MetaChest: Generalized few-shot learning of pathologies from chest X-rays

Berenice Montalvo-Lezama[1,2*] and Gibran Fuentes-Pineda[2]

[1*]Posgrado en Ciencia e Ingeniería de la Computación.
[2]Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Circuito Escolar s/n, Ciudad Universitaria, Coyoacán, 04510, CDMX, Mexico.

*Corresponding author(s). E-mail(s): bereml@turing.iimas.unam.mx;
Contributing authors: gibranfp@unam.mx;

Abstract

The limited availability of annotated data presents a major challenge for applying deep learning methods to medical image analysis. Few-shot learning methods aim to recognize new classes from only a small number of labeled examples. These methods are typically studied under the standard few-shot learning setting, where all classes in a task are new. However, medical applications such as pathology classification from chest X-rays often require learning new classes while simultaneously leveraging knowledge of previously known ones, a scenario more closely aligned with generalized few-shot classification. Despite its practical relevance, few-shot learning has been scarcely studied in this context. In this work, we present MetaChest, a large-scale dataset of 479,215 chest X-rays collected from four public databases. MetaChest includes a meta-set partition specifically designed for standard few-shot classification, as well as an algorithm for generating multi-label episodes. We conduct extensive experiments evaluating both a standard transfer learning approach and an extension of ProtoNet across a wide range of few-shot multi-label classification tasks. Our results demonstrate that increasing the number of classes per episode and the number of training examples per class improves classification performance. Notably, the transfer learning approach consistently outperforms the ProtoNet extension, despite not being tailored for few-shot learning. We also show that higher-resolution images improve accuracy at the cost of additional computation, while efficient model architectures achieve comparable performance to larger models with significantly reduced resource requirements.

Keywords: few-shot learning, chest X-ray dataset, chest X-ray multi-label classification, meta-learning, deep learning

1 Introduction

In recent decades, deep learning has revolutionized medical image analysis, particularly in the field of radiology [1–3]. Deep neural networks have enabled the processing of large volumes of radiological data, the extraction of complex features, and the development of models that can enhance the accuracy of medical diagnoses. Despite these advances, a major challenge arises when only limited annotated data are available, as deep learning models typically require large amounts of labeled data to achieve strong performance. This issue is especially relevant to tasks such as pathology classification in chest X-rays, where labeled data can be scarce and difficult to obtain. To address this limitation, early research has explored the use of the standard few-shot classification (SFSC) paradigm, which aims to train models capable of generalizing to new classes using only a few labeled examples per class. However, this paradigm differs significantly from the way pathologies manifest in practice. From a clinical perspective, the objective is not merely to classify entirely new disease categories, but rather to distinguish between a combination of known pathologies and previously unseen ones. This highlights the need for approaches that are more closely aligned with the complexities of clinical settings.

This work aims to investigate the factors that influence the training of pathology classification models under a formulation that more closely reflects clinical scenarios. In particular, we study how different task instance configurations within the generalized few-shot learning (GFSL) paradigm affect model performance. In addition, we compare two training methods derived from the standard transfer learning and standard few-shot learning paradigms, evaluating them on tasks with a generalized few-shot learning formulation. Finally, we analyze the impact of image resolution and neural network architecture on classification performance.

To address these objectives, this paper makes the following contributions:

- We introduce MetaChest, a dataset comprising 479,215 chest X-ray images collected from four public databases, along with a meta-set partition specifically designed for standard few-shot classification.
- We provide an algorithm to generate multi-label episodes, enabling few-shot learning in multi-label settings.
- We propose ProtoNet-ML, an extension of ProtoNet for multi-label classification tasks.
- We conduct a comprehensive comparison of two methods, one based on standard transfer learning and the other on standard few-shot learning, across a wide range of tasks with varying complexity.
- We analyze the influence of image resolution and model architecture on pathology classification performance.

The rest of this article is organized as follows. Section 2 reviews prior work on pathology classification in chest X-rays using deep learning, including approaches based on transfer learning and meta-learning. Section 3 introduces the MetaChest dataset, detailing its pathology distribution and a partitioning scheme suitable for standard few-shot classification. Section 4 outlines the key differences between standard transfer learning, standard few-shot classification, and generalized few-shot classification. Section 5 proposes using generalized few-shot classification to generate tasks that more closely resemble clinical practice in pathology classification. It also introduces a multi-label episode generation algorithm and describes the two classification methods used in this work: BatchBased and ProtoNet-ML. Section 6 details the experimental setup and evaluation methodology, and presents and analyzes the experimental results. Finally, Section 7 summarizes the conclusions and outlines directions for future research.

2 Related work

In this section, we review related works on chest X-ray classification using deep learning techniques. We also discuss relevant transfer learning and meta-learning approaches, as well as their applications to the medical image domain.

2.1 Deep learning for chest X-ray classification

Deep neural networks, coupled with large-scale datasets, have enabled significant progress in several computer vision fields. Over the past few years, in an effort to take advantage of deep neural networks, datasets of increasingly specialized domains have been made publicly available. For instance, in the medical domain, multiple chest X-ray datasets have been introduced, such as CheXpert [4], Chest X-ray8 [5], Chest X-ray14 [5], MIMIC [6], MIMIC-CXR-JPG [7], OpenI [8], and PadChest [9]. These datasets have been fundamental in the development of deep learning models for chest X-ray analysis and generation tasks. In contrast to ImageNet [10], the scale of these datasets is at least one order of magnitude smaller. In addition, the distributions of these datasets are greatly heterogeneous, e.g. the number and kind of pathologies, the class imbalance, the collection and labeling procedure, the quality of the images, and the patient population.

With the introduction of these datasets, several studies addressing pathology classification from chest X-rays using deep learning started to appear. For binary classification (i.e. presence or absence), Lakhani and Sundaram [11] focused on tuberculosis identification, whereas Mabrouk et al. [12] targeted pneumonia identification. Since the X-ray of a patient may exhibit signs of multiple diseases, the identification of pathologies from chest X-rays has often been formulated as a multi-label classification problem. For instance, Baltruschat et al. [13] used a ResNet-50 [14] architecture to classify fourteen pathologies in the ChestX-ray14 dataset, where each X-ray could be assigned to more than one pathology. Similarly, Irvin et al. [4] compared various ConvNet architectures for multi-label chest X-ray classification using CheXpert, finding that DenseNet121 outperformed ResNet152 [14], Inception-v4 [15], and SE-ResNeXt101 [16].

2.2 Standard transfer learning

Transfer learning (TL) is a cornerstone of deep learning for image analysis, since it can reduce the amount of data and the computational resources required to train a model for a target task by leveraging the representations learned from one or multiple source tasks. In TL, multiple strategies to adapt the representations from the source task to the target task have been proposed. In practice, the most widely used transfer strategy has been standard transfer learning, which consists of pre-training models using a conventional batch-based training (as opposed to other training schemes). Specifically, ImageNet pre-training has been a standard practice for a wide variety of natural image tasks, including classification [17, 18], segmentation [19, 20] and object detection [21].

Due to the widespread use of ImageNet in practice, multiple works have studied the transferability of the learned representations from ImageNet to other natural image tasks [22–25]. Surprisingly, even though some studies have suggested that the source and target datasets must be closely related for an effective knowledge transfer [23], ImageNet pre-training has been used with seeming success for wildly dissimilar image domains (e.g. medical images [2, 5, 26]). In contrast, there are transferability studies in specific domains where ImageNet pre-training has not provided any improvement over random initialization [1, 3, 26].

Other transferability studies have focused on analyzing the effect of the architecture size and the scale of the training dataset on the effectiveness of STL. In intra-domain scenarios, where the source and target datasets are closely related, studies have been mainly focused on natural image datasets. For example, Kolesnikov et al. [18] and Zhai et al. [27] analyzed how the pre-training dataset size and the architecture depth influence knowledge transfer when both the source and target datasets are composed of natural images. The results from these studies have consistently shown better performance with larger architectures and pre-training datasets.

In inter-domain scenarios, where the source and target datasets belong to different domains (e.g., natural images and chest X-rays), existing studies are scarce, not very systematic and report mixed results. Raghu et al. [26] did not find significant differences on chest X-ray and retinal image classification performance using a ResNet-50 architecture when comparing ImageNet-1k pre-training with random initialization. Ke et al. [2] studied the effect of ImageNet-1k pre-training on chest X-ray classification performance using ConvNet architectures of different sizes. Their results showed a slight performance improvement when using deeper pre-trained architectures. On the other hand, Mustafa et al. [1] studied the influence of ImageNet-1k, ImageNet-21k, and JFT-300M pre-training on classification performance using ResNets of different sizes. The target tasks considered in this study were cancer identification from mammograms, pathology classification from chest X-rays, and skin condition from dermatological images. The results were far from conclusive, observing performance improvements with larger pre-training datasets and architectures only in some target tasks. Similarly, Cherti and Jitsev [3] carried out a comparative study of ResNet models pre-trained on ImageNet-1k, ImageNet-21k, and a combination of different chest X-ray datasets for pathology classification. They reported small improvements

in performance when models were pre-trained on larger source datasets and transferred to larger target datasets. However, no performance improvement was observed when transferring for smaller target datasets, regardless of the size of the pre-training dataset.

2.3 Few-shot classification

Meta-learning is a transfer learning strategy which aims to generate models that can be quickly adapted to new tasks [28]. As opposed to STL, in meta-learning, new tasks are commonly known as episodes and are typically small with respect to both the number of classes and the number of examples per class. The most widely studied problem in meta-learning is few-shot classification, which is a multi-class classification problem where a few examples per class are available for training (typically, 1 or 5).

The earliest works in meta-learning proposed methods for multi-class classification on natural image datasets [29–33], such as MiniImageNet [30] and FC100 [34]. These datasets are reduced versions of ImageNet and CIFAR100 and were created to facilitate episodic training.

More recent works have applied meta-learning to domain-specific problems, particularly through datasets of various medical imaging modalities. For instance, meta-learning methods have been studied in skin disease classification from dermatological images [35], COVID-19 classification from chest CT scans [36], and cancer classification from histological images [37, 38]. Moreover, meta-learning methods have also been used for image segmentation in CT scans, magnetic resonance images [39], and dermatological images [40].

3 MetaChest dataset

Over the past decades, several chest X-ray datasets have been collected, which vary in aspects such as number of examples, population of study, labeling strategy, period of time, pathologies, and source institution. Table 1 shows a comparison of publicly available chest X-ray datasets. In general, these datasets exhibit heterogeneous characteristics, ranging from a few thousand to hundreds of thousand of images collected for periods of a few years and up to a few decades. One key factor influencing the distribution of pathologies in a dataset is the patient population from which the chest X-rays were obtained. As can be observed in Table 1, most publicly available datasets were collected from medical institutions in the United States, albeit from different hospitals and regions. However, there are two datasets from other countries: PadChest, from Hospital San Juan in Spain, and VinDr-CXR, from multiple hospitals in Vietnam.

Clinical data collection is a complex process involving several tasks which can require a considerable amount of time and resources. Data labeling is one of the tasks that can generate greater variability among chest X-ray datasets; the rightmost column in Table 1 summarizes the labeling strategy employed by each dataset. Most datasets derived annotations automatically from radiology reports using natural language processing (NLP) methods, with the exception of PadChest which was annotated by expert radiologists. The specific strategy and tool used for annotating the chest X-rays directly influences the distribution of labels. For instance, MIMIC provides two sets

Dataset	#Pathol	#Images	Period	Source	Labelig Pipeline
OpenI	18	7,470	NA	Indiana Network for Patient Care, Indiana, USA	MeSH
chestX-ray8	8	108,948	1992-2015	National Institutes of Health, USA	MetaMap, DNorm, custom negation rules
ChestX-ray14	14	112,120	1992-2015	National Institutes of Health, USA	MetaMap, DNorm, custom negation rules
CheXpert	14	224,316	2002-2017	Stanford Hospital, California, USA	CheXpert
MIMIC	14	377,110	2011-2016	Beth Israel Deaconess Medical Center, Massachusetts, USA	CheXpert/NegBio
PadChest	19	168,861	2009-2017	Hospital San Juan Alicante, Spain	Physicians
VinDr-CXR	14	18,000	2018-2020	Hanoi Medical University Hospital and Hospital 108, Vietnam	VinDr Lab

Table 1: Comparison of publicly available chest X-ray datasets. Here, MIMIC [6] refers to MIMIC-CXR-JPG [7].

of labels that have different distributions: one generated by NegBio [41] and another by CheXpert [4].

