Evaluation of Transfer Learning in Pneumonia Classification

By

Azhar Uddin Molla

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

December 12, 2020

Nashville, Tennessee

Approved:

Bennett A. Landman, Ph.D.

Yuankai Huo, Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Pneumonia is a form of acute respiratory infection that inflames the air sacs of one or both lungs and causes significant damage to lung tissue [1]. Every year, approximately 7% of the world population gets infected by pneumonia, out of which, 4 million patients face fatal risks [2]. Despite recent advancements in medical treatments, pneumonia remains to be the leading cause of deaths in children under the age of five worldwide [3]. The key to successfully treating pneumonia lies in early detection. When diagnosed early, it can be treated at a low cost.

Typical well identified symptoms of pneumonia include shortness of breath, chest pain, and cough. To diagnose pneumonia one of the following tests is required: chest X-ray, CT of the lungs, MRI of the chest and needle biopsy of the lung. In children, the diagnosis poses a difficulty due to the low sensitivity of microbiological tests [4]. Therefore, the chest X-ray has become an important diagnostic tool for the diagnosis of pneumonia in children. Additionally, X-ray is the imaging technique that is widely available even in underdeveloped regions in contrast to CT and MRI. Analyzing and diagnosing chest X-rays is not only time consuming but also requires professional radiologist and healthcare workers [5]. Moreover, there is a scarcity of radiologists and healthcare workers in many parts of the world. This necessitates the development of computer-aided diagnosis not only to help prevent pneumonia-related mortality but also to empower and augment clinical decision making.

1.1 Scope of the proposed work

The deep learning models have ensured superior results with high generalization performances compared to traditional methods for a wide variety of applications, including the domain of medical imaging. Deep convolutional neural networks (CNNs) have achieved tremendous success in solving image classification, object recognition, and image segmentation problems [6]–[8] and therefore CNNS are widely used in the research community. It has yielded remarkable results compared to the state-of-the-art models in the field of biomedical image analysis. Grewal et al. proposed a deep learning technique that achieved radiologist level accuracy for brain hemorrhage detection in CT scans [9]. Gulshan et al. used a deep learning method for detecting diabetic retinopathy in retinal fundus photographs [10]. A deep learning model that achieved dermatologist-level classification of skin cancer was proposed by Andre et al. [11]. Y. Bar [12] discussed pathology detection in chest X-rays using deep learning. Other application of CNNs in this area includes detection of mitosis cells from microscopic images [13], [14], tumor detection [15], tuberculosis detection [16] , detection and classification of immune cells [17], segmentation of neural membranes [18] and quantization of mass in mammograms [19].

However, deep learning models require much more data than that of traditional machine learning systems. In other words, deep networks are heavily reliant on big data to avoid overfitting [20]. Unfortunately, many application domains do not have readily available big data because of the data being inaccessible, or the data being rare or expensive to collect and label. To cope up with the limited availability of labeled data for a current task, the knowledge learned in another task of different domain where data is abundantly available, can be utilized [21]. Moreover, in order to achieve a higher generalization ability and also to reduce the computational costs, the concept of transfer learning is exploited [22]. A deep transfer learning based framework for the detection and classification of breast cancer was proposed by Khan et al. [23]. Fauw et al. devised a clinically applicable diagnosis and referral for retinal disease that utilizes transfer learning [24].

Much work has already been done on chest X-ray data specifically on pneumonia classification. Khatri et al. [25] used the earth mover's distance to identify infected pneumonia lungs using chest X-Ray images. Rajpurkar et al. [26] devised a deep CNN called CheXNeXt to detect fourteen different pathologies including pneumonia. Rahib et al. [27] and Okeke et al. [28] used CNN models for pneumonia classification. Cohen et al. [29] and Rajaraman et al. [30] employed customized DenseNet-121 and VGG16 respectively, and reported assuring results. Saraiva et al. [31] and Ayan et al. [32] used deep learning based methods for pneumonia diagnosis. Rahman et al. achieved an accuracy of 98% for pneumonia classification using deep transfer learning on DenseNet201 architecture [33]. Vikash et al. [34] proposed a transfer learning based approach that ensembled multiple pretrained network using majority voting. Hashmi et al. [35] came up with a weighted classifier based approach that combines prediction from five state-of-the-art pretrained CNN models.

In this work, a transfer learning based framework is applied for pneumonia classification on the chest X-ray dataset that was made publicly available by Kermany et al. [36]. The prior works on this dataset discussed in Section 3.1 implies better performance for deep learning models that are pre-trained compared to the others. For this study, an existing convolutional neural network architecture (DenseNet), which is originally designed for classification task in natural image dataset (ImageNet), is adapted along with the pretrained weights of the network, and subsequently fine-tuned for X-ray dataset for pneumonia diagnosis. Thus, the framework uses deep transfer learning which extracts the features from the X-ray image that aids in identifying whether it is a case of pneumonia. Further, experiments were performed to find out the optimal number of dense blocks that should be finetuned for the transfer learning without losing on the performance.

1.2 Organization of the thesis

The thesis in its entirety is organized into six chapters. Chapter 1 provides an introduction to the thesis. Chapter 2 furnishes background information on CNN and transfer learning. Chapter 3 deals with the database description. Chapter 4 is primarily concerned with the methodology. Chapter 5 discusses the experimental findings and paves the road for Chapter 6 which concludes the thesis.

Chapter 2

Background

In this chapter, we first discuss about the convolutional neural networks (CNNs) and DenseNet architecture. The discussion is then further extended to transfer learning. Finally, the performance metrics used in evaluating the models in the experimental studies are briefly explained.

2.1 Convolutional Neural Network

Recent advancement in deep learning has made a huge impact in the biomedical field. Deep CNN has pronounced its name for providing with higher generalization performances than other machine learning paradigms and has yielded state of the art models producing remarkable performance for many difficult problems, especially in the area of image classification and segmentation [6]–[8], [27]. CNNs are designed to learn spatial hierarchies of features automatically through backpropagation algorithm. Multiple basic building blocks such as convolution, pooling and fully connected (FC) layers are used to construct CNN [37]. The task of the convolutional layer is to apply convolution operation on the input image with the learnable filters in a moving window fashion, thereby creating the input volume for the next layer. The size of the filter is usually $3 \times 3, 5 \times 5$ or 7×7. The output $Y_i^{(l)}$ of convolutional layer l consists of m_1^l feature maps of size $m_2^{(l)} \times m_3^{(l)}$. The i^{th} feature map, represented as $Y_i^{(l)}$ is calculated according to Eq. (1), where $B_i^{(l)}$ represents the bias matrix, $K_{i,j}^{(l)}$ represents the filter associated with i^{th} output feature map $Y_i^{(l)}$ and convolves on previous layer's j^{th} feature map $Y_j^{(l-1)}$, $*$ denotes the convolution operation and f is the activation function.

$$Y_i^{(l)} = f\left(B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{i,j}^{(l)} * Y_j^{(l-1)} \right) \tag{1}$$

The convolution layer can overwhelmingly increase the size of the input tensor for the next layer. That is where the pooling layer comes in. It reduces the resolution of the feature maps and keeps important feature and information intact [38]. In addition, it decreases the number of parameters [39]. The pooling layer (l) is characterized by two parameters: the spatial size of the filter, $F^{(l)}$ and the step or stride, $S^{(l)}$. It receives input data in the size of $m_1^{(l-1)} \times m_2^{(l-1)} \times m_3^{(l-1)}$ and produces an output volume of $m_1^{(l)} \times m_2^{(l)} \times m_3^{(l)}$. Briefly, the size reduction of the pooling layer is shown in Eqs. (2), (3) and (4).

$$m_1^{(l)} = m_1^{(l-1)} \tag{2}$$

$$m_2^{(l)} = (m_2^{(l-1)} - F^{(l)})/S^{(l)} + 1 \tag{3}$$

$$m_3^{(l)} = (m_3^{(l-1)} - F^{(l)})/S^{(l)} + 1 \tag{4}$$

A FC layer is applied to a single vector obtained by flattening the features from the previous layer and provides the final predictive values for each label [37]. The i^{th} output $Y_i^{(l)}$of the FC layer l is shown in Eq. (5), where w, y and b represents the weight vector, feature vector, and bias respectively.

$$Y_i^{(l)} = f(Z_i^{(l)}) \text{ with } Z_i^{(l)} = \sum_{j=1}^{m_1^{(l-1)}} w_{i,j}^{(l)} y_j^{(l-1)} + b_i^{(l)} \tag{5}$$

A CNN comprises an input layer, an output layer and multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that is subsequently followed by pooling layers and FC layers.

