NIH Chest X-ray Classification of Thorax Diseases

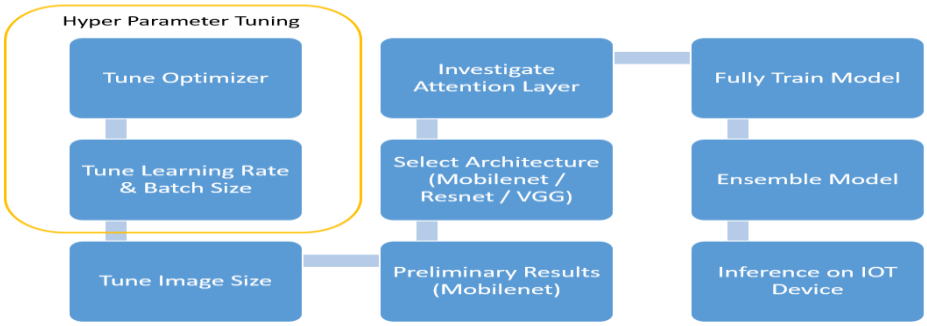
Varun Khatri

{khatri.va@northeastern.edu}

***Abstract*—The chest X-ray is one of the most widely used radiological tests for lung disease screening and diagnosis. There are a lot of X-rays Radiology reports are required for imaging tests gathered and kept in the Picture Archiving and Communication Systems of many contemporary hospitals (PACS). On the other hand, it remains unclear how a hospital-sized knowledge database including essential imaging informatics may be leveraged to help data-hungry deep learning paradigms in the development of really large-scale, high-precision computer-aided diagnostic (CAD) systems.**

I. INTRODUCTION

The goal of this project is to employ a deep neural net architecture to categorize the NIH chest x-ray dataset. We improve our model in little increments. We adjust hyperparameters first, then try out other architectures.
The National Institutes of Health (NIH) compiled this database, which comprises over 100,000 anonymised chest x-ray pictures from over 30,000 people. The information is based on NLP analysis of radiological records and may contain places where diagnoses are less certain. As a simplistic assumption, we'll suppose that the dataset is accurate in diagnoses because of its size. One of the problems with this situation is that the data lacks a "diagnostic confidence" feature. In addition to a chest X-ray, diagnosis involves patient presentation and history. Further, some physician's diagnoses will not be agreed upon by others. Therefore, it is likely that some of the images are mislabeled.
The figure below shows the roadmap to create model.



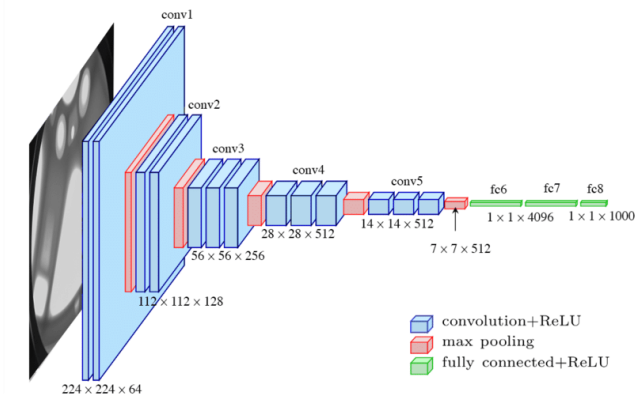
II. TECHNOLOGIES

Convolutional Neural Networks:

CNNs (Convolutional Neural Networks) are a form of neural network that is frequently used to solve image processing difficulties. The convolutional layer, which connects neurons to pixels only in their receptive areas rather than every single pixel, distinguishes them from standard neural networks. As a consequence, features may be extracted while dimensionality is greatly reduced. CNN began when David Hubel and Torsten Wiesel, two neurophysiologists, released groundbreaking research on the response of a cat's visual cortical neurons to stimuli. Researchers took some of the findings as motivation to add a convolution layer to neural networks shortly after. Yann LeCun, Leon Bottou, Yosuha Bengio, and Patrick Haffner invented the LeNet-5 method for identifying handwritten numbers in the 1990s. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning system that can take an input picture, assign relevance (learnable weights and biases) to various aspects/objects in the image, and distinguish between them. When compared to other classification methods, the amount of pre-processing required by a ConvNet is significantly less. While basic approaches need hand-engineering of filters, ConvNets can learn these filters/characteristics with enough training. The design of a ConvNet is inspired by the organization of the Visual Cortex and is akin to the connection

pattern of Neurons in the Human Brain. Individual neurons can only respond to stimuli in a small area of the visual field called the Receptive Field. A number of similar fields can be stacked on top of each other to span the full visual field.

