Single Thoracic Disease Prediction on Chest X-Ray

Aditya Vikram Singh

Deepti Gupta

Manisha Sharma

Musarath Jahan Rahamathullah

Pallava Arasu Pari

Roozbeh Sadeghian

ANLY 535

November 22, 2019

Single Thoracic Disease Prediction on Chest X-Ray

Introduction

Motivation

The use of diagnostic imaging has increased dramatically in recent years. A substantial number are chest X-Rays used to diagnose a plethora of conditions. Chest X-ray exam is one of the most frequent and cost-effective medical imaging examinations. These diagnoses are still primarily done by radiologists manually poring over each scan, with no automated triaging or assistance. This study will help the first-pass specialist examination by predicting the disease categories with efficiency greater than the manual analysis. Our aim is to use deep learning to predict thorax disease categories using chest X-rays to decrease manual intervention during x-ray study and concluding better and faster results.

Objective

Our problem can be casted as a multiclass image classification problem with 15 different labels on NIH Chest X-Ray14 dataset and predicting the single most probable thorax disease categories with the greater efficiency than the manual examination.

Situation and Challenges

The first of many problems that were confronted to work with this dataset was its huge data and system incompatibility to deal with it. The data includes 112,120 images with size 1024 x 1024 which summed up to 42 GB. Leave alone model processing with multiple batches and epochs, the data preprocessing and exploratory data analysis took iteratively huge efforts. To deal with it, we chose Google Colab Notebooks for their tremendously efficient GPU processors and sliced the data into smaller section of 4 GB and worked on 5606 images. Since this study is only focusing on image data and not considering the patient's metadata, each of the chest x-rays is independent and identically distributed, there should not be bias in the data.

The second issue was to decide the number of output classes in the multiclass output. The data has 14 disease classes namely; Cardiomegaly, Emphysema, Effusion, Hernia, Nodule, Pneumothorax, Atelectasis, Pleural Thickening, Mass, Edema, Consolidation, Infiltration, Fibrosis, Pneumonia and a 15th output class as No Finding. The first action plan was to keep 15 output labels for 14 diseases and "No Finding" state, by using one hot encoding, the multi-class labels can be converted for the respective diseases which was further converted into 14 output classes. Even then, since 50% data has output "No Finding" and 20% of the rest of the 50% data has class output Infiltration, the data may not enough to train the model for all the diseases and the model may not perform very well.

The next big issue was to decide on the correct loss function whether to use Binary Cross entropy since the data is one hot encoded and the value would be either 0 (not the disease) or 1 (has the disease) for that specific output disease class, or to use Categorical Cross Entropy since the data has multiclass multilabel output. It was concluded that Binary Cross Entropy would be a valid and better choice after brainstorming and web research.

Why Analytics would Help?

The NIH Chest X-ray dataset is extracted from the clinical PACS database at the National Institutes of Health Clinical Center and consists of ~60% of all frontal chest x-rays in the hospital. Therefore, this dataset is expected to be significantly more representative to the real patient population distributions and realistic clinical diagnosis challenges, than any previous chest x-ray datasets. Of course, the size of this dataset, in terms of the total numbers of images and thorax disease frequencies, would better facilitate deep neural network training (Bar, Diamant, Wolf, & Greenspan, 2015).

Clinical diagnosis of chest X-ray can be challenging, and sometimes believed to be harder than diagnosis via chest CT imaging. Even some promising work has been reported in the past, and especially in recent deep learning work on Tuberculosis (TB) classification. To achieve clinically relevant computer-aided detection and diagnosis (CAD) in real world medical sites on all data settings of chest X-rays is still very difficult, if not impossible when only several thousands of images are employed for study. We aim to use deep learning to predict thorax disease categories using chest x-rays with greater than first-pass specialist accuracy.

Related Work

Background

“Thoracic diseases are very serious health problems that plague a large number of people. Chest X- ray is currently one of the most popular methods to diagnose thoracic diseases, playing an important role in the healthcare workflow” (Mao, Yao, Pan, Luo, & Zeng, 2019, p. 2). Detecting the thoracic diseases early and correctly can help clinicians to improve patient treatment effectively. Chest X-ray (CXR), also known as chest radiograph, is a projection radiograph of the chest and is used to diagnose conditions affecting the chest, its contents, and nearby structures. Due to its affordable price and quick turnaround, CXR is currently one of the most popular radiological examinations to diagnose thoracic diseases. Currently, reading CXR and giving an accurate diagnosis rely on expert knowledge and medical experience of radiologists. With the increasing amount of CXR images, to handle the heavy and tedious workload of reading the CXR images with subtle texture changes, even the most experienced expert may be prone to make mistakes. Therefore, it is important to develop a Computer-Aided

Diagnosis (CAD) system to automatically detect different types of thoracic diseases by reading patients' CXR images. This is all founded on a well-designed transfer of human knowledge to machine intelligence (Kumar, Grewal, & Srivastava, 2018; Mao et al., 2019; Yan, Yao, Li, Xu, & Huang, 2018).

