Chest X-Ray Diagnosis Using CheXpert

Angeliki Dimitriou
AILS Laboratory
National Technical University of Athens
Athens, Greece
angelikidim@islab.ntua.gr

Vaia Kontopoulou
BIOMIG Laboratory
National Technical University of Athens
Athens, Greece
vaiakontop@biomed.ntua.gr

Panagiotis Lamprakis
AILS Laboratory
National Technical University of Athens
Athens, Greece
panoslamprakis@mail.ntua.gr

***Abstract*—Large, labeled datasets are a driving force for many medical imaging tasks. In this context, we choose to work with CheXpert [8], a very large and complex dataset, published by the Standford research team that contains 224,316 chest radiographs of 65,240 patients. Our research goal is to detect the existence of five major observations in patients' radiographs: Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion. This form of diagnosis is formulated as a multilabel classification task that can be performed using a variety of different algorithms. Experiments are carried out using a downsampled version of the original dataset paired with further sampling and preprocessing techniques in order to alleviate problems stemming from the size and intricacy of the input. We benchmark classic machine learning algorithms against state-of-the-art models and evaluate them on a test dataset of 234 chest X-rays, using the ROC-AUC score as a metric. We find out that an ensemble using the DenseNet algorithm performs better on the dataset among those that we experimented with, confirming already existing literature. Code for the reported experiments is available at https://github.com/PanosLam/ML-CheXpert-NTUA.**

I. Introduction

A chest X-ray involves exposing the chest to a minimal amount of ionizing radiation, which enables the creation of internal images of the lungs, heart, and chest wall. This imaging technique is helpful in diagnosing various medical conditions, including shortness of breath, persistent cough, fever, chest pain, and injuries. Moreover, it assists in the diagnosis and monitoring of treatment for numerous lung-related ailments like pneumonia, emphysema, and cancer. The speed and ease of chest X-rays make them a valuable tool in emergency diagnosis and treatment.

The use of chest radiography as a primary imaging examination is widespread worldwide, playing a crucial role in screening, diagnosing, and managing various life-threatening illnesses. Implementing automated chest radiograph interpretation by radiologists is highly advantageous in numerous medical contexts, including enhancing workflow prioritization and clinical decision-making support, facilitating large-scale screening initiatives, and supporting global population health programs. The tremendous significance of this task and our keen interest in biomedical engineering problems are key factors for the choice of this topic.

In this work, we use the CheXpert dataset [8], which was specifically created as a standard for automated Chest X-ray interpretation. To this end, it contains a large number of high-quality radiograph images sourced from thousands of patients and labeled by experts in the field. We choose a subset of the labels provided, based on their clinical significance and prevalence in the dataset, to evaluate an array of algorithms on this task.

The original CheXpert dataset is too large for our resources to process and handle. Given this limitation, we leverage a smaller version found in Kaggle [1], on which we apply further scaling and sampling techniques. A central problem for our data is that several labels are uncertain. To this end, we utilize findings from the original CheXpert paper and substitute uncertainty labels accordingly. Details on this can be found in Section III.

We tackle radiograph interpretation as a multilabel classification problem. The input to all algorithms is a Chest X-ray image, which can be represented as a flattened one-dimensional vector or a 2-D array of pixels according to the model used. We use both classic algorithms, such as Gaussian Naive Bayes and K-Nearest Neighbors, as well as deep models like Multi-Layer Perceptron or Convolutional Neural Networks. Specifically, the convolutional models we benchmarked are ResNet152 and DenseNet121. The output we seek to obtain is the existence or not of the five selected observation labels: Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion. It is possible for several observations to co-exist or for no finding to be present.

In this work, we apply all aforementioned algorithms to the CheXpert dataset, evaluate them using the ROC AUC metric and compare them to each other. We provide quantitative results as well as produced heatmaps for the best model, in an attempt to visually explain its decisions. In the end, we are able to confirm that CNN-based approaches dominate and specifically the DenseNet model achieves the best performance with an average 0.87 AUC score on the 5 labels, competing with models reported in the literature. Our findings give us significant insights pertaining to the task at hand as well as the CheXpert dataset.

II. Related work

Early attempts on CheXpert include the work by Rajpurkar et al. [17] who proposed the CheXNet model, a DenseNet121 trained on frontal radiographs of the CheXpert dataset. In their publication, they first focus on the prediction of the pneumonia diagnosis by using batch training and the Adam optimizer. Then, they generalize their model to predict all

of the 14 observations of the dataset by changing the output layer and modifying their loss function. It's noteworthy that the publication employs normalization of input images using the mean and standard deviation of ImageNet instances, which is a sensible approach given that their model is pre-trained on ImageNet. Moreover, Hasan et al. [3] proved that on top of the 5 target pathology labels of CheXpert, classification can be performed on the radiographs for the tuberculosis disease without accuracy degradation for the overall model. Additionally, they confirmed the value of using transfer learning on CNN models for this problem. In our work, we also utilize pretrained systems successfully and perform standard normalization techniques.

Bressem et al. [10] compare and evaluate different deep learning architectures for classification on two radiograph datasets, CheXpert and COVID-19. Their work is similar to ours in terms of the focus on model exploration. However, we do not specifically compare different families of Deep Learning CNNs. Instead, we incorporate classic ML algorithms with a limited choice of CNNs. This work aided us in choosing the CNN architectures to test, ResNet152 and DenseNet121 since they were among the best-performing ones on CheXpert. Their work is quite exhaustive; on top of training an abundance of models, they performed data augmentation, used several batch sizes, and employed a novel training technique. An interesting takeaway from their work is that shallow models can have comparable or better performance than deep ones and that the handling of uncertainty labels is of great importance. The work by Ke et al. [9] agrees with the findings by Bressem et al. [10]. In fact, they go one step further and prove that the performance on CheXpert is more correlated to the family of the selected CNN algorithm rather than the size of the model. In this light, models can be more parameter efficient by a rate of 3.25x without leading to a statistically significant decrease in their performance. Once again, they showed that ImageNet pretraining boosts model performance, which we decided to employ. Ke et al. also suggested truncating modern architectures by removing layers starting from the end, which we do not do in our research.

