DETECTING ABNORMALITIES IN CHEST X-RAYS USING DEEP LEARNING

ADWOA BRAKO

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

abrako64@gwu.edu

Table of Contents

1. Introduction *4*

1.1. Motivation 4

1.2. Problem 5

1.3. Objective 5

Outline of shared Work 5

2. Background Information on Proposed Models *6*

2.1. Convolutional Neural Networks (CNN) 6

2.2. Transfer Learning Using MobileNet 7

3. Detailed Description of Work *8*

3.1. Data Preprocessing 8

3.2. CNN Model 8

3.4. Training of Models 10

4. Results *11*

5. Summary *13*

6. Conclusion *14*

7. Code *14*

REFERENCES *15*

ABSTRACT

Analyzing respiratory diseases or infections cannot be done merely by checking symptoms of patients. Diagnostic Radiology, being a branch of medicine that uses non-invasive imaging technology, plays an integral role in the diagnosis and treatment of diseases. Although various forms of imaging technology can be used to detect diseases, chest X-Rays are prominent in identifying respiratory diseases because of how cost-effective they are. As more technologically advanced techniques emerge, with Deep Learning playing an integral role in this advancement, disease detection can be done efficiently, with little to no misrepresentations and without delays.

Combining Deep Learning with big data, will aid in automatic detection and classification of diseases. As a result of this, image reading time for radiologists will be reduced and patients will be able to obtain timely diagnosis of diseases or infections. In this work, I propose Deep Learning techniques that seek to identify diseases related to chest X-ray scans. Classification techniques discussed include Convolutional Neural Networks (CNN) and Transfer Learning using MobileNet pre-trained models.

Keywords - ***MobileNet, CNN, classification, chest-X-rays, hyperparameter tuning, network***

1. Introduction

1.1. Motivation

In recent years, medical image analysis has evolved into major areas such as Computer Vision, Pattern Recognition, Machine & Deep Learning, among other areas. Related to the advancement of medical image analysis, Computer-Aided Diagnosis (CAD) aids in the automated detection and classification of diseases.

Diagnostic Radiology, being a branch of medicine that uses non-invasive imaging technology, plays an integral role in the diagnosis and treatment of diseases. One crucial aspect of a radiologist's work is producing error-free results in a timely manner.

Most infectious diseases related to the respiratory system rely on X-Ray scans and over 40% of these scans are related to chest infections or diseases. An integral aspect of CAD advancement is the automatic detection and classification of diseases, as this will maximize the accuracy and regularity of radiological diagnosis and minimize image reading time. Thus, the overall productivity and the quality of radiologists' duties will improve.

Deep Learning has seen tremendous growth and improvement in recent years and combining this technique with readily available big data, allows for classification of images based on associated classes assigned.

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.

Outline of shared Work

Russell Moncrief	**Adwoa Brako**
● Data Cleaning Dropping irrelevant columns ○ Adding new column with image paths to map images with respective labels ○ One-Hot encoding target labels ● Data preprocessing ○ Augmenting data(images) for training; preprocessing such as flipping, zooming, rotating and normalizing modified images	● Data Preprocessing ○ Augmenting data(images) for ○ training; preprocessing such as flipping, zooming, rotating and normalizing modified images. ● Training of proposed models ○ Models trained are of **CNN** and **MobileNet** architecture

2. Background Information on Proposed Models

For image classification, an increase in the number of layers can cause the number of parameters in a neural network to grow rapidly. This may in turn lead to heavy computational expense during training and tuning many of these parameters can be a burden. Considering these factors and limitations, I decided to work with networks that are capable of addressing these issues without compromising effectiveness or accuracy.

2.1. Convolutional Neural Networks (CNN)

Given CNN architecture as being fully connected feedforward neural networks, they are also extremely effective in reducing the number of parameters of spatial data, without diminishing the quality of the model(s). With a *convolutional layer* being the first part of a CNN, this layer is able to extract features of images before the images are passed on to other layers. The *pooling layer* scales down the amount of information generated by the convolutional layer for each feature, while maintaining the essential information. The *fully connected input layer* flattens the output generated by previous layers by turning them into a single vector as input for the next layer. The *fully connected layer* is responsible for applying weights over inputs generated and the *fully connected output layer* generates the final probabilities to determine the class(es) of images.