An inherent characteristic of medical datasets is class imbalance, that is to say, the number of examples associated with one pathology is significantly larger than the number of examples associated with other pathologies. This is due to multiple factors, including the prevalence of each pathology in the population of study or even the severity of the pathology (which could lead to multiple subsequent chest X-rays).

3.1 Data

In order to have a dataset with a more general epidemiological distribution for evaluating pathology classification models trained on a few examples, we propose MetaChest, a combination of CheXpert, MIMIC, ChestX-ray14, and PadChest, which provides a meta-learning oriented partitioning suitable for few-shot learning scenarios. Only patients between 10 and 80 years were considered, and incomplete records as well as corrupted images were discarded. Overall, MetaChest comprises 479,215 chest X-ray images, of which 322,475 are multi-labeled. Each of these images is associated with one or more of the 15 most common pathologies across the four original datasets, resulting in a total of 596,494 different pathology instances. On the other hand, 156,740 images are normal or labeled as *not finding*, indicating that no specific abnormalities were observed in the original datasets.

The frequency of each pathology in MetaChest is shown in Figure 1. As can be observed, there is a pronounced class imbalance, with the most frequent pathology (Effusion) occurring nearly two orders of magnitude more often than the least frequent one (Hernia). With respect to labeling, MetaChest has a label cardinality (average number of labels per image) of 1.84 and a label density (label over the total number of labels; see Tsoumakas et al. [42]) of 0.12.

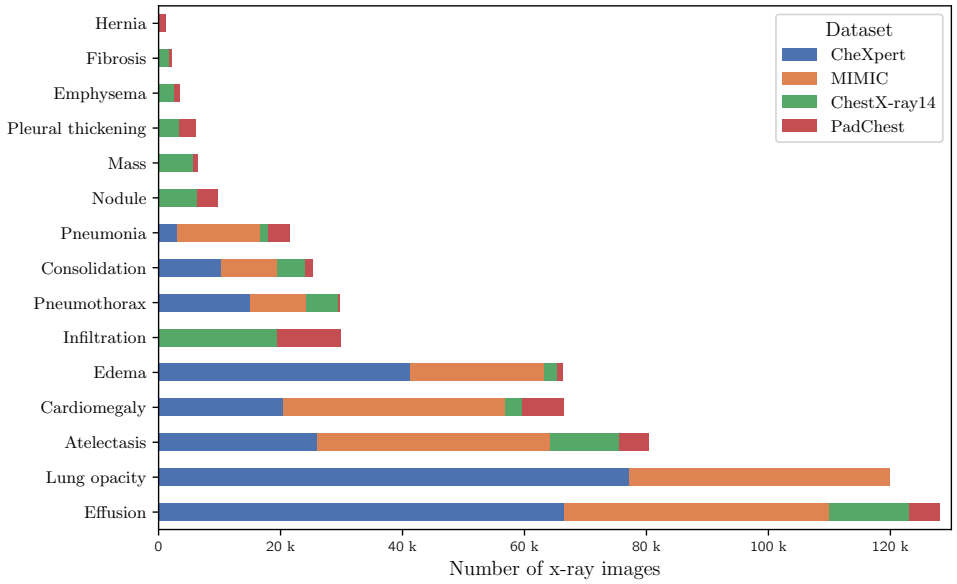

Fig. 1: Distribution of labels per pathology and dataset in MetaChest.

Label co-occurrence in MetaChest is illustrated in Figure 2. The most frequently co-occurring pathology pairs are Lung Opacity-Effusion, Effusion-Atelectasis and Effusion-Edema. Note that although Lung Opacity is the second most frequent pathology in MetaChest and frequently occurs together with five pathologies, there are seven pathologies with which it never presents together. Moreover, Hernia is the pathology that less commonly occurs together with other pathologies, which is expected since it is also the less frequent pathology in MetaChest.

The code used to generate MetaChest are publicly available at https://github.com/bereml/metachest and on the dataset's website at https://bereml.github.io/metachest/.

3.2 Meta-learning partition

In this work, we focus on pathology classification using small datasets with few classes and few examples per class. In particular, we consider an episode-based setting similar to SFSL, where the classifier is trained and evaluated across multiple episodes to study the model's behavior in scenarios with a small number of classes and few examples.

For this reason, we partition MetaChest classes into meta-training $\mathcal{C}_{meta-trn}$, meta-validation $\mathcal{C}_{meta-val}$, and meta-test $\mathcal{C}_{meta-tst}$ sets using the following procedure. First, for $\mathcal{C}_{meta-tst}$ we select the five pathologies with the fewest images that are present in all four original datasets. This allows the study of dataset shift and its impact on classification performance. Then, from the remaining nine pathologies, we select the five with the largest number of images for $\mathcal{C}_{meta-trn}$ and the other four for $\mathcal{C}_{meta-val}$. Unlike

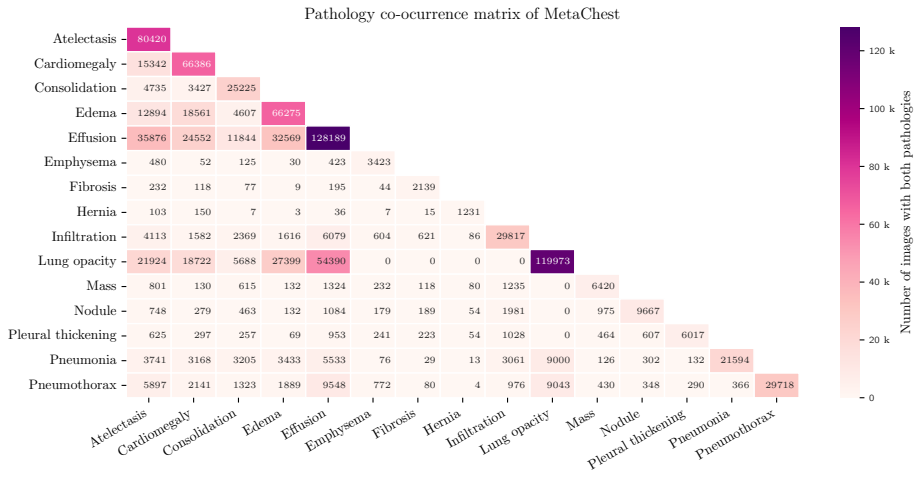

Fig. 2: Co-occurrence matrix of MetaChest pathologies.

the meta-test set, the meta-training and meta-validation sets contain pathologies that are not available in all four original datasets.

Table 2 shows the classes associated with each meta-set, along with the number of examples contributed by each original dataset. In general, CheXpert and MIMIC provide the largest number of labels for the meta-training and meta-test sets; together, these two datasets account for 77.33% and 86.24% of the total labels in the meta-training and meta-test sets, respectively. On the other hand, ChestX-ray14 and PadChest supply all the examples in the meta-validation set, as well as all the examples of several pathologies in the meta-training set. This is due to the absence of these pathologies in CheXpert and MIMIC. Note that although CheXpert contributes only three different pathologies (Effusion, Lung Opacity, and Atelectasis) to the meta-training set, it accounts for 44% of the total labels in this meta-set. In contrast, both PadChest and ChestX-ray14 contribute six out of seven pathologies, yet cover only the 7.14% and 15.51%, respectively, of the total labels in the meta-training set. Regarding normal images, there are 99,983 in the meta-training set, 1,788 in the meta-validation set, and 54,969 in the meta-test set.

Although Cherti and Jitsev [3] used a chest X-ray dataset that combined multiple datasets, it is not publicly available, its generation procedure is not described, and it does not provide appropriate partitions for meta-learning evaluation. Similarly, TorchXRayVision [43] is a library that allows the combination of different chest X-ray datasets, but it does not consider SFSL scenarios either. Conversely, MetaChest employs a disjoint class partition, enabling experimentation in SFSL settings. In addition, $\mathcal{C}_{meta-tst}$ is composed of the classes with the fewest examples available across the four original datasets, which is convenient for evaluating classification methods on images collected from multiple hospitals.

Classes	**MetaChest**	Datasets			
		CheXpert	MIMIC	ChestX-ray14	PadChest
$\mathcal{C}_{meta-trn}$					
Effusion	128,189	66,484	43,544	13,086	5,075
Lung opacity	119,973	77,194	42,779		
Atelectasis	80,420	25,980	38,297	11,335	4,808
Infiltration	29,817			19,362	10,455
Nodule	9,667			6,238	3,429
Mass	6,420			5,682	738
Pleural thickening	6,017			3,326	2,691
Total	*380,503*	*169,658*	*124,620*	*59,029*	*27,196*
$\mathcal{C}_{meta-val}$					
Emphysema	3,423			2,484	939
Fibrosis	2,139			1,650	489
Hernia	1,231			197	1,034
Total	*6,793*			*4,331*	*2,462*
$\mathcal{C}_{meta-tst}$					
Cardiomegaly	66,386	20,391	36,512	2,701	6,782
Edema	66,275	41,247	21,894	2,269	865
Pneumothorax	29,718	14,977	9,215	5,220	306
Consolidation	25,225	10,340	9,183	4,505	1,197
Pneumonia	21,594	2,986	13,679	1,381	3,548
Total	*209,198*	*89,941*	*90,483*	*16,076*	*12,698*

Table 2: Meta-training, meta-validation, and meta-test class sets with the corresponding number of label instances per pathology.

4 Transfer learning strategies

In this section, we describe the two transfer learning strategies used in this work and highlight their differences.

4.1 Standard transfer learning

Standard transfer learning (STL) is the most widely studied and spread strategy for computer vision tasks. When performing STL, we can identify two main stages [44]:

- Pre-training, which aims to acquire transferable knowledge from a source dataset $\mathcal{S}$.
- Adaptation, which leverages the knowledge acquired during pre-training to solve a task on a target dataset $\mathcal{T}$.

In the pre-training stage, $\mathcal{S}$ is divided into training $\mathcal{S}_{trn}$, validation $\mathcal{S}_{val}$ and test $\mathcal{S}_{tst}$ subsets. A randomly initialized neural network is then trained using batches $\mathcal{B}_{trn}$ sampled from $\mathcal{S}_{trn}$ and validated with batches $\mathcal{B}_{val}$ sampled from $\mathcal{S}_{val}$ to produce a pre-trained model. This process is commonly repeated with different hyperparameter configurations, yielding multiple pre-trained models. A single pre-trained model is

subsequently selected based on its performance on the validation subset $\mathcal{S}_{val}$. In some cases, the selected pre-trained model is also evaluated on the test subset $\mathcal{S}_{tst}$.

In the adaptation stage, the target dataset $\mathcal{T}$ is typically divided into train $\mathcal{T}_{trn}$, validation $\mathcal{T}_{val}$, and test $\mathcal{T}_{tst}$ subsets. In order to transfer the knowledge acquired from the source dataset, a pre-trained neural network is first assembled: the feature extraction layers (also known as the backbone) are preserved with their original weights and biases, while the layers specific to the pre-training task are replaced with randomly initialized layers tailored to the target task. Then, the assembled neural network is trained using batches $\mathcal{B}_{trn}$ sampled from $\mathcal{T}_{trn}$ and validated with batches sampled from $\mathcal{T}_{val}$ to produce the model for the target task. As in the pre-training stage, multiple models can be produced with different hyperparameter configurations, from which a single model is selected based on its performance on the validation subset $\mathcal{T}_{val}$. Finally, the performance of the selected model is estimated using the test subset $\mathcal{T}_{tst}$.