The two state-of-the-art models achieve a 0.93 ROC AUC on all 14 labels, by focusing on two main aspects: uncertainty label handling and maximization of AUC through deep models. Pham et al. [16] apply a label smoothing technique to the original uncertainty handling methods. They perform conditional training and utilize ensembles, exploiting predictions from several CNN models. Inspired by the results of Pham et al. and many others, we also decide to leverage ensembling techniques. However, in our assignment, we assume loose label correlation, as seen in Figure 1. Even if our assumption is valid, Pham et al. utilize hidden correlations and manage a remarkable increase in performance. Finally, Yuan et al. [19] propose a novel surrogate loss function for training deep neural networks that maximizes the AUC ROC for binary classifiers. The proposed method is designed to be robust to label noise and class imbalance, making it suitable for large-scale medical image classification tasks exactly like CheXpert.

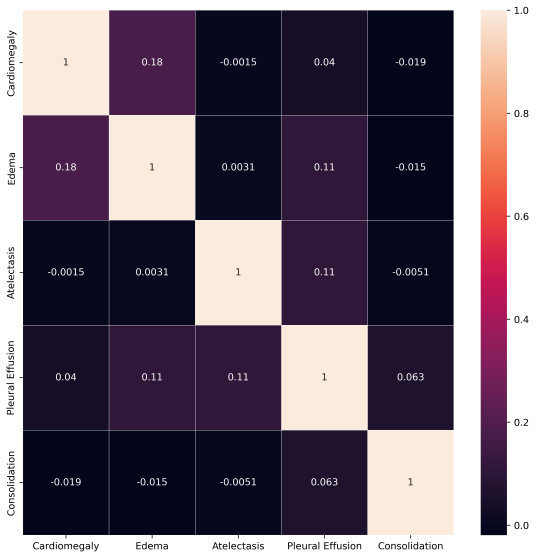

Fig. 1. Label correlation

This approach is too sophisticated to use in this context.

III. Dataset & Features

The CheXpert dataset consists of 224,316 images of chest X-rays. These samples were sourced from a total of 65,240 patients in the span of 200 separate studies. The original image resolution is variable, e.g some images are of size 390×320, while others are of size 320×369, etc. Information about the patient id, age, and gender as well as the study id and type of radiograph (frontal/lateral) can be found in the 'csv' file that accompanies the raw image data. In the same file, we can find the labels which correspond to the potential existence of 14 different observations, which link the radiographs to heart and lung diseases. A sample of the dataset images can be seen in Figure 2, along with the respective present observations. Out of the 14 labels of the dataset, 11 of them are pathologies, 1 is a clinical diagnosis and 2 are auxiliary observations. Table I lists all dataset labels and their respective type.

Fig. 2. Sample of radiograph images. Here, an empty string corresponds to the 'No Finding' observation.

The labeling process of the data is a tricky aspect of this dataset. To be more precise, in the original study [8], researchers had to find an automated way to extract labels pertaining to the 14 selected observations from free-form text provided by 3 radiologist experts. In fact, they tried several

labelers and evaluated them against the final results. The dataset that we use follows their labeling approach named *Mention Aggregation*. This method dictates that when an observation is classified as positive at least once in the report, it gets assigned the label (1), whereas if it is classified as negative it is labeled as (0). An observation is assigned an uncertain (u or -1) label if it has no positively classified mentions and at least one uncertain mention. Lastly, missing labels (blank) represent the lack of a reference in the diagnosis to the pathologies.

The handling of uncertainty labels is a challenging topic, with which many related works have experimented. In the scope of our research work, we selected to go with a simpler approach, based on the findings of the original paper. All missing label values are replaced with 0. For pathologies Atelectasis and Edema, we used the 'U-Ones' technique, replacing uncertain values with (1). For the other three observations, we employ the 'U-Zeros' method and set their uncertain values to (0).

TABLE I
LABEL SEGREGATION BASED ON TYPE

Category	Type
Enlarged Cardiomediastinum	Pathology
Cardiomegaly	Pathology
Lung Opacity	Pathology
Lung Lesion	Pathology
Edema	Pathology
Consolidation	Pathology
Atelectasis	Pathology
Pneumothorax	Pathology
Pleural Effusion	Pathology
Pleural Other	Pathology
Fracture	Pathoogy
Pneumonia	Clinical Diagnosis
No Finding	Other
Support Devices	Other

The original dataset is too large to process with the resources that we have in place, thus we use a smaller version of it from Kaggle [1]. This version is smaller in the sense that all images are downsampled compared to their original sizes (to variable dimensions close to $400 \times 400 \times 3$). However, even so, further scaling techniques must be employed. Especially when working with algorithms that require tabular data as input (kNN, GNB, MLP), flattened 2-D arrays need to have manageable sizes to prevent filling the system's RAM. In these cases, all images are rescaled to $90 \times 90 \times 1$ dimensions and transformed into a 1-D vector. We choose not to involve additional features, such as patient information, in the training process because they should have little to no interference for inference. For CNN-based approaches, images are scaled according to the input required by each pretrained model. Both DenseNet and ResNet models take grayscale images of size around 320×320 and rescale them to 224×224 to leverage them as input.

In addition to scaling, sampling is also a necessary measure. The percentage of samples we keep is variable. Specifically, for classic algorithms, we have to remove a sizeable amount of images due to our lack of resources. Therefore, a mere 15% of the dataset is kept. For CNN-based models, we kept 100% of the small dataset. It is apparent that we had no shortage of data for our experiments. Normally, for medical imaging tasks, we would have needed to employ data augmentation techniques, such as image rotation, flipping, blurring, etc. However, CheXpert has managed to solve this problem and allowed us to use as much data as possible to solve the task at hand.

The CheXpert dataset is already split into a training and validation set, while the test set is not available to the public. Given the fact that we have already sampled out part of the data, we have to redefine these sets. To this end, we decide to use the validation set as our test set. This gives us a chance to have a form of reference with existing literature since we cannot directly use the original test data. Our new test set has 234 samples, while the number of training and validation samples varies according to the algorithm. For classic algorithms, we take 10% of the training samples and set them as validation data to assess overfitting. Therefore, we used about 30k training samples and 3k validation samples. For the CNN approaches we used all 224k images for training since the model is unlikely to overfit for reasons we explain in Section V.