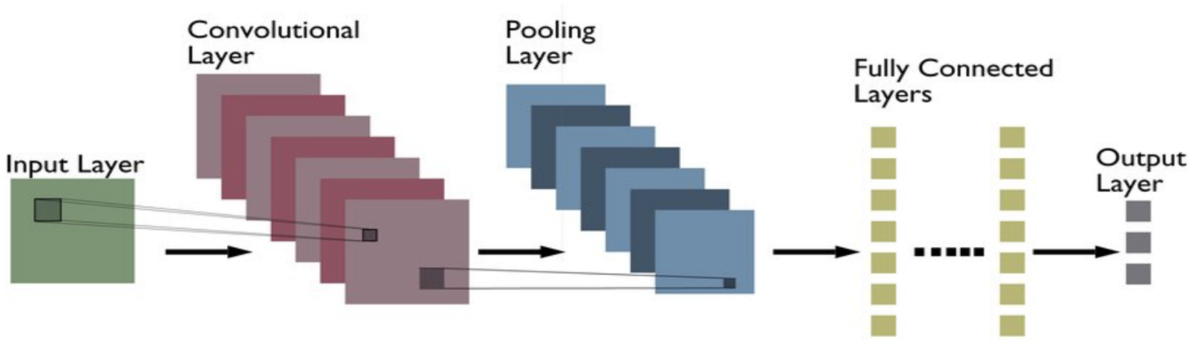

Fig 1: Representation of CNN architecture

2.2. Transfer Learning Using MobileNet

In transfer learning, a model developed for a specific task or problem is reused as a starting point for a different model to solve a related problem. This optimization improves performance and allows rapid progress when training or modelling a different task. To narrow down transfer learning, I used MobileNet as the base for the second model. MobileNet was chosen because it is based on a streamlined architecture that uses depth-wise separable convolutions to build light-weight deep neural networks, which reduces model size and complexity.

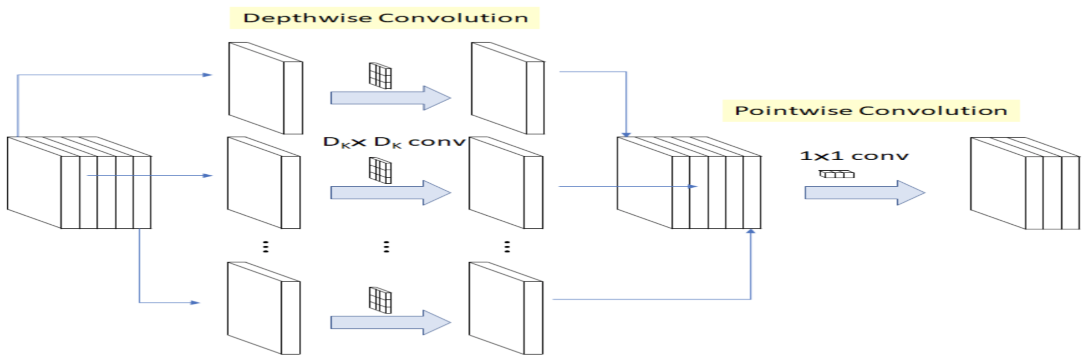

Fig 2: Representation of MobileNet architecture

3. Detailed Description of Work

3.1. Data Preprocessing

Before passing the input data (images) to the models, they had to be preprocessed to "clean" or prepare the images for training. For this project, the Keras framework was implemented for preprocessing and structuring the models' architecture. As part of this stage, the images, in grayscale, were resized to (224 x 224) pixels from the original dimension of (1024 x 1024) pixels. Keras *ImageDataGenerator* was used for preprocessing the images, to improve the data and suppress unwanted distortions. The *ImageDataGenerator* was responsible for augmenting the data, thus expanding the training dataset in order to improve performance and ability of the models to generalize.

