficient for measuring the variability in training a network, as highlighted by the bootstrap analysis. On average, the network needs to be trained a little over 15 times to adequately estimate the variability. After 15 training sessions, the variation appears to converge for all methods and metrics. This estimate also shows the probability of getting a more “lucky” run by re-training the same network one more time. When having less than 5 training runs, the probability that the training session having the lowest errors has already been trained is smaller and it may be worthwhile to train the network a few more times to achieve a training session that will result in better performance. However, after about 15 training sessions, finding a run that will have much better (or worse) performance is not likely.

As previously mentioned, Gibbs oscillations are an artifact of the CP-FFT solver used to generate the datasets, and all the training data contains these features. Since the oscillations are an artifact of a simulation, one could argue that it could be advantageous that a network would ignore them. However, the opposing argument is that the Gibbs oscillations appear in the training dataset, therefore they should show up in the network predictions. Furthermore, if a network were to be trained using microstructures having high-frequency features similar to Gibbs oscillations, a representation of those features would be necessary for the network to provide accurate predictions. Regardless, it is shown by Figure 4 that models trained using the same dataset, hyperparameters, and loss function can result in not only different prediction quality but different feature representations, especially concerning the Gibbs oscillations feature.

Whether a network represents Gibbs oscillations in its predictions depends on the capability of the network to replicate them and when the lowest MSE_σ occurs. It

was shown that the networks typically reduce the MSE_σ before the network learns the Gibbs oscillations. Ref. [35] showed that neural networks will tend to learn lower frequency features (stress fields that scale with microstructure phase scale) before learning higher frequency features (such as the smaller scale Gibbs oscillations), a trend that is possibly observed for the models studied here. However, there seems to be a difference in how often and how well models replicate the Gibbs oscillations. The simple addition model replicates them very well and often, while the sigmoid model can replicate them well at the end of training (see Supplemental Figures), but not nearly as often. The baseline and $\tan^{-1}$ models do not replicate the Gibbs oscillations very well, and the $\tan^{-1}$ method does so more often than the sigmoid and baseline models. There is the possibility that the training sessions that did not replicate the Gibbs oscillations just needed more iterations to learn the high-frequency features, as [10] showed that the same network can have different convergence times for different features. A PBR term enforcing similar high-frequency features between targets and predictions could possibly reduce the variability in feature representation between different training sessions; however, that possibility was not investigated.

The Pix2Pix network has shown to be an exemplar for studying the effects of PBR on training variability. However, our findings and general approach are not specific to the Pix2Pix architecture discussed in this work. The following is a more general discussion on the variability in training a ML network that is applicable to any ML model. The consequences of a training process having high variability, when to consider the variability, and suggested best practices are discussed, using the above results as an example.

What are the implications of high training variability?

Measuring the variability in a model across different training sessions evaluates the reliability of the training method. A higher variability in the average performance of a model means that the reproducibility of the training process is low. Someone using that training process is less likely to achieve similar results to those reported without running the model multiple times and, depending on the model, this could end up being an unreasonable amount of computation time. A repeatable process for training a network is as important as curating a good dataset for training, and the reliability of a training process should be reported to evaluate the reproducibility. The main consequence of a model having high-performance variation is that the performance of a network may not be what is expected, which could lead to misleading conclusions about networks or training algorithms.

Considerations for choosing the best algorithm/training method

When comparing and choosing the best method (in this case the network loss function), one needs to consider not only which method reduces the errors most efficiently, but how reliably the errors are reduced. One could simply train a method many times and choose the version that reduces the error the most. In this case, the simple addition may be the best pick, since it reduces both MSE_σ and $\text{MSE}_{\text{equil}}$ (the baseline and $\tan^{-1}$ reduce the MSE_σ more, but have greater equilibrium errors). However, training a network many times is often not practical depending on how expensive the

network is to train. Training a network that takes days, or even months, many times to get the best version is unreasonable. In this case, a low variability network would be more desirable so that training sessions can be minimized to achieve the expected outcome. Low training variability could possibly be advantageous if a network is going to be trained on different datasets for the same reasons, although this should be studied in more depth. The feature representation in network predictions also needs to be considered, as some models may more reliably replicate features than others.

In summary, choosing the best overall network and training algorithm depends on the computational resources available for training, variability, and accuracy of the network’s predictions, and may not be the same network in every scenario. For example, the simple addition may be the best pick if training times are quick and/or sufficient computational resources are available. The simple addition method sufficiently reduces both MSE_σ and $\text{MSE}_{\text{equil}}$ (the baseline and $\tan^{-1}$ reduce the MSE_σ more, but have greater equilibrium errors) where the network can be trained many times until a training session is successful. Alternatively, if computing resources are limited, the sigmoid method may be the best option. The sigmoid method has the lowest MSE_σ among the worst-performing models and is consistent in reproducing similar $\text{MSE}_{\text{equil}}$ values.

Suggested practices

Reporting the average performance variation of a network becomes especially important when comparing deep learning methods. A large motivation behind incorporating physics into a deep learning network is to train the network more efficiently and/or result in a more accurate network. Unless the variability is reported, the comparative improvement in accuracy/efficiency becomes obscured. This work clearly demonstrates that comparing just one or a small number of training runs for a method may result in misleading conclusions about the performance of a network. For example, comparing only the best performances of the sigmoid and $\tan^{-1}$ methods would lead to the conclusion that the $\tan^{-1}$ method converges more slowly than the sigmoid method but achieves a lower error, when the opposite is true on average (see Figure 3 and Table 3). Comparing the baseline and $\tan^{-1}$ methods, both have similar average performance in stress errors, yet the baseline method’s best-performing model has a lower MSE_σ than the $\tan^{-1}$ method’s best-performing model. A similar situation is seen comparing the best-performing models for the simple addition and sigmoid methods. The simple addition method’s best-performing model achieves lower errors than the sigmoid method’s best, but the sigmoid method will on average achieve a lower MSE_σ .

