# EE 456 Final Project Report

Arya Keni, Sung Chang

November 22, 2022

Design and Implementation of a Convolutional NN pair for Detection of Pneumonia Characteristics, with comparative analysis of a DCNN parallelly with VGG-16

## *Abstract*

The Final Project for EE456 includes a working Convolutional Neural Network, that involves backpropagation and binary classification to identify Pneumonia from X-Ray Image inputs. There is a large dataset of X-Ray Images for training and validation present, that is stored for the neural network to form a prediction from (for a sample example image).

The key idea is that the X-Ray image of a pneumonia ridden patient is vastly different than that of a non-pneumonia ridden patient. There are discernable differences around the thoracic cavity (lungs area) of the patient, near the ribs and in the color gradation near that region as well. Through a Convolutional Neural Network (CNN) (the best network architecture for image analysis for feature extraction and classification, exactly what is being done right now), we aim to identify pneumonia as a disease. This disease is chosen as it has a tested and vast existing dataset and is simpler to a fair degree to detect with a sophisticated (within certain bounds) Neural net, unlike COVID-19 or more complex imaging tasks as far as diseases go.

The typical architecture of a CNN is followed (derived from a base set of MLPs), with an 80%-20% or 90%-10% training to validation splitting, of the data. The data is in png files, and are grayscale, to provide for easy data vector computations/weight updating. There are nonlinear activation functions of a radial nature, that provide for sharper contrast determination within the context of boundary setting to identify the features that contribute to the pneumonia label of extraction (another label being normal). There are about 60000 images in total, with certain number towards either category of data in total (Pneumonia or Normal).

A Deep CNN (DCNN, or deep convolutional network), which involves the typical CNN architecture was also designed but with fuller (associative) connections between all the hidden layers, with a general increase in said parameter count. This can be trained on the same data with the same settings of activation, splitting of the dataset and so on. The classification as per the problem statement shall be the same in this case.

A VGG (Visual Geometry Group) Network was also designed for further analysis of a unique layer set architecture for the network and to analyze the effectiveness of learning specific applied tensors that involved these grayscale images. This was unique as a leveraging factor to compare a completely different system for analyzing a typical convolutional platform for the dataset, especially pertaining to images. This is because its prominent level of feature extracting capabilities employed within many layers (16 in this case), is interesting to apply into an image processing and feature extraction scenario.

The dataset is obtained from a large open-source hospital network leveraging the data to Kaggle, a Machine Learning hub for dataset storage, from the borrowed platform of the official medical center: **Chinese National Hospital in Guangzhou.** An interesting factor to note is all X-Rays are adult X-Rays, as pediatric X-Rays are off scope for this project due to them presenting completely different sets of features to analyze.

## *Outline:*

Typically, the mode of experimentation consists of cross comparisons on CNN (or multilayer VGG) and DCNN, to gauge the increase or trends in accuracy, along with loss metrics over epochs taken for both models. Additionally, there are 2 forms of MLP (Multilayer Perceptron) as simplistic models with naïve learning mechanisms that can be compared to cross verify the rate

of increase, along with a standard 16-layer VGG for image processing of the cost of increase in complexity to CNN (or VGG of mid-level layering) or higher to a DCNN is interesting to check as well.

## *Introduction:*

The type of DCNN is typically a modification of AlexNet, and the type of CNN is a type of LeNet (with the other version of the CNN being a modified set from slightly changed hyperparameters for the same LeNet base architecture for both CNN models), following typical architecture hyperparameters, and structural (neuron, layer, vector, weight, and bias) connotations (for both nets). Additionally, a VGG is an interesting parallel between the Lenet and AlexNet to compare and mark observations within and along vast architectural differences.

For a dense dataset of 256x256 pixel area per grayscale X-Ray image, the physically defining characteristics of pneumonia being large clusters of light-dark contrast and gradation in features of the ribcage calls for feature extraction to classify binary-wise if it is indeed pneumonia or not. A python notebook was used to leverage an advanced GPU from google to use for training, testing, and analysis. (Code in .ipynb file, with relevant folder calls for data).

A python notebook is used to leverage a google server GPU to reduce runtime of training exponentially, and hence all data and folder calls for input files is through the google drive, and for plots storage for results as well. The interactive model of the file provides for lookups of the tables, plots, and prints generated of the statistics, metrics, analyses, probabilities, and plot trends (along with plotted metrics, such as confusion matrix, and so on...)

## *Theory:*

Ideally, we hope to see a considerable accuracy in detection of Pneumonia solely from a large data set and X-Ray images, with verification from existing data that it is the case. It is also interesting to compete the performances of the CNN and DCN, with intuitive senses suggesting the DCN outperforming the CNN, but depending on the solution and data, that may not be the case (example overfitting situations).

The performances are computed by the typical metrics of MSE, or that of accuracy in MLP vectors by weight error corrections, (in training and testing), and the learning rate will be annealed linearly at a small rate of decease to ensure optimal error surface functional energy minimization (for the given local minima for the dataset).