2.2 DenseNet

In a conventional CNN, a non-linear transformation H_l is applied to produce the output of the l^{th} layer x_l from the previous layer's output x_{l-1},

$$x_l = H_l(x_{l-1}) \tag{6}$$

where H is defined as a convolution followed by a rectified non-linearity (ReLU) and a dropout [40], [41].

In order to alleviate the training of a very deep network and to address the issue of the vanishing gradient, residual block is introduced ResNets [42]. The identity mapping of the input is added to the output of a layer and the output x_l is given by

$$x_l = H_l(x_{l-1}) + x_{l-1} \tag{7}$$

The reuse of features permits the direct gradient to flow to earlier layers. Here, H is realized by 2 or 3 repeated blocks which is composed of Batch Normalization (BN) [43], ReLU and convolution. DenseNet [44] provides a more sophisticated connectivity pattern to further improve information flow between layers. Here, the outputs of all preceding layers are used as the input for the current layer:

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}]), \tag{8}$$

where $[x_0, x_1, \ldots, x_{l-1}]$ represents a single tensor obtained by concatenating the outputs of the previous layers. This enables even the last layer to get input from the first layer and all layers get direct gradient flow from the loss function.

Here, $H_l(\cdot)$ represents the sequence: BN-ReLU-Conv(1×1)-BN-ReLU-Conv(3×3). Each function H_l produces k feature maps which is called the growth rate of the network. The l^{th} layer in such case has $k_0 + k \times (l - 1)$ input feature maps. Here, the number of channels in the input layer is represented by k_0. A very pivotal difference between DenseNet and other existing architectures is that DenseNet can achieve state-of-the-art results even with very narrow layers i.e. less number of feature-maps per layer (e.g. k= 12) [44].

Even though each layer produces only k feature maps, it has considerably more inputs than other network architectures because of the dense connectivity. A 1×1 convolution is introduced as *bottleneck layer* before the 3×3 convolution to reduce the number of input feature-maps, and this improves the computational efficiency as noted in [42], [45].

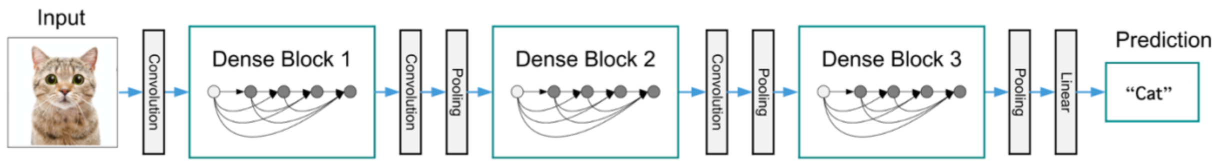
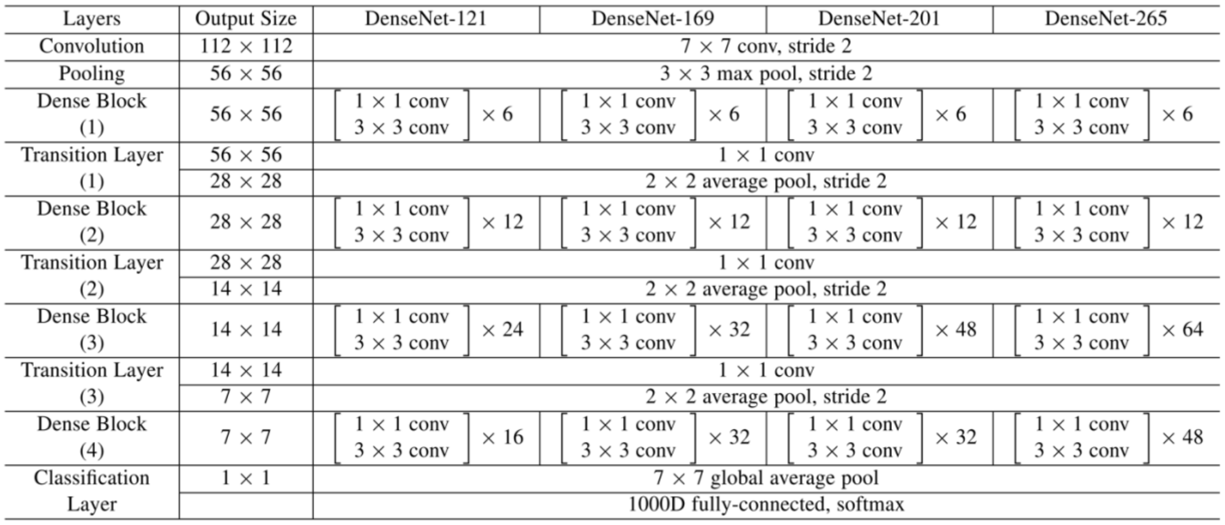


Figure 2.1 A deep DenseNet with three dense blocks (Huang et al [44])

To facilitate pooling the network is divided into multiple densely connected dense blocks as demonstrated in Figure **2.1**. The spatial dimensionality of the feature maps is reduced by introducing *transition layers* between two adjacent blocks. It is composed of a 1×1 convolution that preserves the number of feature maps, followed by a 2×2 pooling operation.
To make the model more compact, *compression* factor $0 < \theta \leq 1$ is introduced. This allows the transition layers to generate $\lfloor \theta m \rfloor$ output feature maps, where m is number of the feature-maps produced by the previous dense block.

The standard DenseNet architecture uses four dense blocks with 224×224 input images. The initial convolution layer consists of 2×k convolutions of size 7×7 and stride 2, where k is the growth rate. The standard network configurations are shown in Figure **2.1**. The growth rate, k, for all the networks is 32. The first "conv" layer with filter size 7×7 corresponds to Conv-BN-ReLU. Rest of the "conv" layer represents the sequence BN-ReLU-Conv. The particular variant used in this current study is DenseNet-121 which is discussed in Chapter 4.

Table 2.1 DenseNet architectures (Huang et al. [46])



Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-265
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

2.3 Transfer Learning

Transfer learning focuses on transferring the knowledge from one domain to other. It is a machine learning methodology that consists of adopting features learned on the source problem and leveraging them on a new but similar target problem [21]. The conception of transfer learning comes from educational psychology. For example, a person who has learned to ride a bicycle would find it easier to learn riding a motorbike than others, because both the bicycle and the motorbike are two wheelers, and the two tasks require some shared common knowledge. Some intuitive examples of transfer learning are shown in Figure **2**.**2**. Transfer learning takes inspiration from the sheer ability of human beings to transfer knowledge from one domain to another and aims on leveraging the knowledge acquired from the source domain to improve the learning in a target domain. It can also help to learn a new task in target domain with limited number of labeled examples. The transferred knowledge, however, might not always bring positive effect to a new task. If two domains do not share very much in common, knowledge transfer from one to other could fail. For example, knowledge of playing a piano would not facilitate the learning process to riding a bicycle.

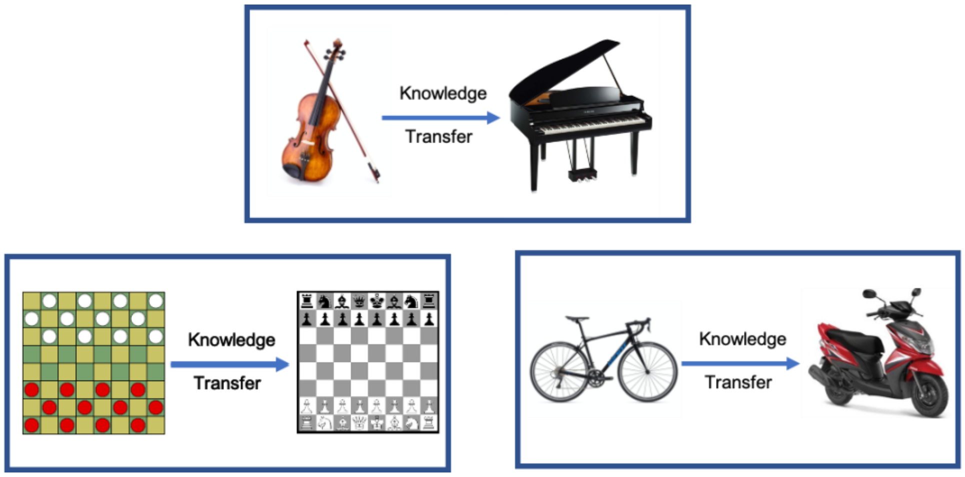


Figure 2.2 Intuitive examples about transfer learning

In machine learning theory, independent and identically distributed (i.i.d). assumption is often made for training datasets which implies that the training data must be i.i.d. with the test data. This hypothesis is relaxed in Transfer learning. This works as a motivation to utilize transfer learning against the problem of insufficient training data [47]. Since the training data and test data are not required to be i.i.d. for transfer learning, and the model in target domain does not need to be trained from scratch. Therefore, the demand of huge training data is significantly reduced and the computation cost for training in the target domain is also decreased.