A bootstrap analysis was useful in estimating the variation of each metric for each network, as well as determining the number of training sessions needed to estimate these variations. When computing resources allow, the authors suggest performing this analysis to ensure the variation values are converged. However, we acknowledge that with limited computing resources, or networks requiring very long training times, the number of training sessions to converge the variation measurement may not be reasonable. This poses an issue for fair comparisons of networks with expensive training times.

Arguably, reporting the variability of a model may become less important when a comparison among deep learning methods, parameters, or loss functions is *not* made, and code and/or a trained model is provided. Although a measure of the variability would still be helpful for someone else wanting to repeat the training process, measuring the variability of a network might be difficult to justify if computing resources are limited. For the most reproducible model, providing code and hyper-parameters used for training is the most straightforward method.

5 Conclusions

The model variation was studied across different training sessions of networks having, and without, physics-informed losses to enforce stress equilibrium. All physics-informed losses reduced the variation in equilibrium error compared to the baseline, and two of the three physics-informed losses also reduced the variation in stress errors. This shows that a physics-informed loss can create a more reliable and repeatable network. Two of the three physics-informed losses reduced the convergence and variation of convergence iteration of the stress error compared to the baseline. Also shown is that networks will vary with regard to which features are captured for a given training session and that some methods were more likely to reproduce high-frequency features than others. Therefore, reporting model/network variability is important for comparing models. Our understanding is that the aspects of model variability during training, and the resultant influences on the outcome of a fully trained model as observed in the present study, are likely applicable any time that deep learning tools are developed for the relatively small datasets that are common in the materials sciences. Ideally, these aspects should be reported along with model results.

Acknowledgements. This study was funded by National Science Foundation award CMMI 1826149 (Quantitative Representation of Microstructure). AL and SRN also received funding from the National Science Foundation award EEC 2133630 (NSF Engineering Research Center for Hybrid Autonomous Manufacturing Moving from Evolution to Revolution (ERC-HAMMER)). The funder played no role in study design, data collection, analysis and interpretation of data, or the writing of this manuscript. This work was performed, in part, at the Center for Integrated Nanotechnologies, an Office of Science User Facility operated for the U.S. Department of Energy (DOE) Office of Science by Los Alamos National Laboratory (Contract 89233218CNA000001) and Sandia National Laboratories (Contract DE-NA-0003525).

Author Contributions. AL developed and implemented the ML Network and performed the numerical experiments. All authors contributed to the initial experimental design and subsequent iterations. All authors contributed to discussions and interpretation of the results. AL composed the early drafts of the manuscript. SRN and DMD provided revision and feedback on early draft and final revisions on submitted version. All authors reviewed the final manuscript and approve its submission.

Competing Interests. All authors declare no financial or non-financial competing interests.

Code Availability. The underlying code for this study is available in the PB-GAN repository and can be accessed via this link <https://github.com/mesoOSU/PB-GAN>. All training, validation, and test datasets necessary to reproduce the current study are also available through the PB-GAN repository.

Ethics approval and consent to participate. Not applicable.

Consent for publication. Not applicable.

Materials availability. Not applicable.

Data availability. Not applicable (with code availability).

References

- [1] Ge, M., Su, F., Zhao, Z., Su, D.: Deep learning analysis on microscopic imaging in materials science. *Materials Today Nano* **11**, 100087 (2020) <https://doi.org/10.1016/j.mtnano.2020.100087>
- [2] Jacobs, R.: Deep learning object detection in materials science: Current state and future directions. *Computational Materials Science* **211**, 111527 (2022) <https://doi.org/10.1016/j.commatsci.2022.111527>
- [3] Merchant, A., Batzner, S., Schoenholz, S.S., Aykol, M., Cheon, G., Cubuk, E.D.: Scaling deep learning for materials discovery. *Nature* **624**, 80–85 (2023)
- [4] Zuo, Y., Qin, M., Chen, C., Ye, W., Li, X., Luo, J., Ong, S.P.: Accelerating materials discovery with bayesian optimization and graph deep learning. *Materials Today* **51**, 126–135 (2021) <https://doi.org/10.1016/j.mattod.2021.08.012>
- [5] Li, X., Liu, Z., Cui, S., Luo, C., Li, C., Zhuang, Z.: Predicting the effective mechanical property of heterogeneous materials by image based modeling and deep learning. *Computer Methods in Applied Mechanics and Engineering* **347**, 735–753 (2019) <https://doi.org/10.1016/j.cma.2019.01.005>
- [6] Jha, D., Gupta, V., Liao, W.-K., Choudhary, A., Agrawal, A.: Moving closer to experimental level materials property prediction using ai. *Sci Rep* **12**(3), 11953 (2022) <https://doi.org/10.1038/s41598-022-15816-0>
- [7] Henkes, A., Wessels, H.: Three-dimensional microstructure generation using generative adversarial neural networks in the context of continuum micromechanics. *Computer Methods in Applied Mechanics and Engineering* **400**, 115497 (2022) <https://doi.org/10.1016/j.cma.2022.115497>
- [8] Chun, S., Roy, S., Nguyen, Y.T., Choi, J.B., Udaykumar, H.S., Baek, S.S.: Deep learning for synthetic microstructure generation in a materials-by-design framework for heterogeneous energetic materials. *Sci Rep* **10**, 13307 (2020) <https://doi.org/10.1038/s41598-020-70149-0>