Related Research and their Work

There has been a significant amount of research performed to support clinical decision making and diagnostics with the help of machine learning algorithms and deep neural networks. Advancements and new researches in medical imaging and visualization provide improved mechanisms of analysis and diagnostics to the specialists to have a high resolution and in-depth insights (Rampasek & Goldenberg, 2018).

Before CXR dataset was released, there were also some works about thoracic disease classification on some relatively small dataset. Bar et al. (2015) applied a pre-trained Decaf Convolutional Neural Network (CNN) (Convolutional neural network, 2019) model to classify 8 thoracic diseases on a dataset of 433 images. Since Wang et al. (2019) released the datasets CXR, there has been an increasing amount of research on CXR analysis using deep neural networks. Wang et al. (2019) also evaluated the performance of four classic deep learning architectures (i.e., AlexNet (AlexNet, 2019), VGGNet (Simonyan & Zisserman, 2015), GoogLeNet (Szegedy et al., 2015) and ResNet (He, Zhang, Ren, & Sun, 2016)) to diagnose 14 thoracic diseases from CXR images. Kumar et al. (2018) explored suitable loss functions to train a convolutional neural network (CNN) from scratch and presented a boosted cascaded CNN for multi-label CXR classification. Chen et al. (2019) presented a deep Hierarchical Multi-Label Classification (HMLC) approach to model conditional probability on detecting 14 abnormality labels from the PLCO dataset, which comprises 198,000 manually annotated CXRs. Mao et al.

(2019) have worked on Deep Generative Classifier (DGC) on Chest X-Ray14 dataset and calculated the Area Under ROC (AUC) for each disease.

Relation of our Work with the Others

Commented [DG1]: To be covered in the end

We applied deep learning techniques to the CXR dataset and have presented our findings in the report. Considering the popularity of CNNs in working with images and their overall effectiveness, we used a simple architecture of CNNs for our purpose. We have also applied other methods such as LSTMs and Dense-Nets for the sake of research on how these models would perform. Taking light from other's research we experimented with different cost functions and settled with Binary Cross-Entropy for our multi-label problem.

Data

Dataset Overview

Chest X-ray (CXR) dataset from the National Institutes of Health (NIH) Clinical Center comprises 112,120 frontal-view X-ray images of 30,805 unique patients with the text-mined fourteen disease image labels (where each image can have multi-labels), mined from the associated radiological reports using natural language processing (Baltruschat, Nickisch, Grass, Knopp, & Saalbach, 2019). The input X-ray images are of size 1024×1024 and png format, along with the metadata on age, gender and number of visits to the hospital. Fourteen common thoracic pathologies include Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural thickening, Cardiomegaly, Nodule, Mass and Hernia. The text-mined disease labels are expected to have an accuracy >90% (Chen et al., 2019).

Dataset Limitation

The limitation with data is, (a) the image labels are NLP extracted so there would be some erroneous labels, but the NLP labelling accuracy is estimated to be >90%, (b) there are very limited numbers of disease region bounding boxes, and (c) chest x-ray radiology reports are not anticipated to be publicly shared. Parties who use this public dataset are encouraged to share their “updated” image labels and/or new bounding boxes in their own studied later, maybe through manual annotation.

Dataset Pre-Processing

After data pre-processing which included removing the outliers in age columns and checking for bad data, exploratory data analysis was performed to understand the length and breadth of the data. Data was also subset in order to perform analysis on multiple approaches of modeling and implementing CNN, Dense-Nets and LSTMs on the multi-class multi-labelled Chest X-ray dataset.

Commented [DG2]: Rephrase Later, add more on both the approaches

Add on findings from VGG-LSTM

Exploratory Data Analysis

Out of 112,120 data, 60,361 had outcome “No Finding”, 9,547 had “Infiltration”, 4,215 were diagnosed with “Atelectasis” and the rest of the 12 output classes were even lower in numbers which is an indication of unbalanced data.

From below picture, Figure 1A. depicts the frequencies of different diseases and it shows the diseases - Infiltration, Effusion, Atelectasis and Nodule are significant among the patients and, Hernia, Pneumonia, Fibrosis and Edema have the lowest number of patients. Figure 1B. describes the gender gap in the thorax diseases with men being more prone to be suffering from them then women. Figure 1C. depicts age as one of the most important contributing factors to the disease. The age bracket of 40-65 years has a sharp incline in the number of diseased patients, and from age 20 till 60 years, the disease curve is higher than the normal curve but after 60 years, there is an equal proportion of diseased and normal patients, which implies that the probability of thorax diseases increases after the age of 60. From the histogram of Age in Figure 1D., most of the patient's lie in the age bracket of 40-75 years of age.