Pan and Yang [48] used domain, task, and marginal probabilities to propose a framework for better understanding the transfer learning. A domain is represented by $D = \{\chi, P(X)\}$, which consists of two parts, namely the feature space χ and the marginal probability distribution $P(X)$ where $X = \{x_1, \ldots, x_n\} \in \chi$. A task can be represented by $T = \{y, f(x)\}$ where y is label space and $f(x)$ is target prediction function. $f(x)$ can also be considered as a conditional probability function $P(y|x)$. Then, the transfer learning can be formally defined as follows:
Given a source domain D_s with a corresponding source task T_s and a target domain D_t with a corresponding task T_t, transfer learning is the process of improving the target predictive function $f_t(\cdot)$ by using the related information from D_s and T_s, where $D_s \neq D_t$ or $T_s \neq T_t$. Homogeneous transfer learning refers to the case where $\chi_s = \chi_t$, conversely $\chi_s \neq \chi_t$ is the case of heterogeneous transfer learning.

In transfer learning, a model that was previously trained for a specific task is taken as the starting point to solve another task. In other words, instead of going through the lengthy process of training a new model with randomly initialized weights, which requires substantial amount a time and computation resources, pre-trained models are used as the starting point for the new task. In this work, deep learning model, pre-trained on ImageNet [49] dataset, was used for pneumonia classification. In order to achieve further improvements, the pre-trained model was fine-tuned. The details related to fine-tuning are discussed in Chapter 4.

2.4 Performance Metrics

The model in each experiment was tested on the test dataset after the completion of the training phase. The performance was validated using the accuracy, recall, precision, F1 score, and area under the curve (AUC) score. All the performance metrics used in this work are discussed below.

In the below-mentioned definitions and equations, while classifying healthy and pneumonia patients, true positive (TP) denotes the number of pneumonia images identified as pneumonia, true negative (TN) denotes the number of normal images identified as normal (healthy), false positive (FP) denotes the number of normal images incorrectly identified as pneumonia images, and false negative (FN) denotes the number of pneumonia images incorrectly identified as normal.

In the context of statistical analysis, Recall is referred to as the True Positive Rate or Sensitivity. It calculates the proportion of actual positives which are correctly classified (e.g. the percentage of pneumonia that are correctly identified as pneumonia). Recall is the ratio of true positives to the summation of true positives and false negatives.

$$\boldsymbol{Recall = \frac{TP}{TP + FN}} \tag{9}$$

Precision also referred to as the positive predictive value is the proportion of positive test results that are true positives. A small precision value for example of the order of 15% or below, indicates that many of the positive results from this testing procedure are false positives. Thereby indicating the fact that, it will be necessary to follow up any positive result with a more reliable test to obtain a more accurate assessment as to whether the null hypothesis holds or not. Precision is the ratio of the true positives to the summation of true positives and false positives.

$$\boldsymbol{Precision = \frac{TP}{TP + FP}} \tag{10}$$

Accuracy can be defined as a systems degree of veracity and is defined as

$$\boldsymbol{Accuracy = \frac{TP + TN}{TP + FP + TN + FN}} \tag{11}$$

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F1-measure or balanced F1-score. F1-score is a factor in which the Recall and Precision scores are evenly weighted. F1-score is an important measure while considering two class problems, and often outscores the measure of Accuracy. F1-score is calculated as follows-

$$\boldsymbol{F1 = 2.\left(\frac{Recall * Precesion}{Recall + Precision}\right)} \tag{12}$$

ROC curve and AUC score: The receiver operating characteristics (ROC) curve is a probability curve. It is the plot of sensitivity (true positive rate) against false positive rate as the binary classifier’s discrimination threshold is varied. The area under the ROC curve (AUC) represents the degree of separability.

Chapter 3

Dataset Description

The dataset [36] used in this study comprised a total of 5,856 chest X-ray images divided into two sub-sets: a training set and a test set, each having two categories: Normal and Pneumonia (see Table **3.1**). Each image in the dataset was two dimensional and consisted three channels. The bacterial and viral pneumonia were combined into a single category: pneumonia. There was no case of viral and bacterial co-infection. All chest X-ray images were taken during the routine clinical care of the patients. Two expert physicians then graded the diagnoses for the images before being approved for training the AI system. Additionally, the dataset was also checked by a third expert to account for any grading errors. Figure **3.1** shows two chest X-ray images, one from the normal class and the other from the pneumonia class.

Table 3.1 Distribution of the experimental dataset in its original form

	Training	**Test**	**Total**
Normal	1,349	234	1,583
Pneumonia	3,883	390	4,273

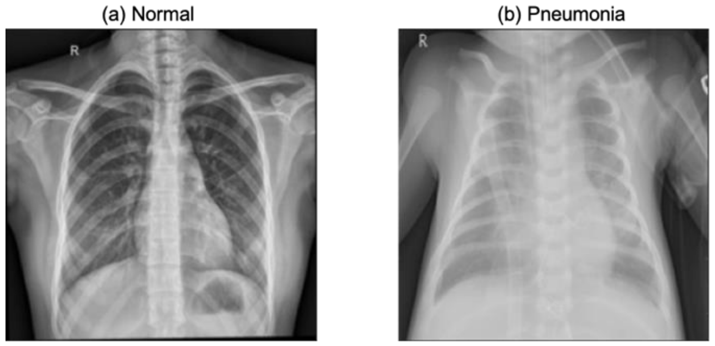


Figure 3.1 Chest Xray with (a) healthy lungs and (b) pneumonia infected lungs

3.1 Prior work on the public dataset

Haloi et al. [50] developed an advanced pneumonia grading system for X-ray images. The system uses a deep CNN trained with end-to-end transfer learning with online random augmentation which provides radiologist-level accurate confidence values for diseases prevalence as the output. Sousa et al. [51] proposed an approach for classification of pneumonia using chest X-ray images on mobile devices. The method employed pre-trained CNN together with a quantization process making use of the TensorFlow Lite platform (with MobileNetV1 and MobileNetV2) to reduce the processing requirement as well as the computational cost.

Islam proposed a scalar-on-image classification model that adopts the ideas of functional data analysis [52]. The model treats images as functional measurements and selects basis functions by exploiting the underlying covariance structures. These basis functions are used in approximating the image profiles and corresponding regression coefficient. The regression model is then applied to classify pneumonia against normal as well as viral pneumonia against bacterial pneumonia.

Multiple deep learning based works for pneumonia classification on this dataset are available in literature. Okeke et al. [28] constructed a CNN model from scratch and achieved a validation accuracy of 93.73%. M.Togacar et al. [53] extracted features from multiple deep learning models and the classifier achieved an accuracy of 96.84%. Vikash et al. [36] used a majority voting based ensemble model on top of multiple pretrained network and achieved 96.4% accuracy.

A weighted classifier was devised by Hashmi et al. [35] that combines the predictions from the five pretrained state-of-the-art deep learning models namely ResNet18, Xception, InceptionV3, DenseNet121, and MobileNetV3. Each of the pretrained model was individually finetuned for pneumonia classification. Then a part of the dataset which was kept aside was used to find the optimal weight for each of the classifiers. The weighted classifier outperformed all the individual models. A partial data augmentation technique was used to increase the size of the training dataset and to reduce the class imbalance. The weighted classifier achieved 98.43% accuracy.

Chapter 4

Methodology

In this work, transfer learning was employed for the detection of pneumonia from chest X-rays and a set of finetuning experiment was evaluated which is detailed in Chapter 5. The network architecture used for the convolutional base is DenseNet121 which consists of four dense blocks with transition layers in between, as shown in Figure **4.1**.