- [9] Agrawal, A., Choudhary, A.: Deep materials informatics: Applications of deep learning in materials science. *MRS Communications* **9**(3), 779–792 (2019) <https://doi.org/10.1557/mrc.2019.73>
- [10] Pham, H.V., Qian, S., Wang, J., Lutellier, T., Rosenthal, J., Tan, L., Yu, Y., Nagappan, N.: Problems and opportunities in training deep learning software systems: an analysis of variance. In: *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. ASE '20*, pp. 771–783. Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3324884.3416545> . <https://doi.org/10.1145/3324884.3416545>
- [11] Khan, M.M.R., Arif, R.B., Siddique, M.A.B., Oishe, M.R.: Study and observation of the variation of accuracies of knn, svm, lmn, enn algorithms on eleven different datasets from uci machine learning repository. In: *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEICT)*, pp. 124–129 (2018). <https://doi.org/10.1109/CEEICT.2018.8628041>
- [12] Bouthillier, X., Delaunay, P., Bronzi, M., Trofimov, A., Nichyporuk, B., Szeto, J., Mohammadi Sepahvand, N., Raff, E., Madan, K., Voleti, V., Ebrahimi Kahou, S., Michalski, V., Arbel, T., Pal, C., Varoquaux, G., Vincent, P.: Accounting for variance in machine learning benchmarks. In: Smola, A., Dimakis, A., Stoica, I. (eds.) *Proceedings of Machine Learning and Systems*, vol. 3, pp. 747–769 (2021). https://proceedings.mlsys.org/paper_files/paper/2021/file/0184b0cd3cfb185989f858a1d9f5c1eb-Paper.pdf
- [13] Alahmari, S.S., Goldgof, D.B., Mouton, P.R., Hall, L.O.: Challenges for the repeatability of deep learning models. *IEEE Access* **8**, 211860–211868 (2020) <https://doi.org/10.1109/ACCESS.2020.3039833>
- [14] Qian, S., Pham, V.H., Lutellier, T., Hu, Z., Kim, J., Tan, L., Yu, Y., Chen, J., Shah, S.: Are my deep learning systems fair? an empirical study of fixed-seed training. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., Vaughan, J.W. (eds.) *Advances in Neural Information Processing Systems*, vol. 34, pp. 30211–30227 (2021). https://proceedings.neurips.cc/paper_files/paper/2021/file/fdda6e957f1e5ee2f3b311fe4f145ae1-Paper.pdf
- [15] Pinto, W.G., Alguacil, A., Bauerheim, M.: On the reproducibility of fully convolutional neural networks for modeling time–space-evolving physical systems. *Data-Centric Engineering* **3**, 19 (2022) <https://doi.org/10.1017/dce.2022.18>
- [16] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d’Alché-Buc, F., Fox, E., Larochelle, H.: Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *J. Mach. Learn. Res.* **22**(1) (2021)
- [17] Wang, A.Y.-T., Murdock, R.J., Kauwe, S.K., Oliynyk, A.O., Gurlo, A., Brgoch, J., Persson, K.A., , Sparks, T.D.: Machine learning for materials scientists: An

- introductory guide toward best practices. *Chemistry of Materials* **32**(12), 4954–4965 (2020) <https://doi.org/10.1021/acs.chemmater.0c01907>
- [18] Shin, D., Alberdi, R., Lebensohn, R.A., Dingreville, R.: Deep material network via a quilting strategy: visualization for explainability and recursive training for improved accuracy. *npj Comput Mater* **9**(128) (2023) <https://doi.org/10.1038/s41524-023-01085-6>
 - [19] Tavazza, F., DeCost, B., Choudhary, K.: Uncertainty prediction for machine learning models of material properties. *ACS Omega* **6**(48), 32431–32440 (2021) <https://doi.org/10.1021/acsomega.1c03752>
 - [20] Olivier, A., Shields, M.D., Graham-Brady, L.: Bayesian neural networks for uncertainty quantification in data-driven materials modeling. *Computer Methods in Applied Mechanics and Engineering* **386**, 114079 (2021) <https://doi.org/10.1016/j.cma.2021.114079>
 - [21] Viana, F.A.C., Subramaniyan, A.K.: A survey of bayesian calibration and physics-informed neural networks in scientific modeling. *Archives of Computational Methods in Engineering* **28**, 3801–3830 (2021)
 - [22] Li, J., Long, X., Deng, X., Jiang, W., Zhou, K., Jiang, C., Zhang, X.: A principled distance-aware uncertainty quantification approach for enhancing the reliability of physics-informed neural network. *Reliability Engineering & System Safety* **245**, 109963 (2024) <https://doi.org/10.1016/j.ress.2024.109963>
 - [23] Lemay, A., Hoebel, K., Bridge, C.P., Befano, B., Sanjosé, S.D., Egemen, D., Rodriguez, A.C., Schiffman, M., Campbell, J.P., Kalpathy-Cramer, J.: Improving the repeatability of deep learning models with Monte Carlo dropout (2022)
 - [24] Bai, Z., Song, S.: Structural reliability analysis based on neural networks with physics-informed training samples. *Engineering Applications of Artificial Intelligence* **126**, 107157 (2023) <https://doi.org/10.1016/j.engappai.2023.107157>
 - [25] Lu, Y., Wang, B., Zhao, Y., Yang, X., Li, L., Dong, M., Lv, Q., Zhou, F., Gu, N., Shang, L.: Physics-informed surrogate modeling for hydro-fracture geometry prediction based on deep learning. *Energy* **253**, 124139 (2022) <https://doi.org/10.1016/j.energy.2022.124139>
 - [26] Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., Mahoney, M.W.: Characterizing possible failure modes in physics-informed neural networks. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., Vaughan, J.W. (eds.) *Advances in Neural Information Processing Systems*, vol. 34, pp. 26548–26560. Curran Associates, Inc., ??? (2021). https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf