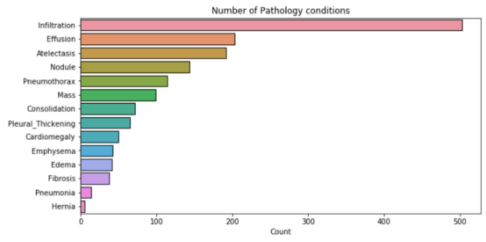


Figure A: Frequencies of different Pathology Conditions

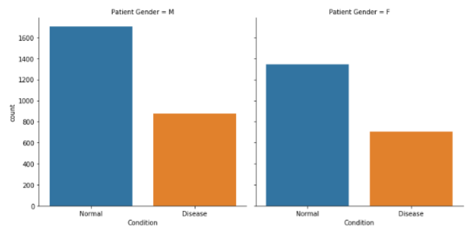


Figure B: Gender-wise Distribution of the Condition (Diseased or not)

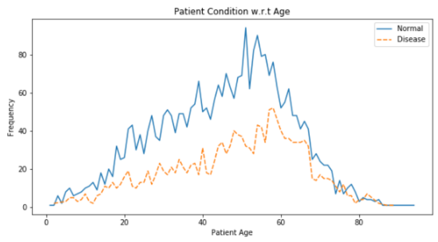


Figure C: Frequencies of Patient's Condition with Age

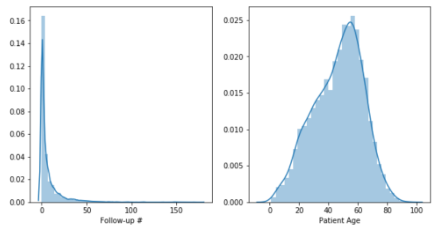


Figure D: Histogram of Patient's Follow up and Age

Figure 1. Data Trend Analysis

The below data analysis describes the patterns of the different diseases w.r.t age and gender. Thorax diseases like Emphysema, Pneumothorax, Mass, Consolidation, Edema and Pneumonia have a clear trend of being more active in females than males. Also, diseases like Effusion, Nodule, Atelectasis, Pleural Thickening, Mass and Infiltration are clearly more significant and concentrated in the age group of 40-65 years.