The methods adopted in augmenting the data include flipping the images horizontally and vertically, rescaling and normalizing, shifting the height and width, as well as rotating the images. Performing data augmentation aided in enhancing image features that are important for training.

3.2. CNN Model

In connection with the general description of CNN models (found in **2.1.**), the detailed architecture I used to train the images is described in the diagram below.

Relu and Sigmoid activations were used between the layers and dropout of (0.2) or (0.5) was used to train the models.

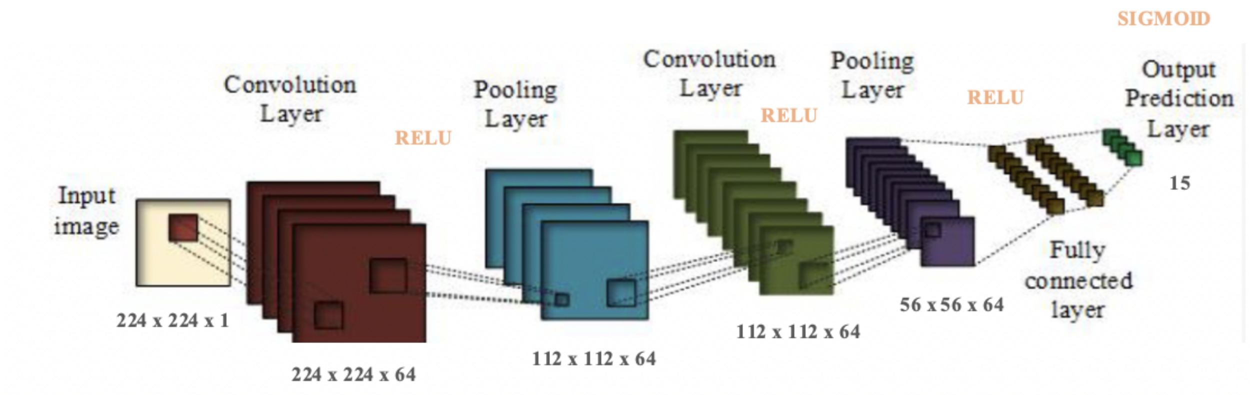

Fig 3: CNN model architecture

MaxPooling2D was chosen over AveragePooling2D because it selects the maximum pixel value in each batch, thus highlighting the most important features of that batch.

3.3. Pre-Trained Model Using MobileNet

Using MobileNet as the starter, I added *GlobalAveragePooling2D* with a dropout of (0.5). After this, I included a fully connected layer with 512 neurons and a dropout of (0.5). The next fully connected layer was the output layer, which had the target labels as input, with sigmoid transfer function. I included *ModelCheckpoint*, *EarlyStopping* and *ReduceLROnPlateau* callbacks in the model's architecture. Adding EarlyStopping was to help

stop training when performance on the validation set starts to degrade. This would ensure improvement in generalization of the network and address or reduce overfitting. To adjust the learning rate when the model's performance starts to stall or plateau, *ReduceLROnPlateau* was added to aid in fine-tuning the model weights.

3.4. Training of Models

The parameters adjusted during training of CNN and MobileNet include *Learning Rate*, *Epochs*, *Batch Size, Dropout.* Parameters chosen for the first CNN model include Learning Rate=0.02, Epochs=2 (with 1227 steps per epoch), Batch Size=64 and Dropout=0.2.

Parameters chosen for the second CNN model 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 the learning rate controls how quickly the model adapts to the problem, the learning rates used in the CNN model were selected to ensure that the model would not converge too quickly (large learning rate) or get stuck in the process, which is caused by extremely small learning rate

Parameters used to train the first MobileNet include Learning Rate=0.02, Epochs=1 (with 1227 steps per epoch), Batch Size=64 and Dropout=0.5.

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

In all CNN and pre-trained trained models, *Adam* was used as the optimizer and *binary_crossentropy* as a Performance Index.