A. *Vgg16*



The convnets are fed a fixed-size 224 by 224 RGB picture during training. The only pre-processing done here is subtracting the mean RGB value derived on the training set from each pixel. The picture is processed through a stack of convolutional (conv.) layers, where filters with a very narrow receptive field are utilized, such as 3 3 (which is the lowest size to capture the notions of left/right, up/down, and center and has the same effective receptive field as one 7 x 7). It's more complex, with more non-linearities and fewer parameters. 1 1 convolution filters, which may be thought of as a linear modification of the input channels (followed by non-linearity), are also used in one of the configurations.

For 3 x 3 convolutional layers, the convolution stride and spatial padding of the conv. layer input are both set to 1 pixel, ensuring that the spatial resolution is kept after convolution. Spatial pooling is aided by five max-pooling layers that follow parts of the convolutional layers. Max-pooling is done with stride 2 across a 2x2 pixel frame.

Following a stack of convolutional layers (which have varying depths in different designs), there are three Fully-Connected (FC) layers: the first two have 4096 channels apiece, while the third performs 1000-way ILSVRC classification and so comprises 1000 channels (one for each class). The soft-max layer is the last layer. In all networks, the completely linked levels are configured in the same way.

B. *MobileNet*

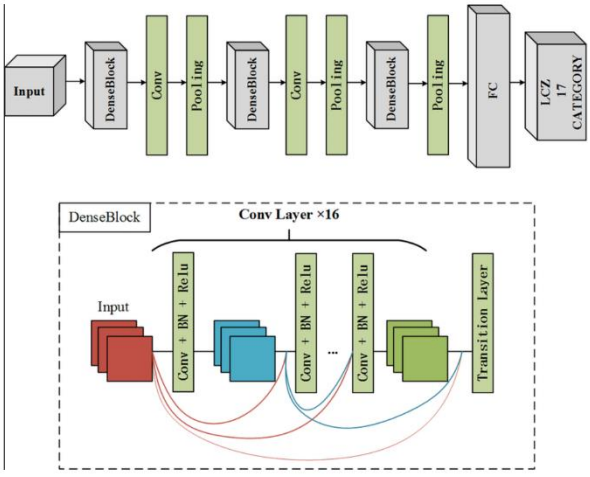
Table 1. MobileNet Body Architecture

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
5× 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

Except for the first layer, mobilenet is based on depthwise separable convolutions. A complete convolutional layer is the first layer. Batch normalization and ReLU non-linearity are applied to all layers. The last layer, on the other hand, is a completely linked layer with no non-linearity that feeds the softmax for classification. Strided convolution is used for both depthwise convolution and the first fully convolutional layer in down sampling.

When depthwise and pointwise convolution are considered independent layers, the total number of layers for mobilenet is 28.

C. *DenseNet*



Each layer in DenseNet receives extra input from all preceding levels and sends its own feature-maps to all future layers. The term "concatenation" is used. Each layer receives "collective knowledge" from the levels above it.

Because each layer gets feature maps from all preceding layers, the network can be thinner and more compact, resulting in fewer channels. The extra number of channels for each layer is the growth rate k.

As a result, it has better computational and memory efficiency. The notion of concatenation during forward propagation is depicted in the diagram below.