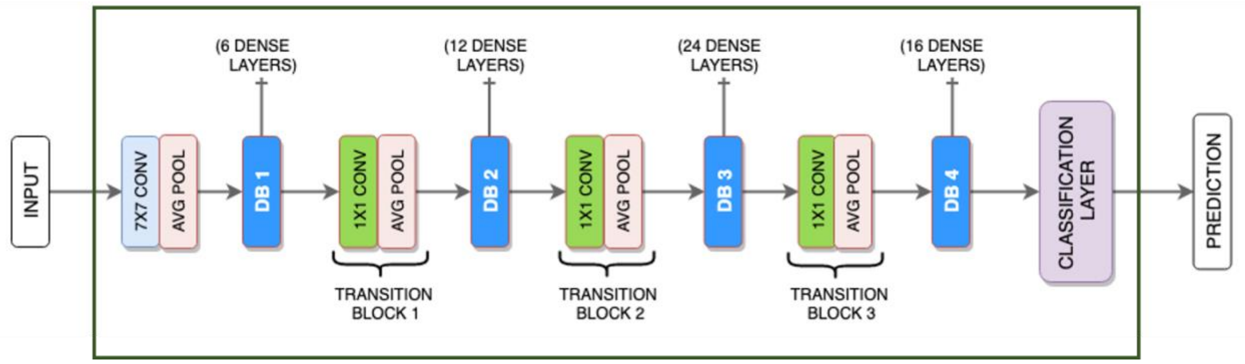


Figure 4.1 Schematic of DenseNet121 architecture

4.1 Data Preprocessing

Each image from the dataset was preprocessed according to requirement of the deep neural network used. The two important steps involved were: resizing and normalization. The standard DenseNet121 architecture requires the input images to be of size 224 × 224, but the dataset consisted images of varying sizes. Hence, all the images were resized to fulfill the requirement. As an additional preprocessing step, RGB values in each image were normalized to be in range [0,1], in order to meet the expectation of the base architecture.

4.2 Oversampling

Since the ratio of two classes in the training and the test set was not consistent, the training and the test set was combined which yielded in 1,583 normal and 4,273 pneumonia samples. After shuffling samples in each of the classes, the dataset was split into training, validation and test sets with cardinality of 70%, 15% and 15% respectively. After such reorganization the resultant training dataset consisted of a total of 1,109 images of the normal (healthy) case and 2,991 images of the pneumonia case which makes the dataset highly imbalanced.

A highly unbalanced dataset can adversely affect the training process that leads to a bias during the prediction phase of the trained model [54]. This means an instance not previously seen by the model is more likely to be classified as the majority class (pneumonia in this case). One way to handle this is to balance out the bias by providing the minority class (normal in this case) with a higher weightage for prediction, based on the cardinality of the two classes. However, this may lead to incorrect predictions if the model was actually able to learn the feature correctly even with the imbalance in the dataset [55], [56].

In the regression model study treating images of the current dataset as functional measurements, Islam [52] drew a random sample of equal size from each class of this dataset so as to get a new balanced set of 2,600 images and then split it into training and test set. While this approach could help in avoiding data imbalance, a considerable amount of data is discarded and unutilized. For the training of a data hungry deep learning models, it might not be wise to discard data especially when there is a limited amount of data available.

While working with this same dataset for the weighted CNN classifier, Hashmi et al. [35] combined the training and test set, and made three splits: one for training the individual classifiers, one for training the weights associated with the classifier, and one meant for testing the individual and weighted classifiers. To handle the imbalance in the dataset, they used transformations on the images belonging to normal class in training set and added them back in the training set along with the original images to increase the size of the normal class and make it comparable in size with the pneumonia class. In has become increasingly common in training a deep learning model to oversample the minority class so as to handle the class imbalance and then it is followed by augmentation transformations to incorporate variance in input feature space [20]. In this work, images form the normal class were oversampled so as to match the cardinality of the pneumonia class in the input training batches.

4.3 Data Augmentation

Adequate training of a deep neural network requires big data. Parameters are undermined when the training data lacks sufficient variance and the trained network generalizes poorly. Data augmentation solves this problem by applying geometric transformations on existing images while preserving their labels. The augmentation step aids in extracting more information from the training dataset and thus helps the network to avoid overfitting [20].

Table 4.1 Setting used for the augmentation operation

Transformation	**Setting**
Vertical Shift	-0.2 to 0.2
Horizontal Shift	-0.2 to 0.2
Shear	0.2 in each direction
Rotation	-45° to 45°

The settings used in image augmentation are shown in Table **2.1**. The resultant images after applying various augmentation transformation on a single original image are shown in Figure **4.2**. While the figure demonstrates only one augmentation operation performed to the base image in each case, during training of the network multiple of these techniques were randomly applied to generate the augmented image.

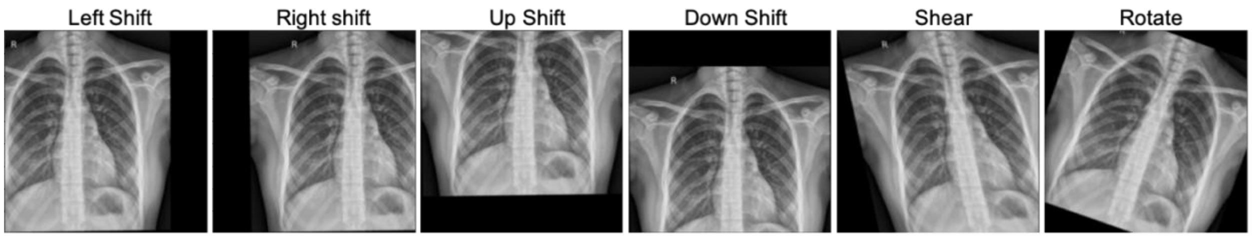


Figure 4.2 Example of resultant image after applying augmentation techniques

4.4 Feature extraction

In the context of deep learning, the most common manifestation of transfer learning is described as a workflow in Table **4**.**2**. The first and second step refer to the feature extraction process. The idea here is to reuse the representations learned by the network on a previous task so as to extract meaningful features from the images of the new dataset. Hence, the convolutional base is set to run only in inference mode, with no weight updates at this stage.

Table 4.2 Deep transfer learning workflow

```
1. Take a previously trained model and freeze its convolutional base. It
   would run in inference mode to work as a feature extractor.
2. Add a new trainable classifier layer on top of the frozen layers. It will
   learn to translate the old feature maps into predictions for the new
   dataset.
3. Train the new classifier on the new dataset.
4. To achieve further improvement, perform finetuning which refers to
   unfreezing the entire or part of the model and re-training it with a very
   low learning rate on the new dataset.
```

In this work, DenseNet121 was used as the convolutional base. And the existing classifier, which was meant for classification of ImageNet natural images consisting of 1000 classes, was replaced with a new classifier that will handle the binary classification of the pneumonia dataset. The new classifier is trained from scratch with the pretrained model working in inference mode so as to adapt the feature maps learned previously into new dataset.

It does not require to train the entire model from scratch because the base convolutional network already learned the kernels to extract the features that are generic for image classification. However, the classifier of the pretrained model could not be reused since it is specific to the previous classification task and is trained to work for a different set of classes. A pictorial representation of the states of the different part of the DenseNet121 during feature extraction phase is shown in Figure **4**.**3**.

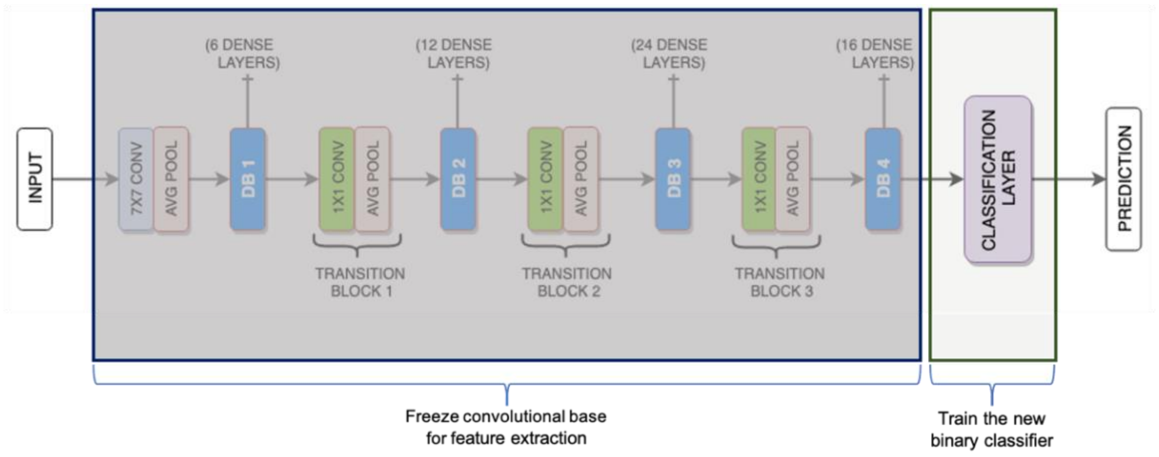


Figure 4.3 Pre-trained DenseNet121 running in feature extraction mode

4.5 Fine-Tuning the model

After training the new classifier with the extracted features obtained by running the pretrained network in inference mode, further enhancement for the model on the new task is achieved by making a few of the top layers of a frozen model base trainable and then jointly training both the new classifier and the last layers of the base model.