Interesting aspects include ideal identification of complex sets of visible characteristics and a clear margin of are mapping pertaining to a real-world medical problem (illness) and its remedy. Using the same principles of PCA, classification, decision boundaries, backpropagation, activations, and layers, and MLP type aspects of hidden layers and connectivity, along with other methods relating to feed forward algorithms, algorithms that involve weigh updates and learning rate updates, and minimizing error energy functions, as well as concepts involving computing error/loss across epochs are all be useful here.

The typical layers involved at the CNN are convolution, dense, ReLU, and Softmax for activation, all in 2D for images. Additional layers include Max Pooling, and Flattening.

The *helping libraries* used are that of in python3: TensorFlow for simplifying matrix computations, sklearn (and skimage) and seaborn for data preprocessing and data mapping (and plotting), pandas for data objectification, NumPy for other mathematical calculations, cv2 for image file reads, and random for random seed generation.

Note that the pathways for data images and stores of results are to the local google drive colabs, to ensure parallel processing between the learnt model and the deliverables. The use of this exponentially reduces runtime from days to minutes.

***Architecture:***

It is important to briefly comment about the architecture of both systems to be employed, and to note stark features they possess:

For the CNN's, the architecture follows a similar pattern:

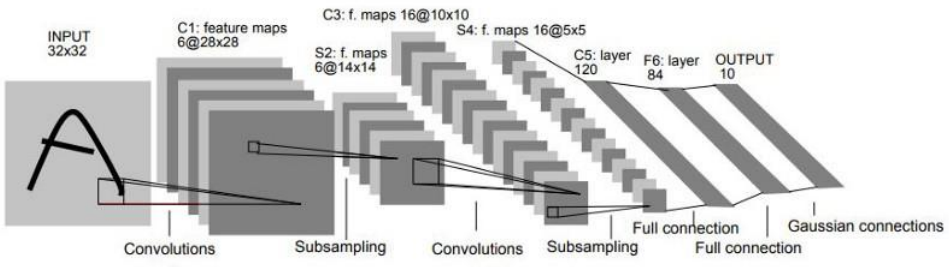



**Figure A: The LeNet architecture for CNN's (Source: medium.com: What are LeNet's?)**

This is used for the solution model for classification for features. The interesting factor about this architecture is the use of feature maps, and reduction in samples for a deduced classification, which provides for a less computationally intensive (relatively) method for problem solving.

For the Deep CNN's, the architecture changes thusly:

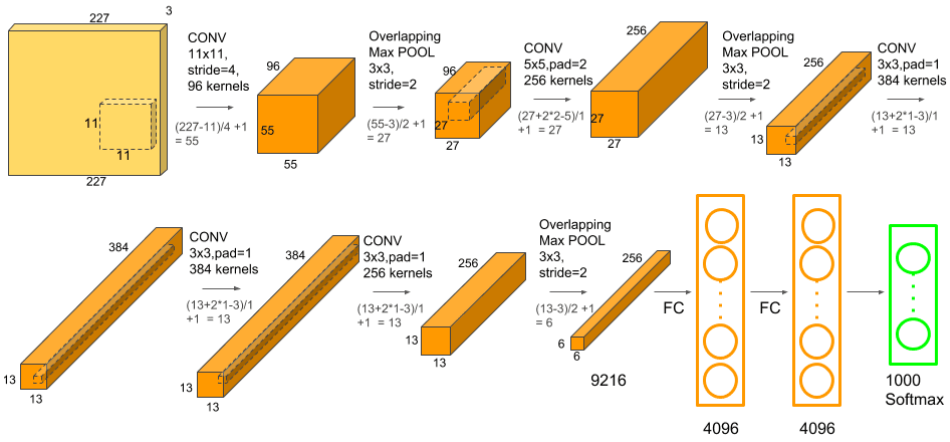



**Figure B: Classical Deep AlexNet, for Data extraction in an image (Source: medium.com: Understanding the famous AlexNet)**

This model, on the other hand, has a tensor-based structure involving multiple processing units of image pixels as nodes, across dimensions and vectors alike as per problem specifications. Convolution of an image in 2 dimensions is a key feature of this net, where the focus is on statistically and computationally regressing an image as a signal of values.

