# DETECTING ABNORMALITIES IN CHEST X-RAYS USING DEEP LEARNING

Adwoa Brako[1], Russell Moncrief[2]

Columbian College of Arts & Sciences, The George Washington University, Washington,D.C.

[1]abrako64@gwu.edu

[2]russellmoncrief@gwu.edu

## ABSTRACT

For years medical experts have used radiological techniques to explore and visualize abnormalities in the human body. Current radiological techniques require human intervention to process and interpret x-ray images and provide diagnoses. These techniques can become cumbersome and are prone to human error. Because of these drawbacks, we suggest automation is needed to increase efficiency and reduce mis-diagnosis. We investigate the use of the Convolution Neural Network (CNN) and the pre-trained MobileNet (MNT) models as candidate solutions for disease classification. We show that CNN achieves an accuracy of 90 percent and MNT achieves an accuracy of 92 percent. Because of the high rate of classification accuracy and the increase in processing efficiency, we suggest that CNN and MNT are viable substitutes for human intervention. Further analysis showing the AUC-ROC curves of the disease types is highlighted in this paper.

**Keywords -** ***MobileNet, CNN, classification, chest-X-rays, hyperparameter tuning, Network architecture, convolution layer, callback***

## 1. Introduction

### 1.1. Motivation

Medical experts have used radiography as a technique to explore and visualize abnormalities and fractures in the body for several decades. Research has shown that over 40% of X-ray scans produced are of the chest. As a crucial diagnostic imaging tool for identifying abnormalities in the chest, chest radiographs / X-rays are used by health professionals. Chest X-rays are one of the most frequent and cost-effective medical imaging examinations available and can be used to detect many diseases. With an increased rate and heavy reliance on chest X-rays, classifying abnormalities in chest X-ray images can be a difficult task for radiologists, as the functionality of these scans can be limited by challenges in interpretation. This may lead to delay in producing results and/or wrong diagnosis of diseases, which will in turn have adverse effects on patients.

In recent years, Deep Learning has achieved profound recognition and immense breakthroughs in a vast number of computer vision applications, including natural and medical images classification. To curb challenges faced by medical experts, Deep Learning, coupled with large volumes of data, has the ability to efficiently classify images and related disease labels.

### 1.2. Problem

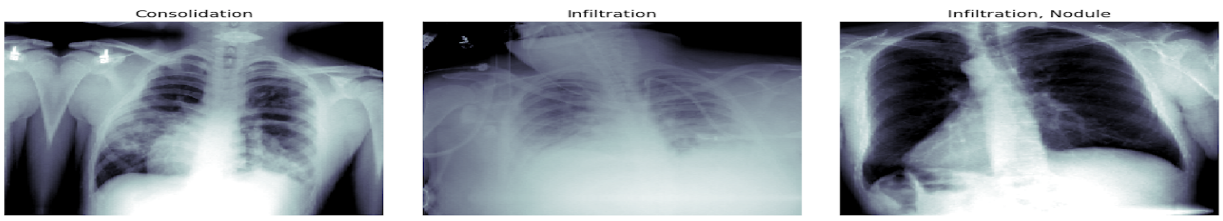
Given the high rate of chest X-Ray scans performed, how can radiologists make objective diagnosis, with less errors, within a timely manner?

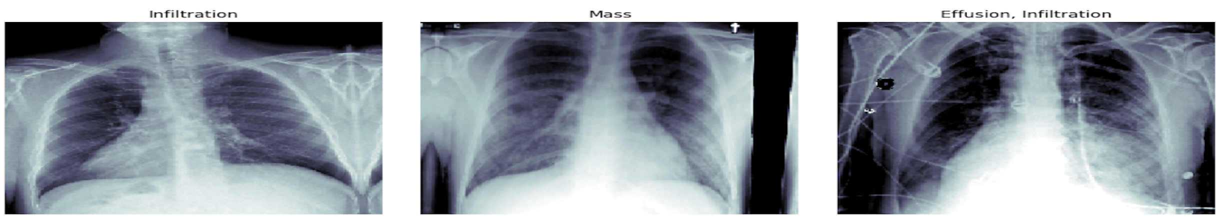
### 1.3. Objective

The goals of this project are (1) to develop an algorithm to aid in classifying Chest X-Ray images based on the type of disease and (2) to accurately classify these images, which will help with the early detection of diseases and adequate timely diagnosis.