More sophisticated preprocessing methods were only used in the case of classic algorithms. Specifically, as part of the optimization process, we tried Standard and Min-Max normalization, dimensionality reduction with PCA, and feature selection with Variance Threshold. Our findings for the most suitable preprocessing techniques will be described in detail in the experimental section of this report.

Last but not least, we would like to emphasize the fact that we trained our model to predict the existence of only 5 of the 14 observations. These are Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion, which were proven to be the most significant by the original paper. It is crucial to report that after exploration, we were able to find that dataset labels are very loosely correlated, as discernible from Figure 1. Therefore, we have the opportunity to treat this task as a multilabel classification problem, solved by independent One-Vs-Rest Classifiers.

IV. METHODS

In this section, we will list and describe the methods we used in our experiments to tackle the problem of multilabel classification. We attempt to use both classic algorithms that require tabular data as input, as well as convolutional algorithms catering to input represented as an image. Classic algorithms are used in the One-Vs-Rest setting, while convolutional ones are trained using categorical Cross-Entropy loss on the output logits for the five classes, which are the output of a sigmoid activation function. For CNNs, we employed ensembles, a technique in which multiple models are combined to achieve better predictive performance than

any individual model. We used this method with a number of better-performing variations of the same model.

a) ***Gaussian Naive Bayes (GNB):*** [13] a probabilistic machine learning algorithm that is often used for classification tasks. The GNB classifier assumes that the probability distribution of the features given the class labels is Gaussian (i.e., follows a normal distribution). It also assumes that the features are independent of each other given the class label, which is why it is called "naive." To classify a new data point, the GNB algorithm calculates the posterior probability of each class given the observed features using Bayes' theorem. It does this by first estimating the mean and variance of each feature for each class from the training data. It then uses these estimates to calculate the likelihood of the observed features given each class. Finally, it combines the likelihoods with the prior probabilities of the classes to obtain the posterior probabilities. The class with the highest posterior probability is then chosen as the predicted class for the input data point. In the case of multilabel classification, this process is repeated for each of the target classes. The equations for GNB are:

$$p(C_k|x_1, ..., x_n) \propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{1}$$

$$p(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(x_i-\mu_k)^2}{2\sigma_k^2}} \tag{2}$$

where C_k is is the k-th class, x_i are the features, n is the number of features, μ_k is the mean and σ_k is the standard deviation of each class distribution.

The GNB classifier is a simple yet effective method that can work well when the probabilistic assumptions hold. In our case, where features are image pixels the naive assumption can be safely made. We choose the Gaussian variant of NB classifiers due to the nature of CheXpert image features, which are continuous in the range $[0, 255]$.

b) ***k Nearest Neighbors (KNN):*** [2] a type of supervised machine learning algorithm used for classification tasks. It is based on the idea that data points that are close to each other in feature space should belong to the same class. The KNN algorithm works by finding the k-nearest neighbors of a given data point in the training set and assigning the most common class label among those neighbors as the predicted label for the input data point. To determine the k-nearest neighbors, the algorithm calculates the distance between the input data point and all the points in the training set using a distance metric such as Euclidean distance like in Equation 3. The neighbors are then selected as the k points in the training set with the smallest distances to the input point. The value of k is a hyperparameter that must be specified by the user, and it controls the size of the neighborhood used for classification. Other distance metrics can also be used like Minkowski or Manhattan distance, especially when the dataset is characterized by many features.

$$\|\mathbf{x^a} - \mathbf{x^b}\|_2 = \sqrt{\sum_{k=1}^{n} (x_k^a - x_k^b)^2} \tag{3}$$

where x^a, x^b are two data points and x_k^a, x_k^b their respective features.

KNN is a simple yet powerful method that can work well for a wide range of classification tasks. However, it can be computationally expensive for large datasets and may not perform well when the feature space is high-dimensional. A mitigation to the performance issue is to use efficient data structures for the distances, e.g KD-Trees. Given the nature of our dataset, we do not expect high performance and are benchmarking this algorithm to determine whether data samples follow the KNN assumption mentioned above.

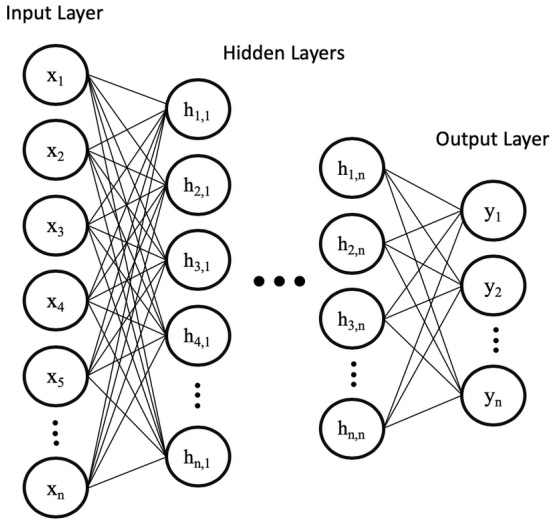

Fig. 3. MLP/FeedForward Neural Network architecture

c) ***Multi-Layer Perceptron (MLP) :*** [4] a feedforward neural network (FFNN) with one or more hidden layers between the input and output layers, as seen in Figure 3. The MLP classifier uses a set of weights and biases to transform the input features into a set of hidden features, which are then transformed into the output class probabilities. Each neuron in the MLP has a set of parameters, which are learned from the training data using backpropagation and adjusted during training to minimize the difference between the predicted class probabilities and the true class labels. The output equation for each neuron (perceptron) is given below:

$$f(x) = \alpha(w_0 + \sum_{i=1}^{N} w_i x_i) \tag{4}$$

where w_0 the bias, w_i the weights, N the number of features and α the activation function. The activation function transforms the output in a meaningful way and can be one of the following examples: sigmoid, tanh, ReLU. The ReLU function, with equation $\alpha(x) = max(0, x)$, is the most commonly used since it tackles the problem of vanishing gradients [6].

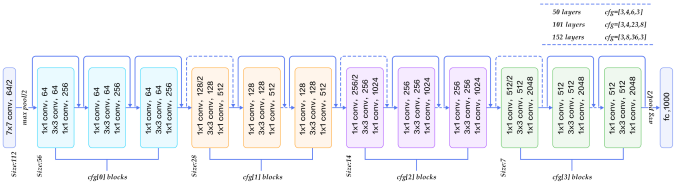

Fig. 4. ResNet architecture

The sigmoid function with equation $\sigma(x) = \frac{1}{1+e^{-x}}$, despite its known limitations, is still used in many applications, such as this one.

