Transfer learning for detecting Pneumonia from Chest X-Rays

Vinay Ambre
University of Central Florida
vinay.ambre@knights.ucf.edu

Ganesh Aravind
University of Central Florida
ganesharavind@knights.ucf.edu

ABSTRACT

Pneumonia is considered as one of the fatal diseases which can cause severe consequences within a short span of time. More than a million adults are hospitalized with pneumonia in US alone. The best possible method to detect pneumonia currently in individuals is using chest x-ray images. In this work, we present a method using transfer learning to automatically detect pneumonia in chest x-rays. Transfer learning involves using pre-trained model weights to carry out the feature extraction functionality from the images without training the entire network. Our work include partially using pre-trained model weights of VGG19, DenseNet, MobileNet and InceptionResNet and partially using our stack of convolution layers for classifying the images.

KEYWORDS

Chest X-Rays, Neural Networks, Transfer Learning, Convolution, Transductive Learning

1 INTRODUCTION

About 450 million people worldwide get affected by pneumonia every year and about 4 million die from the disease. It is caused due to inflammatory response from the lung sacs called alveoli. As the germs like bacteria, fungi, virus reach the lunch the white blood cells acts against them causing inflammation. Hence, the alveoli get filled with pneuomonia fluid and this fluid causes symptoms like coughing, trouble in breathing, etc. Today, one of the most conventional technique of detecting pneumonia is using the analysis of Chest X-Rays [6]. The image produced on the metal surface is used to identify the infection if any. The radiologists analyze the white spots which depicts the high intensity of the fluid in the lungs. The white spots produced in the X-ray images look similar for healthy lungs as well as lungs in early stages of pneumonia. Hence, It sometimes becomes difficult to detect pneumonia in early stages using this technique. Moreover, the radiologists must have sufficiently trained eyes to tell the correct diagnosis. If there is an error by the human eye to differentiate between the categories then it may fall into either false negative or false positive which may ultimately have impacts on human body. As we can see, Figure 1 and Figure 10 displays different lungs belonging to normal and pneumonia category but the imperceptibility of the normal vs pneumonia images can also be seen which portrays that well trained eyes must be necessary to differentiate between the x-rays images.

Aside from the traditional method for detecting pneumonia via x-ray images, diagnosing the lower respiratory tract infection with the help of techniques like bronchoalveolar lavage are used. It is a medical procedure in which bronchoscope is passed in through mouth or nose and fluid is squirted into a tube. This method is also called lung biopsy in which tissues or cells are removed from the body to examine under the microscope. While the mentioned radiological and biopsy methods might be effective, we provide a deep learning approach using transfer learning to this pneumonia classification. There were two previous implementations that we reviewed which involved usage of deep learning for classification. One was the early diagnosis of pneumonia which was done on a public dataset on kaggle. Their work consisted of using residual network layer along with the convolution network layer for classification. There was also some image pre-processing on the chest x-rays before classification. The other model was trained on the