## 2. Data

This project is based on data acquired from the National Institutes of Health (NIH) Chest-X-ray dataset. The dataset is composed of over 100,000 frontal-view grayscale chest x-ray images in PNG format (with sizes of 1024 x 1024 pixels) from over 30,000 unique patients. The diseases are labeled with 15 different classes, namely, **Infiltration, Fibrosis, Cardiomegaly, Pneumonia, Atelectasis, Nodule, Mass, Consolidation, Pneumothorax, Edema, Emphysema, Effusion, Pleural thickening ,Hernia and "No findings".** The image labels are produced with Natural Language Processing and associated radiological reports and  are thought to be greater than 90% accuracy.





Sample of chest X-rays in data

## 3. Methodology and Training

Image classification can be a difficult task, given that it can be computationally expensive and hyper-parameter tuning of models can be a challenge. In order to minimize this, the deep learning network adopted will be very crucial in performing the desired task and efficiently classifying labels.

### 3.1. Overview of Convolutional Neural Networks

Convolutional Neural Networks (CNN), being a category of Neural Networks, have proven extremely effective in areas of image recognition and classification. These networks perceive images as dimensional objects, rather than flat objects measured by width and height. These networks are capable of minimizing the parameters in images, while storing the necessary features.

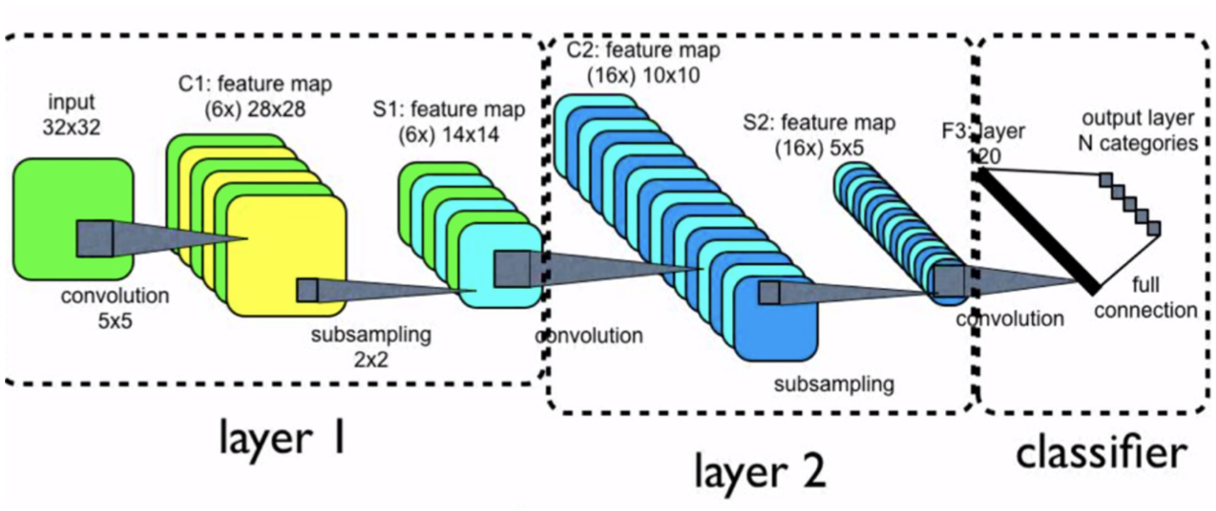


**Fig1. Representation of CNN architecture**

With the *convolution layer* being the building block of CNN, this layer is capable of extracting features of the input using a *convolution filter.* After convolution, pooling is performed to reduce the dimensionality of the input. Optional hyperparameters in the architecture include setting the *filter size*, *stride* and *padding. Fully connected layers* are added after *convolution* and *pooling layers*. Because the *fully connected layer* expects a 1D vector, the output of the final *pooling layer* is converted or flattened and passed as an input to the *fully connected layer.*

Given an input image $I$ and a kernel(filter) $K$ of dimension$k_1$x $k_2$, the convolution operation is:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} K_{m,n} \cdot I_{i+m,j+n} + b$$

3.2. Experimental Setup of Proposed Convolutional Neural Network Model

Multiple CNN model architectures were utilized during training. The input images for the initial network had a dimension of (224 x 224 x 1).