- [27] Meng, C., Griesemer, S., Cao, D., Seo, S., Liu, Y.: When physics meets machine learning: a survey of physics-informed machine learning. *Mach. Learn. Comput. Sci. Eng* **1**(20) (2025)
- [28] Cuomo, S., Cola, V.S.D., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F.: Scientific machine learning through physics-informed neural networks: Where we are and what’s next. *J Sci Comput* **92**(88) (2022)
- [29] Blechschmidt, J., Ernst, O.G.: Three ways to solve partial differential equations with neural networks — a review. *GAMM-Mitteilungen* **44**(2), 202100006 (2021) <https://doi.org/10.1002/gamm.202100006>
<https://onlinelibrary.wiley.com/doi/pdf/10.1002/gamm.202100006>
- [30] Pang, G., Lu, L., Karniadakis, G.E.: fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing* **41**(4), 2603–2626 (2019) <https://doi.org/10.1137/18M1229845>
- [31] Lenau, A., Dimiduk, D.M., Niezgoda, S.R.: Importance of hyper-parameter optimization during training of physics-informed deep learning networks. *Integr Mater Manuf Innov* **14**, 115–135 (2025) <https://doi.org/10.1007/s40192-025-00394-6>
- [32] Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.A.: Image-to-Image Translation with Conditional Adversarial Networks (2018)
- [33] Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman and Hall/CRC, New York (1994)
- [34] Carslaw, H.S.: Introduction to the Theory of Fourier’s Series and Integrals (Third Ed.), Chapter IX. Dover Publications Inc., London (1930)
- [35] Basri, R., Jacobs, D.W., Kasten, Y., Kritchman, S.: The convergence rate of neural networks for learned functions of different frequencies. *CoRR* **abs/1906.00425** (2019) [1906.00425](https://arxiv.org/abs/1906.00425)

6 Supplemental Figures

6.1 Examples of training sessions reproducing Gibbs oscillations for each method

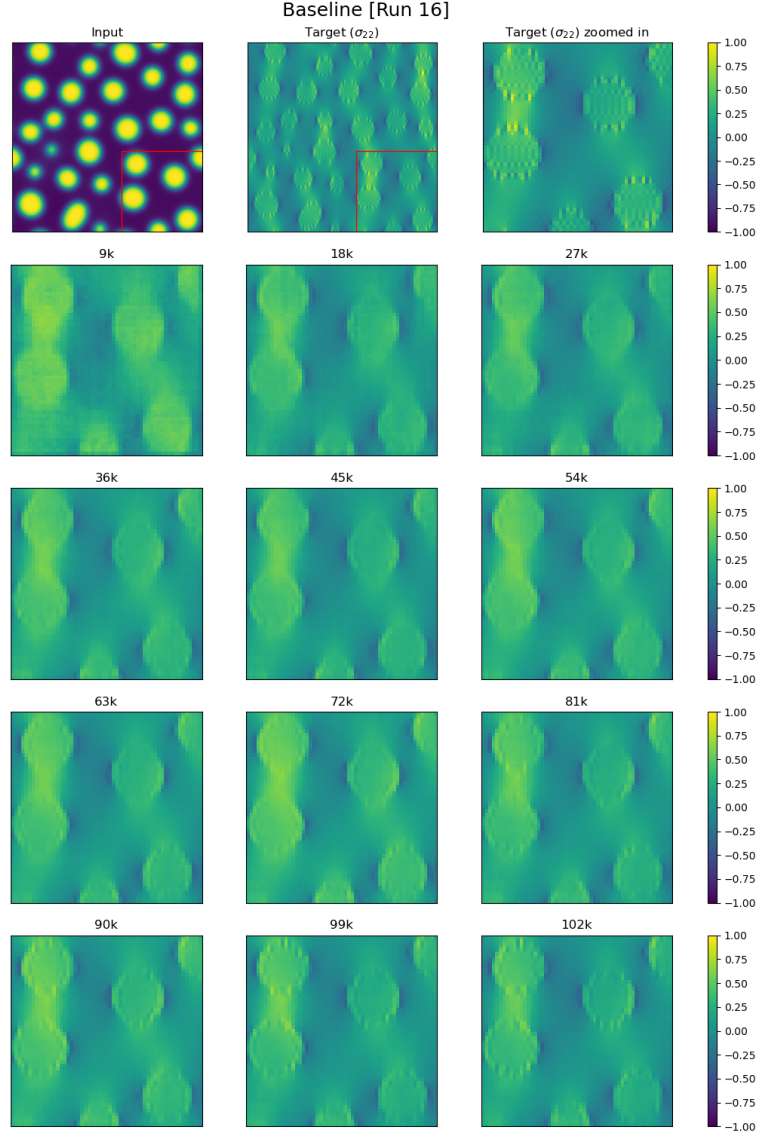


Fig. 6: Baseline predicted σ_{22} field from different iterations saved throughout a single training. Replicates Gibbs oscillations.