During the fine-tuning process, the learning rate needs to be very low since a large learning rate might drastically change the weights of the network leading to quick overfitting thereby losing the knowledge it learned from the previous training on source domain. Very slow update allows the network to fine-tune the feature maps so that they become more relevant for the current task in hand. Thus, the goal here is to achieve meaningful improvements by incrementally readapting the pretrained features to the new dataset.

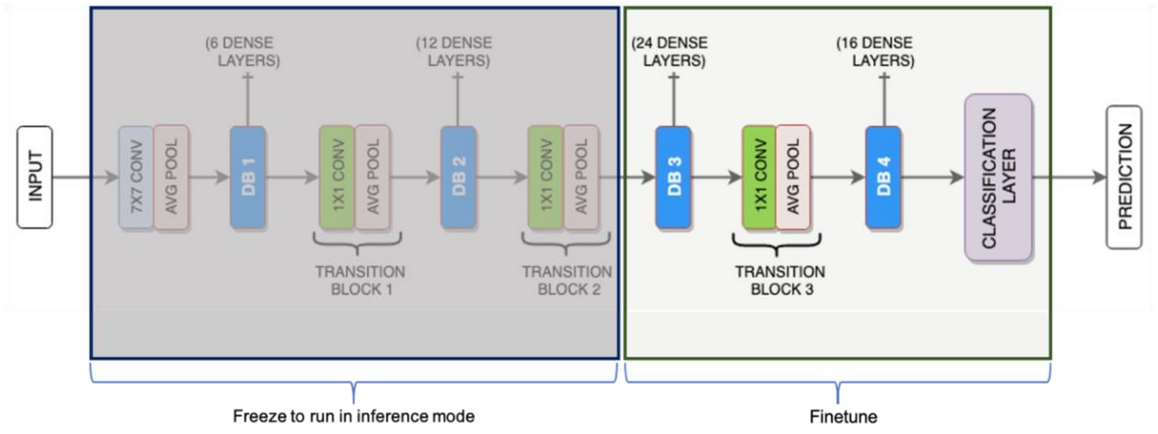


Figure 4.4 Fine-tuning of pre-trained DenseNet121

The finetuning of top 2 dense blocks of DenseNet121 is depicted in Figure **4.4**. DenseNet121 consists of four dense blocks. In the current work, a number of experiments on finetuning the pretrained DenseNet121 model were performed with top n number of dense blocks set to trainable, where $n \in \{1,2,3,4\}$. The results are reported in next chapter.

Chapter 5

Experimental Results

In this section, we discuss about the experimental setup, evaluation results and analyze the performance of each of the models trained under different finetuning setups.

5.1 Experimental setup

This chapter deals with pneumonia classification experiments and evaluation of the model which is trained following different techniques. The chest X-ray image dataset from [57] was reorganized as discussed in Section 4.2 and was used for all the experiments. The Keras open-source deep learning framework with TensorFlow in the backend was used to load the architecture of DenseNet121 and the existing multiclass classifier was replaced with a binary classifier to create a new model from the base model. The network was then trained using multiple experimental setup which is discussed below. In each setup, the pre-processed and normalized chest X-ray images were used for the training. Data oversampling and augmentation techniques were employed to address the imbalance in dataset and to avoid overfitting. For the training, Adam optimizer [58] was used along with the binary crossentropy loss function. All these configurations were chosen based on empirical trials. All the computation work was done on Google Colaboratory that provides with a 12GB of RAM, NVIDIA Tesla T4 16 GB GPU, and Intel(R) Xeon(R) CPU @ 2.20GHz processor.

While constructing the new model on top of Densenet121, the weights of the convolutional base was initialized with the ImageNet pretrained weights. And the weights of the newly added classifier were randomly initialized. The output tensor of the base model was passed through a global average pooling layer to obtain the feature vector, followed by a dropout layer for regularization with a rate of 0.5, and finally the fully connected classifier layer. The transfer learning process is then carried out in two steps: (a) training of the classifier while the convolutional base is set to work in inference mode only and (b) fine-tuning the entire or part of the network.

The transfer learning experiment was run in four variants: (i) Model1: Finetune top 1 dense block, (ii) Model2: Finetune top 2 dense blocks, (iii) Model3: Finetune top 3 dense blocks, and (iv) Model4: Finetune all 4 dense blocks.

5.2 Result in Terms of Testing Accuracy and Testing Loss

Figure **5.1** shows the accuracy and loss curves obtained while training the models for 100 epochs both for training and validation sets. The learning rate during training of the classifier and the finetuning were kept at 1e-5 and 1e-6 respectively. The validation accuracy in most cases is higher than the training accuracy because both batch normalization and dropout affect the training accuracy, and these are disabled by the network while evaluating the validation set. The vertical dashed line indicates the epoch where finetuning starts.

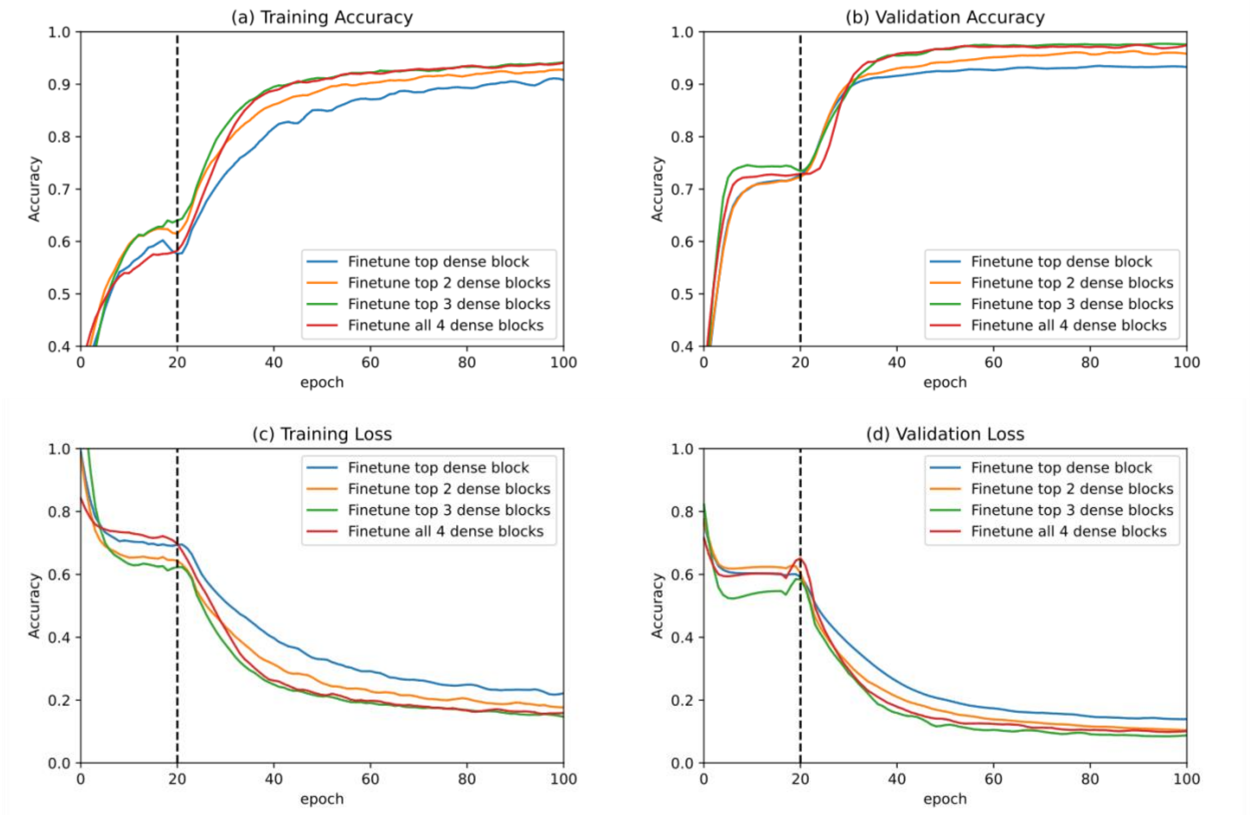


Figure 5.1 Training accuracy and training loss curves for the finetune variants

Table **5.1** summarizes the testing accuracy and testing loss for each variant of finetuning. All test images were pre-processed in the same way the training images were processed to follow the size and normalization requirement of the network. The test accuracy increased as more dense blocks were included for finetuning. The maximum testing accuracy and the minimum testing loss was attained when top 3 dense blocks were finetuned. And inclusion of all four dense blocks for finetuning caused a slight decline in test accuracy.