**Test1:** The initial CNN model trained consisted of two convolutional layers with Relu transfer function between these layers, with *maxpooling2D* of size (2x2). The first convolutional layer had a kernel size of (6 x 6), while the second convolutional layer had a kernel size of (2 x 2). After these convolutional layers, the input was flattened to 1D, then passed on to two fully connected layers. Dropout of (0.2) was added to the first fully connected layer, with Relu as the transfer function. The final fully connected layer had the target labels as input, with sigmoid transfer function. The architecture of this network is below:

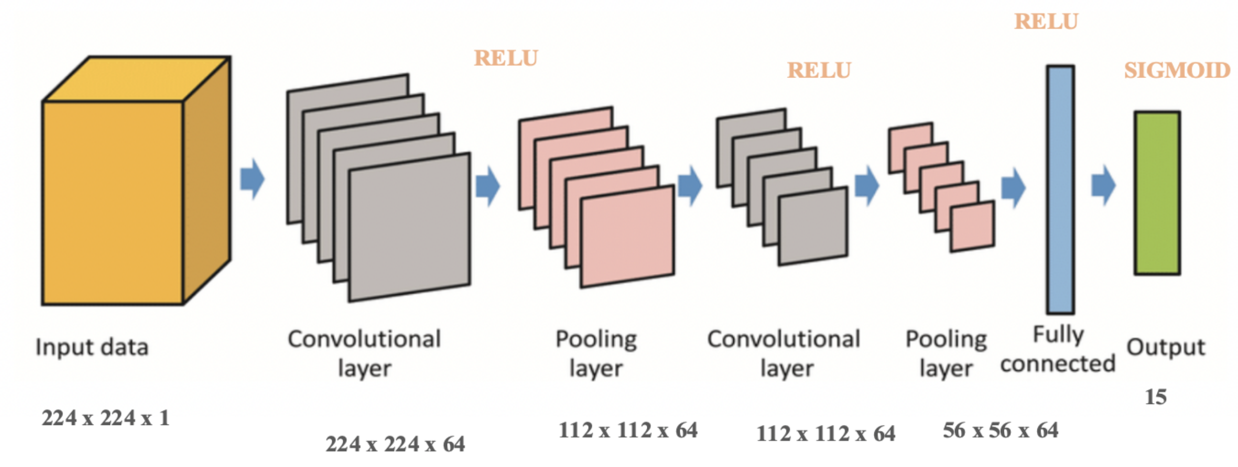


**Fig2. CNN model architecture - Test1**

The model's performance was assessed with a 20% held out set of the data. Overfitting was monitored by comparing *binary cross entropy loss* and *accuracy*. *Binary cross entropy* was used because it is ideal for multi-label classification problems. The loss was calculated as follows, where ŷ is the predicted value:

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{i=0}^{N}(y * log(\hat{y}_i) + (1-y) * log(1-\hat{y}_i))$$

Parameters chosen for the first model in **Test1** include Learning Rate=0.02, Epochs=2 (with 1227 steps per epoch), Batch Size=64 and Dropout=0.2.

Parameters chosen for the second model in **Test1** include Learning Rate=0.001, Epochs=3 (with 1227 steps per epoch), Batch Size=64 and Dropout=0.5.