4. Results

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

The table above shows the parameters I used to train the models and the results of the models with associated metrics used.

The first CNN model trained with associated parameters and *Accuracy* as the metric obtained an accuracy of 55%. In an effort to improve performance, I decided to adjust the hyperparameters for a second round of training. Reducing the learning rate to 0.001, increasing the dropout to (0.5) and adding 1 epoch (with 1227 steps per epoch) caused to model's accuracy to improve to about 90%. This indicates that the model needed more training to "learn" the data or images.

With respect to training the MobileNet pre-trained model, the set parameters with *Accuracy* as the metric resulted in an accuracy of 75%. To increase the accuracy of the pre-trained model, I made slight adjustments to the hyperparameters by increasing the number of epochs from 1 (with 1227 steps per epoch) to 3 (with 1227 steps per epoch), while maintaining all other parameters. Upon testing the model on the held-out test set, the model obtained an accuracy of 92%. This again was an indication that the model required more training on the images. I judged the performance of the third MobileNet pre-trained model using *Area Under Curve (AUC)*.

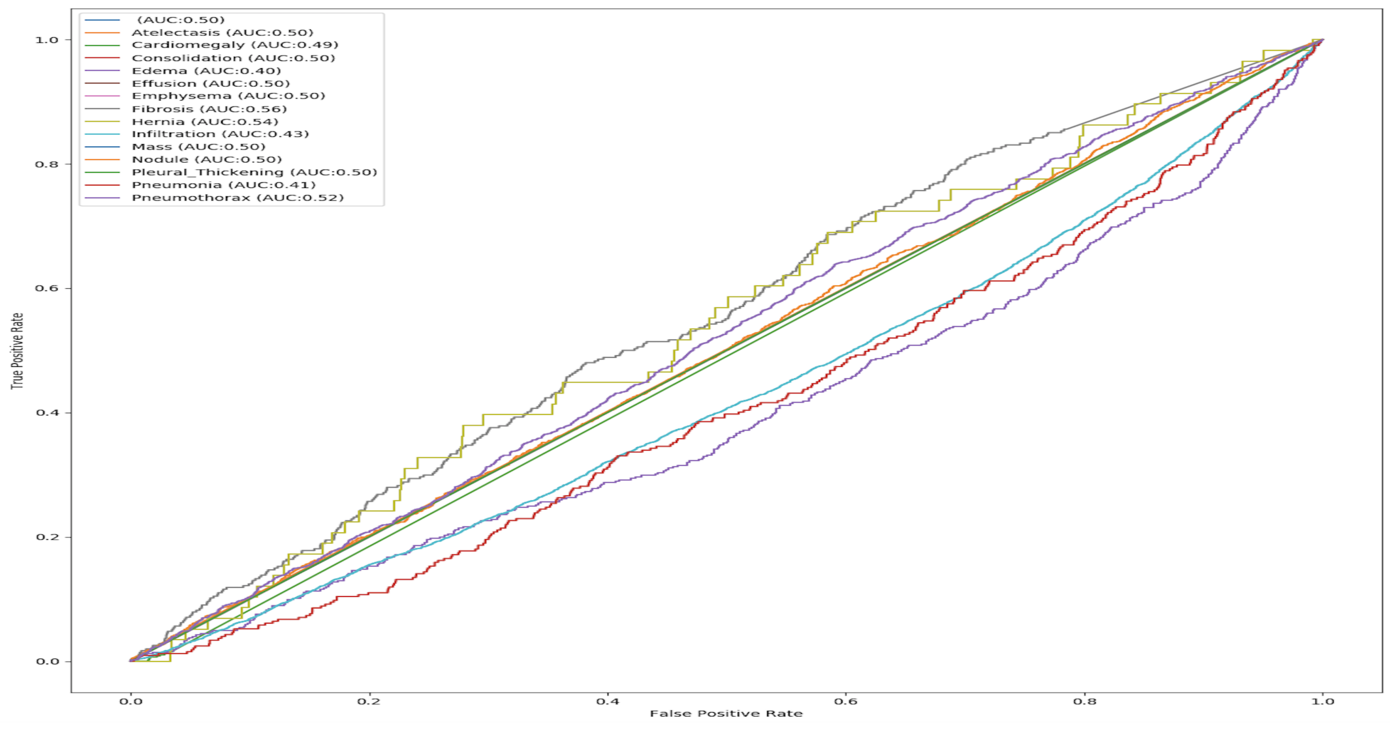

Fig4: AUC-ROC plot of third MobileNet pre-trained model

The AUC-ROC curve is ideal for classification problems, as it is able to distinguish between classes. A higher AUC indicates that the model distinguishes between classes well.

The AUC obtained for the various class labels is as follows: **Atelectasis (AUC 0.50), Cardiomegaly (AUC 0.49), consolidation (AUC 0.50), Edema (AUC 0.40), Effusion (AUC 0.50), Emphysema (AUC 0.50), Fibrosis (AUC 0.56), Hernia (AUC 0.54), Infiltration (AUC 0.43), Mass (AUC 0.50), Nodule (AUC 0.50), Pleural_Thickening (AUC 0.50), Pneumonia (AUC 0.41) and Pneumothorax (AUC 0.52).**

Hernia and Pneumothorax were the only two classes that had an AUC above 0.50. The AUC scores obtained for the classes indicate that the model did not generalize well on the test set, which implies that adjustments have to be made to the model and hyperparameters in order to improve performance. With further training, the models would be able to achieve a higher AUC for the class labels.

5. Summary