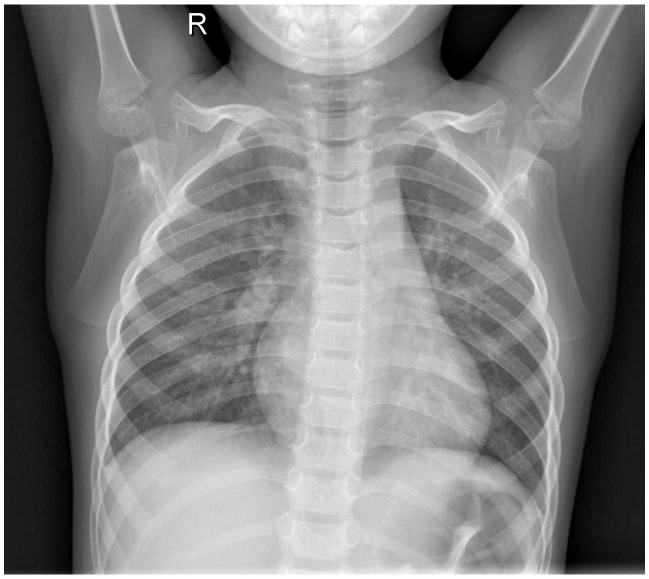

Figure 1: Normal (Healthy) Lungs

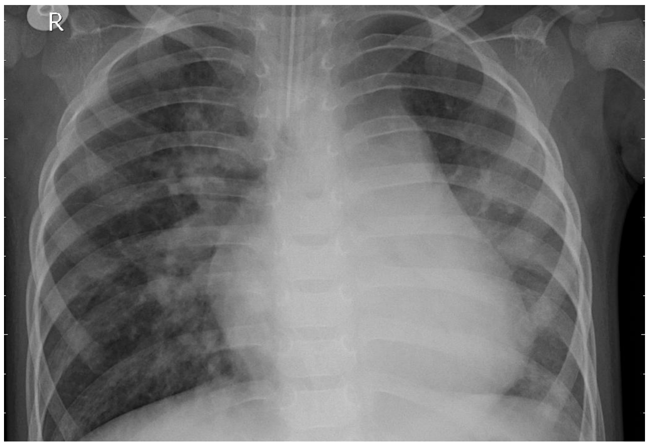

Figure 2: Pneumonia (Diseased) Lungs

CheXNet dataset. Their work included usage of a 121 dense layer convolution neural network.

2 TRANSFER LEARNING

2.1 Introduction

Transfer Learning involves using a knowledge base which is trained on some domain A for some task and using the same knowledge base for performing a task from another domain B[9]. This provides a optimised and improved performance for the prediction task. The principle behind this is that we don't need to train our model from scratch rather the pre-trained weights are capable of finding distinct features from the images at the initial layers and hence improving the speed. This is analogous to a human brain which sometimes leverages the knowledge from the past experiences. Example, while learning to drive a car, the brain can takes some previous assumptions from it's motorcycle riding experience[5]. For this purpose, pre-trained model weights which are trained on enormous datasets are used as source domain for knowledge base. It is not obvious that there is a benefit to using transfer learning until after the model has been developed and evaluated.

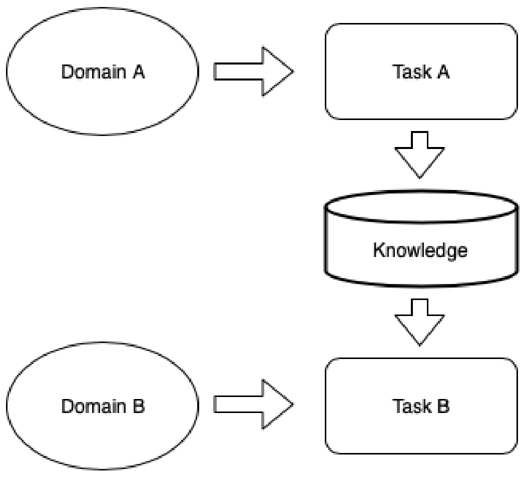

Figure 3: Transfer Learning

2.2 Transductive Learning

Transductive learning is a category in deep transfer learning in which the source domain and target domain are different but the task to performed is same. Our work categorizes into such type. We have considered 4 pre-trained models belonging to a domain different from our target task which is classifying whether the diagnosis is normal or pneumonia. But, the challenge that comes is the knowledge base from a different domain. This is overcome by freezing some layers from the source model and appending custom layers at the bottom so that the new customized model can learn the weights specific to the target domain. This new model is then trained on the target domain dataset so that weights are updated for that particular domain. This avoids training a lot of initial layers whose tasks is to identify image features thus, saving computational time. When the layers are frozen their weights don't get updated during the back propagation.