A batch size of 64, which is quite small in comparison with the amount of data, was chosen because smaller batches generally work well, are able to offer a regularizing effect and lower generalization error. In subsequent training, the batch size was increased to determine if performance would improve. As learning rate controls how quickly the model adapts to the problem, the learning rates used in **Test1** were selected to ensure that the model would not converge too quickly (large learning rate) or get stuck in the process, caused by extremely small learning rate)

**Test2**: Both convolutional layers in this network had a kernel size of (2 x 2), with *maxpooling2D* of size (2x2) in both layers. After these convolutional layers, the input was flattened to 1D, then passed on to two fully connected layers. Dropout of (0.5) was added to the first fully connected layer, with LeakyRelu as the transfer function. Further research revealed that sigmoid is not an ideal transfer function for classification problems, so Relu transfer function was used in the output layer of this trained model. The architecture of this network is below:

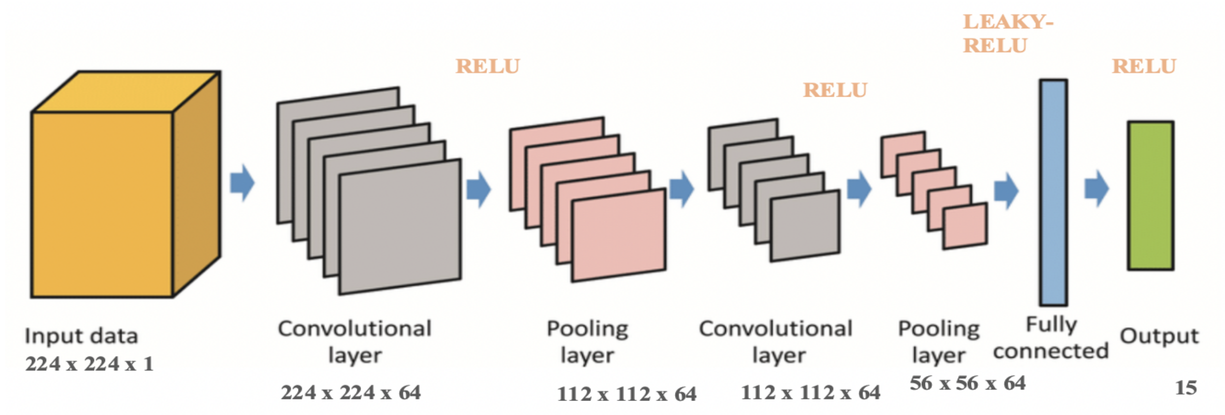


**Fig3. CNN model architecture - Test2**

Parameters chosen for the trained model in **Test2** include Learning Rate=0.02, Epochs=2 (with 1227 steps per epoch), Batch Size=64 and Dropout=0.2.

## 3.3. Overview of Transfer Learning

Transfer Learning involves using a model trained on a problem as a starting point on a related problem. One or more layers from the trained model are used in a different model. This flexibility allows using pre-trained models directly and integrating them into entirely new models. Transfer learning has the benefit of reducing training time for a neural network model and is capable of resulting in lower generalization error. There are several models for Transfer Learning, including Residual Network (eg. ResNet50) , VGG (eg. VGG16 or VGG19), MobileNet, among others.

## 3.4. Experimental Setup of Proposed Pre-Trained Model

MobileNet was used as the base for training another set of models. This pre-trained model was selected because of its streamlined architecture of using depth-wise separable convolutions to build lightweight networks.

| Type / Stride | Filter Shape | Input Size |
|---|---|---|
| Conv / s2 | 3 × 3 × 3 × 32 | 224 × 224 × 3 |
| Conv dw / s1 | 3 × 3 × 32 dw | 112 × 112 × 32 |
| Conv / s1 | 1 × 1 × 32 × 64 | 112 × 112 × 32 |
| Conv dw / s2 | 3 × 3 × 64 dw | 112 × 112 × 64 |
| Conv / s1 | 1 × 1 × 64 × 128 | 56 × 56 × 64 |
| Conv dw / s1 | 3 × 3 × 128 dw | 56 × 56 × 128 |
| Conv / s1 | 1 × 1 × 128 × 128 | 56 × 56 × 128 |
| Conv dw / s2 | 3 × 3 × 128 dw | 56 × 56 × 128 |
| Conv / s1 | 1 × 1 × 128 × 256 | 28 × 28 × 128 |
| Conv dw / s1 | 3 × 3 × 256 dw | 28 × 28 × 256 |
| Conv / s1 | 1 × 1 × 256 × 256 | 28 × 28 × 256 |
| Conv dw / s2 | 3 × 3 × 256 dw | 28 × 28 × 256 |
| Conv / s1 | 1 × 1 × 256 × 512 | 14 × 14 × 256 |
| 5× Conv dw / s1 | 3 × 3 × 512 dw | 14 × 14 × 512 |
| Conv / s1 | 1 × 1 × 512 × 512 | 14 × 14 × 512 |
| Conv dw / s2 | 3 × 3 × 512 dw | 14 × 14 × 512 |
| Conv / s1 | 1 × 1 × 512 × 1024 | 7 × 7 × 512 |
| Conv dw / s2 | 3 × 3 × 1024 dw | 7 × 7 × 1024 |
| Conv / s1 | 1 × 1 × 1024 × 1024 | 7 × 7 × 1024 |
| Avg Pool / s1 | Pool 7 × 7 | 7 × 7 × 1024 |
| FC / s1 | 1024 × 1000 | 1 × 1 × 1024 |
| Softmax / s1 | Classifier | 1 × 1 × 1000 |