For the VGG NN, the architecture is as follows:

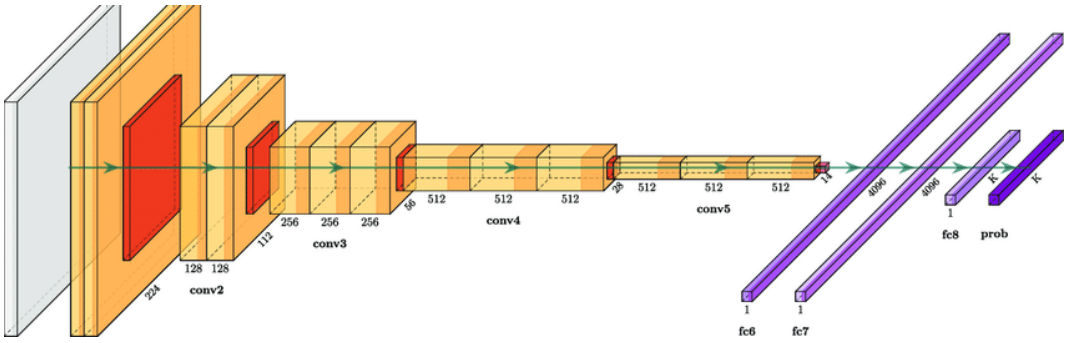


**Figure C: A VGG or Deep-Deep AlexNet architecture, simplified for layered processing (Source: Cognition, Volume 208)**

This model utilizes an even deeper preface from an AlexNet, employing multiple convolutions in series and parallel, which have a high learning rate and accuracy metric (on average, in theory) for object recognition parametric.

**For all the aforementioned architectures, a key point is that:** The binary classification implied the final layer being a 1 node output (with binary capabilities of one hotness), and not a final layer vector of probable confidences per classification (n-identifications, for "n" more than 2).

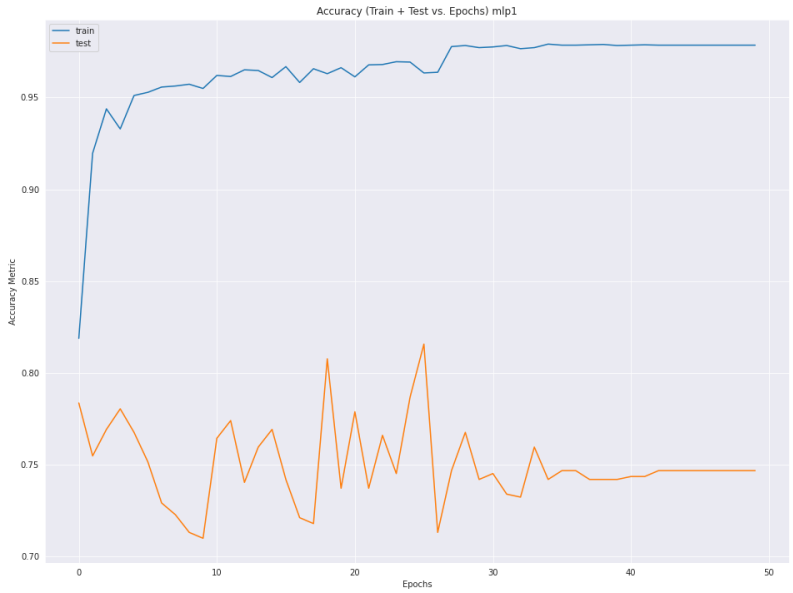
## *Results:*

For the DCNN and CNN's, these are the results: It is interesting to observe the following sample outputs generated, and what they may mean:

### For the CNN 1:

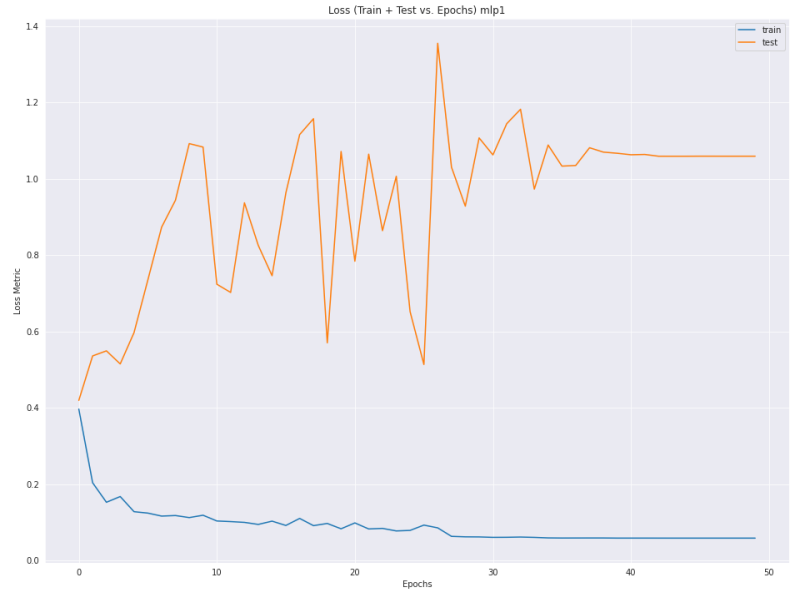
The following are the results:

Accuracy Metric for Epochs



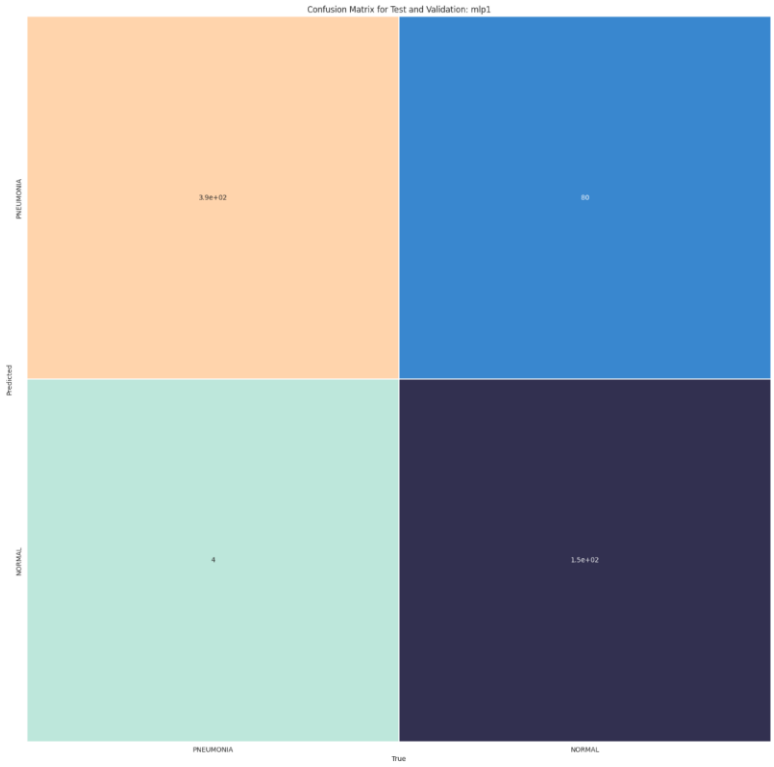

This plot shows the accuracy as a percentage normalized for train and test for CNN model 1

Loss Metric for Epochs



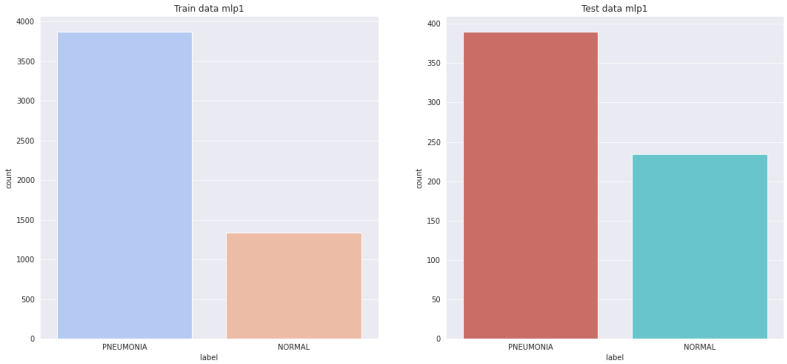

This plot shows the loss as a metric value for train and test for CNN model 1

Confusion Matrix



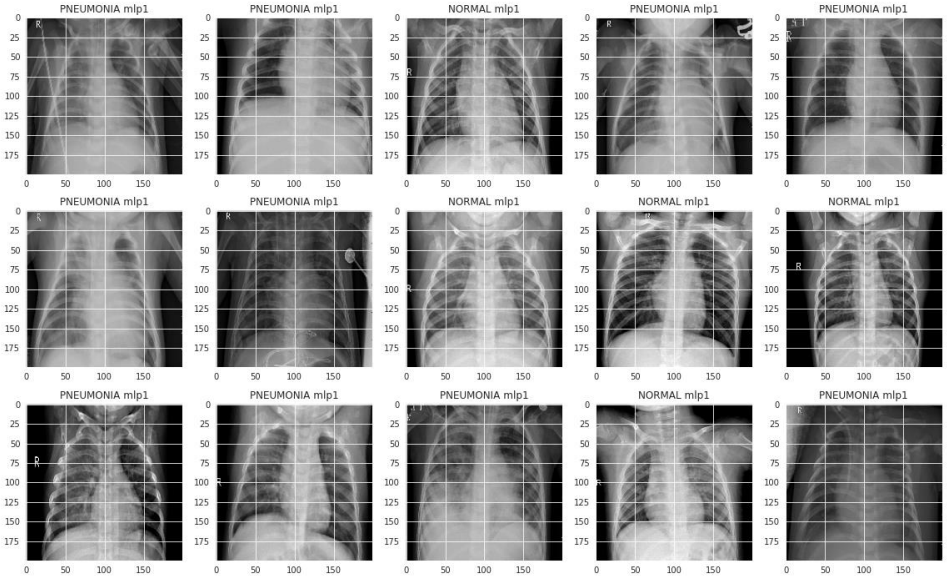

The confusion matrix for the CNN Model 1 test cases, showing true to true Normal and Pneumonia Image Counts

Sample of Train Test Split



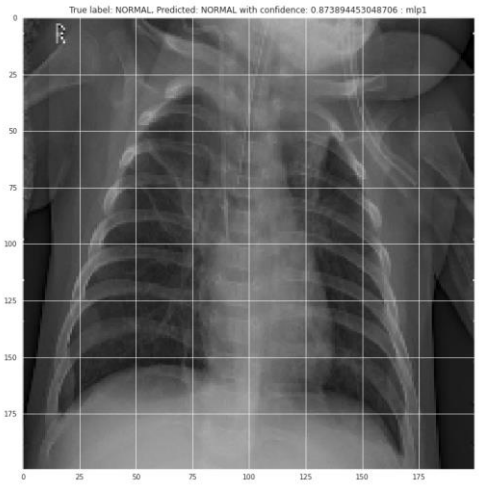
This shows most of the images are pneumonia labelled, in train and test.