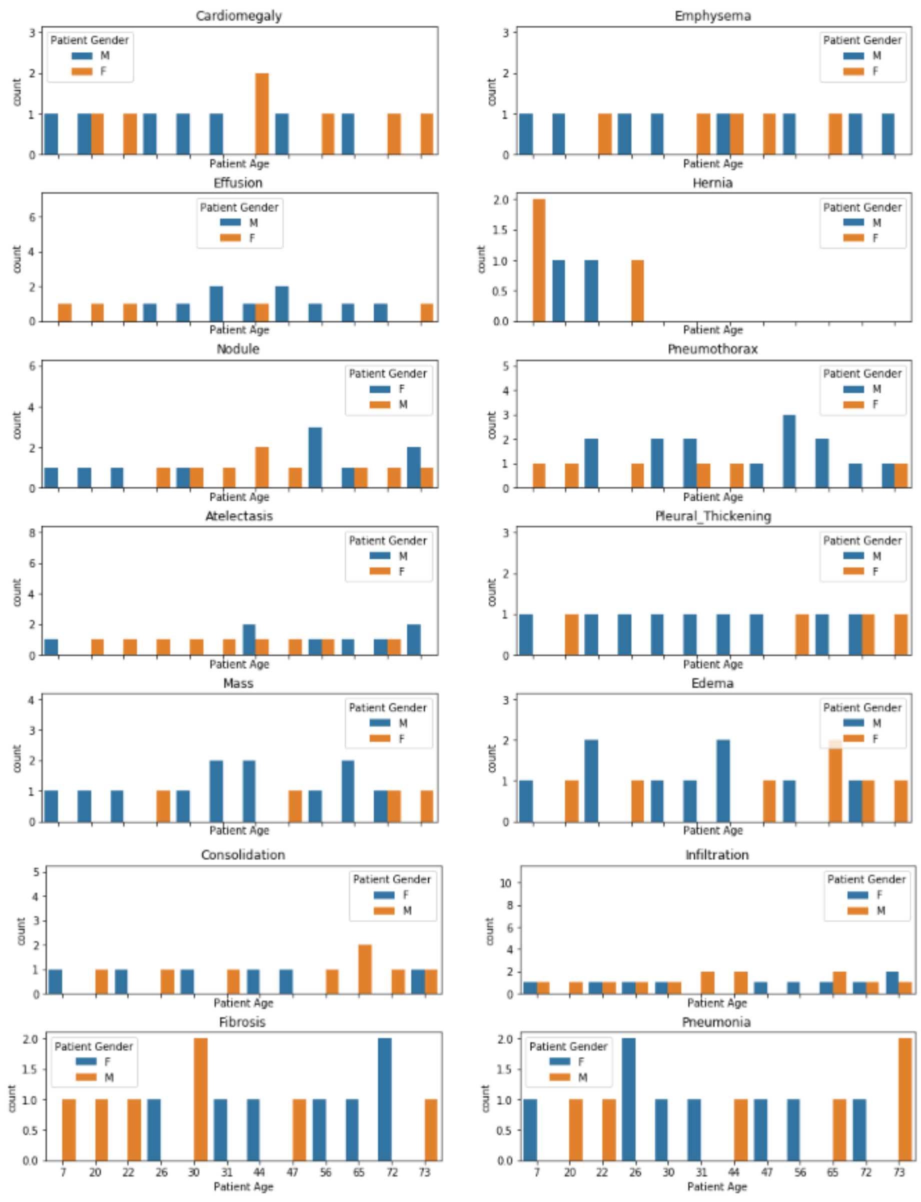


Figure 2. Relation of Diseases with Patient's Age

Technical Approach

Detailed Description of Complete Process

This study includes two approaches for NIH chest x-ray classification. In the first approach, a 14GB data sample comprised of 34,154 images is factored into multilabel and multiclass target variable for classification. Multilabel classification is implemented to recognize all the diseases that occur together in single x-ray as there could be more than one clinical diagnosis. Post data cleaning, 14 columns indicating the presence of each disease are added as one-hot encoders to sample cleaned dataset along with the array of images name and disease-indicator columns (one-hot encoded) is created. Disease-indicator columns depict the presence of diseases in the corresponding image. A train-test split of the dataset is performed in the ratio of 80:20. Data Augmentation is used to avoid overfitting and enable our models to generalize better and to constantly change our training dataset. Fit generator function is used to accept the batch of data randomly created using user-defined function (generate_nn_batch) and perform backpropagation and update the weights in our model. This approach yields good accuracy although the probability of finding a class was very less. Hence, we explored the sample dataset with other models with the second approach mentioned below.

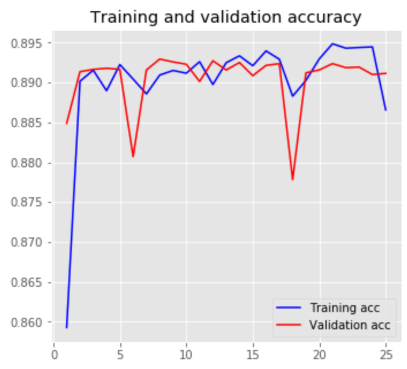
In the second approach, 4GB sample data comprising 5606 images is considered. In this approach, data is further resampled to consider images with only “one pathology condition”. As mentioned in previous approach, one hot encoding is performed for the 14 diseases creating columns with 1/0 values for each of the diseases. Images are resized to 128x128 and they along with their labels are converted into an array for classification and further, saved into a compressed format for reusability. Train_test_split method is used to split the dataset into a train and test set in an 80:20 ratio. Different approaches such as Deep neural network, Convolutional

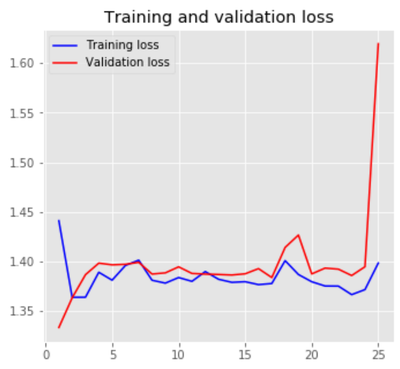
Neural/ Network with/without drop out, batch normalization, data augmentation and Recurrent Neural network/LSTM, VGG16 are implemented and then compared. In order to avoid overfitting, drop out and early stopping are also used. The predictions are cross validated against the actual diseases using CNN with batch normalization.