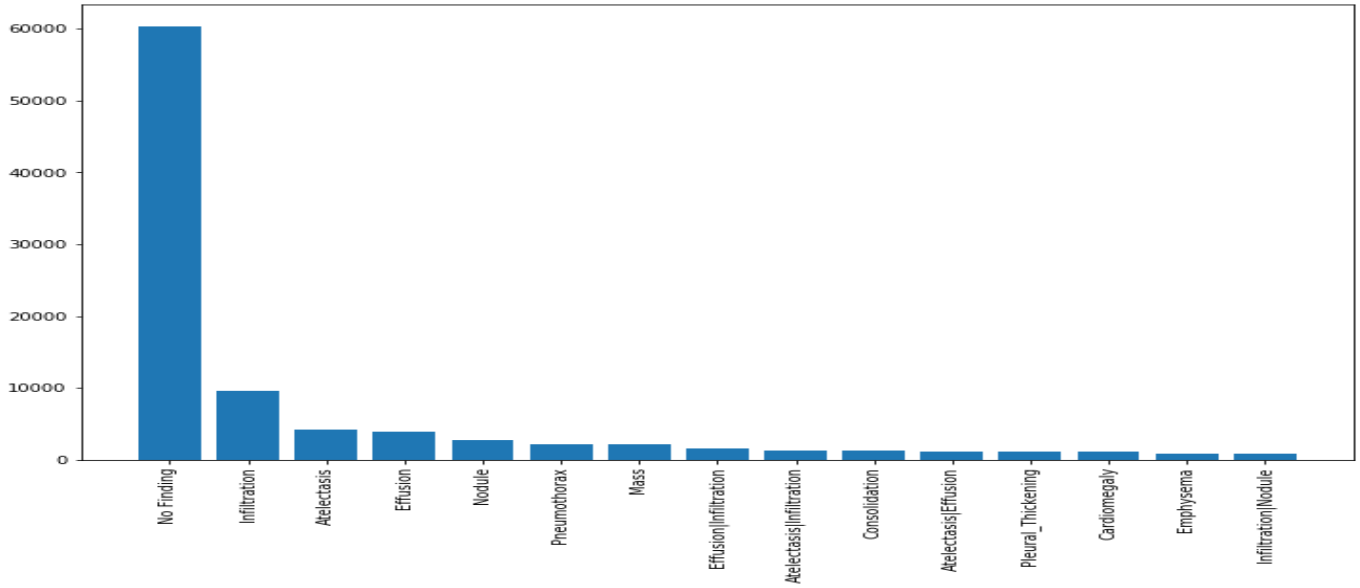
The transition layers between two contiguous dense blocks are 1x1 Conv followed by 2x2 average pooling.

Within the dense block, feature map sizes are uniform, allowing them to be readily concatenated. A global average pooling is conducted at the conclusion of the last dense block, and then a softmax classifier is

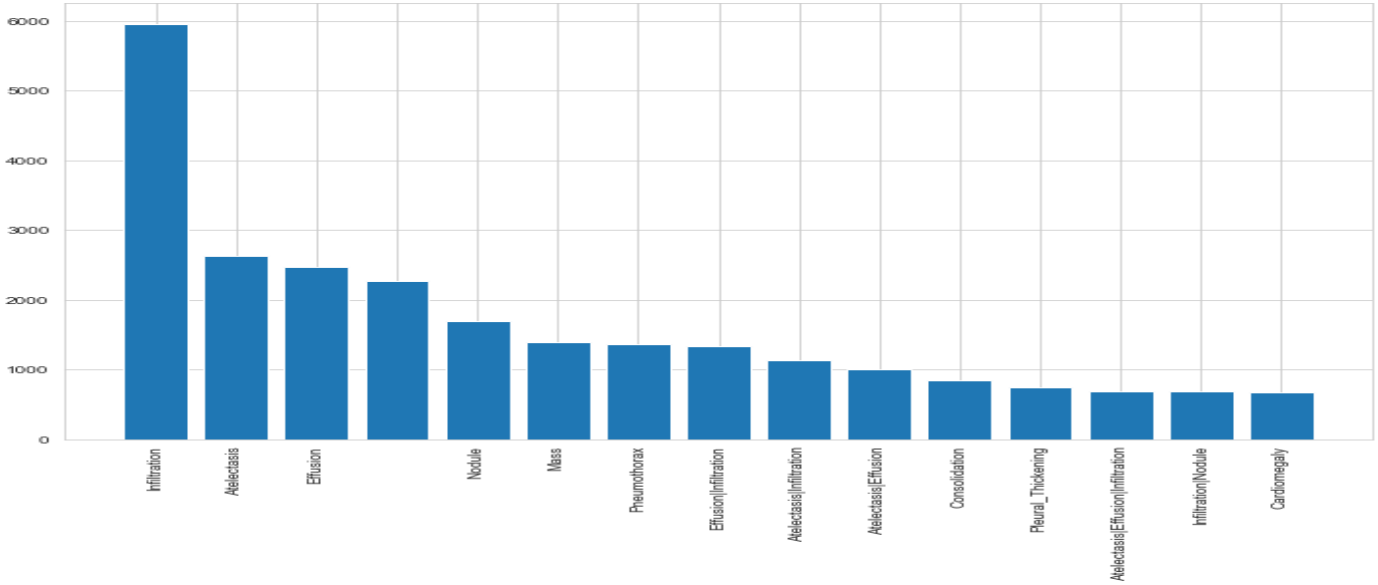
added. The error signal can be conveyed more directly to earlier tiers. As previous layers can get direct supervision from the final classification layer, this is a form of implicit deep supervision.