Sampling for Image Data

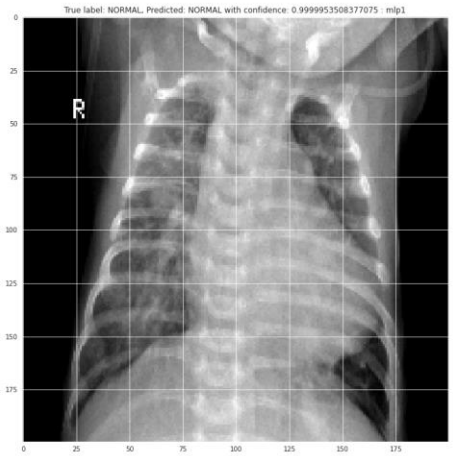


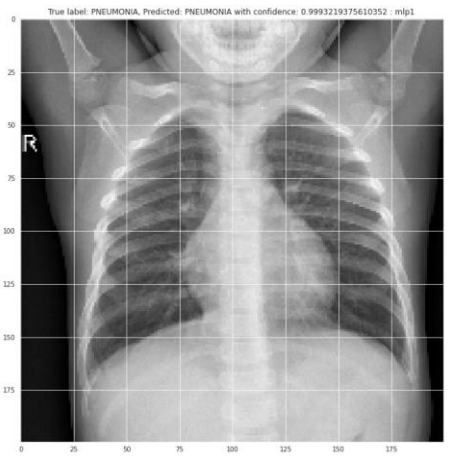

These are some base images from the sample set for view.

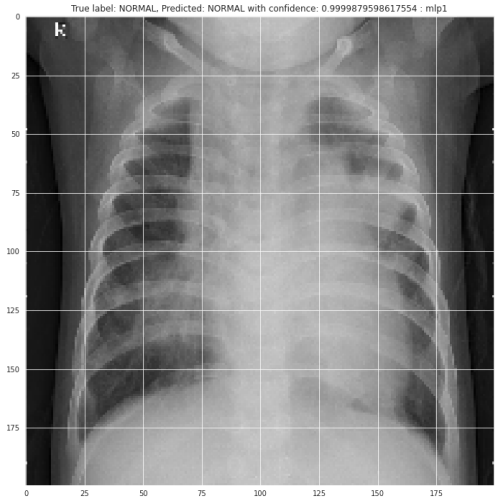
Random 10 Images sampled and predicted

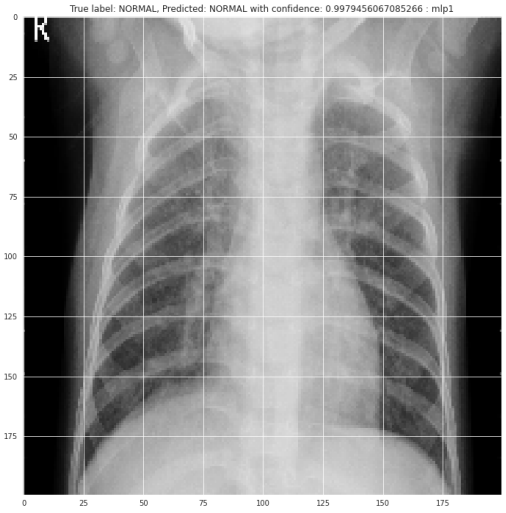

True label: NORMAL, Predicted: NORMAL with confidence: 0.873894453048706 : mlp1

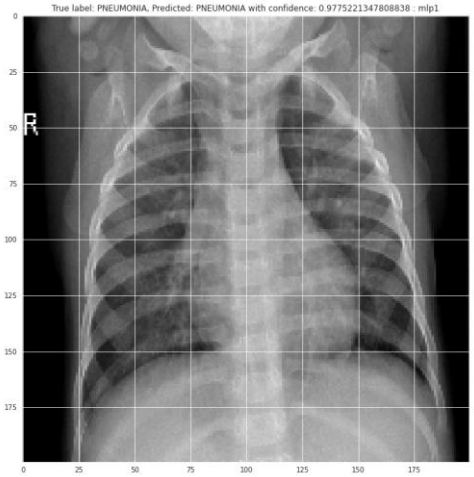

True label: PNEUMONIA, Predicted: NORMAL with confidence: 0.8547824621200562 : mlp1