An MLP has a large number of hyperparameters including the number of hidden layers and neurons in each layer that can be tuned to improve its performance. Increasing these hyperparameters leads to a more complex model that can capture more complex patterns. The MLP classifier is a powerful and flexible method that can work well for a wide range of classification tasks with non-linear decision boundaries. However, it can be computationally expensive to train and may require a large amount of training data to avoid overfitting.

d) ***ResNet152*** *:* [5] The ResNet CNN-based architecture aims to battle the accuracy saturation and degradation which emerges as a result of the increasingly difficult training of the neural networks and the problems they are assigned to solve. The main idea behind their development is a shift in learning perspective: instead of the features of the input, the ResNet layers aim to discover its residual, meaning the subtraction of the feature learned from the input. This is rendered feasible by adding shortcut connections between layers, which simplifies the training process and resolves the problem of degradation in accuracy.

The equation describing a ResNet building block is the following:

$$y = F(x, W_i) + W_s x \tag{5}$$

where x and y are the network layers' input and output feature vectors and the function $F(x, W_i)$ represents the residual mapping to be learned. In the case of non-matching dimensions, a linear projection W_s is used.

The development of the ResNet was based on a simple implementation of VGG nets, with the addition of shortcut connections (Figure: 4). With respect to the relation between layers' input and output dimensions, the network performs an additional adjustment on the size of the propagated feature maps (dotted lines in Figure 4). There are three types of shortcut connections covering the increase in feature map dimensions through a network layer:

- Shortcut connections perform identity mapping and zero padding is used for increasing dimensions, a strategy that requires no extra parameters.
- Use of the projection shortcut (5) for an increase in dimensions with the other shortcuts remaining equal to the identity, a strategy requiring extra parameters.
- All shortcuts are used as projections, with the need for more additional parameters.

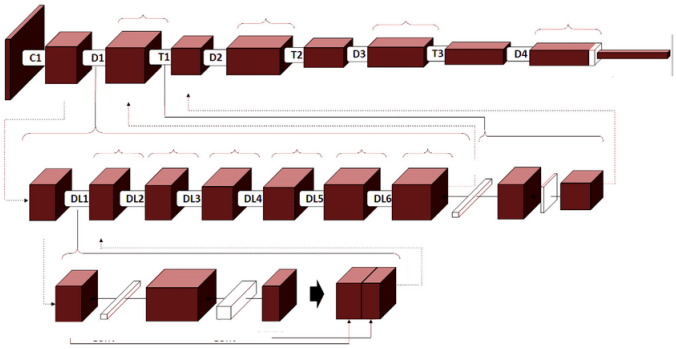

Fig. 5. DenseNet Architecture

The network depth is the determining factor for the naming of the ResNet152, which is comprised of 152 layers/ResNet blocks.

e) ***DenseNet121****:* [7] The DenseNet architecture addresses the vanishing gradient problem of the preceding ResNet architecture by incorporating direct connections between each of the CNN's layers. As a result, a DenseNet neural network of N layers realizes N(N+1)/2 links. Regarding the network connectivity, each layer receives as input x_l, a concatenation of the feature maps produced by previous layers $H_l([x_0, x_1, \ldots, x_{l-1}])$, a characteristic than decreases the number of trainable parameters and thus simplifies the training process.

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}]) \tag{6}$$

The function $H_l(.)$ mentioned above, is a composition of three consecutive operations: batch normalization, linear rectification and 3x3 convolution. In order to render (6) feasible, it is essential that the feature maps' sizes remain of constant shape through the network, which is not the case. As a result, the DenseNet architecture makes use of DenseBlocks, inside of which the dimensions of the feature maps remain constant by changing the number of filters applied between them. Transition layers are the network layers applied between the DenseBlocks, which also downsample the inputs to half their size. Another important feature of the DenseNet architecture is the parameter k, also known as "growth rate". The growth rate is essentially the amount of information added to the global state of the network (aka the feature maps) by each of its layers. Also added to the DenseNet basic architecture are the bottleneck layers, which are applied in the form of a 1x1 convolution before the 3x3 convolution stages in order to increase the efficiency and speed of the training phase. A form of compression is also introduced to the system by reducing the number of feature maps between the DenseBlocks, at transition layers.

The DenseNet121 architecture used in our study is an implementation of the DenseNet architecture for the ImageNet database, consisting of the following layers:

- 1 7x7 Convolution layer
- 58 3x3 Convolution layers

- 61 1x1 Convolution layers
- 4 Average Pooling layers
- 1 Fully Connected layer

The requirement of DenseNet architectures for a smaller number of trainable parameters and their tolerance for reusage of the developed feature maps, has resulted in a compact model architecture and an overall improvement of performance, in comparison to the standard CNN and ResNet architectures. Additionally, DenseNet has been shown to be better at handling imbalanced datasets, which is important for medical diagnosis tasks like ours where some diseases are much rarer than others.

V. Experiments

We conducted experiments with the 5 different machine learning algorithms described in Section IV. Classic ML tests were run on Kaggle and Colab resources while both CNN experiments were realized on a desktop with an Intel i7 core and 32GB RAM.

The primary evaluation metric we used was ROC AUC for each target class, as suggested by the original paper of the dataset [8]. This measure is a common metric for diagnostic tests and medical classification applications. The ROC curve is a graphical representation of the performance of a binary classification model, which plots the true positive rate (TPR / sensitivity) against the false positive rate (FPR / 1-specificity) at various classification thresholds. The AUC is the area under this curve, which ranges from 0 to 1, with a higher AUC indicating better performance. In other words, the ROC AUC score measures the ability of a classification model to distinguish between positive and negative classes, and it summarizes its performance. In our setting, each model has 5 ROC curves and 5 scores for each target class. FPR and TPR are defined in Equation 7.

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{TN + FP} \tag{7}$$

where TP is true positives, FP false positives, TN true negatives and FN false negatives.

In the CNN experiments together with the ROC AUC metric, we used the Precision-Recall (PR) curve which summarizes the trade-off between the true positive rate and the positive predictive value for each model, using different probability thresholds.