Table 5.1 Final testing accuracy and testing loss of the finetune variants

Experiment variant	Testing Accuracy	Testing Loss
Finetune top Dense block	93.28%	0.1396
Finetune top 2 Dense blocks	95.90%	0.1038
Finetune top 3 Dense blocks	97.61%	0.0876
Finetune all 4 Dense blocks	97.04%	0.0957

5.3 Comparison with the baseline

Table 5.2 compares the performance of the four models, in terms of accuracy, sensitivity and specificity on the test dataset, with the baseline obtained and reported by Kermany et al. [36] using transfer learning on Inception V3 architecture that was pretrained on the ImageNet dataset. All the models in this study attains better values for each of the performance metrics, except the sensitivity of Model1.

Table 5.2 Performance comparison with the baseline

Experiment variant	Testing Accuracy	Sensitivity	Specificity
Finetune top Dense block	93.28%	91.89%	97.05%
Finetune top 2 Dense blocks	95.90%	96.10%	95.36%
Finetune top 3 Dense blocks	97.61%	98.60%	94.94%
Finetune all 4 Dense blocks	97.04%	97.66%	95.36%
Kermany et al [36]	92.80%	93.20%	90.10%

5.4 Performance analysis of the models

To assess the variability in prediction for each of the models, 95% confidence interval (CI) were computed using bootstraps with two thousand resampling where samples of random sizes were drawn with replacement from the test dataset. To dichotomize the predictions of the models, a threshold of 0.5 was used. The mean, lower limit and the upper limit of 95% CI for the models are shown in Table **5.3**.

Table 5.3 95% confidence intervals for the testing accuracy

Experimental model	Lower Limit	Mean	Upper Limit
Model1	0.9169	0.9328	0.9487
Model2	0.9487	0.9590	0.9715
Model3	0.9658	0.9761	0.9852
Model4	0.9590	0.9704	0.9806

Further, in order to assess the performance of one model against another, McNemar's test [59] was used to evaluate whether there were significant differences. Contingency matrices constructed for the test for each consecutive pair of models are shown in Figure **5.2**. It was found that the p-value for first two consecutive pair of models was well under 0.05. The p-values of Model1 against Model2 and Model2 against Model3 were 0.000003 and 0.000488 respectively. While the inclusion of the all four dense blocks for finetuning caused a slight decline in test accuracy for Model4 compared to Model3, it was not statistically significant (p-value 0.143463).

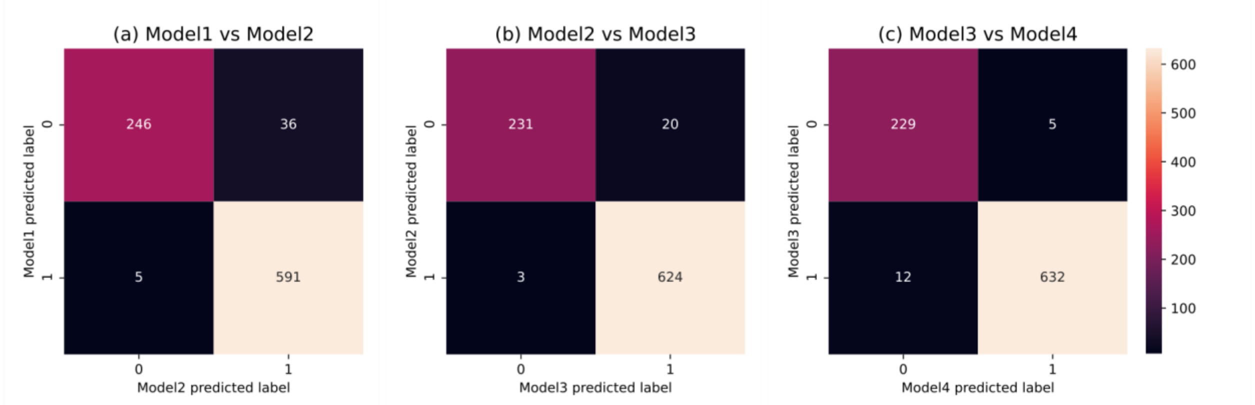


Figure 5.2 Contingency matrices for consecutive pair of models

Figure 5.3 shows the receiver operating characteristics (ROC) curves of the models. To assess the performance of the models on AUC metric, DeLong's test for two correlated ROC curves was used [60], [61]. The AUC score 0.992 (95% CI 0.9874-0.9963) of Model2 was better compared to Model1's at 0.988 (95% CI 0.9817-0.9923) with p-value = 0.00334. Model3 achieved AUC of 0.995 (95% CI 0.9917-0.9983). Although it was only slightly higher than that of Model2, the difference was statistically significant (p-value = 0.003182). The AUC score for Model4 was 0.993 (95% CI 0.9876-0.9985), which was a small decline from what Model3 had, but it was not statistically significant (p-value = 0.248).

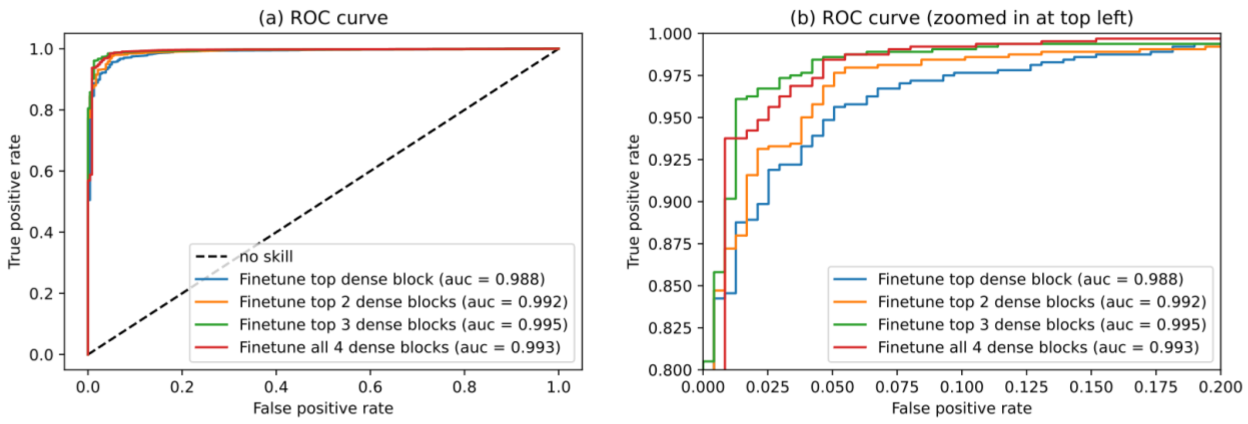


Figure 5.3 Receiver operating characteristics (ROC) curves for the finetune variants

While the actual differences in the AUC score for models were negligible, it is worth remembering that this metric might not be apt for performance evaluation in this case because the test dataset is imbalanced with higher number of pneumonia cases than the normal cases.

The precision-recall curves are usually more informative in case of imbalanced test set. Figure **5.4** shows the precision-recall curves for all the models. The dashed line represents the baseline for a random predictor with no skill. The area under the curve for the models shows a very negligible difference and attains maximum value for Model3 where top 3 dense layers are finetuned.

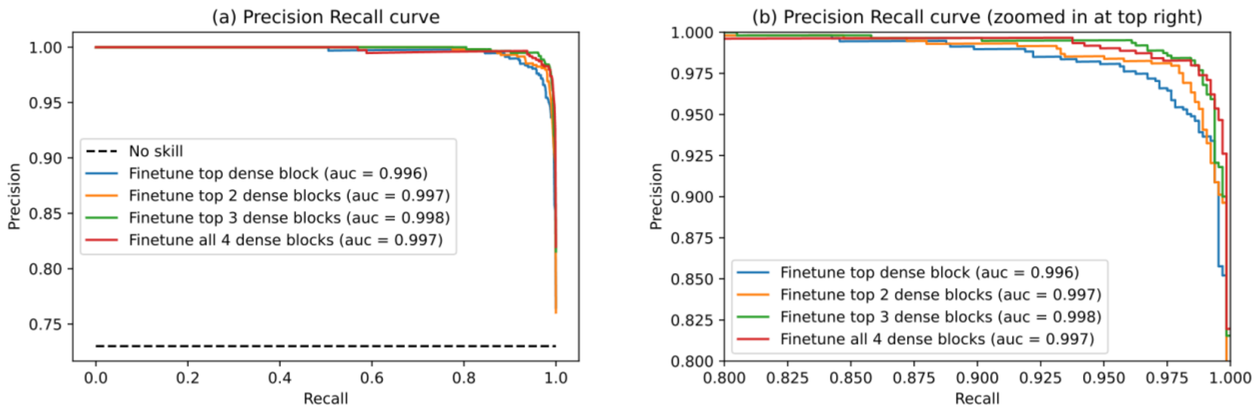


Figure 5.4 Precision-Recall curves for the finetune variants

For further evaluation, confusion matrix on the test dataset for each of the models were obtained. It can be inferred that the Model3 has the least false negative rate by comparing the confusion matrices in Figure **5.5**.