Based on the results produced after training, the last MobileNet model was selected as ideal, as it was able to obtain the actual performance of the model. However, wrong predictions were made, which led to false positives, indicating that the model was not totally accurate. I was unable to train another CNN model using AUC metric due to training time and would be able to pass a better judgement on performance of CNN and the pre-trained models in future work. Undertaking this project has broadened my knowledge in Deep Learning, precisely, Neural Networks for image classification. I have a better understanding of the math behind CNN model architectures and how critical it is to choose the right dimensions, coupled with knowing which transfer functions and metrics are ideal for a particular Neural Network problem.

6. Conclusion

Deep Learning is certainly capable of classifying images, as long as adequate cleaning and preprocessing techniques are utilized. Hyperparameter tuning also influences performance of models to a large extent and need to be carefully tuned for optimal effect.

Future work: To improve performance of models, measures to consider include (1) Adopting more preprocessing methods to create more modified versions of images. (2) Increase complexity of network architectures to improve performance. (3) Conduct further research to determine ways of better classifying images with effective hyperparameter tuning. Although the models are not fit for medical use yet, I believe they have potential for functional classification of chest X-rays, based on disease type and with more work, will be able to assist radiologists in diagnosing diseases within a timely manner.

7. Code

$$\frac{80-15}{80+12} * 100 = \frac{65}{92} * 100 = 70\%$$

70% of the codes I used for this analysis were obtained online.

REFERENCES

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

Rahmati, .T., Ismaili, .A. & Alimani, .S. (2018). Chest X-Rays Image Classification in Medical Image Analysis. Retrieved from https://www.researchgate.net/publication/330105218_CHEST_X-RAYS_IMAGE_CLASSIFICATION_IN_MEDICAL_IMAGE_ANALYSIS

MedLine Plus. Imaging and Radiology. Retrieved from: https://medlineplus.gov/ency/article/007451.htm

The Sequential Model API, Keras. Retrieved from: https://keras.io/models/sequential/

Convolutional Layers, Keras. Retrieved from https://keras.io/layers/convolutional/

Sivasamy, .J. & Subashini, .T.S. Classification and predictions of Lung Diseases from Chest X-Rays Using MobileNet. Retrieved from http://ijaema.com/gallery/70-ijaema-march-3557.pdf

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

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 X-ray CNN, Kaggle. Retrieved from https://www.kaggle.com/kmader/train-simple-xray-cnn