A. Classic Machine Learning Algorithms

We conducted multilabel classification using implementations of three classic ML algorithms from the scikit-learn library [15]: GNB, KNN and MLP, as a deep FFNN approach. The models were wrapped with a One-Vs-Rest classifier to provide predictions for all 5 target classes. Optimization was achieved with a mix of GridSearch on the respective hyperparameter space and trial-and-error when a complete search was deemed too expensive. Given the large size of our dataset, which as described has been sampled and scaled in order to fit into the available RAM during computation, we conducted cross-validation with only 2 folds combined with the GridSearch. When choosing hyperparameters by tinkering with different values, we utilized the validation set and confirmed the best chosen model on the test set.

TABLE II
Hyperparameter Values for classic ML algorithms

	Hyperparameter	Values
Preprocessing	Variance Threshold	0.005, 0.009, 0.01, 0.02
	PCA Components	40, 50, 100, 1000, 2000
KNN	Neighbors k	50, 100, 200, 300
	Distance Metric	Euclidean, Minkowski
MLP	Solver	ADAM, SGD
	Activation	ReLU, tanh
	Hidden Layer Sizes	(100), (200), (50, 50)
	Batch Size	64, 128, 200

Hyperparameter selection was based on the model at hand. For example, GNB has no hyperparameters so optimization focused on the appropriate preprocessing techniques used in the pipeline. In all cases, we tested normalization techniques, such as Standard and MinMax scaler, feature selection with Variance Threshold, and dimensionality reduction with PCA. The Variance Threshold selector has the threshold hyperparameter, while PCA has the number of components. We primarily used GridSearchCV to find optimal values for them in the case of GNB. For KNN, we selected to optimize the distance metric and number of neighbors (k) using the search method and guessed-and-checked for the preprocessing techniques. In contrast, for the MLP model, we could only tinker with different values and run multiple independent experiments. This process was preferred because the deep model takes about an hour to produce results given the size of the input. The MLP parameters we tuned were the batch size, solver, hidden layer size, and activation. We highlight that we used shallow MLPs due to limited time and resources. Both for KNN as well as MLP the hyperparameters were picked due to their significance for the model's performance, while the search space was kept relatively small because of time restrictions. Values for preprocessing techniques were picked after data exploration. A complete list of hyperparameter values is available in Table II

The obtained best models for each of the algorithms were the following. GNB performs best when paired with feature selection with a variance larger than 0.01, while KNN leads to the best ROC AUC scores when the number of neighbors is 200, the distance metric is Minkowski and the features are scaled to values between 0 and 1. Lastly, the best MLP model has ReLU as the activation function, an ADAM solver, batch size of 64 and hidden layer sizes of (200). It is best paired with feature selection by variance threshold (threshold = 0.01).

B. Convolutional Neural Networks

In approaching the dataset analysis using convolutional neural networks we used the DenseNet121 and ResNet152 CNN architectures, realizing the uncertainty approach described in section III, for each of the 5 selected pathologies. The models were provided by PyTorch [14], a library that allows the effortless use of pretrained models. In both cases, the networks were trained using images of size 320×320 pixels, which were internally rescaled to 224×224 by the models. We experimented with several hyperparameters and in the end, used the following configuration. The Adam optimizer was used [11] with default β-parameters of values $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and the learning rate was constant at $1\text{x}10^{-4}$ for the duration of the training phase. We used a fixed batch size of 16 images. All three parameters were chosen in accordance with the reference paper [8].

We decided to use the pretrained ImageNet weights in the first step of the training phase and then we fine-tuned the entirety of each network's weights. This was a conscious decision based on the fact that we have a large dataset with samples in the form of Chest X-ray images that are not very close to ImageNet images. Although in the original Chexpert paper, the training phase lasted 3 epochs, in our experiments we used a single-epoch training scheme due to the large training time required by the networks. In the case of the ResNet152, the number of training parameters was 58.154.053 and the training lasted approximately 31h while in the case of the DenseNet152, we trained 6.958.981 parameters for a total duration of 13h.

As in the classic machine learning approach, we used the train.csv of the small Chexpert dataset for training and the valid.csv for testing the classification results. We evaluated the approach using the area under the receiver operating characteristic curve (AUC) metric. We focused on the evaluation of the 5 observations mentioned in Section III. Checkpoints of the network state were saved every 4800 iterations and the final model evaluation yielded the results reported in the following section, which are an ensemble of the 10 best checkpoints on the validation set, computing the mean of the output scores over the 10 models.

C. Results & Discussion

1) Quantitative Results: Comparative ROC AUC results for each of the target classes are available in Table III for all algorithms, where the best results are in bold. Firstly, observation of the results of the classic ML algorithms shows that GNB performs best among them. Despite its simplicity and time efficiency, it is able to surpass KNN and MLP for almost all target classes with an average $9.4-11.2\%$ increase. In contrast, the KNN algorithm manages to produce top ROC AUC only for the Pleural Effusion class, but dummy results for the three first classes. Due to its nature, we can assume that images belonging to the classes with 0.5 AUC score do not follow the assumption of KNN; images with small Minkowski distance do not necessarily belong to the same class. The poor performance of the MLP can be attributed to its complexity.

TABLE III
OVERALL ROC AUC RESULTS FOR THE 5 TARGET CLASSES

	Atelectasis	**Cardiomegaly**	**Consolidation**	**Edema**	**Pleural Effusion**
GNB	0.649	0.675	0.722	0.6328	0.716
KNN	0.5	0.5	0.5	0.621	0.724
MLP	0.556	0.61	0.5	0.601	0.653
ResNet	0.787	**0.817**	**0.875**	0.912	0.926
DenseNet	**0.79**	0.813	0.87	**0.927**	**0.938**

Experimentation with this model was difficult due to limited resources and therefore we were not able to perform a full search on the hyperparameter space. GNB has the advantage that it requires a small training set and little time. Thus, with GridSearchCV we were able to optimize it more accurately and achieve satisfactory results because pixel values can be assumed to be independent (naive assumption).

In Figure 6 we provide ROC curves for the classic algorithms. It is apparent that the thresholds deemed important by the respective scikit-learn functions were few. However, we can draw some interesting conclusions. The better performance of the GNB algorithm is obvious through the AUC visualization which is overall larger. As expected, the other two algorithms exhibit performance closer to a random classifier (dotted line), especially for the first three labels. Specifically, KNN's incompetence is completely clear.