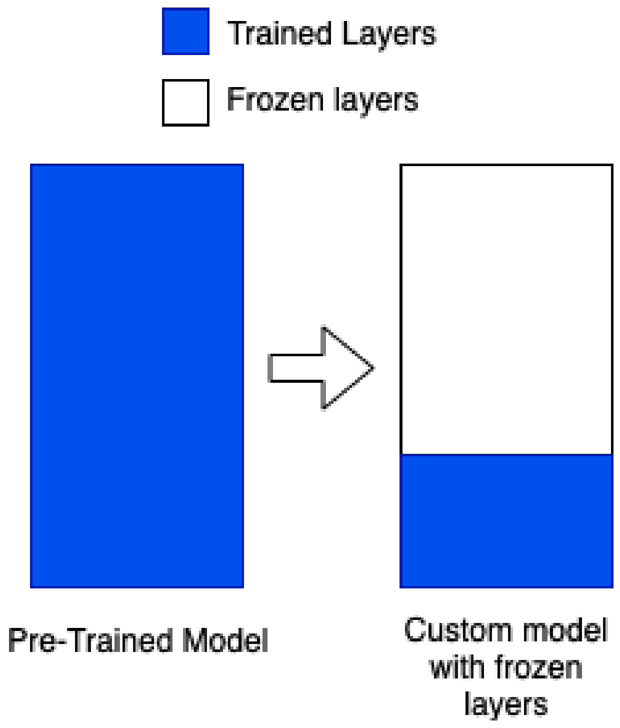

Figure 4: Frozen layers for Training

We have frozen various layers across the 4 models and appended them with our 4 dense custom layers. This custom layer helps in learning weights for our specific domain for the classification purpose. This helps in achieving our goal of using the pre-trained weights for identifying features and then predicting the diagnosis using the trained layers.

3 DATASET

The dataset that we used in our work is from the Mendeley research nertwork. The name of datast is Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification [1]. It consists of OCT and Chest X-Ray images. We considered the Chest X-Rays part for our work. The dataset was divided in two directories, training and testing images. The directories further contained two directories belonging to respective two classes.ie. Normal lungs and Pneumonia lungs.

The dataset contained 5234 training images and 626 testing images and was missing a validation dataset. We extracted some images from training and testing directories to create a validation dataset. Finally, the training dataset consisted of 5116 images, validation dataset consisted of 200 images and testing dataset consisted of 540 images. The training dataset is imbalanced just like most of the medical imaging dataset. Either the normal images will be more or the diseased. This dataset contained more pneumonia diagnosed images than the normal diagnosed images. The ratio was almost

3:1. The dimensions of the images also was varying and in a larger range then normal. The images were also in the RGB format.

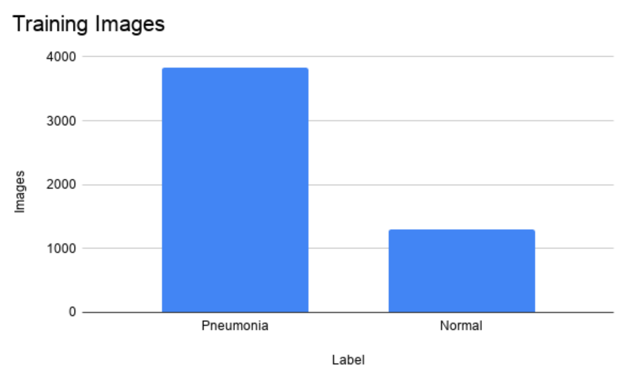

Figure 5: Distribution of Training images

4 IMPLEMENTATION

Our work includes use of Tensorflow Keras for implementing the neural network model. The 4 models that were chosen as base model were:-

(1) VGG
(2) InceptionResNet
(3) DenseNet
(4) MobileNet

4.1 Preprocessing and Hyperparameters

The dataset was kept in the same format as the source.i.e in the sub directories with the label names as the directory names. the preprocessing pipeline of the base models were used which are available in the Keras library. Input size of the images was reduced to 224×224. The images were taken as input for processing in batches of 30 with a class mode of categorical.

Adam optimizer was chosen as the optimizer with a varying learning rate. Learning rate was reduced if the model is not learning after epochs. binary crossentropy was used as the loss function with accuracy as the metric for compiling. The best model was saved among all the epochs. The training of model was done for 50 epochs with a batch size of 32. Since the training dataset was uneven.i.e normal images were thrice less then the pneumonia images, class weights were added in ratio 3:1.i.e. giving 3 times more weight to the normal images than the pneumonia diagnosed images during the training process. Few layers from the base layer were frozen for the purpose of learning new weights for our task. The top layer.i.e classification layer from the base layer was removed and replaced by a custom stack of layers. The base models imported were loaded with the weights that were learned when the models were trained on the 'imagenet' dataset.