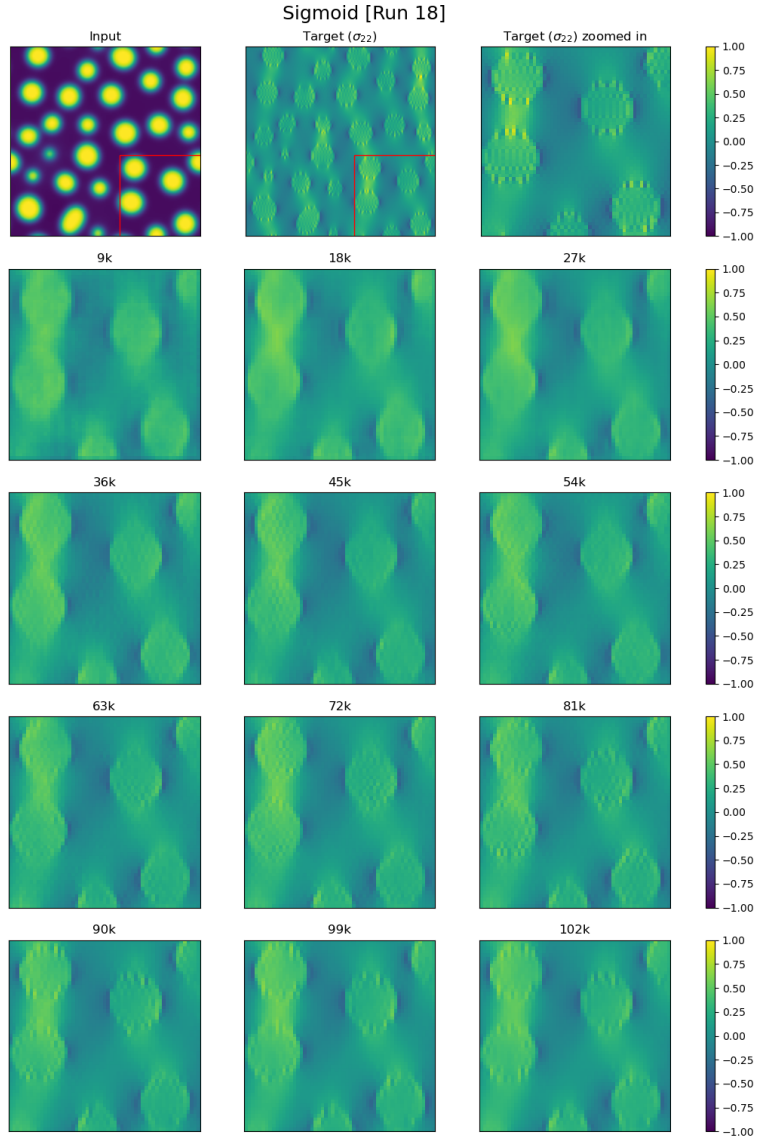


Fig. 7: Sigmoid predicted σ_{22} field from different iterations saved throughout a single training. Replicates Gibbs oscillations.