Note that the pre-training and adaptation stages in STL have some distinctive characteristics that are worth mentioning:

- The classes in the source dataset $\mathcal{S}$ and the target dataset $\mathcal{T}$ are typically different; that is to say, the classes encountered during the adaptation stage were not seen during pre-training.
- Training is a batch-based iterative process, in which all classes within $\mathcal{S}$ are considered.
- Although the target dataset is smaller than the source dataset, it typically contains examples on the order of hundreds or even thousands per category.
- The STL performance is evaluated on a single task T that considers all classes and examples in $\mathcal{T}_{tst}$.

4.2 Meta-learning

Meta-learning (MTL) is an alternative paradigm to STL, which aims to obtain models that can adapt to novel tasks with unseen classes and very few examples per class [44]. In other words, the objective of MTL is to achieve a more efficient transfer in terms of data. Similar to STL, the transfer process of MTL consists of a pre-training stage followed by an adaptation stage. Although in MTL these stages are commonly referred to as meta-training and meta-test [45], for the sake of consistency and clarity, we use the terms pre-training and adaptation for both STL and MTL. In this work, we are interested in two MTL formulations for classification: standard few-shot classification (SFSC) and generalized few-shot classification.

In SFSC, the pre-training stage is equipped with the meta-training $\mathcal{D}_{meta-trn}$ and meta-validation $\mathcal{D}_{meta-val}$ datasets, while the adaptation stage uses the meta-test dataset $\mathcal{D}_{meta-tst}$. During the pre-training stage an iterative training process is carried out. In each iteration, a classification task $E_{meta-trn}$ is randomly generated. This task is known as episode and is used to train the neural network. Each episode $E_{meta-trn}$ is composed of a training D_{trn} subset and a test D_{tst} subset, which share the same classes. To generate a meta-training episode $E_{meta-trn}$, n classes (known as n-way) are randomly selected from the set of meta-training classes $\mathcal{C}_{meta-trn}$. For each selected class, k_{trn} and k_{tst} examples are randomly sampled from $\mathcal{D}_{meta-trn}$ to

form the D_{trn} and D_{tst} subsets. Typically, an episode is 5-way, and the number of samples per class are $k_{trn} = 5$ and $k_{tst} = 15$. Once the model is trained with a meta-training episode $E_{meta-trn}$, its performance is evaluated with an episode $E_{meta-val}$ sampled from the meta-validation set $\mathcal{D}_{meta-val}$. This pre-training process is known as episodic training.

As opposed to STL, the adaptation stage in SFSC follows a similar iterative process as pre-training, except that the meta-test episodes $E_{meta-tst}$ are sampled from $\mathcal{D}_{meta-tst}$. The model's performance in the adaptation stage is estimated by averaging performance scores over hundreds or thousands of episodes. Note that while STL focuses on assessing the capacity of the model to adapt to a single task T that comprises all the examples and classes in the test subset of the target dataset, SFSC assesses the model's capacity to adapt to a large number of small episodes $E_{meta-tst}$ sampled from $\mathcal{D}_{meta-tst}$. In other words, SFSC aims to estimate the adaptability of the model to tasks with novel classes and a few examples per class.

On the other hand, the difference between SFSC and GFSC lies in the classes and examples that constitute the $\mathcal{D}_{meta-val}$ and $\mathcal{D}_{meta-tst}$ sets. In SFSC, the set of classes for $\mathcal{D}_{meta-val}$ ($\mathcal{D}_{meta-tst}$) is equal to $\mathcal{C}_{meta-val}$ ($\mathcal{C}_{meta-tst}$), which is disjoint from the set of classes for $\mathcal{D}_{meta-trn}$. In contrast, in GFSL the set of classes for $\mathcal{D}_{meta-val}$ ($\mathcal{D}_{meta-tst}$) is equal to $\mathcal{C}_{meta-trn} \cup \mathcal{C}_{meta-val}$ ($\mathcal{C}_{meta-trn} \cup \mathcal{C}_{meta-tst}$). Thus, GFSC can be regarded as a generalization of SFSC in which evaluation episodes are comprised not only of unseen classes sampled from $\mathcal{C}_{meta-val}$ ($\mathcal{C}_{meta-tst}$), but also seen classes from $\mathcal{C}_{meta-trn}$.

5 Methodology

In this section, we present a formulation of few-shot multi-label classification for chest X-rays and describe a transfer learning method and a meta-learning method, which will be compared through empirical experiments.

5.1 Few-shot multi-label classification for chest X-rays

In this work, we focus on generalized few-shot classification, since this formulation allows modeling common medical scenarios in which one seeks to classify opacities in an X-ray image that are associated with a combination of well-known pathologies and uncommon or even novel pathologies. Recall that in GFSC, a meta-validation or meta-test episode is composed of two types of classes. The first type are the seen classes, which are used in the meta-training episodes during the pre-training phase. In this sense, the seen classes are regarded as known information, even if the examples have not been previously seen. The second type is the unseen classes, which are completely new and appear only in meta-validation episodes during pre-training or meta-testing in the adaptation stage. These classes and examples are considered completely novel information. The greater the number of unseen classes, the more difficult the episode, due to the higher amount of novel information, reaching a limit at the SFSC formulation (i.e., when all classes in the episode are unseen). However, in medical scenarios, an X-ray image presents opacities that are mostly expected to be associated with known pathologies, which contrasts with SFSC, where all pathologies are unknown.

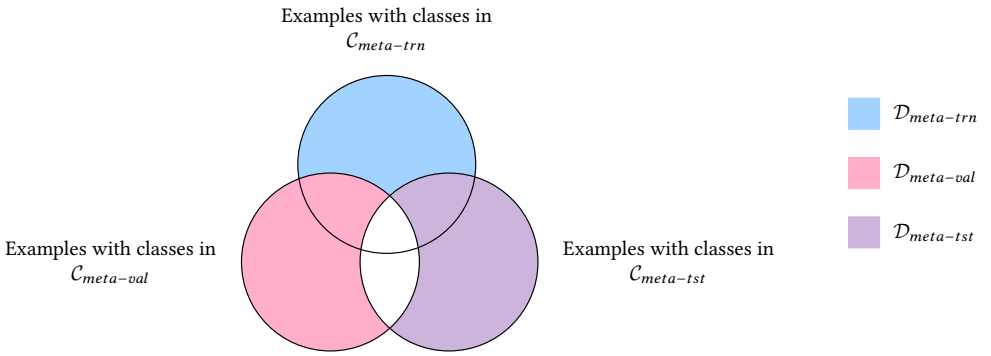

Fig. 3: Venn diagram illustrating the relationship between meta-training $\mathcal{C}_{meta-trn}$, meta-validation $\mathcal{C}_{meta-val}$, and meta-test $\mathcal{C}_{meta-tst}$ class sets. Blue indicates examples in $\mathcal{D}_{meta-trn}$, pink in $\mathcal{D}_{meta-val}$, and purple in $\mathcal{D}_{meta-tst}$. Since these sets are disjoint and episodes are generated from only one set at a time, the meta-validation and the meta-test episodes contain examples that are not used during meta-training.

To study this, we propose Algorithm 1, which generates multi-labeled episodes and allows control over the number of seen and unseen classes, as well as the minimum number of examples per class. Due to the multi-label nature of MetaChest, for the generation of episodes, the data is divided into $\mathcal{D}_{meta-trn}$, $\mathcal{D}_{meta-val}$, and $\mathcal{D}_{meta-tst}$ sets, as shown in Figure 3. This division ensures that no examples are shared between meta-training, meta-validation, and meta-test episodes, making the classification task more challenging and contributing to a more robust evaluation.

The pseudocode in Algorithm 1 outlines the meta-validation episode generation process, which can be applied similarly to meta-test episodes. First, we sample a set C_{seen} of n_{seen} classes from the meta-training classes $\mathcal{C}_{meta-trn}$, and sort them in ascending order according to their frequency in MetaChest (lines 1 and 2). Analogously, we sample the set C_{unseen} for the unseen classes (lines 3 and 4). Given the multi-label nature of the data, we identify the set of excluded classes $C_{excluded}$ (line 5), which keeps track of the classes that do not belong to C_{seen} or C_{unseen}. Then, we generate the sample set $\mathcal{D}'$ consisting of examples x in MetaChest that are not labeled with any of the excluded classes $C_{excluded}$ (line 6), thereby avoiding the introduction of additional classes into the episodes. Next, we generate the training subset D_{trn} (lines 8 to 15). For each class c in $(C_{seen} \cup C_{unseen})$, we determine the number of missing examples $k_{missing}$ in D_{trn} needed to reach k_{trn} (lines 9 and 11). After that, we sample a set D_c with $k_{missing}$ examples from $\mathcal{D}'$ with class c and add it to D_{trn} (lines 12 and 13). Finally, we add a *not finding* X-ray example (line 15) to ensure that for every class there will be a negative example in D_{trn}, which allows calculation the ROC-based metrics used in this work. The test subset D_{tst} is generated in an analogous manner.

5.2 Classification methods

Let us denote the set of examples that are labeled with a class c in the training episode as $D_c = \{(\mathbf{x}, \mathbf{y}) \mid (\mathbf{x}, \mathbf{y}) \in D_{trn} \text{ and } \mathbf{y}[c] = 1\}$, where $\mathbf{x} \in \mathbb{R}^{h \times w \times 3}$ is a $h \times w$ image,

Algorithm 1: Meta-validation episode generator.

Data: MetaChest dataset $\mathcal{D}$
Data: Sets of classes $\mathcal{C}_{meta-trn}$ and $\mathcal{C}_{meta-val}$
Data: Number of seen n_{seen} and unseen n_{unseen} classes
Data: Number of traning k_{trn} and test k_{tst} examples per class
Result: Meta-validation episode $E_{meta-val}$

$C_{seen} \leftarrow$ sample n_{seen} classes from $\mathcal{C}_{meta-trn}$;
$C_{seen} \leftarrow$ sort C_{seen} by increasing frequency in $\mathcal{D}$;
$C_{unseen} \leftarrow$ sample n_{unseen} classes from $\mathcal{C}_{meta-val}$;
$C_{unseen} \leftarrow$ sort C_{unseen} by increasing frequency in $\mathcal{D}$;
$C_{excluded} \leftarrow \overline{(C_{seen} \cup C_{unseen})}$;
$\mathcal{D}' \leftarrow \{x \mid x \in \mathcal{D}$ and x is not labeled with a class $c \in C_{excluded}\}$;
$D_{trn} \leftarrow \emptyset$, $D_{tst} \leftarrow \emptyset$;
foreach $(D, k) \in \{(D_{trn}, k_{trn}), (D_{tst}, k_{tst})\}$ **do**
    foreach $c \in C_{seen} \cup C_{unseen}$ **do**
        $k_{present} \leftarrow |\{x \mid x \in D$ and x is labeled with $c\}|$;
        $k_{missing} \leftarrow k - k_{present}$;
        $D_c \leftarrow$ sample $k_{missing}$ examples labeled with c from $\mathcal{D}'$;
        $D \leftarrow D \cup D_c$;
        $\mathcal{D}' \leftarrow \mathcal{D}' - D_c$;
    $D \leftarrow D \cup \{1$ *not finding* image sampled from $\mathcal{D}\}$;
$E_{meta-val} \leftarrow (D_{trn}, D_{tst})$;
return $E_{meta-val}$

and $\mathbf{y} \in \{0,1\}^{n-\text{way}}$ is the associated multi-label vector. Furthermore, we denote $f_\phi(\mathbf{x}) \in \mathbb{R}^D$ as the D-dimensional vector representation of $\mathbf{x}$, computed by a backbone f_ϕ with trainable parameters ϕ.