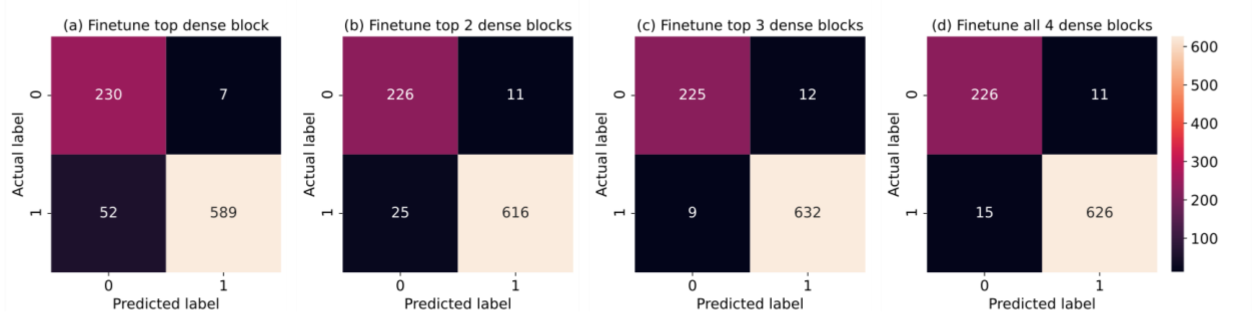


Figure 5.5 Confusion matrix for the finetune variants

Finally, accuracy, precision, recall, and F1 score for all the models were computed from the respective confusion matrices. Table **5.4** compares the accuracy, precision, recall and F1 score for the four models. It is evident that Model3 outperforms Model1 and Model2 in all four evaluation metrics. Model4 gets a tiny improvement in precision over Model3 but at the same time recall is decreased. In this case, it can be deduced that Model3 is better than Model4 by comparing the F1 score that combines precision and recall.

Table 5.4 Accuracy, precision, recall, and F1 score for the finetune variants

Experiment variant	Accuracy	Precision	Recall	F1 Score
Finetune top Dense block	0.9328	0.9883	0.9189	0.9523
Finetune top two Dense blocks	0.9590	0.9825	0.9610	0.9716
Finetune top three Dense blocks	0.9761	0.9814	0.9860	0.9837
Finetune all four Dense blocks	0.9704	0.9827	0.9766	0.9797

5.5 Comparison with existing methods

Cohen et al. [29] used a DenseNet-121 based model that attained 92.8% accuracy. Rajaraman et al. [30] reported a test accuracy of 96.2% for customized VGG16 to detect pneumonia. Saraiva et al. [31] and Ayan et al. [32] used deep learning based methods and achieved an accuracy of 94.4% and 87% respectively. Rahman et al. compared the performance of deep transfer learning on AlexNet, ResNet18, DenseNet201 and SqueezeNet. The DenseNet201 model attained the highest accuracy of 98% for pneumonia classification [33]. Vikash et al. [34] used a majority voting based ensemble model on top of multiple pretrained network and achieved an 96.4% accuracy. Hashmi et al. [35] achieved a 98.43% accuracy with their weighted classifier that combine prediction from five state-of-the-art pretrained CNN models. The Model3 of this study attained comparable results with other existing methods and in many cases surpassed them. All these results are summarized in Table **5**.**5**.

Table 5.5 Comparison with existing methods (all measures in percentage)

Experiment variant	Accuracy	Precision	Recall	F1 Score	AUC
Cohen et al. [29]	92.8	90.1	93.2	91.62	99.0
Rajaraman et al. [30]	96.2	97	99.5	98.23	99.0
Saraiva et al. [31]	94.4	94.3	94.5	94.40	94.5
Ayan et al. [32]	87	91.3	89.1	90.19	87.0
Rahman et al. [33]	98	97.0	99.0	97.99	98.0
Vikas et al. [34]	96.4	93.28	99.6	96.34	99.34
Hashmi et al [35]	98.43	98.26	99.00	98.63	99.76
Model3	97.61	98.14	98.60	98.37	99.5

5.6 Discussion

The underlying architecture for each variant of the finetuning experiment was same. In each case the convolutional base was initialized with pretrained weights and the newly added classifier was trained for 20 epochs. And in the next 80 epochs, finetuning was performed for up to top n number of dense blocks, where $n \in \{1,2,3,4\}$. The performance of the classifier for the models in this study got better as n increased but only up to $n = 3$. There was a small decline in values of performance metrics evaluated on the test dataset when the fourth dense block was included for finetuning, but it was not statistically significant. This indicates that in deep transfer learning, the finetuning phase does not need to be extended all the way down to the layers at the lowest level. As supported by the results of this study, a model that performs finetuning of enough number of layers, can achieve a performance equivalent to another model that implements finetuning for all the existing layer in the underlying architecture. Thus, finetuning only up to the optimal number of layers, instead of carrying it out for all the layers, can save us computational cost without compromising the performance.

The optimal number of layers, that requires finetuning, is expected to vary based on the underlying network architecture and also the kind of the new task being learned by the network. For pneumonia classification on DenseNet121 architecture, the optimal number of top dense blocks to finetune is found to be three in this study. Therefore, while applying transfer learning in future for another similar image classification task on the same architecture, it would be a good strategy to start with top three dense blocks, and then the optimal number of layers can be established by including a few top layers of fourth dense block or excluding a few bottom layers of third block for the finetuning process.

Furthermore, lower layers of deep CNNs learn the generic features that generalize to almost all types of images. As we go higher up, the learned features are increasingly more specific to the dataset on which the model is trained [62]. These low-level spatial features are better learned with big data because the network gets to explore through enough variation of input feature space enabling it to capture the underlying characteristics [63]. And finetuning these lower layers during transfer leaning without slow enough learning rate, can alter these low-level kernels causing it to overfit on the small dataset of the new task being learned. In this study we learned that finetuning the lower layers does not significantly change the performance of the classifier. Such outcome, thus, corroborates that the lower layers in a pre-trained network carries the knowledge of extracting generic features which carries important information for other similar tasks, and it might be best for the transfer learning to use them unaltered.

Chapter 6

Conclusion

In this work, a deep transfer learning based framework was applied for automatic detection of pneumonia in chest X-ray images. A state-of-the-art deep learning network (DenseNet) was employed for this study. Transfer learning and data augmentation were used to solve the problem of overfitting which is prevalent in case of insufficient training data. Further, the finetuning experiments were performed up to different levels of dense blocks in the network and it was found that finetuning the top three dense block yields best performance and no further gain was achieved when fourth dense block was included. In transfer learning involving deep learning network, generally only a top few layers are finetuned to avoid the computation cost and reach a desired level of performance. Many work in literature uses the approach to finetune the entire network in an attempt to extract the last bit of improvement [35]. This study suggests that finetuning the entire network might not be the best approach for the target task.

6.1 Limitations and scope for future work

Since the finetuning experiments were run only on the pneumonia dataset, this study cannot affirm that optimal number of dense blocks to be finetuned remains same for similar transfer learning tasks. In future, it would be interesting to study (i) how much of low-level spatial characteristics needs to be preserved for a given target task, and (ii) how the optimal number of layers, that should be finetuned, depends on the underlying network architecture and target task. Also, the dense blocks in the network were set to trainable or untrainable during finetuning in the current study. Future work can be devoted to find out how the performance of the network changes when all the dense blocks are finetuned but with the dense blocks assigned with decreasing rate of weight updates as we go lower in the network.

References

[1] J. G. Bartlett, R. F. Breiman, L. A. Mandell, and T. M. File Jr, "Community-acquired pneumonia in adults: guidelines for management," *Clin. Infect. Dis.*, vol. 26, no. 4, pp. 811–838, 1998.

[2] M. P. Girard, T. Cherian, Y. Pervikov, and M. P. Kieny, "A review of vaccine research and development: human acute respiratory infections," *Vaccine*, vol. 23, no. 50, pp. 5708–5724, 2005.

[3] Y. Ye, E. Zulu, M. Mutisya, B. Orindi, J. Emina, and C. Kyobutungi, "Seasonal pattern of pneumonia mortality among under-five children in Nairobi's informal settlements," *Am. J. Trop. Med. Hyg.*, vol. 81, no. 5, pp. 770–775, 2009.

[4] L. L. G. Oliveira, S. A. e Silva, L. H. V. Ribeiro, R. M. de Oliveira, C. J. Coelho, and A. L. S. S. Andrade, "Computer-aided diagnosis in chest radiography for detection of childhood pneumonia," *Int. J. Med. Inform.*, vol. 77, no. 8, pp. 555–564, 2008.