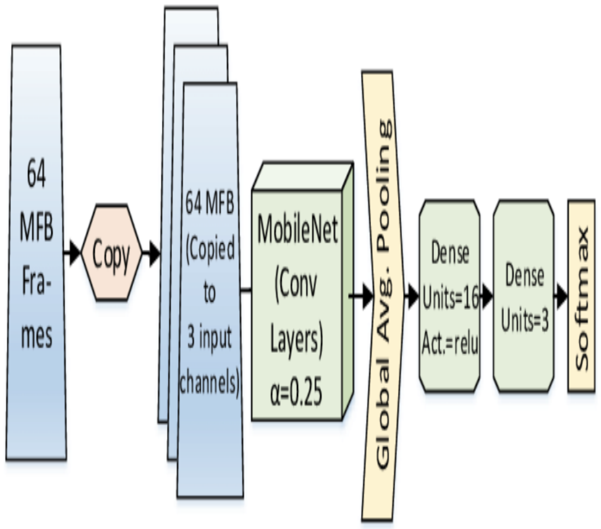


**Fig4. General architecture of Mobilenet**

Multiple model architectures utilising MobileNet were used during training. The input images for the initial network had a dimension of (224 x 224 x 1).

**Pre-TrainedTest1:** After initialising MobileNet as the base model, *GlobalAveragePooling2D* was added, with a dropout of (0.5). Next was a fully connected layer with 512 neurons and with a dropout of (0.5). The fully connected output layer had the target labels as input, with sigmoid transfer function. *ModelCheckpoint*, *EarlyStopping* and *ReduceLROnPlateau* callbacks were included in the model's architecture. EarlyStopping was added to stop training at the point when performance on the validation set starts to degrade. This was done to address or reduce overfitting and improve generalization of the network. *ReduceLROnPlateau* was added to adjust the learning rate when the model's performance starts to stall or plateau. This was done to aid in fine-tuning model weights. Parameters chosen for the first model in **Pre-TrainedTest1** include Learning Rate=0.02, Epochs=1 (with 1227 steps per epoch), Batch Size=64 and Dropout=0.5.

Another model with the same network architecture was trained, but with slightly adjusted parameters. Parameters chosen for the second model in **Pre-TrainedTest1** include Learning Rate=0.02, Epochs=3 (with 1227 steps per epoch), Batch Size=64 and Dropout=0.5.

1. Results. Describe the results of your experiments, using figures and tables wherever possible. Include all results (including all figures and tables) in the main body of the report, not in appendices. Provide an explanation of each figure and table that you include. Your discussions in this section will be the most important part of the report.