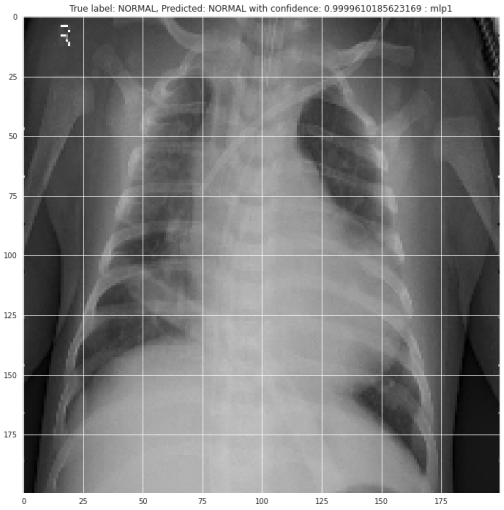

True label: NORMAL, Predicted: NORMAL with confidence: 0.9999953508377075 : mlp1

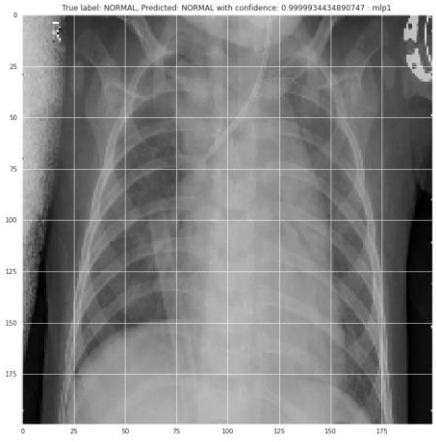

True label: PNEUMONIA, Predicted: PNEUMONIA with confidence: 0.9993219375610352 : mlp1

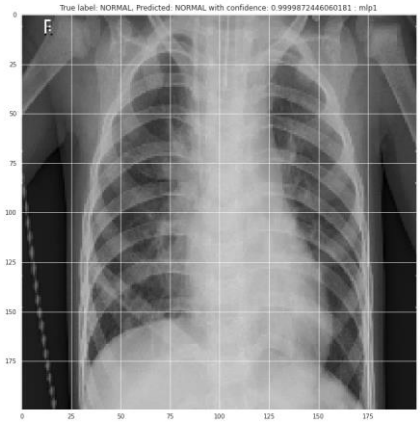

True label: NORMAL, Predicted: NORMAL with confidence: 0.9999879598617554 : mlp1


True label: NORMAL, Predicted: NORMAL with confidence: 0.9979456067085266 : mlp1


True label: PNEUMONIA, Predicted: PNEUMONIA with confidence: 0.9775221347808838 : mlp1




These 10 images depict with some confidence level predictions of true to given labels for the X-Ray.

Precision, Recall and Overall Statistics

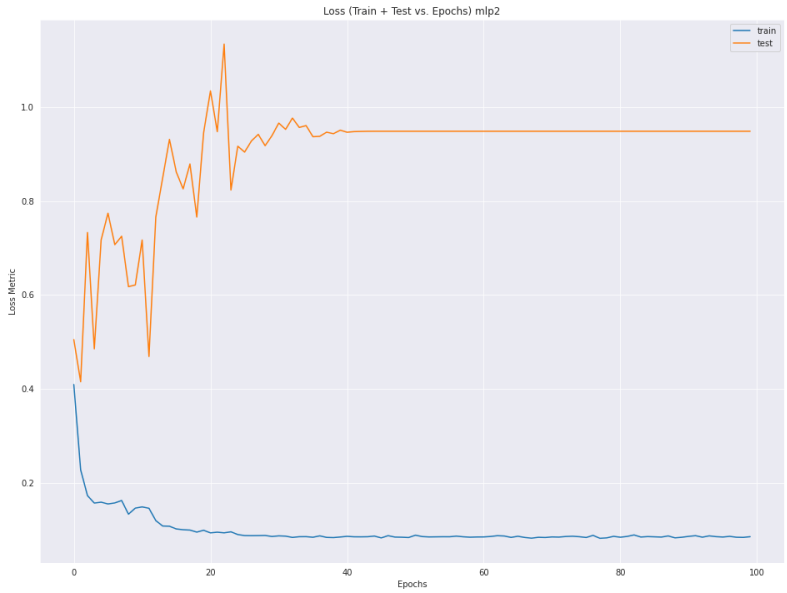
```
Overall Precision Accuracy for model mlp1: 0.7467948794364929
Overall Recall Accuracy for model mlp1: 0.7467948794364929
              precision    recall  f1-score   support

   PNEUMONIA       0.95      0.34      0.50       234
      NORMAL       0.71      0.99      0.83       390

    accuracy                           0.75       624
   macro avg       0.83      0.67      0.67       624
weighted avg       0.80      0.75      0.71       624

Test loss for model mlp1: 105.8800458908081%
Test accuracy for model mlp1: 74.67948794364929%
MLP Error for model mlp1: 25.320512056350708
```

Binary classification statistics and metrics tell a lot about a net.