The AUC ROC curves for the CNN models (Figures 7 and 8) exhibit a good classifier performance for each of the findings, with the DenseNet model displaying a better overall performance than the ResNet (Table III results). Also in relation to the classic algorithm results, we observe that the CNN algorithms have a significant advantage for every finding, with the distance between best classic/worst CNN algorithm being: 0.138 for Atelectasis and Cardiomegaly, 0.148 for Consolidation, 0.202 for Pleural Effusion, and 0.279 for Edema.

Regarding the Precision-Recall curves in Figures 7,8, in the cases of Edema and Pleural Effusion we get good classification results in both DenseNet and ResNet models while in the case of Atelectasis, the classifier performance diminishes in both models. The model performance in the case of Cardiomegaly is significantly reduced for an increased recall percentage and finally, in the case of Consolidation, both models' behaviour is that of a random classifier.

In attempting an interpretation of the aforementioned results, we can explain the difference in performance indicated by the ROC and PR curves as an imbalance of the findings' classes in the validation dataset, which is already very small in size, and which may have influenced the models' behaviour. In summary, we can say that DenseNet performs better than ResNet on the CheXpert dataset because it is better suited for handling fine-grained analysis of medical images and for dealing with class imbalance.

We would like to highlight that the results reported here are not directly comparable to top results reported in the existing literature. This is due to the fact that many existing papers train

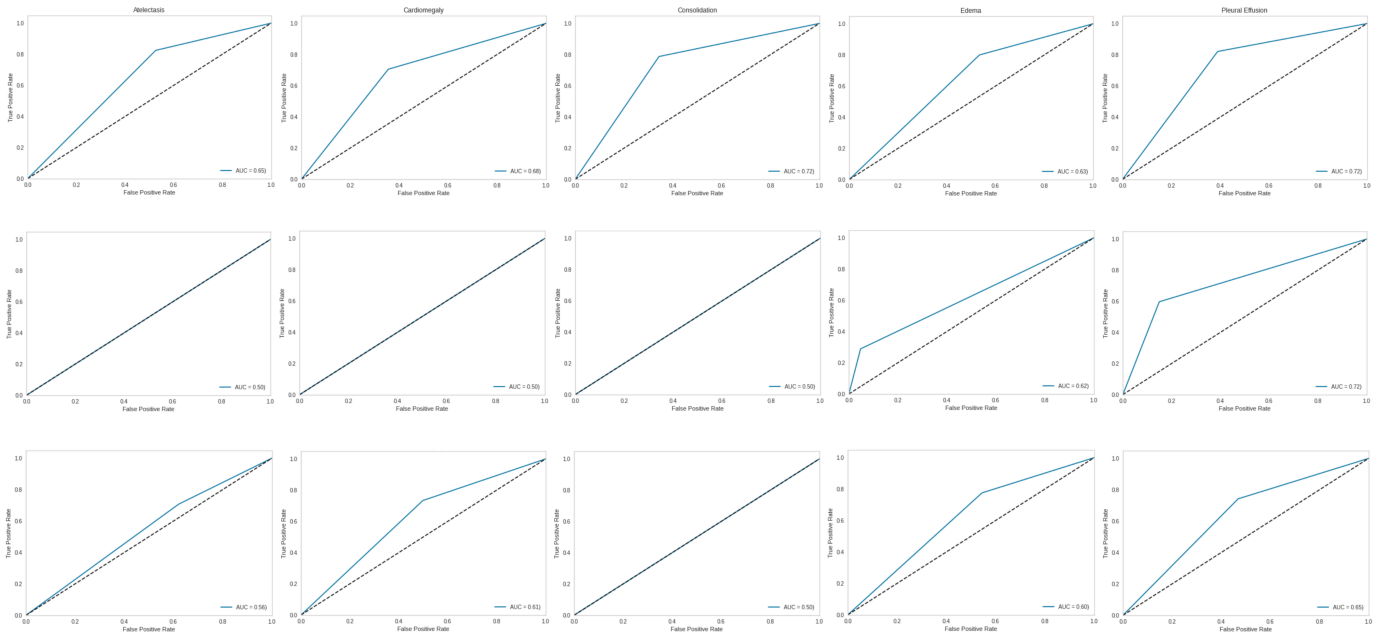

Fig. 6. ROC plots on the original validation set (our test set) using the classic ML algorithms. From top to bottom: GNB, KNN, MLP.

and evaluate using all 14 CheXpert labels. However, there is a discernible agreement with previous work confirming that we produced results similar to state-of-the-art models.

We are confident in the fact that the models we trained did not overfit. To be more precise, in the case of classic algorithms, we used a validation set to keep track of the model's performance and were able to see that training and validation ROC AUC scores did not diverge. For CNN approaches, we used pretrained models and only trained the classification head for one epoch. This fact alone is enough to rule out overfitting on the training samples. Besides, for all the employed algorithms, performance on the training set was reflected in the much smaller test set which reinforces the generalization capabilities of the models.

2) Qualitative Analysis: In Figure 9, qualitative results are provided for three sample images, belonging to one of the three following cases each: images classified with Pleural Effusion only, images with no findings, and images with multiple findings. To avoid confusion, we report that No Findings/ Multiple Findings do not correspond to specific CheXpert labels, but to the prediction of no pathology and of more than one pathology respectively. We have produced heatmaps utilizing the GradCAM algorithm [18]. Specifically, we visualize the DenseNet model's convolutional layer, using the gradients of the final linear layer as weights and performing a weighted sum of the final feature maps with them. In the heatmap, red areas correspond to dense regions and depict the parts of the image which the model focused on to predict the potential existence of the pathology.

The first row of Figure 9, provides us with an understanding of where the CNN algorithm pinpoints the existence of Pleural Effusion. The visual results are sensible to non-experts who do not know whether this pathology is truly detected in the highlighted red areas. However, by looking at the third column, we observe that different pathologies are detected in different areas of the radiograph. For example, in the second pair of images of this column, the top activated class is Edema, which seems to be detected in areas closer to the patient's neck in comparison to Pleural Effusion. Finally, the second column, regarding X-rays with no findings, gives us confusing insights. The DenseNet model highlights contributing regions for the top class in a similar way to when the model actually finds pathologies. This observation leads us to believe that the interpretation of heatmaps should be done only with the knowledge of the predicted label in order to avoid misconceptions.