4.2 Custom Dense Layers

The custom layers that we used at the bottom of the base model were a stack of dense layers. The role of these layers were to learn weights associated for classifying the images on two categories. Usually in transfer learning it is recommended to use a Maxpool layer with a output layer or a stak of dense layer with a output layer. We used a average pooling layer right after the last layer of the base model. This pooling layer was connected to a stack of dense layers as shown in figure 6. The dense layers have 'ReLu' activation function. The stack of fully connected dense layers was finally connected to an output layer with two nodes and 'softmax' activation. The two nodes depict the two categories of classification.

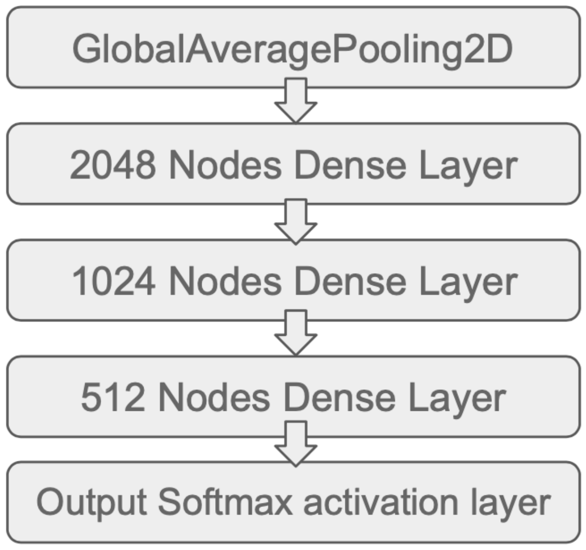

Figure 6: Custom Stack of layers

5 MODELS CONSIDERED

5.1 VGG

VGG stands for Visual geometry group at the Oxford university. This was developed for the 2014 imagnet challenge. It had 16 weight layer version and 19 weight layer version. The design of VGG is similar to Alex Net and Lenet. One thing we know is as we go deeper into the network the feature maps increase thereby making the network wider. They also have stuck to one filter size of 3×3. They have around 140 million parameters. This takes input of RGB and size 224×224. It consists of 3×3 convolutional layers followed by max pooling layers which reduces the size of feature maps to 112×112. Each of these convolutional layer has 64 feature maps. This is followed by another set of convolution and max pooling until it reaches the final 2 layers which consists of 3 convolutional layers followed again by a max pooling layer. This is later followed by fully connected layers and the output is a 1 among 1000 classification.

Having a closer look, we can understand that a layer in a VGG is basically a bunch of convolution layers. To be more precise, a sequence of layers is used as one layer. Between these convolution layers there exist no max pooling but non linearity still exists.

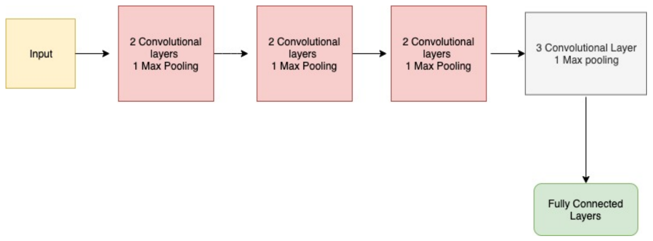
Input
2 Convolutional layers 1 Max Pooling
2 Convolutional layers 1 Max Pooling
2 Convolutional layers 1 Max Pooling
3 Convolutional Layer 1 Max pooling
Fully Connected Layers