For the CNN 2:

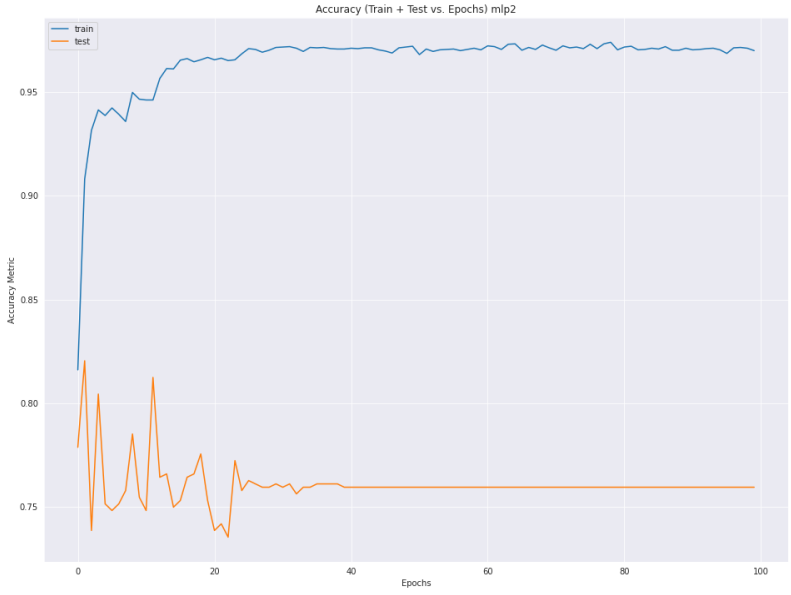
The following are the results:

Loss Metric Per Epoch



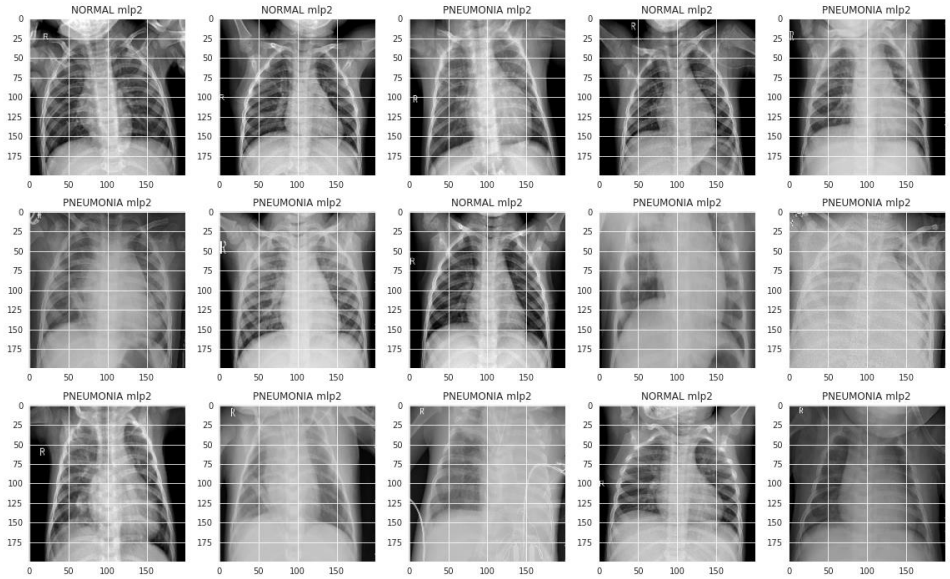

Loss metric trend of test and train per epoch

Accuracy metric Per Epoch



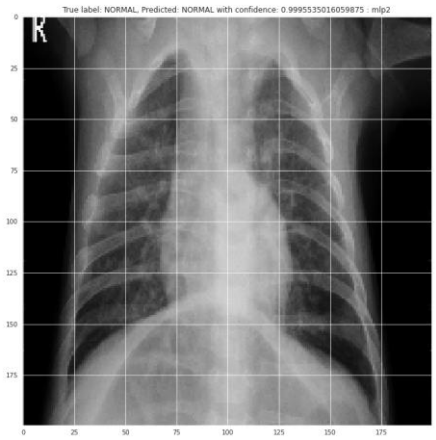
Accuracy trend of test and train per epoch.