VI. Conclusion/Future work

To conclude, we performed multilabel classification on the CheXpert dataset and attempted to detect the existence of 5 major observations: Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion. We benchmarked 3 classic ML algorithms - GNB, KNN and MLP - against 2 CNN ones - ResNet152 and DenseNet121. The difference in performance between classic ML algorithms compared to CNNs is noticeable and definitely not a matter to be ignored. Especially in the medical domain where predictions are critical and associated with human life, automated diagnosis should be as accurate as possible. The DenseNet model proved to perform better than ResNet, as expected given the existing literature. DenseNet is more appropriate for fine-grained analysis of medical images and for managing imbalanced datasets, such as CheXpert. Apart from its increased performance, this model leads to

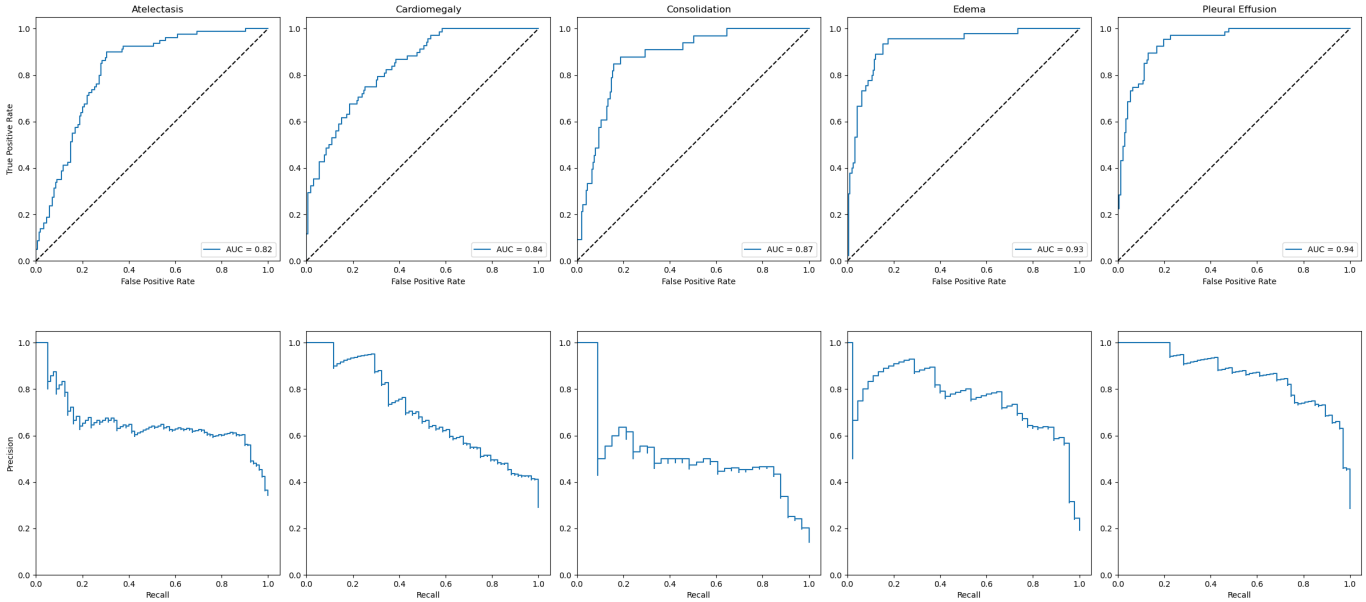

Fig. 7. ROC and Precision-Recall plots on the original validation set (our test set) using the pretrained ResNet152 architecture.

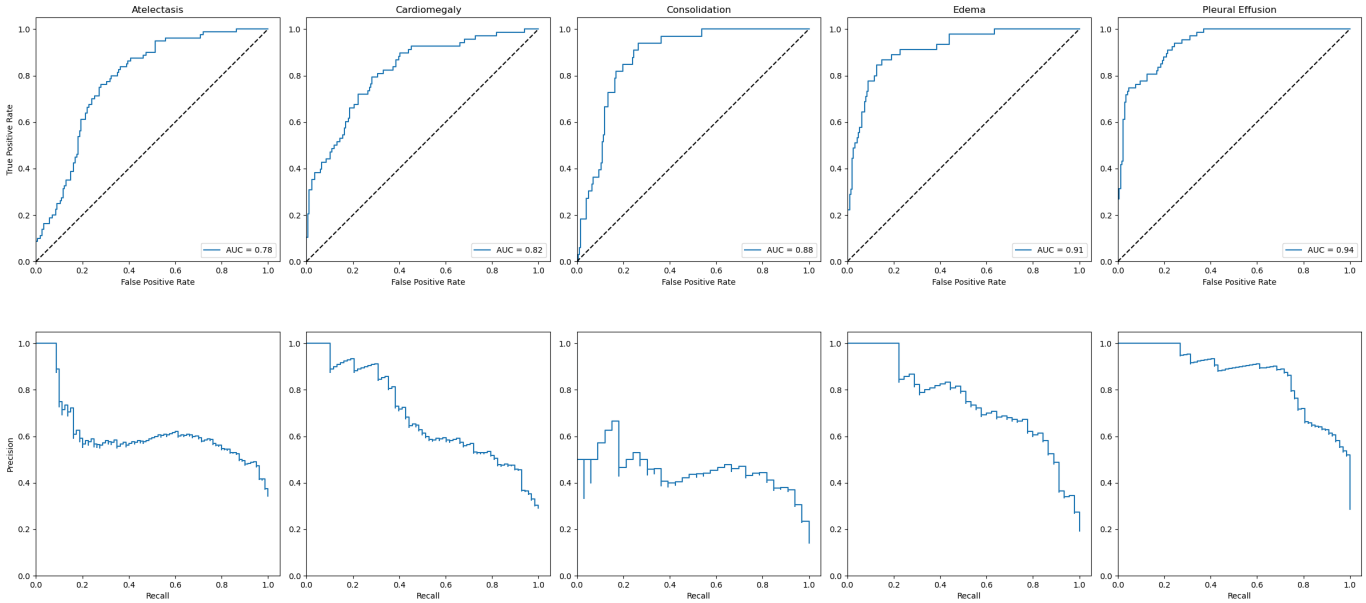

Fig. 8. ROC and Precision-Recall plots on the original validation set (our test set) using the pretrained DenseNet121 architecture.

lower training times by a factor of 2 and simpler complexity with $8\times$ fewer parameters. In our experiments, we confirmed the importance of combining CNNs with ensembling and were able to produce results close to the state-of-the-art (SOTA). Visual interpretation of our results leads us to the interesting finding that explaining model predictions, although challenging, can prove to be crucial for high-risk applications.