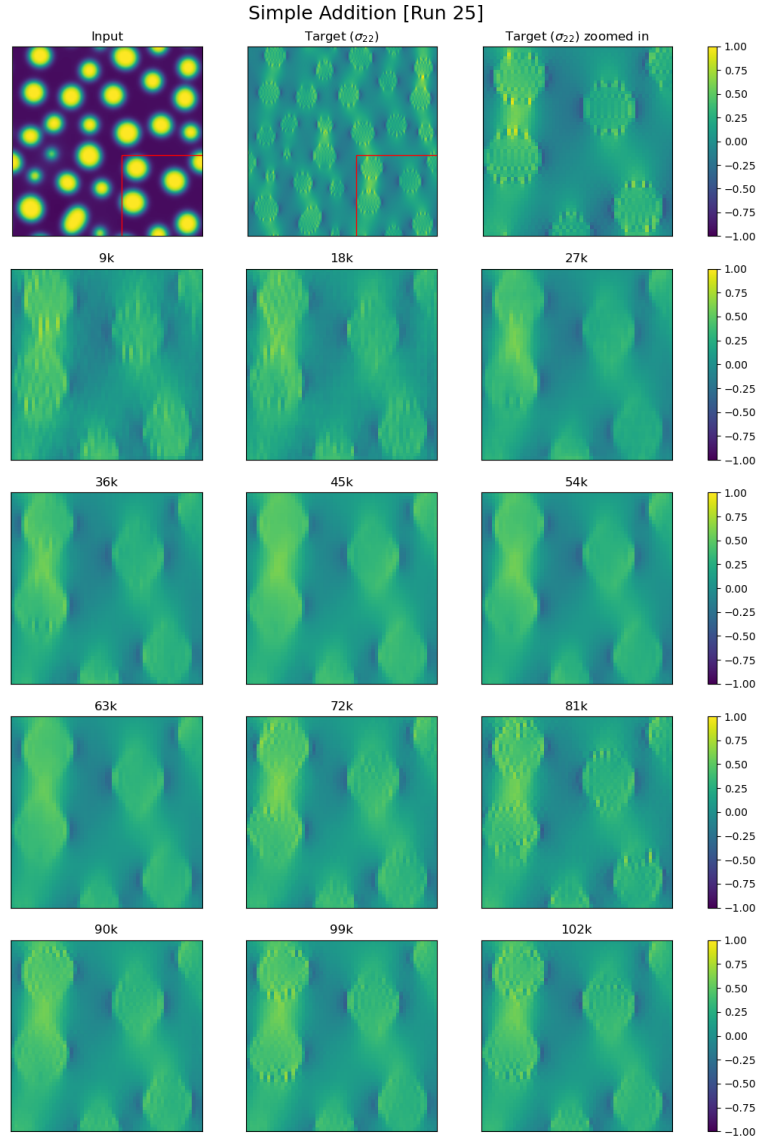


Fig. 8: Simple addition predicted σ_{22} field from different iterations saved throughout a single training. Replicates Gibbs oscillations.

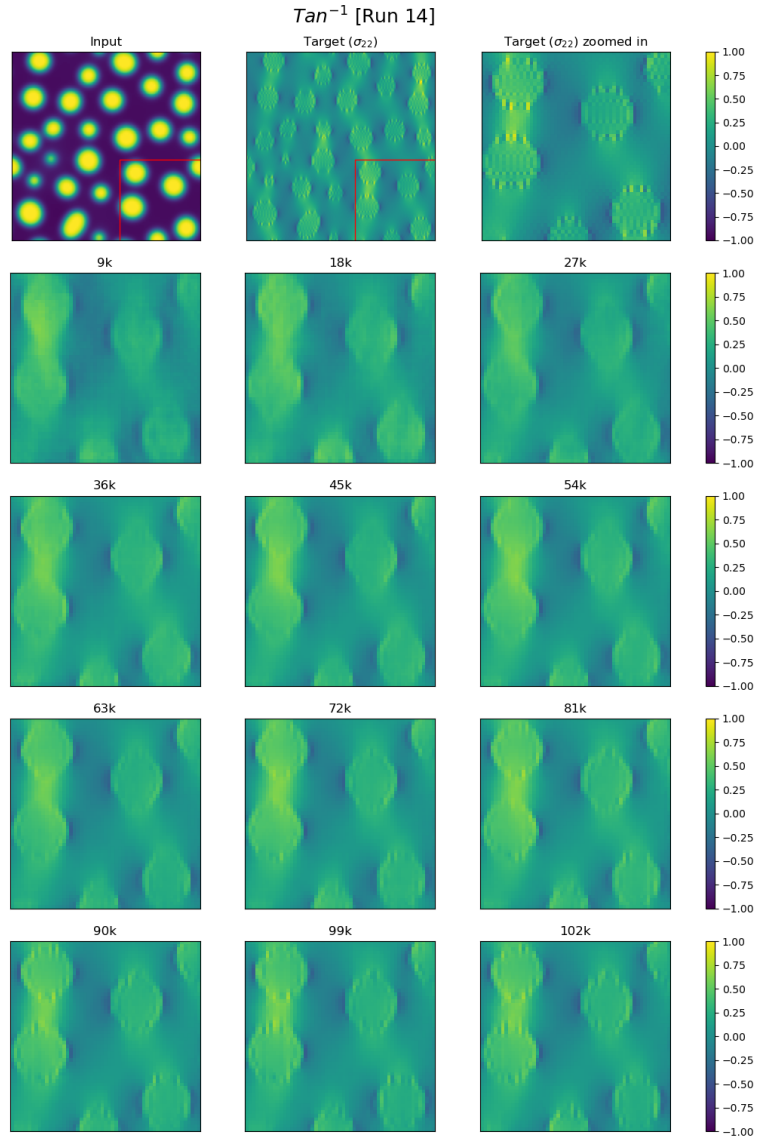


Fig. 9: Tan^{-1} predicted σ_{22} field from different iterations saved throughout a single training. Replicates Gibbs oscillations.