Figure 7: VGG Architecture

When we look at a succession of 3×3 convolutions, we see the non linearity which leads to greater discrimination. When we have many non linear features we find the classes tend to become linearly separable. But here we incorporate it with the help of a succession of non linearities. Therefore this gives better discriminatory power in the network. This is a compelling advantage of VGG. Another interesting feature is in terms of the receptive field. When we look at the receptive field of the first layer, it has 3×3 receptive field. This gives rise to 64 feature maps. Looking at the second the filter kernel size is 3×3, but the receptive field on the original image is 5×5. Similarly when we look at the receptive field on the final 3 successive convolutions it is a 7×7 receptive field. The computational gain observed here is when we perform a succession of 3×3 convolutions, the number of computations involved to produce one element in the output feature map is much lesser compared to performing a 7×7 convolution. Also we have a larger receptive field on the input which is desirable in most cases.

Apart from these, the use of successive convolutions instead of using a filter with a larger receptive field saves the number of parameters. For instance if we use a 7×7 filter on 3 channels, the number of parameters would be 49×9. On the other hand if we use a succession of 3 convolutional layers to get the same receptive field with 3×3 filter sizes, then the number of parameters would be 27×9. This explains the decrease in parameters compared to other deeper networks[7].In our approach we have freezed 17 layers.

5.2 Inception Resnet

This is a hybrid architecture taking into the positive factors of Inception and Resnet Modules. When we have a look at Resnet and its working structure they had enormous layers making it a very deep networks. It also gave a better error rate than human. The gradient flow is one major issue when it comes to deep neural network. To overcome this the concept of skip connections were used. These skip connections are basically identity mappings. They provide an alternate path for the gradient flow thereby making training possible. For the ImageNet challenge, an 152 layer architecture was made. It also goes without saying that the number of parameters were also drastically increased in this kind of approach. In our approach we have freezed 509 layers. The principal behind residual networks is that a deeper network can be constructed from a shallow network by copying weights from the latter. When we come across any gaps we perform an identity mapping. The general idea to be kept in mind is when we train a network the training error should not be higher than that of the shallow network. Such a block is a residual

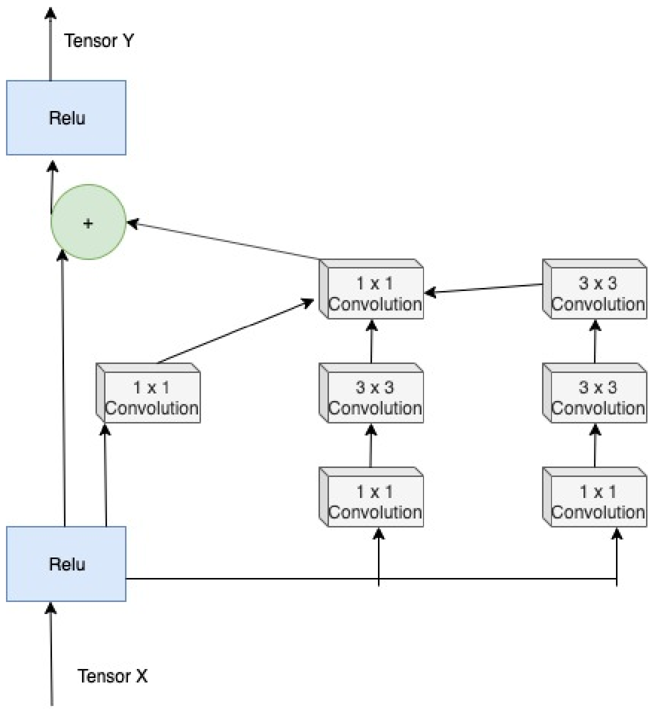

Figure 8: Inception Resnet Architecture

block. We combine one such residual block with a inception module to make use of the advantages of both resnet and inception.[8]

5.3 DenseNet

It is one of the most recent architecture used in the Imagenet classification challenges. It has shown great accuracies in classification despite having fewer parameters. One issue with resnets is as networks started getting deeper the gradients started vanishing. In resnets this was addressed by skipping a layer and adding feature maps from the previous layer. This was because one of the key observations made was by creating short paths from layers closer to the input to the later layers closer to the output, the gradient propagation is significantly improved. This naturally improves the classification too. This helps to train very deep networks, like even 100 layers.