Implementation, Test and Evaluation

Dense Neural Net

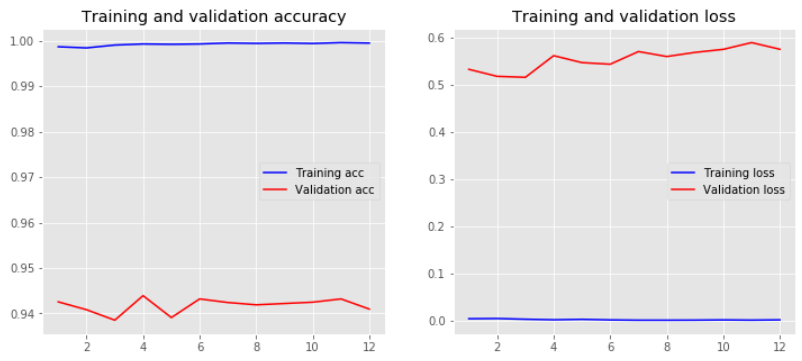
The dense neural network architecture is presented in Appendix 1a. In this neural network, three dense layers (2 hidden layers) are used which reflected approx. 89% accuracy on the training and test data.



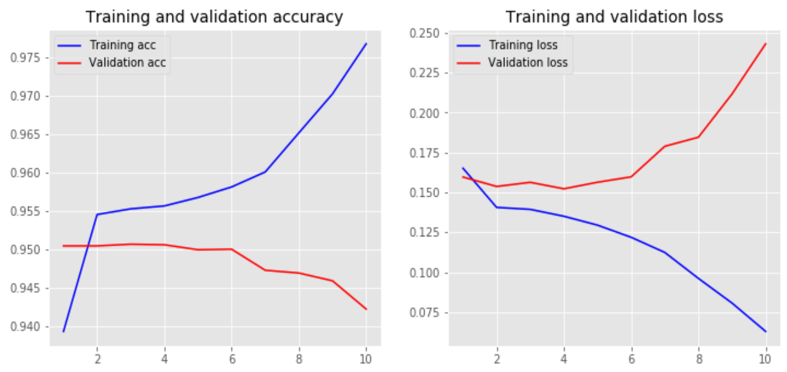


Convolutional Neural Net

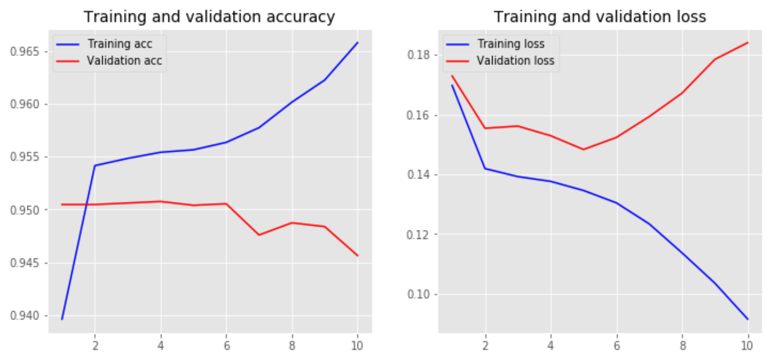
The convolutional neural network architecture is presented in Appendix 1.b. In this neural network, 4 layers of CNNs (3 hidden layers) with 3x3 filters, and 2 dense layers for compilation and flattening are used. The final fully connected layer is activated with “sigmoid” activation, and the entire model is compiled with Adam optimizer (with default settings) and “binary cross-entropy” loss function. After running 12 epochs with the CNN model, a max accuracy was found to be ≈ 100% for training data and 94% for validation data.



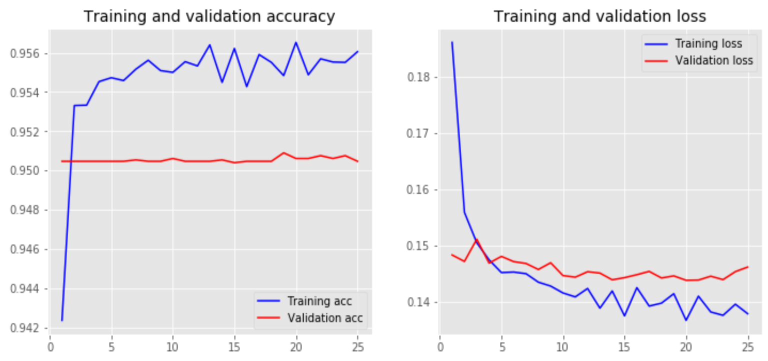
To see if over fitting exists, dropouts are implemented after every two layers on the CNN model, with values 0.2 and 0.1 consecutively. From the training results shown below, it is analyzed that validation accuracy starts off better at 95% than training without dropout, but then starts declining with consecutive epochs.



To further try improving the model accuracy, batch normalization is added after every two CNN layers. The batch normalization should ensure gradient descent sanity with small loss function values. From the results below, we see that there's not much improvement in validation loss after training through some epochs; the validation loss starts increasing but at a slower rate than before batch normalization.



Next, data augmentation is tried to supplement the lack of data, to analyze any improvements in accuracy. The data augmentation function augments data pixels, image size and position to generation “new” datasets. As a result, an improvement is noted in the stability of validation accuracy through epochs, but training accuracy is compromised which probably needs better training.