6.2 Examples of training sessions NOT reproducing Gibbs oscillations for each method

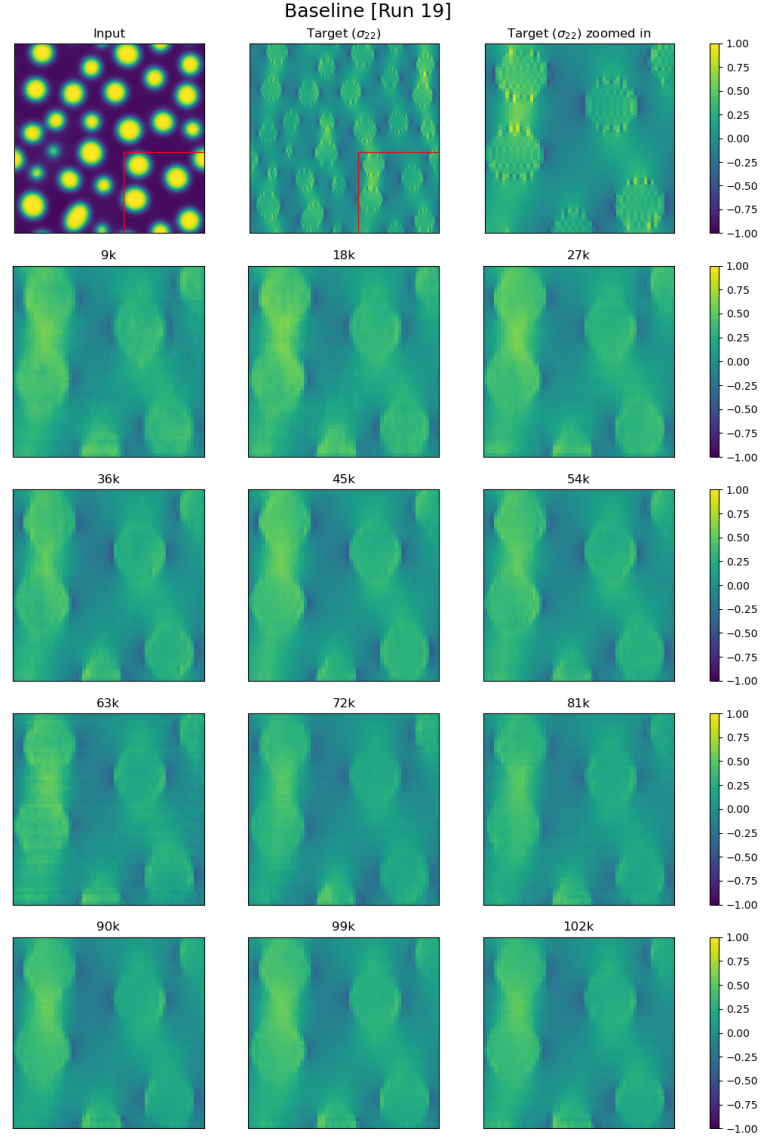


Fig. 10: Baseline predicted σ_{22} field from different iterations saved throughout a single training. Does not replicate Gibbs oscillations.

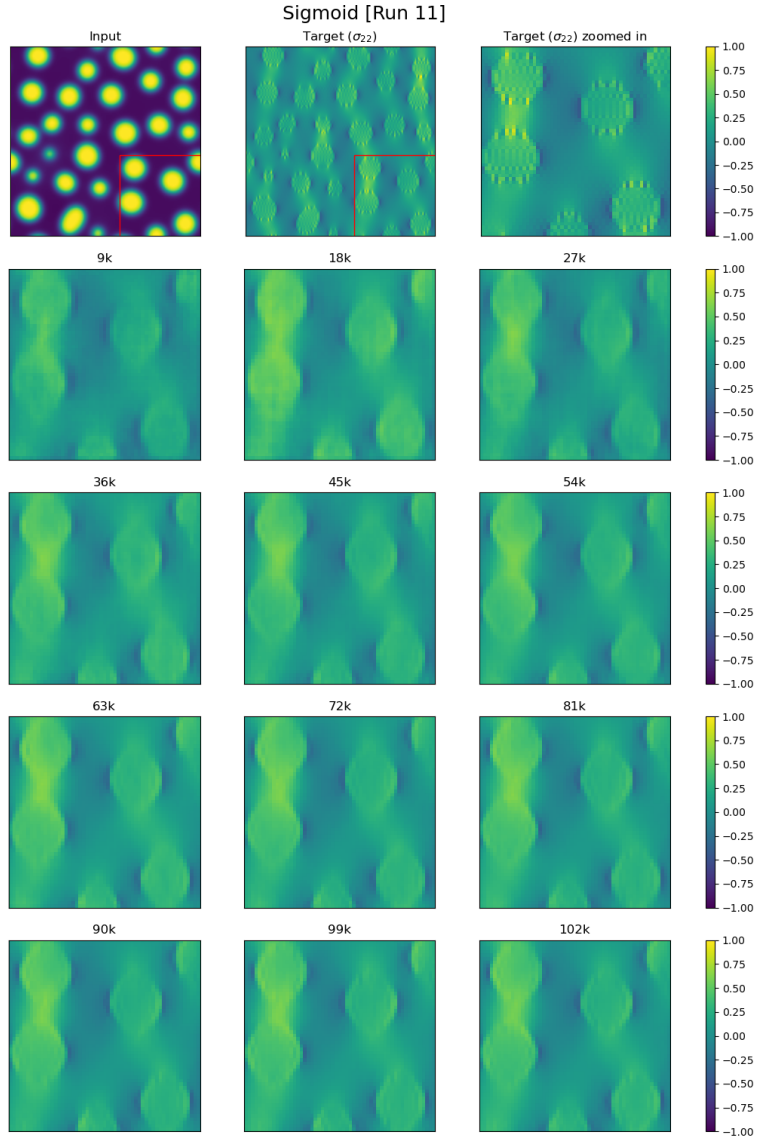


Fig. 11: Sigmoid predicted σ_{22} field from different iterations saved throughout a single training. Does not replicate Gibbs oscillations.