Now according to the dense net architecture, it improves the gradient propagation by connecting all layers directly with each other. A typical CNN having n layers has n connection. However in densenet, since every layer is connected to every other layer, it has n(n+1)/2 connections.

Now on closely observing, we can see if there exists 10 layers the 10th layer would get feature maps from all the preceding 9 layers. Now if each layer outputs 256 feature maps, this could arise in explosion of feature maps. So to overcome this problem we fix the number of output maps for each layer. We also create dense blocks. Each of these dense blocks has prefixed number of layers inside them. Among these layers the feature maps are shared between them. The output from such a dense block is given to a transition layer. This transition layer uses a bottleneck concept like 1 × 1 convolution followed by a max pooling to reduce the size of the feature maps.

One interesting advantage of denseness are the parameter efficiency. Since we have fixed the outputs for the layers, very few kernels are learnt per layer. Another compelling advantage with densenets is implicit deep supervision and feature reuse. When we consider inception net they use auxiliary cost functions using feature maps from the intermediate layers. This improves the learning in the sense, it has the features learnt to be discriminative to improve the auxiliary cost function. There are many approaches to it. One such approach may have which takes feature maps from the intermediate layers and feed it to SVM as an input. It then does the classification and then that error is backpropagated. However in car of a densenet where the feature maps are concatenated from the preceding layers, they have a direct connection with the error function. They might be separated by a few dense blocks, but still have access to the error function, thereby improving training and learning discriminative features[3].In our approach we have freezed 313 layers.

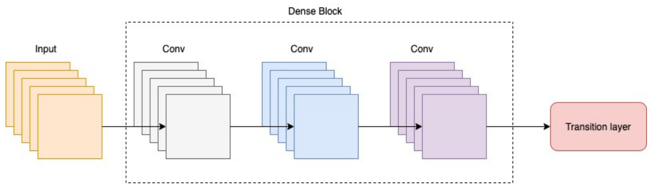

Figure 9: Densenet Architecture

5.4 MobileNet

The field of Mobile applications and Mobile Embedded devices have some constraints like memory and power constraints. Considering these the first version of MobileNet was built. This has shown greater accuracy and computational power was also significantly reduced. This uses depthwise separable convolution. The idea behind this is to reduce the computational strains on the machine.

In standard convolution, the application of filters across all input channels and the combination of these values are done on a single step. Depthwise convolution on the other hand breaks these into 2 steps. It initially performs the depth wise convolution or the filtering stage and later the pointless convolution that does the combining stage. Depthwise convolution applies convolution to a single input channel at a time. This is different from the standard convolution that applies convolution to all channels at the same time. Let us take a inout volume F having a shape Df × Df × M. For depth wise convolution we use filters or kernels k of shape Dk × Dk × 1. Here Dk is the width and height of the square kernel and it has a depth of 1, because this convolution is only applied to a channel unlike standard convolution which is applied throughout the entire depth. Since we apply one kernel to a single input channel, we require m kernels as mentioned earlier over the entire input volume F. For each of these m convolutions we end up with m number of Dg × Dg × 1 outputs. Now on stacking these outputs together, we have an output volume of G which is of shape Dg × Dg × M. Here ends us the first phase. This is then succeeded by a pointless convolution stage.

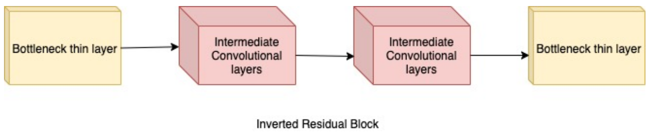
Bottleneck thin layer
Intermediate Convolutional layers
Intermediate Convolutional layers
Bottleneck thin layer
Inverted Residual Block

Figure 10: MobileNet Architecture

The pointless convolution stage performs the linear combination of each of these layers. The input is the volume of shape Dg × Dg × M. The filter would have a shape 1 × 1 × M. This is basically a 1 × 1 convolution operation over all M layers. The output will thus have the same input width and height as the input Dg × Dg for each filter. Assuming that we want to use N such filters, the output volume becomes Dg × Dg × N.