## 4. Results

| MODEL | PARAMETERS | RESULTS |
|---|---|---|
| **CNN** | Learning Rate: 0.02<br>Epochs: 2 (with 1227 steps per epoch)<br>Batch Size: 64<br>Dropout: 0.2<br>Performance Index:<br>*binary_crossentropy* | Metric: Accuracy<br>Result: 55% |
| **CNN** | Learning Rate: 0.001<br>Epochs: 3 (with 1227 steps per epoch)<br>Batch Size: 64<br>Dropout: 0.5<br>Performance Index:<br>*binary_crossentropy* | Metric: Accuracy<br>Result: 90% |
| **CNN** | Learning Rate: 0.02<br>Epochs: 3 (with 1227 steps per epoch)<br>Batch Size: 64<br>Dropout: 0.5<br>Performance Index:<br>*binary_crossentropy* | Metric: AUC |
| **MobileNet Pre-Trained** | Learning Rate: 0.02<br>Epochs: 1 (with 1227 steps per epoch)<br>Batch Size: 64<br>Dropout: 0.5<br>Performance Index:<br>*binary_crossentropy* | Metric: Accuracy<br>Result: 75% |
| **MobileNet Pre-Trained** | Learning Rate: 0.02<br>Epochs: 3 (with 1227 steps per epoch)<br>Batch Size: 64<br>Dropout: 0.5<br>Performance Index:<br>*binary_crossentropy* | Metric: Accuracy<br>Result: 92% |
| **MobileNet Pre-Trained** | Learning Rate: 0.02<br>Epochs: 3 (with 1227 steps per epoch)<br>Batch Size: 64<br>Dropout: 0.5<br>Performance Index:<br>*binary_crossentropy* | Metric: AUC |

2. Summary and conclusions. Summarize the results you obtained, explain what you have learned, and suggest improvements that could be made in the future.

## 5. Summary and Conclusion

The CNN and pre-trained network architectures utilised in this project

Limitations encountered while working include computational cost and long periods of training, which limited exploring different architectures and hyperparameters for optimal performance.

Future work that could lead to improvement in the models would be to enhance preprocessing techniques and research further on how to fine-tune hyperparameters to improve performance and classification of images. Building more complex models by adding  more layers and adjusting weights would also be considered, to help achieve better results.

## References

NIH Chest X-Ray Image Classification Dataset. Retrieved from https://nihcc.app.box.com/v/ChestXray-NIHCC

Culfaz, .F. (2018). Transfer Learning Using MobileNet and Keras. Retrieved from https://towardsdatascience.com/transfer-learning-using-mobilenet-and-keras-c75daf7ff299

Jefkine (2016). Backpropagation In Convolutional Neural Networks. Retrieved from https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Abdelfattah, .A. (2017). Image Classification using Deep Neural Networks — A beginner friendly approach using TensorFlow. Retrieved from https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4

Wang, .H. & Xia, .Y. ChestNet: A Deep Neural Network for Classification of Thoracic Diseases on Chest Radiography. Retrieved  from https://arxiv.org/pdf/1807.03058.pdf

Abiyev, .R.H. & Ma'aitah, .M. (2018).  Deep Convolutional Neural Networks for Chest Diseases Detection. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6093039/

Rosebrock,  .A. (2017). Image classification with Keras and Deep Learning. Retrieved from https://www.pyimagesearch.com/2017/12/11/image-classification-with-keras-and-deep-learning/

Image Preprocessing, Keras. Retrieved from https://keras.io/preprocessing/image/#flow_from_dataframe

Dertat, .A.(2017). Applied Deep Learning - Part 4: Convolutional Neural Networks. Retrieved from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Rajan, .U. (2018). Basics of Image Classification in Machine Learning Using Open Source Frameworks in IBM PowerAI (Part 2). Retrieved from https://towardsdatascience.com/basics-of-image-classification-in-machine-learning-using-open-source-frameworks-in-ibm-powerai-b4291dc40d25

Saha, .S. (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Retrieved from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

NIH Chest X-Ray Image classification, Kaggle. Retrieved from https://www.kaggle.com/rishi0497/chest-x-ray

Sagar, .A. (2019). Deep Learning for Detecting Pneumonia from X-ray Images. Retrieved from https://towardsdatascience.com/deep-learning-for-detecting-pneumonia-from-x-ray-images-fc9a3d9fdba8

Kabre, .S. (2020). Step by Step Solution of Deep Learning for Pneumonia Detection from Chest X-Ray Images. Retrieved from https://medium.com/@shubhamkabre/step-by-step-solution-of-deep-learning-for-pneumonia-detection-from-chest-x-ray-images-5bb272eb8548

Train Simple XRay CNN, Kaggle. Retrieved from https://www.kaggle.com/kmader/train-simple-xray-cnn