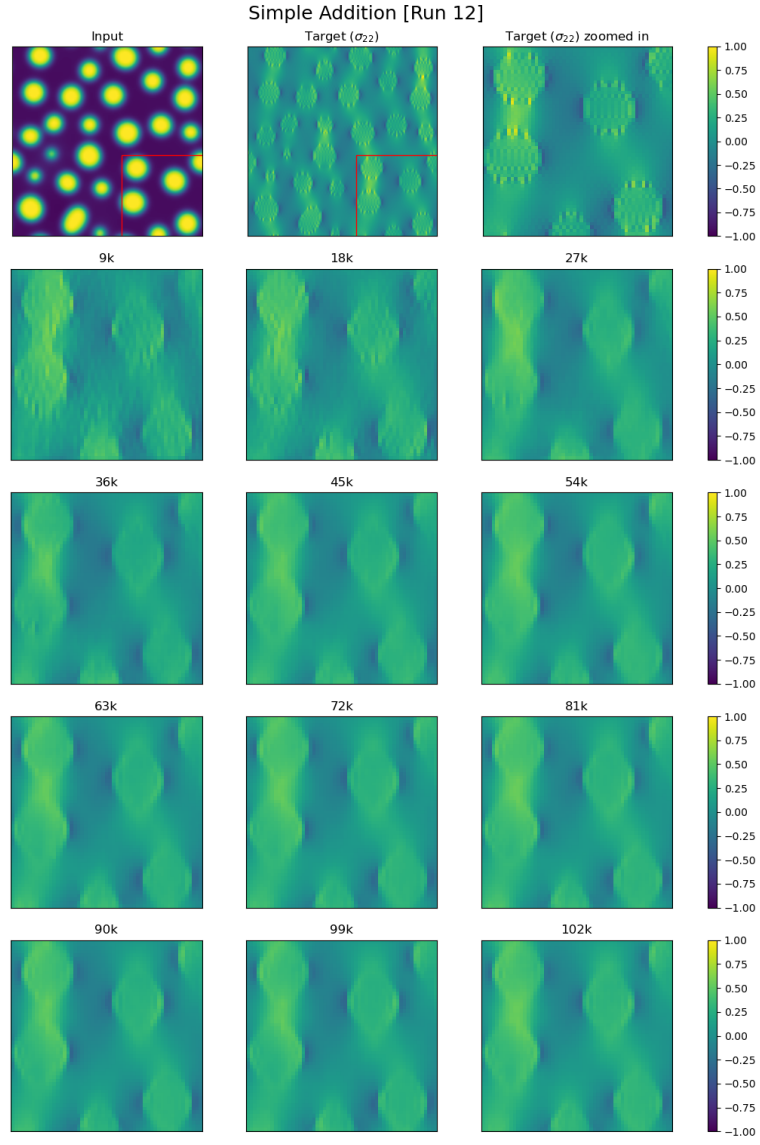


Fig. 12: Simple addition predicted σ_{22} field from different iterations saved throughout a single training. Does not replicate Gibbs oscillations.

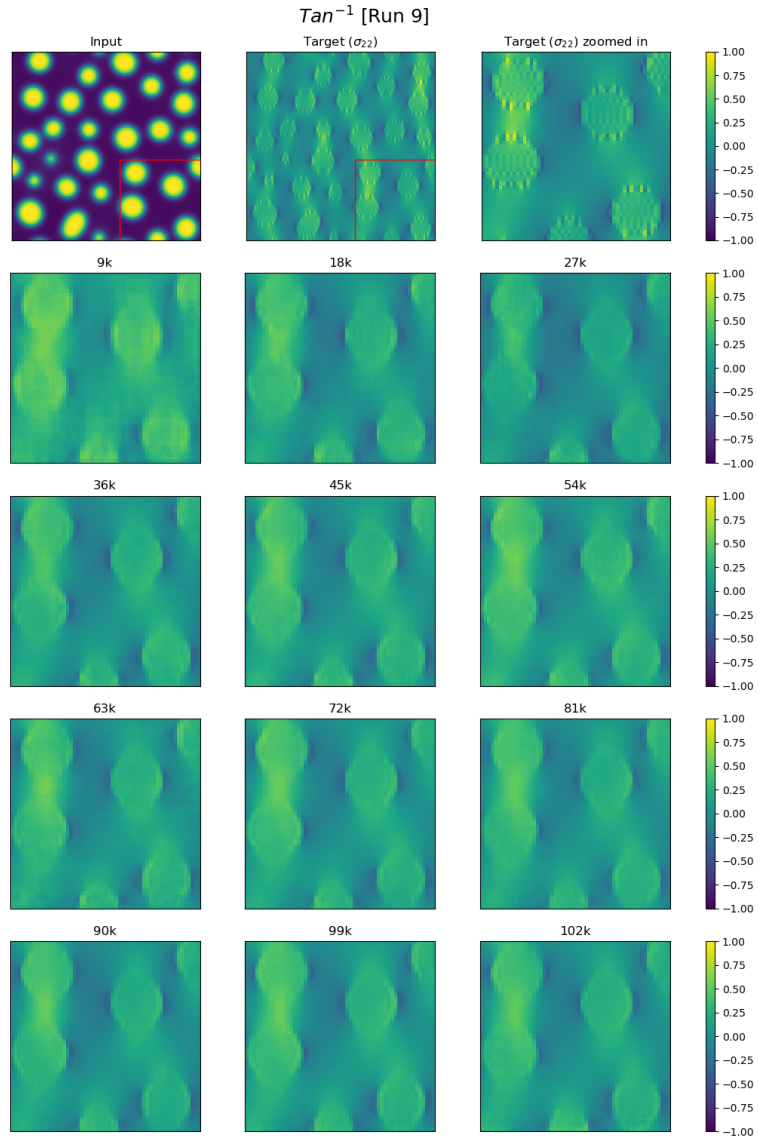


Fig. 13: Tan^{-1} predicted σ_{22} field from different iterations saved throughout a single training. Does not replicate Gibbs oscillations.