The research work performed and reported in this paper was conducted under strict time constraints and limited resources. In the future, we plan to incorporate the Pham et al. [16] techniques of conditional training and label smoothing in order to achieve a performance increase in the models reported. Changing the output activation function from sigmoid to softmax is another experiment we would like to conduct [12]. Moreover, performing Deep AUC Maximization (DAM) [19] seems like an intuitive extension, since ROC AUC is the employed evaluation metric. In addition to incorporating techniques introduced by SOTA models, we deem that introducing attention to our model could be very beneficial. Motivated by the limitations of the simple GradCAM visualizations, we believe that producing attention maps will lead to more interpretable results, which is of great importance for this

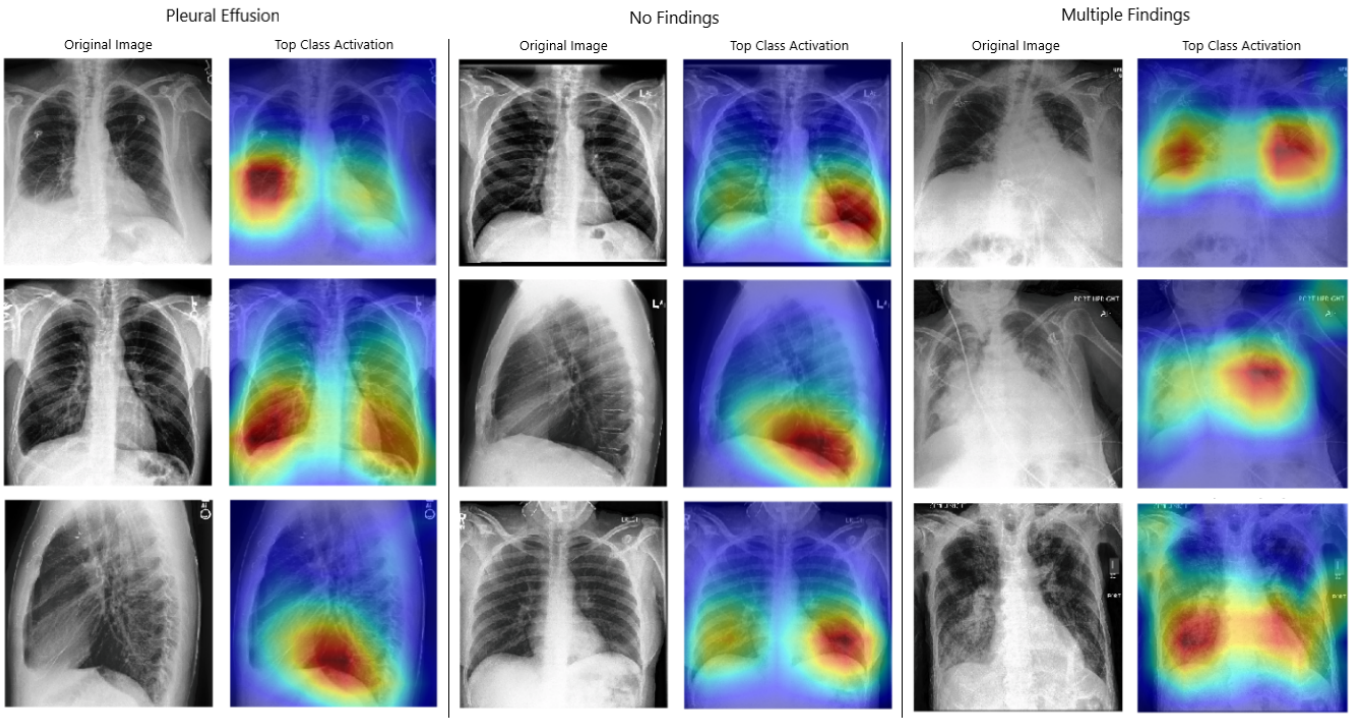

Fig. 9. Heatmaps produced from the DenseNet model. Red areas highlight important regions for the algorithm's predictions.

application.

REFERENCES

[1] "Chexpert-v1.0-small," https://www.kaggle.com/datasets/willarevalo/chexpert-v10-small, accessed: 2023-02-20.

[2] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[3] S. Hasan Nabeel, S. Usman Ullah, and A. K. Saifulnizam, "Classification of chest diseases from x-ray images on the chexpert dataset," 2021.

[4] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016.

[6] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2017.

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

[9] A. Ke, W. Ellsworth, O. Banerjee, A. Y. Ng, and P. Rajpurkar, "Chextransfer: Performance and parameter efficiency of imagenet models for chest x-ray interpretation," 2021.

[10] K. B. Keno, L. C. Adams, C. Erxlebel, B. Hamm, S. M. Niehues, and J. L. Vahldiek, "Comparing different deep learning architectures for classification of chest radiographs," 2020.

[11] D. P. Kingma, J. A. Ba, and J. Adam, "A method for stochastic optimization. arxiv 2014," *arXiv preprint arXiv:1412.6980*, vol. 106, 2020.

[12] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van der Maaten, "Exploring the limits of weakly supervised pretraining," 2018.

[13] A. McCallum, "Graphical models, lecture2: Bayesian network representation," 2019.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] H. H. Pham, T. T. Le, D. Q. Tran, D. T. Ngo, and H. Q. Nguyen, "Interpreting chest x-rays via cnns that exploit hierarchical disease dependencies and uncertainty labels," 2021.

[17] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, R. L. Ball, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," 2017.

[18] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[19] Z. Yuan, Y. Yan, M. Sonka, and T. Yang, "Large-scale robust deep auc maximization: A new surrogate loss and empirical studies on medical image classification," 2021.