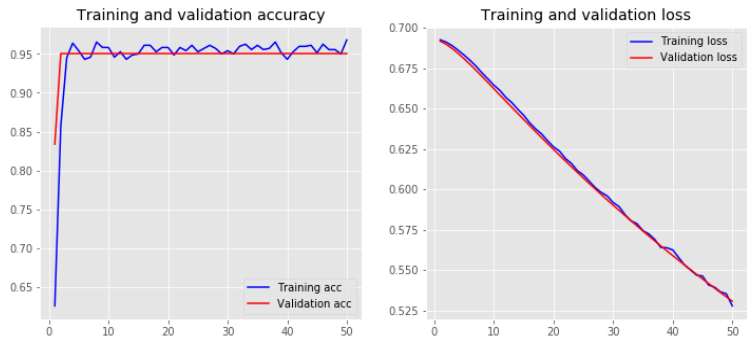


CNN with LSTM

The architecture is presented in Appendix 1c. In this design, four convolution layers passed their output to LSTM. An LSTM network requires the input to be in the shape samples, time steps, features; here samples are the number of data points, time steps represents the number of steps dependent on time, and features refers to the number of variables. The LSTM layer has

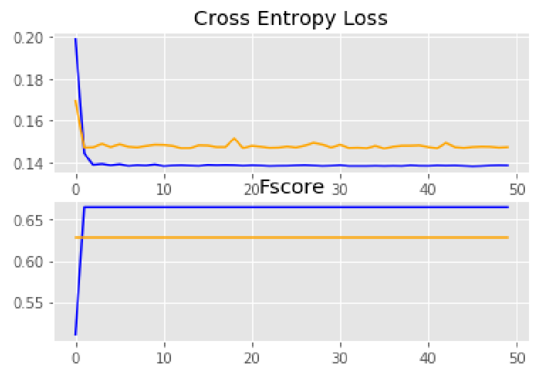
200 memory units and it returns sequences, so that the preceding layers receive sequences and not randomly scattered data. A dropout of 20% is applied at each LSTM layer to avoid the overfitting of the model. Further, a fully connected layer is used with a “softmax” activation to four layers of deep neural network and a “sigmoid” activation as the last layer. The SGD optimizer is applied with learning rate of 0.01, momentum of 0.9 and decay 0.000001. The model is fitted over 50 epochs with a batch size of 1000 on a data generated with 10% rotation, zoom (0.1), randomly shift images vertically and horizontally by 0.1. The prediction from the model gives probability of each disease from an x-ray image.

From the learning curve, it is analyzed that the loss is decreasing with an increase in epochs. The validation accuracy is 95.05%. But the main problem that lies here as the model predicts only Class 10 i.e. “No Finding” cases with f1-score of 0.77.

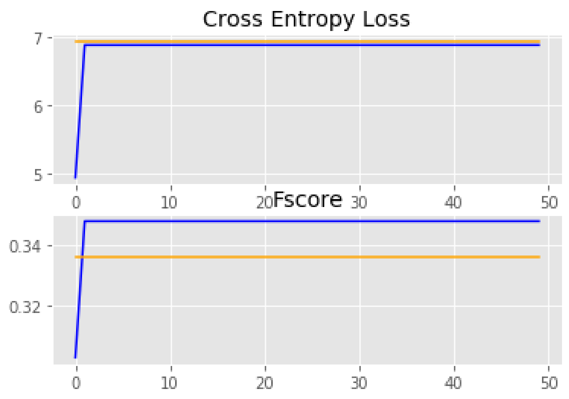


VGG

The architecture for this model is given in appendix 1.d. A pretrained VGG16 neural network has been used here. Our model has hidden layers with 49,152 neurons and an output layer with 15 neurons (since we have 15 disease conditions to predict). The learning curves for Vgg16 are as follows with loss as 0.147 and f-score as 0.629.



From F1 score of 0.63, it can be noted that model predicts 60% of classes correctly and the cases represented using this model belong to one class which has high number of records. Tuning the parameters in VGG16 didn't help either as the model score derailed with loss as 6.933 and f-score as 0.336.



Results and Conclusions

A. Below is the summary of different models with their respective accuracy and ranks -

Table 1. Assessing the performance of the different models

Name of the Model	A) Accuracy	B) Rank
CNN	95%	1
Dense Layer Network	89%	2
CNN with LSTM	95.05% with lower scope of data	Low
VGG net	60%	Low

The CNN model performs best with 95% accuracy. The Dense model is found to be performing decently compared to the rest of the models other than the CNN. Most surprisingly, the VGG-net performed worse compared to CNN, even though being a deeper model. This probably can be related to the lack of training data and the poor performance of the training of the number of weights in VGG net. CNN with LSTM, since according to expectations, the LSTM did not have much to learn with the “non-time series” natured batch data.