III. DATA PROCESSING

Data processing is a method of evaluating a dataset in a CSV file using a graph to extract elements impacting the prediction of outcomes and to better understand the dataset in order to choose the appropriate Machine Learning Algorithm for the job. Matplotlib, seaborn, pandas, NumPy, and sklearn are some of the libraries we utilize for data processing. CSV files and photos are included in the dataset (X-ray). There are 11 columns in a CSV file. Patient ID, age, gender, view position, image width, image height, pixel space x, and pixel space y are all included in the picture index. The first step is to create a column name path, which contains the locations of pictures in the column image index. When training our CNN models, we require a route column to direct the position of pictures. Then we look at how many illnesses labels exist in our database. There are 14 disease labels (Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural thickening, Cardiomegaly, Nodule Mass, Hernia) and one label of "No findings"(indicate no disease in X-ray image). We do One-hot encoding after gathering information about categorical variables in the dataset: we create a new column (also known as a dummy variable) for each category of a feature to indicate whether a particular row matches to that category. Then, using the sklearn package, we partition the dataset into training and test groups, and using the path column, we split and resize photos to fit in CNN models.



We also reject any unusual diagnoses, meaning those with less than 1000 occurrences, because neural networks rely on huge training sets. The distribution of diagnoses that resulted is depicted in the diagram below.



RESULT

Cnn models were constructed using TensorFlow and the Keras package once the data was processed. To begin, we used image training data to train the CNN models. We utilized our test data of pictures to anticipate the results and calculate the model's accuracy score after training the CNN models. VGG16, MobileNet, and DenseNet were the CNN models employed for this challenge. Then, based on their design, they built Convlutanol layers, MaxPooling layers, and thick layers. model. The predict function was used to forecast test outcomes. The sklearn library's 'roc score' was utilized to determine the model's accuracy score. There are several approaches to increase the CNN model's accuracy score.

On first run of train dataset in this three CNN models results were not so good got accuracy score of 40% in Densenet, 53.67% in MobileNet and 50.24% in VGG16. then some changes were made in Model like changing the number of Epochs and number of steps in epochs. Even made changes in CNN models (reduce of model parameters in dense/maxpool layers and output of last dense layer)to tune it accordingly. this steps were performed many times and after good analyses the best possible values were decided for each model, for all model shape of output of last dense layer was 15. and epochs, step size for VGG16:20,100; DenseNet: 30,60; MobileNet: 20:150. using this results accuracy score achievable on test dataset was: VGG16:67.890%; DenseNet:72.35%; MobileNet:65.67%.

CONCLUSION

As per the roc_acu_score score of all three CNN models, DenseNet performed the best still it is only 72.35% accurate but it's more in comparison to the other two models. Running such Big CNN models on normal pc GPU is very hard it took me a minimum of 4 hr to train each model every time I change the values to test and improve new accuracy. Higher accuracy can be achieved using these models for NIH chest X-ray Dataset using powerful GPU and more time. Due to time constrain it was not possible to achieve the max potential of these three CNN models. But training with tuning on different values for these models' accuracy of 90%+ can be achieved. But this project helped me understand how CNN works and how we can create CNN using Tensorflow and Keras in python. What are the input values for the CNN model and make a change/ add layers in the CNN model. It provided me with a deep understanding of how CNN works and how to build my own CNN model wising python libraries.