5.2.1 ProtoNet-ML

ProtoNet [31] is a multi-class classification method that has been widely studied in the SFSC literature. In this work, we propose an extension to handle multi-label classification, which we call ProtoNet-ML. Following the original method, ProtoNet-ML computes a D-dimensional prototype $\mathbf{z}_c \in \mathbb{R}^D$ for each class c as follows:

$$\mathbf{z}_c = \frac{1}{|D_c|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in D_c} f_\phi(\mathbf{x}_i)$$

The original multi-class ProtoNet estimates class probabilities by applying a softmax over the negative distances between a test example and the class prototypes, implicitly associating the test example to the closest prototype. To enable associations with multiple prototypes, ProtoNet-ML introduces a transformation function over the

distances. Specifically, the transformation function $t : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ between a test example $(\mathbf{x}, \mathbf{y}) \in D_{tst}$ and the prototype $\mathbf{z}_c$ for the class c is defined as:

$$t(f_\phi(\mathbf{x}), \mathbf{z}_c) = \mu_c - d(f_\phi(\mathbf{x}), \mathbf{z}_c)$$

where $d(f_\phi(\mathbf{x}), \mathbf{z}_c) = ||f_\phi(\mathbf{x}) - \mathbf{z}_c||$ is the Euclidean distance, and μ_c is the mean distance between the prototype for the class c and all training examples in the episode, i.e.,

$$\mu_c = \frac{1}{|D_{trn}|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in D_{trn}} d(f_\phi(\mathbf{x}_i), \mathbf{z}_c)$$

Subtracting the example-prototype distance from the mean distance maps examples closer than the mean to increasingly positive values, while those farther away are mapped to increasingly negative values. This transformation can be used to compute a probability distribution for a test example $\mathbf{x}$ belonging to class c as follows:

$$p(\mathbf{y}[c] = 1 \mid \mathbf{x}) = \sigma(t(f_\phi(\mathbf{x}), \mathbf{z}_c))$$

where σ denotes the sigmoid function. Unlike multi-class prototypes, which partition the D-dimensional representation space into disjoint subspaces, multi-label prototypes correspond to subspaces that may overlap. This allows the representation of a single example to fall into more than one subspace at the same time, as shown in Figure 4.

Beyond the Euclidean distance, ProtoNet-ML can be instantiated with other functions, including the Minkowski distance and, with slight modifications, the cosine distance. Moreover, ProtoNet-ML is a flexible method that supports arbitrary activation functions and can operate directly on logits. In our experiments, however, we employ the sigmoid function, as it is the conventional and most natural choice for binary classification.

5.2.2 BatchBased

BatchBased is a method inspired by [46] that employs STL-based training while maintaining MTL-based episode evaluation. On top of the backbone f_ϕ, BatchBased adds a head module g_φ (single fully connected layer) with trainable parameters φ. The probability distribution for a test example $\mathbf{x}$ given a class c is computed as:

$$p(\mathbf{y}[c] = 1 \mid \mathbf{x}) = \sigma(g_\varphi(f_\phi(\mathbf{x})))$$

As in STL, the learning process is carried out in epochs, during which input data is fed to the model in batches. Note that STL batches are constructed from all classes in $\mathcal{C}_{meta-trn}$, whereas MTL episodes involve only a subset of these classes. The model parameters ϕ and φ are updated with each batch by backpropagating through the entire network. After the epoch is completed, an episode-based evaluation is conducted following the MTL paradigm. Specifically, for each $E_{meta-val}$ or $E_{meta-tst}$ episode, the f_ϕ parameters are frozen, while the head module g_φ is replaced and updated on D_{trn}. To update the head parameters φ, an iterative process is repeated t_{steps} steps. At each step, a subset of examples M is randomly sampled from D_{trn}, where $|M|$ is a proportion ptc_{trn} of D_{trn}. Then, the head parameters φ are updated with a

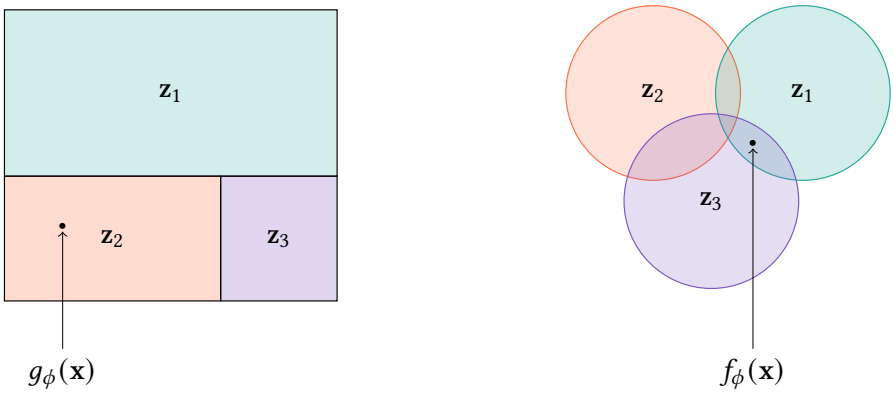

Fig. 4: Representation subspaces for multi-class ProtoNet (left) and ProtoNet-ML (right). In multi-class ProtoNet, a representation vector $g_\phi(\mathbf{x})$ is associated with only one prototype, whereas in ProtoNet-ML, a representation vector $f_\phi(\mathbf{x})$ may be associated with one or more prototypes. Note that $f_\phi(\cdot)$ is the network backbone followed by an encoding layer to reduce representation vector dimensionality.

learning rate lr_{head} through backpropagation using M. Here, t_{steps}, ptc_{trn}, and lr_{head} are considered as hyperparameters.

6 Results and discussion

In this section, we analyze the adaptation process of different models across various formulations of the multi-label classification task using few examples per pathology. First, we compare how two distinct learning paradigms leverage ImageNet pre-training. Next, we examine their behavior across a broad set of few-shot learning tasks designed to reflect challenges commonly encountered in medical settings. In addition, we investigate factors that influence the adaptation process, including image resolution, and variations in architectural connectivity patterns. Finally, we study the influence of hyperparameters on classification performance and provide illustrative examples of the resulting model predictions.

6.1 Experimental setup

For empirical evaluation, certain training and method hyperparameters are fixed, while others are varied to assess their impact on classification performance. The experimental setup is described in detail below. The code to reproduce our main findings is publicly available at https://github.com/bereml/meta-cxr.

Training

The default hyperparameter configurations are listed in Table 3. Unless otherwise specified, the reported results correspond to the BatchBased configuration.

Parameter	Configuration
Data	
Distribution	Complete
Image size	384
Task	
Training batch size	64
Training n-way, k_{trn} k_{tst}	3, 30, 30
Validation n-way, k_{trn} k_{tst}	3, 30, 30
Test n-way, k_{trn} k_{tst}	3, 30, 30
Backbone	
Architecture	MobileNetV3Small075
Pre-training	I1K
Training	
Meta-trn, meta-val, meta-tst episodes	1,000, 100, 10,000
Max epochs	150
Optimizer	AdamW
Stop metric, patience	HM, 10
Float precision	16bit
BatchBased	
Meta-trn LR	0.0001
Meta-val t_{steps}, ptc_{trn}, lr_{head}	100, 0.5, 0.05
Meta-tst t_{steps}, ptc_{trn}, lr_{head}	100, 0.5, 0.05
ProtoNet-ML	
Encoding layer type, size	Average pooling, 128
Meta-training LR	0.0001

Table 3: Default hyperparameter configurations.

Evaluation

Model performance is measured as the average over 10,000 episodes sampled from the meta-test set. As is common in GFSL [47], we evaluate seen and unseen classes separately and report the harmonic mean of their scores. However, we adopt AUC–ROC instead of accuracy [48, 49] to align with evaluation standards in the medical domain.

We employ three metrics commonly used in GFSL [47], each reported with a 95% confidence interval: one computed for the seen classes, another for the unseen classes, and a third one for the harmonic mean (HM), which are defined as follows.

- *Seen*: The AUC-ROC of all labels of the seen classes in the episode as a single binary classification task.
- *Unseen*: The AUC-ROC of all labels of the unseen classes in the episode as a single binary classification task.
- HM: The harmonic mean of *Seen* and *Unseen*, i.e.:

$$HM = \frac{2 \times Seen \times Unseen}{Seen + Unseen}$$

Note that the harmonic mean is commonly used in GFSL because it mitigates the dominance of seen classes in the overall performance [47].

6.2 Leveraging ImageNet

We begin the comparison between BatchBased and ProtoNet-ML using models that are randomly initialized and pre-trained on either ImageNet-K or ImageNet-21K. This experiment was performed using a MobileNetV3Large100 architecture, as it is the only pre-trained model available on both versions of ImageNet.

As shown in Table 4, BatchBased consistently outperforms ProtoNet-ML across all models and metrics. For instance, on ImageNet-1K, BatchBased surpasses ProtoNet-ML by 4.31 HM points. When comparing the ImageNet-1K and ImageNet-21K models for BatchBased, the former achieves better results across all metrics. For example, ImageNet-1K yields an improvement of 0.78 HM points compared to ImageNet-21K. Furthermore, BatchBased initialized with ImageNet-21K weights demonstrates a 4.47 HM point gain over randomly initialized models. The literature on few-example regimes in inter-domain scenarios reports inconclusive findings regarding the benefits of using pre-trained models on ImageNet-1K Cherti and Jitsev [3]. However, our results indicate that using pre-trained models consistently improves performance on chest X-ray images.

Model	*Seen* ↑	*Unseen* ↑	*HM* ↑
BatchBased			
Random	82.42±0.14	78.17±0.35	78.83±0.25
ImageNet-1K	**86.49±0.11**	**83.80±0.31**	**84.08±0.22**
ImageNet-21K	85.89±0.12	82.98±0.32	83.30±0.22
ProtoNet-ML			
Random	76.48±0.14	75.69±0.34	74.83±0.23
ImageNet-1K	82.10±0.12	79.45±0.30	79.77±0.20
ImageNet-21K	81.89±0.12	80.18±0.31	80.06±0.21

Table 4: Comparison of randomly initialized and ImageNet-pretrained MobileNetV3Large100 models for BatchBased and ProtoNet-ML.

6.3 Few-shot learning vs transfer learning

Building on the results from the previous subsection, we now examine different aspects inherent to few-shot classification for BatchBased and ProtoNet-ML. Table 5 compares the results of both methods across different task configurations, while Figure 5 illustrates the behavioral trends of each method.

We observe that ProtoNet-ML achieves improved performance in only a limited subset of task configurations. Table 5 shows that these improvements occur primarily in the 1-unseen setting and tend to disappear as the number of shots (k-shot) or classes (n-way) increases. In the remaining task configurations, BatchBased outperforms

k-shot	3-way		4-way		5-way	
	BatchBased	ProtoNet-ML	BatchBased	ProtoNet-ML	BatchBased	ProtoNet-ML
1-unseen						
1	70.32±0.31	**73.28±0.21**	70.61±0.27	**73.44±0.17**	71.42±0.24	**73.56±0.15**
5	75.63±0.29	**79.13±0.18**	79.23±0.20	**79.38±0.13**	**81.41±0.15**	79.56±0.11
15	80.28±0.26	**80.51±0.19**	**83.61±0.14**	81.06±0.12	**84.71±0.10**	81.27±0.10
30	**82.57±0.23**	80.47±0.20	**84.66±0.12**	81.06±0.12	**85.34±0.08**	81.24±0.10
2-unseen						
1	**67.89±0.20**	66.06±0.15	**69.24±0.15**	66.97±0.13	**69.68±0.13**	67.58±0.12
5	**76.22±0.15**	70.70±0.11	**77.69±0.10**	71.81±0.09	**78.22±0.09**	72.87±0.08
15	**80.20±0.14**	71.22±0.11	**81.26±0.09**	72.53±0.08	**81.48±0.07**	73.91±0.07
30	**81.75±0.13**	71.15±0.11	**82.86±0.08**	72.58±0.08	**82.95±0.07**	74.01±0.07
3-unseen						
1	**57.25±0.12**	56.75±0.10	**68.01±0.15**	66.20±0.13	**68.81±0.12**	66.56±0.11
5	**65.08±0.11**	59.59±0.09	**75.31±0.11**	70.89±0.09	**76.51±0.07**	71.52±0.07
15	**71.04±0.09**	60.57±0.08	**78.89±0.10**	71.69±0.08	**79.88±0.06**	72.49±0.07
30	**74.02±0.08**	60.96±0.08	**80.59±0.10**	71.69±0.08	**81.51±0.06**	72.57±0.07
4-unseen						
1			57.86±0.10	**58.03±0.08**	**68.30±0.13**	66.55±0.12
5			**65.13±0.08**	61.06±0.07	**75.12±0.09**	71.57±0.07
15			**70.38±0.07**	62.01±0.06	**78.51±0.08**	72.53±0.07
30			**73.16±0.06**	62.35±0.06	**80.18±0.08**	72.68±0.07
5-unseen						
1					58.79±0.09	**59.34±0.07**
5					**65.73±0.07**	62.59±0.05
15					**70.43±0.05**	63.52±0.04
30					**73.07±0.05**	63.87±0.04

Table 5: Comparison of BatchBased and ProtoNet-ML on pathology classification tasks evaluated with HM.