Potential Future Work

Better Cost Functions: Like the F-Score used in VGG training, other cost function could have been utilized like the Dice Score. The code for Dice Score is given in Appendix 2.a. The F Score is based on a function of Recall and Precision which are presented in the Figure 3 below.

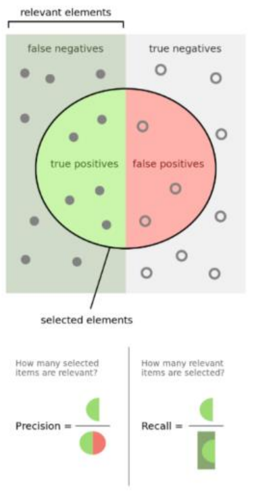


Figure 3. Venn Diagram of Recall and Precision

The Dice function is kind of loss based on modified intersection over union, where the intersection is divided by sum squares of the areas covered by truth and predictions.

The Intersection over Union function works based on the formula (Figure 4):

(Common area of intersecting pixels of predicted and original masks (C))/

(Total area of predicted and original masks (A+B)-Common area of intersecting pixels of predicted and original masks(C))

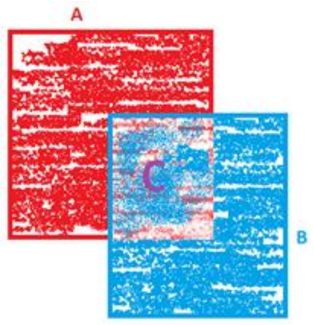


Figure 4. Intersection over Union (IoU) pictorial representation

Training with more data: A larger dataset could have been used to train the models. To achieve this, creating batch generation functions that can randomly select several images for training based on the batch size would have helped. The generator function providing a tensor of the expected output for each image and augment the images for training would have been the next step. A sample code for generator function is provided in Appendix 2b.

References

Baltruschat, I. M., Nickisch, H., Grass, M., Knopp, T., & Saalbach, A. (2019). Comparison of Deep Learning Approaches for Multi-Label Chest X-Ray Classification. *Scientific Reports*, *9*(1). https://doi.org/10.1038/s41598-019-42294-8

Bar, Y., Diamant, I., Wolf, L., & Greenspan, H. (2015). Deep learning with non-medical training used for chest pathology identification. *Medical Imaging 2015: Computer-Aided Diagnosis*. https://doi.org/10.1117/12.2083124

Chen, H., Miao, S., Xu, D., Hager, G. D., Harrison, A. P., & Com, A. P. H. (2019). Deep Hierarchical Multi-label Classification of Chest X-ray Images. In *Proceedings of Machine Learning Research*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2016.90

Kumar, P., Grewal, M., & Srivastava, M. M. (2018). Boosted Cascaded Convnets for Multilabel Classification of Thoracic Diseases in Chest Radiographs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-93000-8_62

Mao, C., Yao, L., Pan, Y., Luo, Y., & Zeng, Z. (2019). Deep Generative Classifiers for Thoracic Disease Diagnosis with Chest X-ray Images. *Proceedings - 2018 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018*. https://doi.org/10.1109/BIBM.2018.8621107

Rampasek, L., & Goldenberg, A. (2018). Learning from Everyday Images Enables Expert-like Diagnosis of Retinal Diseases. *Cell*. https://doi.org/10.1016/j.cell.2018.02.013

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2015.7298594

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., & Summers, R. M. (2019). ChestX-ray: Hospital-Scale Chest X-ray Database and Benchmarks on Weakly Supervised Classification and Localization of Common Thorax Diseases. In *Advances in Computer Vision and Pattern Recognition*. https://doi.org/10.1007/978-3-030-13969-8_18

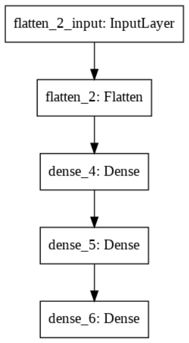
Yan, C., Yao, J., Li, R., Xu, Z., & Huang, J. (2018). Weakly Supervised Deep Learning for Thoracic Disease Classification and Localization on Chest X-rays. *ACM-BCB 2018 - Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. https://doi.org/10.1145/3233547.3233573

AlexNet. (2019, October 18). Retrieved from wikipedia: https://en.wikipedia.org/wiki/AlexNet