[5] M. Lavine, "The early clinical X-ray in the United States: patient experiences and public perceptions," *J. Hist. Med. Allied Sci.*, vol. 67, no. 4, pp. 587–625, 2012.

[6] Y. LeCun *et al.*, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[8] P. Gang *et al.*, "Dimensionality reduction in deep learning for chest X-ray analysis of lung cancer," in *2018 tenth international conference on advanced computational intelligence (ICACI)*, 2018, pp. 878–883.

[9] M. Grewal, M. M. Srivastava, P. Kumar, and S. Varadarajan, "Radnet: Radiologist level accuracy using deep learning for hemorrhage detection in ct scans," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 281–284.

[10] V. Gulshan *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *Jama*, vol. 316, no. 22, pp. 2402–2410, 2016.

[11] A. Esteva *et al.*, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[12] Y. Bar, I. Diamant, L. Wolf, S. Lieberman, E. Konen, and H. Greenspan, “Chest pathology detection using deep learning with non-medical training,” in *2015 IEEE 12th international symposium on biomedical imaging (ISBI)*, 2015, pp. 294–297.

[13] C. D. Malon and E. Cosatto, “Classification of mitotic figures with convolutional neural networks and seeded blob features,” *J. Pathol. Inform.*, vol. 4, 2013.

[14] A. Cruz-Roa *et al.*, “Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks,” in *Medical Imaging 2014: Digital Pathology*, 2014, vol. 9041, p. 904103.

[15] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio, “A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2013, pp. 403–410.

[16] S. Stirenko *et al.*, “Chest X-ray analysis of tuberculosis by deep learning with segmentation and augmentation,” in *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)*, 2018, pp. 422–428.

[17] T. Chen and C. Chefd’Hotel, “Deep learning based automatic immune cell detection for immunohistochemistry images,” in *International workshop on machine learning in medical imaging*, 2014, pp. 17–24.

[18] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” in *Advances in neural information processing systems*, 2012, pp. 2843–2851.

[19] N. Dhungel, G. Carneiro, and A. P. Bradley, “Deep learning and structured prediction for the segmentation of mass in mammograms,” in *International Conference on Medical image computing and computer-assisted intervention*, 2015, pp. 605–612.

[20] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *J. Big Data*, vol. 6, no. 1, p. 60, 2019.

[21] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, *A survey of transfer learning*, vol. 3, no. 1. Springer International Publishing, 2016.

[22] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[23] S. U. Khan, N. Islam, Z. Jan, I. Ud Din, and J. J. P. C. Rodrigues, "A novel deep learning based framework for the detection and classification of breast cancer using transfer learning," *Pattern Recognit. Lett.*, vol. 125, pp. 1–6, 2019, doi: 10.1016/j.patrec.2019.03.022.

[24] J. De Fauw *et al.*, "Clinically applicable deep learning for diagnosis and referral in retinal disease," *Nat. Med.*, vol. 24, no. 9, pp. 1342–1350, 2018.

[25] A. Khatri, R. Jain, H. Vashista, N. Mittal, P. Ranjan, and R. Janardhanan, "Pneumonia Identification in Chest X-Ray Images Using EMD," in *Trends in Communication, Cloud, and Big Data*, Springer, 2020, pp. 87–98.

[26] P. Rajpurkar *et al.*, "Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists," *PLoS Med.*, vol. 15, no. 11, p. e1002686, 2018.

[27] R. H. Abiyev and M. K. S. Ma'aitah, "Deep convolutional neural networks for chest diseases detection," *J. Healthc. Eng.*, vol. 2018, 2018.

[28] O. Stephen, M. Sain, U. J. Maduh, and D.-U. Jeong, "An efficient deep learning approach to pneumonia classification in healthcare," *J. Healthc. Eng.*, vol. 2019, 2019.

[29] J. P. Cohen, P. Bertin, and V. Frappier, "Chester: A Web Delivered Locally Computed Chest X-Ray Disease Prediction System," *arXiv Prepr. arXiv1901.11210*, 2019.

[30] S. Rajaraman, S. Candemir, I. Kim, G. Thoma, and S. Antani, "Visualization and interpretation of convolutional neural network predictions in detecting pneumonia in pediatric chest radiographs," *Appl. Sci.*, vol. 8, no. 10, p. 1715, 2018.

[31] A. A. Saraiva *et al.*, "Models of Learning to Classify X-ray Images for the Detection of Pneumonia using Neural Networks.," in *BIOIMAGING*, 2019, pp. 76–83.

[32] E. Ayan and H. M. Ünver, “Diagnosis of Pneumonia from Chest X-Ray Images Using Deep Learning,” in *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, 2019, pp. 1–5.

[33] T. Rahman *et al.*, “Transfer Learning with Deep Convolutional Neural Network (CNN) for Pneumonia Detection using Chest X-ray,” *Appl. Sci.*, vol. 10, no. 9, p. 3233, 2020.

[34] V. Chouhan *et al.*, “A novel transfer learning based approach for pneumonia detection in chest X-ray images,” *Appl. Sci.*, vol. 10, no. 2, p. 559, 2020.

[35] M. F. Hashmi, S. Katiyar, A. G. Keskar, N. D. Bokde, and Z. W. Geem, “Efficient pneumonia detection in chest xray images using deep transfer learning,” *Diagnostics*, vol. 10, no. 6, pp. 1–23, 2020, doi: 10.3390/diagnostics10060417.

[36] D. S. Kermany *et al.*, “Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning,” *Cell*, vol. 172, no. 5, pp. 1122-1131.e9, 2018, doi: 10.1016/j.cell.2018.02.010.

[37] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights Imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[38] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” *arXiv Prepr. arXiv1511.08458*, 2015.

[39] N. Passalis and A. Tefas, “Learning bag-of-features pooling for deep convolutional neural networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5755–5763.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[41] S. Cai, Y. Shu, G. Chen, B. C. Ooi, W. Wang, and M. Zhang, “Effective and Efficient Dropout for Deep Convolutional Neural Networks,” pp. 1–12, 2019, [Online]. Available: http://arxiv.org/abs/1904.03392.

[42] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in

Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv Prepr. arXiv1502.03167*, 2015.

[44] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[46] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional Networks with Dense Connectivity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8828, no. c, pp. 1–1, 2019, doi: 10.1109/tpami.2019.2918284.

[47] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11141 LNCS, pp. 270–279, 2018, doi: 10.1007/978-3-030-01424-7_27.

[48] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2009.

[49] L. Fei-Fei, "ImageNet: crowdsourcing, benchmarking & other cool things," in *CMU VASC Seminar*, 2010, vol. 16, pp. 18–25.

[50] M. Haloi, K. R. Rajalakshmi, and P. Walia, "Towards Radiologist-Level Accurate Deep Learning System for Pulmonary Screening," *arXiv Prepr. arXiv1807.03120*, 2018.

[51] J. V. M. Sousa, V. R. de Almeida, A. A. Saraiva, D. B. S. Santos, P. M. C. Pimentel, and L. L. de Sousa, "Classification of Pneumonia images on mobile devices with Quantized Neural Network," *Res. Soc. Dev.*, vol. 9, no. 10, pp. e889108382–e889108382, 2020.

[52] M. N. Islam, "Classification of pediatric pneumonia using chest X-rays by functional regression," pp. 1–26, 2020, [Online]. Available: http://arxiv.org/abs/2005.03243.

[53] M. Toğaçar, B. Ergen, Z. Cömert, and F. Özyurt, "A Deep Feature Learning Model for Pneumonia Detection Applying a Combination of mRMR Feature Selection and Machine Learning Models," *Irbm*, vol. 41, no. 4, pp. 212–222, 2020, doi: 10.1016/j.irbm.2019.10.006.

[54] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data," *Data Min. Knowl. Discov.*, vol. 28, no. 1, pp. 92–122, 2014.

[55] J. Burez and D. Van den Poel, "Handling class imbalance in customer churn prediction," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 4626–4636, 2009.

[56] M. Kuhn and K. Johnson, *Applied predictive modeling*, vol. 26. Springer, 2013.

[57] D. S. Kermany *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv Prepr. arXiv1412.6980*, 2014.

[59] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[60] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson, "Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach," *Biometrics*, pp. 837–845, 1988.

[61] X. Robin *et al.*, "pROC: an open-source package for R and S+ to analyze and compare ROC curves," *BMC Bioinformatics*, vol. 12, no. 1, pp. 1–8, 2011.

[62] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3712–3722.

[63] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, 2014, pp. 3320–3328.