ProtoNet-ML. Moreover, as shown in Figure 5, BatchBased demonstrates improved performance as the number of shots increases, whereas ProtoNet-ML's performance remains nearly constant. These findings are consistent with those reported by Luo et al. [50], who compared the performance of conventional models with meta-learning methods such as ProtoNet on SFSC tasks across various natural image datasets. They found that conventional models tend to scale better than meta-learning approaches, particularly on fine-grained datasets. In medical applications, where datasets often include many classes and dozens of examples per class, our results suggest that BatchBased is the more effective approach for training pathology classifiers.

6.4 Pathology Classification Complexity

We evaluate the effectiveness of BatchBased by varying the number of classes per episode, the number of unseen classes, and the number of examples per class.

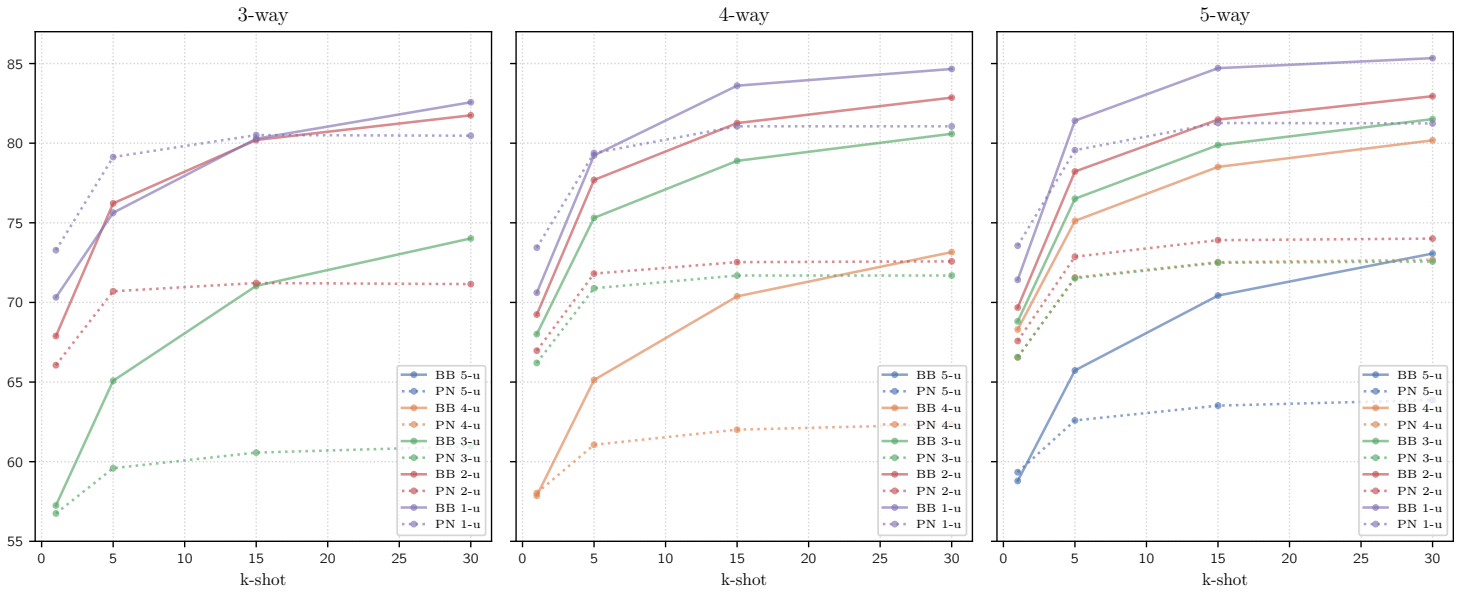

Fig. 5: Performance of BatchBased (BB) and ProtoNet-ML (PN) on pathology classification tasks across varying n-way, n-unseen, and k-shot configurations.

The results of these experiments are presented in Table 6, and the corresponding performance trends are illustrated in Figure 6.

Classes per episode n-way

We observe that, as the number of classes in the episode increases while the number of unseen classes is held constant, performance improves consistently. For example, in Table 6, the 5-way, 3-unseen configuration with 15-shot outperforms the 4-way, 3-unseen configuration with the same number of shots by 0.99 HM points, highlighting the performance gain from adding a single class. As shown in Figure 6, the 4-way configurations (solid orange line) consistently achieve higher HM scores than the 3-way configurations (solid blue line), while the 5-way configurations (solid green line) outperform the 4-way configurations (solid orange line). This suggests that increasing the number of classes per episode, and consequently the number of examples, reduces task complexity and leads to improved performance. Similar results have been reported in the SFSC literature on natural image datasets, where a higher number of classes per episode consistently improves classification performance [50].

Unseen classes n-unseen

We observe that performance decreases as the number of unseen classes increases, as illustrated by the downward trends in the HM curves in Figure 6. From a learning paradigm perspective, this allows us to analyze the complexity of episodes when

n-unseen	1-shot			5-shot		
	Seen ↑	*Unseen* ↑	*HM* ↑	*Seen* ↑	*Unseen* ↑	*HM* ↑
3-way						
1	77.42±0.22	68.15±0.41	70.32±0.31	83.45±0.14	72.41±0.40	75.63±0.29
2	80.26±0.30	60.70±0.20	67.89±0.20	84.18±0.21	70.63±0.17	76.22±0.15
3		57.25±0.12	57.25±0.12		65.08±0.11	65.08±0.11
4-way						
1	77.29±0.17	67.60±0.36	70.61±0.27	83.41±0.11	76.98±0.29	79.23±0.20
2	79.13±0.17	62.53±0.19	69.24±0.15	84.40±0.10	72.47±0.15	77.69±0.10
3	81.13±0.27	59.78±0.14	68.01±0.15	84.66±0.19	68.43±0.11	75.31±0.11
4		57.86±0.10	57.86±0.10		65.13±0.08	65.13±0.08
5-way						
1	77.18±0.14	68.65±0.34	71.42±0.24	83.26±0.10	80.46±0.22	81.41±0.15
2	78.15±0.14	63.60±0.18	69.68±0.13	83.66±0.08	73.78±0.13	78.22±0.09
3	79.84±0.16	61.04±0.14	68.81±0.12	84.76±0.09	69.99±0.10	76.51±0.07
4	81.65±0.25	59.72±0.12	68.30±0.13	85.07±0.18	67.76±0.08	75.12±0.09
5		58.79±0.09	58.79±0.09		65.73±0.07	65.73±0.07

n-unseen	15-shot			30-shot		
	Seen ↑	*Unseen* ↑	*HM* ↑	*Seen* ↑	*Unseen* ↑	*HM* ↑
3-way						
1	85.08±0.12	78.60±0.36	80.28±0.26	85.33±0.12	82.13±0.32	82.57±0.23
2	85.48±0.19	76.29±0.15	80.20±0.14	85.83±0.19	78.70±0.14	81.75±0.13
3		71.04±0.09	71.04±0.09		74.02±0.08	74.02±0.08
4-way						
1	85.29±0.09	82.71±0.21	83.61±0.14	85.60±0.08	84.30±0.18	84.66±0.12
2	86.03±0.09	77.36±0.13	81.26±0.09	86.68±0.09	79.69±0.12	82.86±0.08
3	85.90±0.18	73.44±0.09	78.89±0.10	86.48±0.17	75.90±0.08	80.59±0.10
4		70.38±0.07	70.38±0.07		73.16±0.06	73.16±0.06
5-way						
1	85.22±0.08	84.56±0.15	84.71±0.10	85.59±0.07	85.36±0.13	85.34±0.08
2	85.44±0.07	78.13±0.11	81.48±0.07	86.21±0.06	80.15±0.11	82.95±0.07
3	86.30±0.08	74.52±0.08	79.88±0.06	87.03±0.08	76.80±0.08	81.51±0.06
4	86.34±0.16	72.39±0.07	78.51±0.08	86.91±0.16	74.79±0.06	80.18±0.08
5		70.43±0.05	70.43±0.05		73.07±0.05	73.07±0.05

Table 6: *Seen*, *unseen*, and *HM* metrics for pathology classification tasks across n-way, n-unseen, and k_{trn} configurations.

transitioning from a GFSL formulation (less novel information) to a standard few-shot learning SFSL formulation (entirely novel information). Notably, the performance drop in our experiments is considerably larger when transitioning from GFSL (with at least one seen class) to SFSL (with all classes unseen).

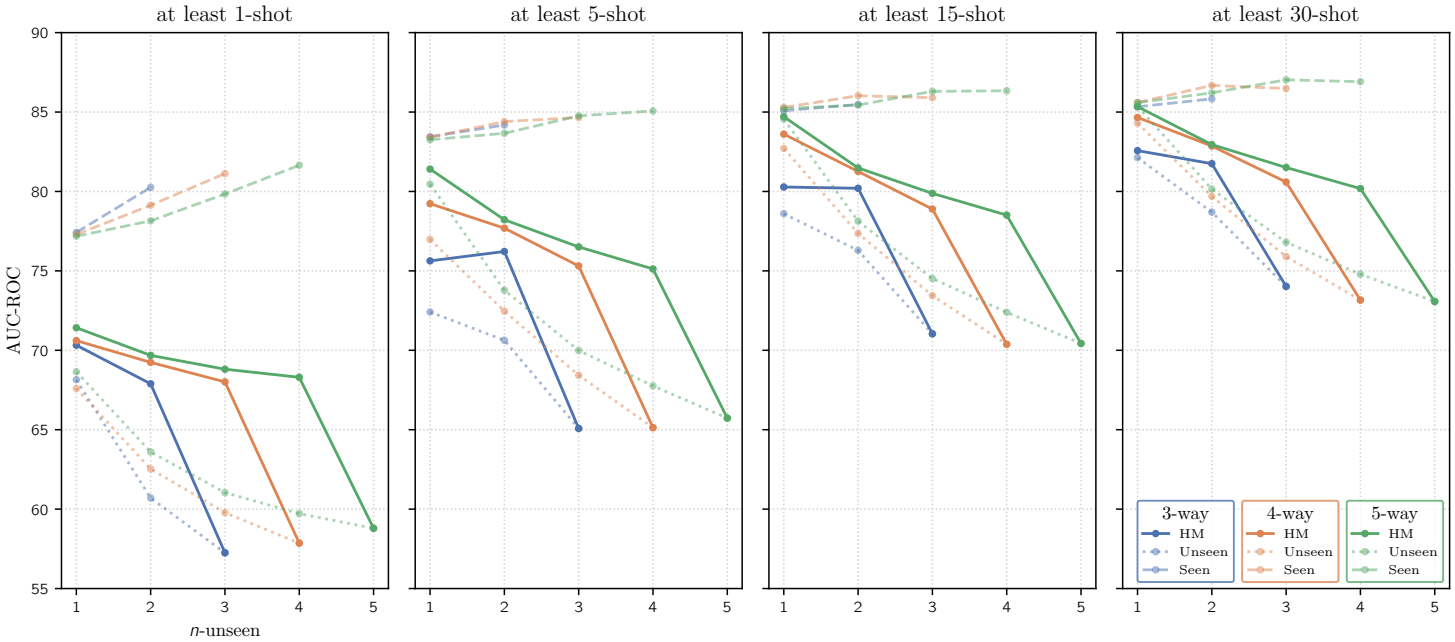

Fig. 6: Performance on pathology classification tasks with few examples, varying the number of classes (n-way) from 3 to 5, the number of unseen classes (n-unseen) from 1 to n-way, and the number of training shots per class (k_{trn}) to at least 5, 10, 15, and 30. Solid lines indicate the harmonic mean (HM), dashed lines indicate the AUC-ROC for seen classes, and dotted lines indicate the AUC-ROC for unseen classes.