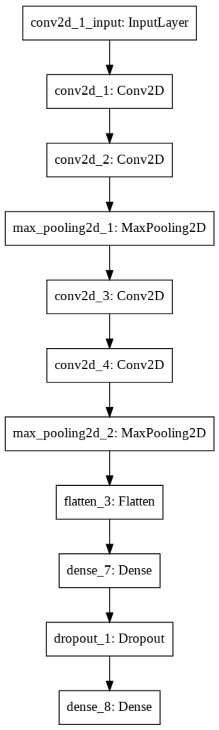
Convolutional neural network. (2019, November 26). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Convolutional_neural_network

Appendix 1

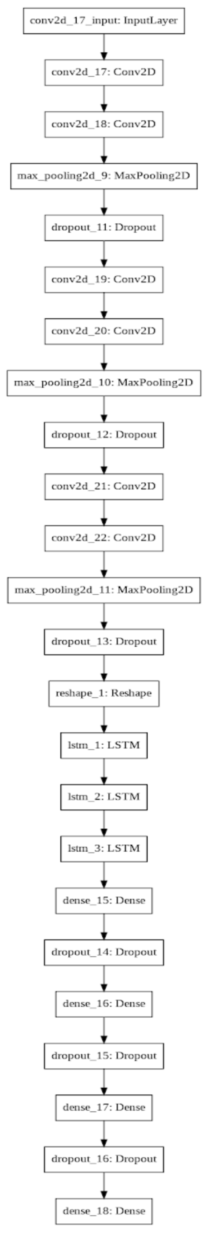
1a. Dense Neural Network Architecture



1b. Simple Convolutional Neural Network



1c. LSTM


conv2d_17_input: InputLayer
conv2d_17: Conv2D
conv2d_18: Conv2D
max_pooling2d_9: MaxPooling2D
dropout_11: Dropout
conv2d_19: Conv2D
conv2d_20: Conv2D
max_pooling2d_10: MaxPooling2D
dropout_12: Dropout
conv2d_21: Conv2D
conv2d_22: Conv2D
max_pooling2d_11: MaxPooling2D
dropout_13: Dropout
reshape_1: Reshape
lstm_1: LSTM
lstm_2: LSTM
lstm_3: LSTM
dense_15: Dense
dropout_14: Dropout
dense_16: Dense
dropout_15: Dropout
dense_17: Dense
dropout_16: Dropout
dense_18: Dense

1d. VGG 16

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_3 (InputLayer)         (None, 128, 128, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 128, 128, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 128, 128, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 64, 64, 64)        0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 64, 64, 128)       73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 64, 64, 128)       147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 32, 32, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 32, 32, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 32, 32, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 32, 32, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 16, 16, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 8, 8, 512)         0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 8192)              0         
_________________________________________________________________
dense_9 (Dense)              (None, 128)               1048704   
_________________________________________________________________
dense_10 (Dense)             (None, 15)                1935      
=================================================================
Total params: 15,765,327
Trainable params: 1,050,639
Non-trainable params: 14,714,688
```

Appendix 2

2a. Dice Loss

```
from keras import backend as K
def dice_coef(y_true, y_pred, smooth=1):
    """
    Dice = (2*|X & Y|)/ (|X|+ |Y|)
         =  2*sum(|A*B|)/(sum(A^2)+sum(B^2))
    ref: https://arxiv.org/pdf/1606.04797v1.pdf
    """
    intersection = K.sum(K.abs(y_true * y_pred), axis=-1)
    return (2. * intersection + smooth) / (K.sum(K.square(y_true),-
1) + K.sum(K.square(y_pred),-1) + smooth)
def dice_coef_loss(y_true, y_pred):
    return 1-dice_coef(y_true, y_pred)
```

2b. Batch Generator

```
#Assign 14 element vector for disease information using this
function.
def mask_all(key):
    masks= np.zeros((14))
    for i in range(0,34151,1):
      if labels_data_model[i,0]==key:
          for k in range(1,14,1):
            masks[k-1]=labels_data_model[i,k]
          return masks
#Assign batch data to X and Y. Adds the image and corresponding
disease array to X[i] and Y[i] for the batch size. Disease array to
Y[i] is assigned using mask_all function.
def create_batch(images_data, batch_size):
    '''
    Creates batches of images and labels in order to feed them to NN
    '''
    X = []
    Y = np.zeros((batch_size, 14))
    for i in range(batch_size):
        disease = np.random.choice(images_data) #randomly select
image name
        image_temp = cv2.imread(disease) #read image from image name
        image_temp = cv2.resize(image_temp, (128, 128))  #resize
image
        X.append(image_temp) #add image to X array of images, of
given batch size.

        Y[i] = mask_all(disease)
    X = np.array(X, dtype="float64")/255.0    #
    return X, Y
def generate_nn_batch(images_data, batch_size):
    '''
    Generates batches of images and corresponding labels
    '''
    while True:
        X, Y = create_batch(images_data, batch_size)
        yield X, Y
```