On computing complexities it has been noticed that standard convolution has 9 times more operations to perform than a depth wise convolution. This is core concept behind MobileNet. It can be considered as neural network architecture that focuses majorly to minimise latency of smaller scale networks, so thereby making computer vision applications run easy on mobile devices. It uses the depth wise convolution in its 28 layer architecture.This compels the use of depth wise convolution in the MobileNetV2[2]. In our approach we have freezed 91 layers.

6 EXPERIMENTAL RESULTS AND CONCLUSIONS

On training with 5234 training images and testing the results on the 626 images over 50 epochs, we observe the results as shown in figure 11 and figure 12. On trying with the 4 proposed pre trained models, VGG19 comes up with the best accuracy of 84.44. The False positive rate for the same can also be viewed in the ROC curve.

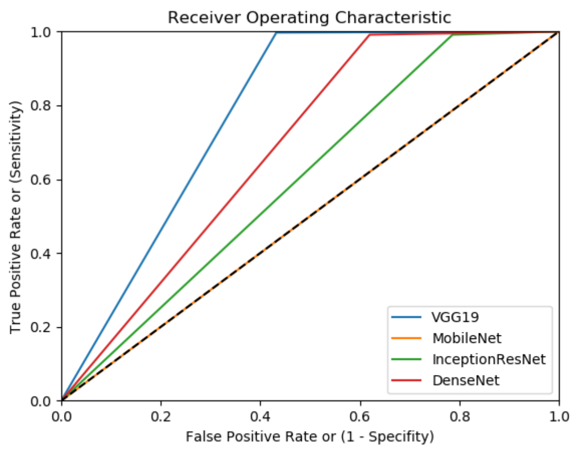

Figure 11: ROC

The Roc curve is constructed by plotting the sensitivity against 1 minus specificity for each possible cutoff. This curve always starts in the lower left corner when sensitivity is zero and specificity is 1. As we can see from figure 11, the false positive rate is the least for VGG19. This makes the precision way better than the other models. The results are significantly better compared to the other models.

On comparing with papers CheXnet: Radiologists-Level Pneumonia Detection on Chest X-Rays with deep Learning[4] and Early diagnosis of pneumonia with deep learning[6] they have achieved a result of 76.80 and 78.73 respectively. VGG19 used in our approach has come up with a accuracy of 84.44 thereby significantly improving classification of pneumonia.

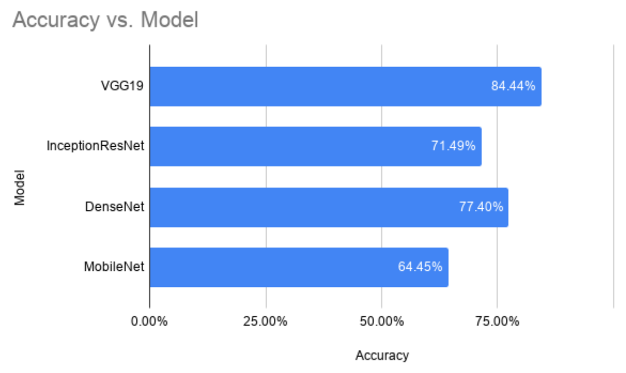

Figure 12: Accuracy for each model

REFERENCES

[1] Daniel Kermany amnd Kang Zhang and Michael Goldbaum. 2018. Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification.

[2] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:cs.CV/1704.04861

[3] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2016. Densely Connected Convolutional Networks. arXiv:cs.CV/1608.06993

[4] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. 2017. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. arXiv:cs.CV/1711.05225

[5] Dipanjan (DJ) Sarkar. 2018. A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning.

[6] Can Jozef Saul, Deniz Yagmur Urey, and Can Doruk Taktakoglu. 2019. Early Diagnosis of Pneumonia with Deep Learning. arXiv:cs.CV/1904.00937

[7] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:cs.CV/1409.1556

[8] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv:cs.CV/1602.07261

[9] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. A Survey on Deep Transfer Learning. arXiv:cs.LG/1808.01974