Examples per class k_{trn}

We observe that performance steadily improves as the number of examples per class increases. This trend is clearly illustrated in Figure 6, which reveals a progressive improvement in performance across the subfigures corresponding to at least 1, 5, 15, and 30 shots per class. For example, in Table 6, under the 5-way 1-unseen configuration, performance improves by 9.99, 13.29, and 13.92 HM points for $k_{trn} = 5$, $k_{trn} = 15$, and $k_{trn} = 30$, respectively, compared to $k_{trn} = 1$. This suggests that increasing the number of examples reduces task complexity, thereby enabling the model to achieve higher performance.

Confidence interval

We consistently observe that increasing the number of classes results in narrower confidence intervals. Similarly, increasing the number of examples per class yields narrower intervals for both seen and unseen classes.

6.5 X-ray resolution

In most cases, deep neural networks used for natural image classification are trained on low-resolution images (typically 224×224 or 256×256 pixels) to reduce computational cost. Such resolutions are adequate for datasets like ImageNet, which involve coarse-grained classification tasks characterized by visually distinct categories (e.g., cars and dogs). Even in few-shot classification tasks on mini-ImageNet, a resolution of 64 and shallow architectures (usually 4 to 6 layers) are commonly used, helping mitigate the parameter explosion.

In contrast, classifying pathologies on chest X-rays is a fine-grained task, as the opacity patterns that distinguish different pathologies are often extremely subtle. The literature on the effect of resolution is limited, particularly in the context of few-shot classification. Consequently, determining the most appropriate resolution for pathology classification on chest X-rays remains an important open research question.

In this experiment, we trained models using three different architectures and systematically varied the X-ray resolution to study its impact. Images were resized using the Lanczos algorithm, a high-quality resampling method known for preserving edge sharpness and fine details [51]. The results are summarized in Table 7.

Resolution	*Seen* ↑	*Unseen* ↑	*HM* ↑
MobileNetV3-Small-0.75			
224	84.29±0.13	81.75±0.32	81.87±0.23
384	85.73±0.12	82.61±0.32	83.03±0.22
512	85.89±0.12	82.53±0.32	83.06±0.22
768	**86.27±0.11**	**82.92±0.31**	**83.49±0.22**
1024	86.39±0.11	82.54±0.32	83.23±0.23
ConvNext-Tiny			
224	87.22±0.11	84.50±0.30	84.88±0.21
384	87.85±0.10	84.58±0.30	85.22±0.21
512	88.09±0.10	84.44±0.30	85.24±0.21
768	**88.16±0.10**	**84.53±0.30**	**85.29±0.22**
DenseNet-121			
224	84.97±0.12	83.27±0.29	83.17±0.21
384	85.04±0.12	82.97±0.29	83.03±0.20
512	**85.39±0.12**	**83.37±0.29**	**83.43±0.20**

Table 7: Comparison of MobileNetV3-Small-0.75, ConvNeXt-Tiny, and DenseNet-121 models on chest X-rays at varying input resolutions (224×224, $384\times$, 512×512, 768×768, and 1024×1024). A fixed batch size of 32 was used to ensure a fair comparison across architectures.

All three evaluated architectures show improved performance at a resolution of 384×384, which is higher than the resolution commonly used in ImageNet. For instance, MobileNetV3-Small-0.75 improves by 1.16 HM points, while ConvNeXt-Tiny and DenseNet-121 achieve gains of 0.34 and 0.14 HM points, respectively.

These results are consistent with previous findings in the medical imaging literature under a complete data regime. In mammography, for example, lesions are detected more accurately in images with a resolution of 1700×2100 pixels [52]. Similarly, in the case of chest X-rays, Rochmawanti and Utaminingrum [53] compared the performance of two models on the ChestX-ray14 dataset using resolutions of 64×64 and 320×320 pixels, finding better performance with higher resolution.

For MobileNetV3-Small-0.75, performance progressively improves as the resolution increases up to 768×768, but begins to decline at higher resolutions, as shown in Figure 7. In contrast, both ConvNeXt-Tiny and DenseNet-121 exhibit consistent improvements with increasing resolution. ConvNeXt-Tiny outperforms the other two architectures across all evaluated resolutions. The highest performance is achieved with this architecture at a resolution of 768×768, although it surpasses the best result of MobileNetV3-Small-0.75 by only 1.8 HM points. ConvNeXt-Tiny improves upon MobileNetV3-Small-0.75 at the default resolution used in this work (384×384) by just 2.26 HM points.

This finding is particularly relevant because increasing image resolution has an strong impact on memory requirements and computational cost for both training and inference. In particular, this affects the memory needed for intermediate computations, gradients, and activations within the neural network, making high-resolution training substantially more demanding. Computational cost also increases sharply, as higher resolutions require more multiply-accumulate operations (MACs) in each layer. In addition, the GPU memory usage grows, limiting batch sizes and potentially slowing training. Model complexity further interacts with image resolution: deeper or wider architectures may struggle to process very high-resolution inputs efficiently without optimization strategies such as mixed precision. Because of these constraints, certain experiments could not be performed. For example, training ConvNeXt-Tiny at 1024×1024 and DenseNet-121 at 768×768 and 1024×1024 was not feasible due to GPU memory limitations and excessive computational cost, highlighting a practical limitation in scaling experiments to very high-resolution images.

6.6 Architectures

We investigate how connectivity patterns and the number of parameters/operations influence pathology classification performance. This is particularly relevant because evidence from the complete data regime in language modeling [54, 55] and computer vision [18, 56] suggests that increasing network size and training data consistently reduces error. For meta-learning, Chen et al. [46] evaluated several convolutional architectures in few-shot multi-class classification on mini-ImageNet and a reduced version of the CUB dataset [57]. Their findings are inconclusive: while deeper architectures improved performance on CUB, gains on mini-ImageNet were observed only in certain cases.

In this experiment, we study the effect of convolutional and attention-based connectivity patterns using popular vision architectures. For both types, we examine efficient architectures with relatively few parameters and operations, as well as larger networks. For efficient convolutional architectures, we focus on ConvNeXt-Atto [58] and lightweight versions of MobileNet [59], while for Transformer-based models, we

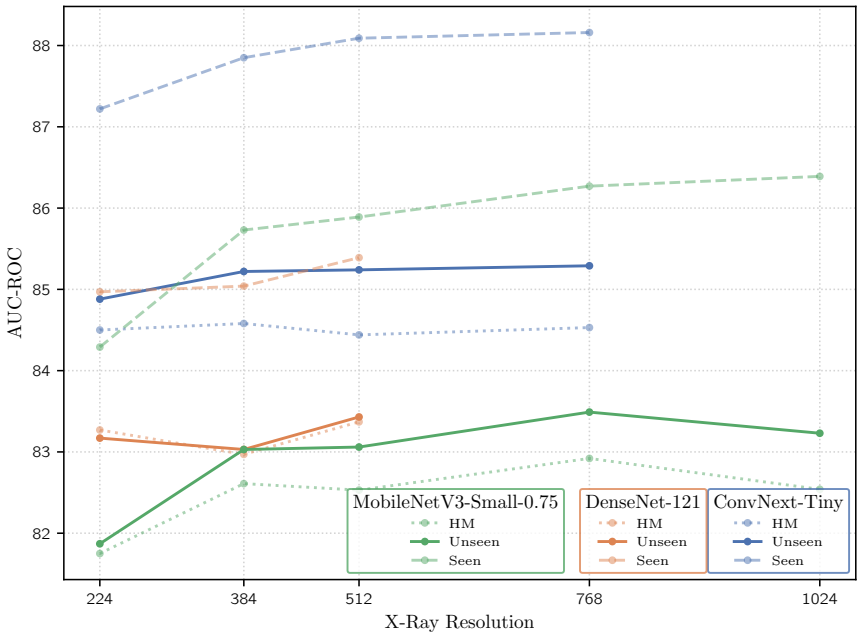

Fig. 7: Comparison of convolutional architectures across progressively increasing chest X-ray resolutions.

use MobileViTV2-1.0 [60]. For larger convolutional architectures, we experiment with DenseNet-121, DenseNet-161 [61], and ConvNeXt-Tiny, and for Transformer-based models, we consider MobileViTV2-2.0 [60]. Table 8 summarizes the results, comparing the performance of these architectures along with their number of parameters and operations.

Among the larger architectures, ConvNeXt-Tiny achieves the highest performance, reaching 85.22 HM points. Notably, it outperforms DenseNet-161 by 1.32 HM points, an architecture previously shown to be effective for medical image analysis problems [12, 53, 62, 63].

Among the efficient architectures, ConvNeXt-Atto achieves the highest performance with 84.71 HM points, followed by MobileNetV3-Large-1.0. Compared to the default architecture in this work (MobileNetV3-Small-0.75), ConvNeXt-Atto offers an improvement of only 1.68 HM points. However, MobileNetV3-Small-0.75 requires just 29.97% of the parameters and 6.83% of the computational operations used by ConvNeXt-Atto. This substantial reduction in resource requirements makes it particularly well-suited for deployment in resource-constrained environments, such as on-device medical image analysis systems.

Interestingly, across both efficient and larger architectures, convolutional models outperform their Transformer-based counterparts. This trend is illustrated in Figure 8,

Architecture	Type	Params (M)↓	MACs (G)↓	*Seen* ↑	*Unseen* ↑	*HM* ↑
Efficient						
MobileNetV3-Small-075	Conv	1.01	0.11	85.73±0.12	82.61±0.32	83.03±0.22
MobileNetV3-Large-1.0	Conv	4.20	0.62	86.75±0.11	84.01±0.30	84.37±0.21
MobileViTV2-1.0	Tsfm	4.38	4.06	86.13±0.11	82.47±0.30	83.21±0.21
ConvNext-Atto	Conv	3.37	1.61	86.88±0.11	84.47±0.30	84.71±0.21
Large						
DenseNet-121	Conv	6.94	8.09	85.04±0.12	82.97±0.29	83.03±0.20
DenseNet-161	Conv	26.46	22.36	86.22±0.11	83.46±0.29	83.90±0.20
ConvNext-Tiny	**Conv**	**27.81**	**18.36**	**87.85±0.10**	**84.58±0.30**	**85.22±0.21**
MobileViTV2-2.0	Tsfm	17.42	16.07	87.15±0.11	84.32±0.30	84.75±0.21

Table 8: Comparison of convolutional and Transformer-based vision architectures. Models are grouped into two categories based on the number of parameters: efficient and large.

which depicts the relationship between model performance and computational efficiency for the evaluated architectures.

Efficient neural network architectures reduce computational requirements without significantly compromising performance, offering several practical benefits. These architectures can run on devices with limited hardware resources, lowering costs and expanding accessibility. Moreover, their efficiency could enable scalable deployment and seamless integration into existing medical infrastructures, including those in remote or resource-constrained regions.

6.7 Hyperparameter analysis

We assess the impact of different hyperparameter configurations on method performance by varying selected hyperparameters and evaluating the resulting classification outcomes. For the BatchBased method, two hyperparameters are tuned during the adaptation phase. The first is the Meta-tst learning rate lr_{head} used to update the head parameters φ. The second is the proportion of Meta-tst ptc_{trn} examples incorporated into the training steps of each episode $E_{meta-tst}$. In the case of ProtoNet-ML, we investigate two types of encoding layer across different output sizes: a fully connected layer and an average pooling layer. The results of these experiments are shown in Table 9.

For BatchBased, the lower learning rate of 0.005 consistently yields the best results, regardless of the chosen ptc_{trn} value. Among the three evaluated configurations of ptc_{trn}, the highest performance is achieved with a value of 0.5. In contrast, ProtoNet-ML obtains its best performance when using an average pooling layer with an output size of 128. Overall, BatchBased shows low sensitivity to hyperparameter variations, maintaining consistent performance across configurations and thus underscoring its robustness.

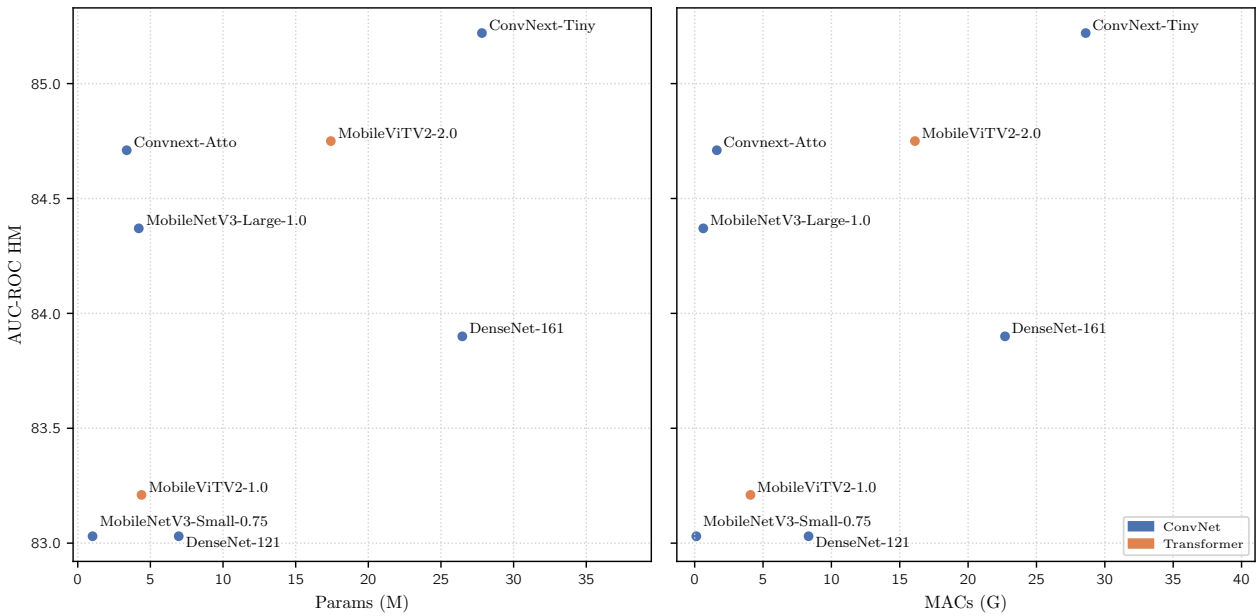

Fig. 8: Trade-off between classification performance and the number of parameters or multiply-accumulate operations (MACs) for different convolutional and Transformer architectures.

6.8 Visualization of model predictions

We visualize a set of model predictions to further examine its behavior qualitatively. Figure 9 presents selected chest X-ray examples along with their corresponding predictions across the four datasets that comprise MetaChest. The examples are arranged from left to right, progressing from correctly classified cases to those exhibiting substantial errors. For example, in the last row, the image in column (a) shows a PadChest X-ray for which the model correctly predicts all four seen classes as well as the unseen class. In contrast, the image in column (d) of the same row illustrates a case in which the model correctly identifies three categories but misclassifies two pathologies, one seen and one unseen, both in red.

7 Conclusions and future work

In this work, we investigated the key factors that influence model training for pathology classification in chest X-rays under few-shot scenarios. To this end, we introduced MetaChest, a benchmark integrating four publicly available chest X-ray datasets. MetaChest provides a meta-set data partition specifically designed for standard few-shot classification, along with a novel multi-label episode generation algorithm. Using MetaChest, we generated diverse classification tasks to compare two representative learning methods: one based on standard transfer learning and another widely adopted

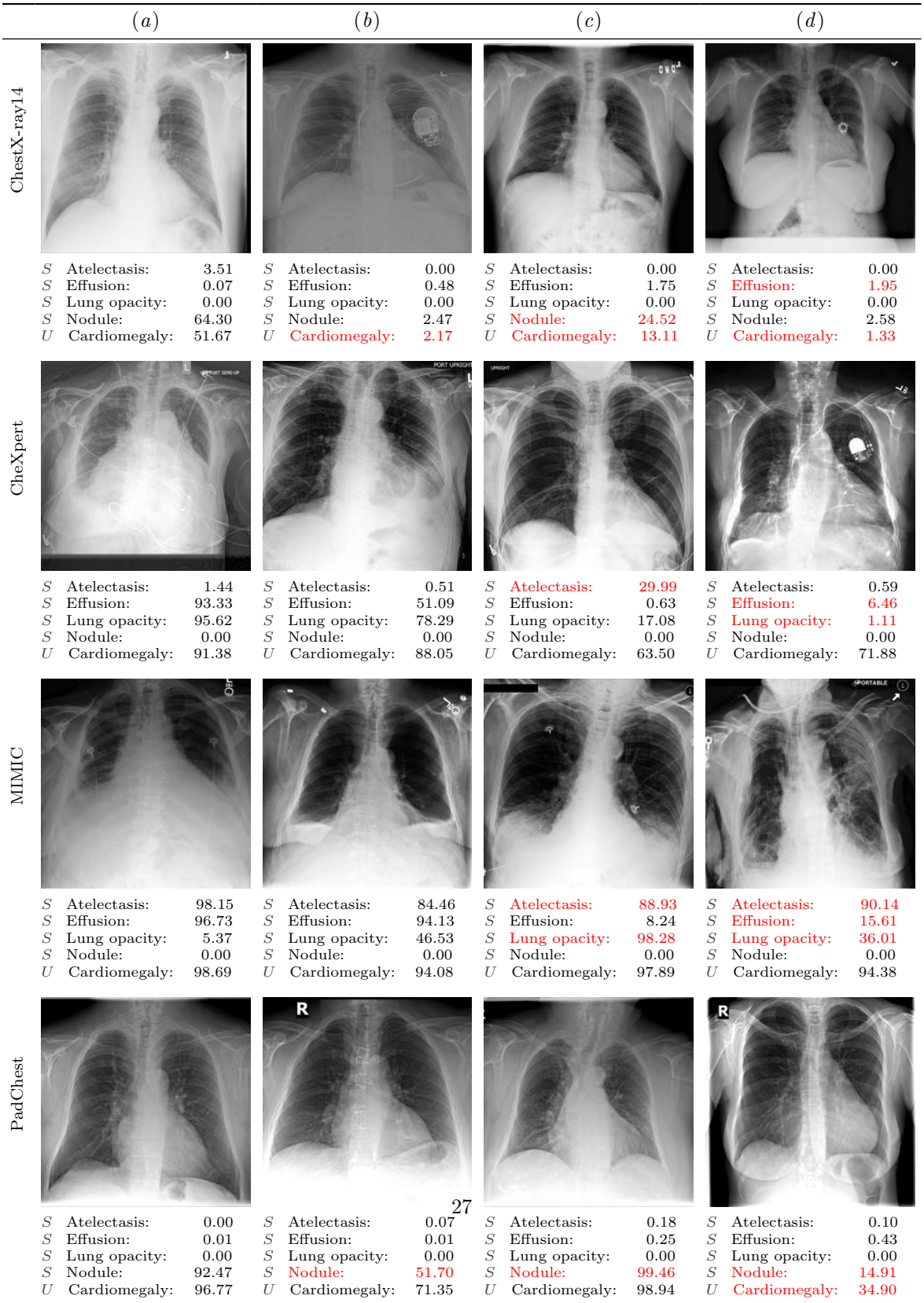

Fig. 9: X-ray examples with predicted labels for the 5-way, 1-unseen classification task with 30-shots per class. Each row shows four examples per dataset in MetaChest. Labels below each chest X-ray are annotated with S and U for seen and unseen classes, respectively. Numbers indicate predicted class probabilities; incorrect predictions are shown in red.

Hyperparameter	*Seen* ↑	*Unseen* ↑	*HM* ↑
BatchBased			
Meta-tst $lr_{head} = 0.01$, $ptc_{trn} =$			
0.25	85.07±0.12	81.50±0.33	82.01±0.24
0.5	85.12±0.12	81.55±0.33	82.07±0.23
0.75	85.11±0.12	81.55±0.33	82.07±0.23
Meta-tst $lr_{head} = 0.005$, $ptc_{trn} =$			
0.25	85.34±0.12	82.15±0.32	82.54±0.23
0.5	**85.33±0.12**	**82.13±0.32**	**82.57±0.23**
0.75	85.32±0.12	82.16±0.32	82.54±0.23
ProtoNet-ML			
Average Pooling			
96	80.16±0.14	79.02±0.36	78.17±0.25
128	**81.88±0.12**	**80.95±0.30**	**80.47±0.20**
144	80.61±0.15	77.90±0.37	77.70±0.27
Fully Connected Layer			
96	80.81±0.14	79.06±0.36	78.51±0.25
128	82.05±0.15	76.20±0.38	77.44±0.27
144	81.12±0.14	77.46±0.37	77.76±0.26

Table 9: Comparison of model performance across BatchBased and ProtoNet-ML hyperparameter configurations.

in standard few-shot classification. We further analyzed how various factors contributing to task complexity impact model performance, including the number of examples per class (k_{trn}), the number of classes per episode (n-way), and the number of unseen classes (n-unseen). Additionally, we explored the effects of image resolution, as well as the connectivity patterns and computational requirements associated with each of the evaluated architectures.

The adoption of the generalized few-shot learning paradigm aligns more closely with the clinical presentation of pathologies in chest X-rays than the standard few-shot classification paradigm. This task formulation is particularly well-suited to specialized medical contexts and useful for a variety of scenarios within the healthcare domain. In addition, the proposed multi-label episode generation algorithm enables the creation of complex classification tasks, further broadening its applicability to real-world medical settings. Interestingly, our results show that BatchBased is an effective classification method in few-shot scenarios, despite being based on standard transfer learning and not specifically designed for few-shot learning. We also observed that increasing the number of classes per episode (n-way) and the number of training examples per class (k_{trn}) improves model performance by enhancing task robustness. With respect to image resolution, we found that using higher resolutions than those commonly applied in natural image tasks leads to better classification performance. This improvement is likely due to the fine-grained nature of pathology classification, where abnormal patterns are subtle and can be overlooked at lower resolutions. However, this performance improvement comes at the cost of higher computational demands and longer training and inference times. In contrast, our results show that efficient architectures

can achieve performance comparable to larger models while substantially reducing computational overhead. This is particularly advantageous in resource-constrained environments, such as remote areas or small hospitals, where these architectures strike a balance between performance, computational efficiency, and practical deployability.

As future work, we envision four main research directions. First, leveraging Vision Foundation Models as a starting point for pathology classification. Pre-trained on large-scale datasets, these models could provide richer and more generalizable feature representations, thereby enhancing classification performance. Second, developing multimodal classification models that integrate complementary information from radiology reports, such as radiologist notes and clinical records. Incorporating this additional contextual information could enrich the diagnostic process and improve overall performance. Third, analyzing the behavior of ProtoNet-ML under different distance and activation functions. Finally, conducting a comparative evaluation between model predictions and expert radiologist assessments. Such a study would enable clinical validation of the results and provide a more accurate and reliable measure of the model's effectiveness in clinical settings.

Acknowledgements

We would like to thank Ricardo Montalvo Lezama for his valuable insights and support in the development of this work. We also extend our heartfelt thanks to Pasita and Chinito for their unwavering support throughout this process.

References

[1] Mustafa, B., Loh, A., Freyberg, J., MacWilliams, P., Wilson, M., McKinney, S.M., Sieniek, M., Winkens, J., Liu, Y., Bui, P., Prabhakara, S., Telang, U., Karthikesalingam, A., Houlsby, N., Natarajan, V.: Supervised transfer learning at scale for medical imaging. arXiv Preprint (2021) arXiv:2101.05913

[2] Ke, A., Ellsworth, W., Banerjee, O., Ng, A.Y., Rajpurkar, P.: Chextransfer: Performance and parameter efficiency of imagenet models for chest x-ray interpretation. arXiv Preprint (2021) arXiv:2101.06871

[3] Cherti, M., Jitsev, J.: Effect of pre-training scale on intra- and inter-domain, full and few-shot transfer learning for natural and x-ray chest images. In: 2022 International Joint Conference on Neural Networks (IJCNN), pp. 1–9 (2022). https://doi.org/10.1109/IJCNN55064.2022.9892393

[4] Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R.L., Shpanskaya, K.S., Seekins, J., Mong, D.A., Halabi, S.S., Sandberg, J.K., Jones, R., Larson, D.B., Langlotz, C.P., Patel, B.N., Lungren, M.P., Ng, A.Y.: Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. arXiv Preprint (2019) arXiv:1901.07031

[5] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., Summers, R.: Chestx-ray8:

Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. arXiv Preprint (2017) arXiv:1705.02315

[6] Johnson, A., Pollard, T., Berkowitz, S., Greenbaum, N., Lungren, M., Deng, C.-y., Mark, R., Horng, S.: Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. Scientific Data, 317 (2019) https://doi.org/10.1038/s41597-019-0322-0

[7] Johnson, A.E.W., Pollard, T.J., Berkowitz, S.J., Greenbaum, N.R., Lungren, M.P., Deng, C., Mark, R.G., Horng, S.: MIMIC-CXR: A large publicly available database of labeled chest radiographs. arXiv Preprint (2019) arXiv:1901.07042

[8] Demner-Fushman, D., Kohli, M.D., Rosenman, M.B., Shooshan, S.E., Rodriguez, L.M., Antani, S.K., Thoma, G.R., McDonald, C.J.: Preparing a collection of radiology examinations for distribution and retrieval. Journal of the American Medical Informatics Association : JAMIA, 304–10 (2016) https://doi.org/10.1093/jamia/ocv080

[9] Bustos, A., Pertusa, A., Salinas, J.-M., de la Iglesia-Vayá, M.: Padchest: A large chest x-ray image dataset with multi-label annotated reports. Medical Image Analysis, 101797 (2020) https://doi.org/10.1016/j.media.2020.101797

[10] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848

[11] Lakhani, P., Sundaram, B.: Deep learning at chest radiography: Automated classification of pulmonary tuberculosis by using convolutional neural networks. Radiology, 162326 (2017) https://doi.org/10.1148/radiol.2017162326

[12] Mabrouk, A., Díaz Redondo, R., Dahou, A., Elsayed Abd Elaziz, M., Kayed, M.: Pneumonia detection on chest x-ray images using ensemble of deep convolutional neural networks. Applied Sciences, 6448 (2022) https://doi.org/10.3390/app12136448

[13] Baltruschat, I., Nickisch, H., Grass, M., Knopp, T., Saalbach, A.: Comparison of deep learning approaches for multi-label chest x-ray classification. Scientific Reports (2019) https://doi.org/10.1038/s41598-019-42294-8

[14] He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '16, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90

[15] Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the

impact of residual connections on learning, 4278–4284 (2017) https://doi.org/10.1609/AAAI.V31I1.11231

[16] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018). https://doi.org/10.1109/CVPR.2018.00745

[17] Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Proceedings of the International Conference on Machine Learning, pp. 6105–6114 (2019). https://doi.org/10.48550/arXiv.1905.11946

[18] Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., Houlsby, N.: Large scale learning of general visual representations for transfer. arXiv Preprint (2019) arXiv:1912.11370

[19] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014). https://doi.org/10.1109/CVPR.2014.81

[20] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016). https://doi.org/10.1109/CVPR.2016.91

[21] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015). https://doi.org/10.1109/CVPR.2015.7298965

[22] Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. arXiv Preprint (2014) arXiv:1403.6382

[23] Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Proceedings of the Advances in Neural Information Processing Systems, vol. 27, pp. 3320–3328 (2014)

[24] Kornblith, S., Shlens, J., Le, Q.V.: Do better ImageNet models transfer better? In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2661–2671 (2019). https://doi.org/10.1109/CVPR.2019.00277

[25] Zhou, H.-Y., Lu, C., Yang, S., Yu, Y.: ConvNets vs. Transformers: Whose visual representations are more transferable? In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pp. 2230–2238 (2021). https://doi.org/10.1109/ICCVW54120.2021.00252

[26] Raghu, M., Zhang, C., Kleinberg, J.M., Bengio, S.: Transfusion: Understanding

transfer learning with applications to medical imaging. arXiv Preprint (2019) arXiv:1902.07208

[27] Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. arXiv Preprint (2021) arXiv:2106.04560

[28] Thrun, S., Pratt, L.: Learning to learn. Kluwer Academic Publishers, USA (1998)

[29] Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML'17, pp. 1126–1135 (2017)

[30] Vinyals, O., Blundell, C., Lillicrap, T.P., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. arXiv Preprint (2016) arXiv:1606.04080

[31] Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17, pp. 4080–4090 (2017). https://doi.org/10.48550/arXiv.1703.05175

[32] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H.S., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. arXiv Preprint (2017) arXiv:1711.06025

[33] Afrasiyabi, A., Larochelle, H., Lalonde, J.-F., Gagné, C.: Matching feature sets for few-shot image classification. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9004–9014 (2022). https://doi.org/10.1109/CVPR52688.2022.00881

[34] Sun, Q., Liu, Y., Chua, T.-S., Schiele, B.: Meta-Transfer Learning for Few-Shot Learning . In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 403–412. IEEE Computer Society, Los Alamitos, CA, USA (2019). https://doi.org/10.1109/CVPR.2019.00049

[35] Mahajan, K., Sharma, M., Vig, L.: Meta-dermdiagnosis: Few-shot skin disease identification using meta-learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 3142–3151 (2020). https://doi.org/10.1109/CVPRW50498.2020.00373

[36] Chen, X., Yao, L., Zhou, T., Dong, J., Zhang, Y.: Momentum contrastive learning for few-shot covid-19 diagnosis from chest ct images. Pattern Recognition, 107826 (2021) https://doi.org/10.1016/j.patcog.2021.107826

[37] Medela, A., Picon, A., Saratxaga, C.L., Belar, O., Cabezón, V., Cicchi, R., Bilbao, R., Glover, B.: Few shot learning in histopathological images:reducing the need of labeled data on biological datasets. In: 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), pp. 1860–1864 (2019). https://doi.org/10.

1109/ISBI.2019.8759182

[38] Shakeri, F., Boudiaf, M., Mohammadi, S., Sheth, I., Havaei, M., Ayed, I.B., Kahou, S.E.: FHIST: A Benchmark for Few-shot Classification of Histological Images (2022)

[39] Tang, H., Liu, X., Sun, S., Yan, X., Xie, X.: Recurrent mask refinement for few-shot medical image segmentation. IEEE/CVF International Conference on Computer Vision (ICCV), 3898–3908 (2021)

[40] Khadka, R., Jha, D., Hicks, S., Thambawita, V., Riegler, M.A., Ali, S., Halvorsen, P.: Meta-learning with implicit gradients in a few-shot setting for medical image segmentation. Computers in Biology and Medicine, 105227 (2022) https://doi.org/10.1016/j.compbiomed.2022.105227

[41] Peng, Y., Wang, X., Lu, L., Bagheri, M., Summers, R.M., Lu, Z.: Negbio: a high-performance tool for negation and uncertainty detection in radiology reports. arXiv Preprint (2017) arXiv:1712.05898

[42] Tsoumakas, G., Katakis, I., Vlahavas, I.: In: Maimon, O., Rokach, L. (eds.) Mining Multi-label Data, pp. 667–685 (2010). https://doi.org/10.1007/978-0-387-09823-4_34

[43] Cohen, J.P., Viviano, J.D., Bertin, P., Morrison, P., Torabian, P., Guarrera, M., Lungren, M.P., Chaudhari, A., Brooks, R., Hashir, M., Bertrand, H.: TorchXRayVision: A library of chest X-ray datasets and models. In: Medical Imaging with Deep Learning (2022). https://github.com/mlmed/torchxrayvision

[44] Jiang, J., Shu, Y., Wang, J., Long, M.: Transferability in deep learning: A survey. arXiv Preprint (2022) arXiv:2201.05867

[45] Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: ICLR (2017)

[46] Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C.F., Huang, J.-B.: A closer look at few-shot classification. In: International Conference on Learning Representations (2019)

[47] Xian, Y., Schiele, B., Akata, Z.: Zero-shot learning - the good, the bad and the ugly. arXiv Preprint (2017) arXiv:1703.04394

[48] Tsuji, T., Hirata, Y., Kusunose, K., Sata, M., Kumagai, S., Shiraishi, K., Kotoku, J.: Classification of chest x-ray images by incorporation of medical domain knowledge into operation branch networks. BMC Medical Imaging (2023) https://doi.org/10.1186/s12880-023-01019-0

[49] Nie, W., Zhang, C., Song, D., Bai, Y., Xie, K., Liu, A.-A.: Chest x-ray image

classification: A causal perspective. In: Greenspan, H., Madabhushi, A., Mousavi, P., Salcudean, S., Duncan, J., Syeda-Mahmood, T., Taylor, R. (eds.) Medical Image Computing and Computer Assisted Intervention – MICCAI 2023, pp. 25–35. Springer, Cham (2023)

[50] Luo, X., Wu, H., Zhang, J., Gao, L., Xu, J., Song, J.: A closer look at few-shot classification again. In: Proceedings of the 40th International Conference on Machine Learning (2023)

[51] Balovsyak, S., Hnatiuk, Y.: Analysis of results of scaling digital images by interpolation algorithms. Security of Infocommunication Systems and Internet of Things, 01007 (2024) https://doi.org/10.31861/sisiot2024.1.01007

[52] Ribli, D., Horváth, A., Unger, Z., Pollner, P., Csabai, I.: Detecting and classifying lesions in mammograms with deep learning. Scientific Reports (2018) https://doi.org/10.1038/s41598-018-22437-z

[53] Rochmawanti, O., Utaminingrum, F.: Chest x-ray image to classify lung diseases in different resolution size using densenet-121 architectures. In: Proceedings of the 6th International Conference on Sustainable Information Engineering and Technology, pp. 327–331 (2021). https://doi.org/10.1145/3479645.3479667

[54] Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models. arXiv Preprint (2020) https://doi.org/10.48550/arXiv.2001.08361 arXiv:2001.08361

[55] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. arXiv Preprint (2020) arXiv:2005.14165

[56] Ridnik, T., Baruch, E.B., Noy, A., Zelnik-Manor, L.: Imagenet-21k pretraining for the masses. arXiv Preprint (2021) arXiv:2104.10972

[57] Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology

[58] Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I.S., Xie, S.: ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders (2023). https://arxiv.org/abs/2301.00808

[59] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for

mobile vision applications. arXiv Preprint (2017) arXiv:1704.04861

[60] Mehta, S., Rastegari, M.: Separable self-attention for mobile vision transformers. Transactions on Machine Learning Research (2023)

[61] Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2261–2269 (2016)

[62] Yao, L., Poblenz, E., Dagunts, D., Covington, B., Bernard, D., Lyman, K.: Learning to diagnose from scratch by exploiting dependencies among labels. arXiv Preprint (2017) arXiv:1710.10501

[63] Cohen, J.P., Hashir, M., Brooks, R., Bertrand, H.: On the limits of cross-domain generalization in automated x-ray prediction. In: Proceedings of the Third Conference on Medical Imaging with Deep Learning. Proceedings of Machine Learning Research, vol. 121, pp. 136–155 (2020)