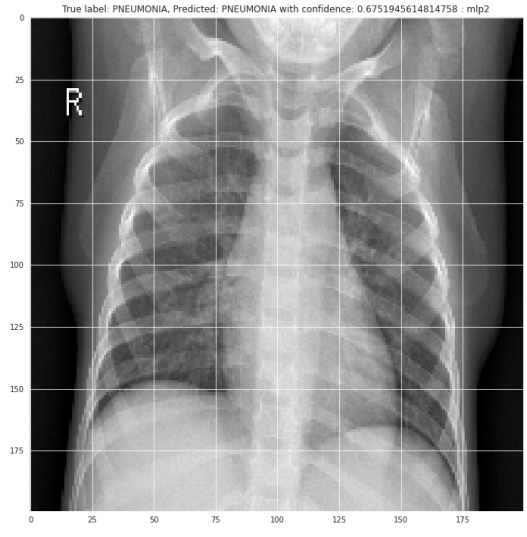
Sample of Input Data

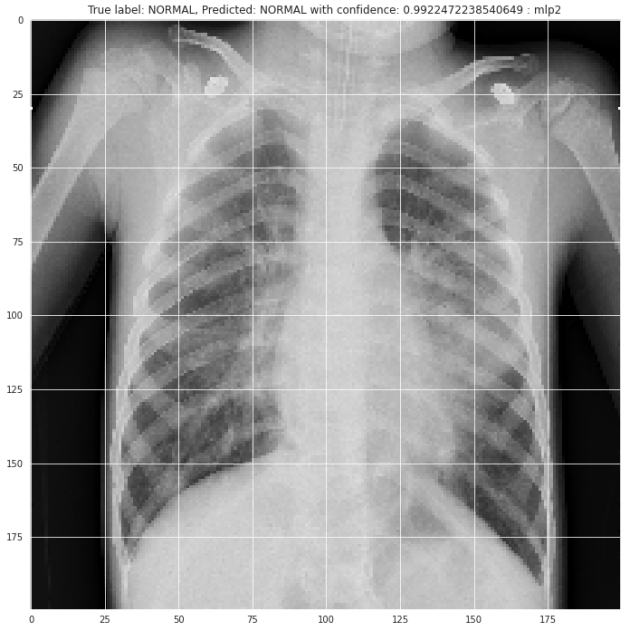


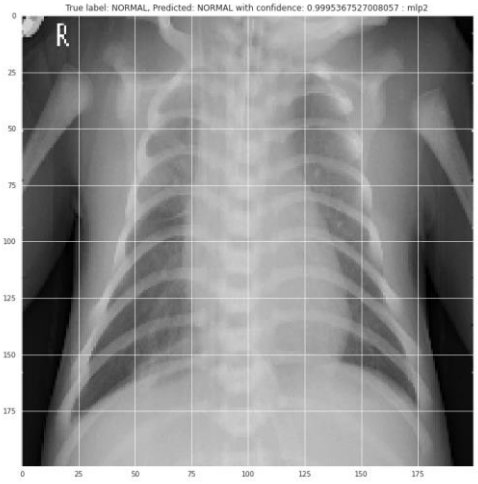

Sample input images to view.

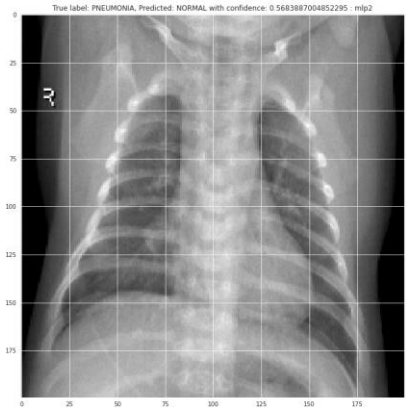
Predictions for 10 random test images

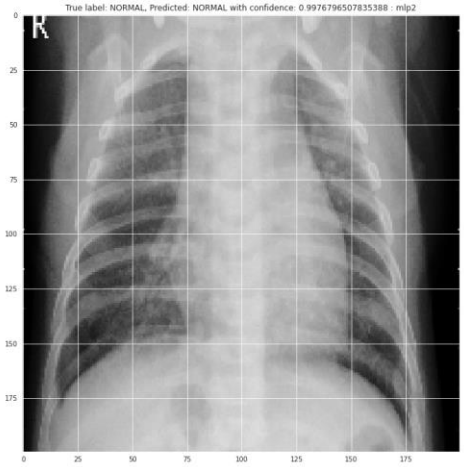

True label: NORMAL, Predicted: NORMAL with confidence: 0.9995535016059875 : mlp2

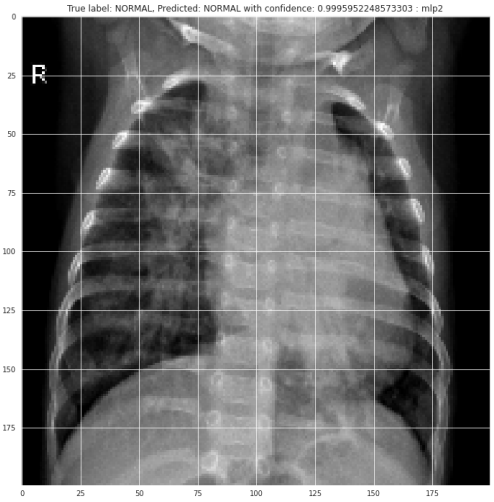

True label: PNEUMONIA, Predicted: PNEUMONIA with confidence: 0.6751945614814758 : mlp2

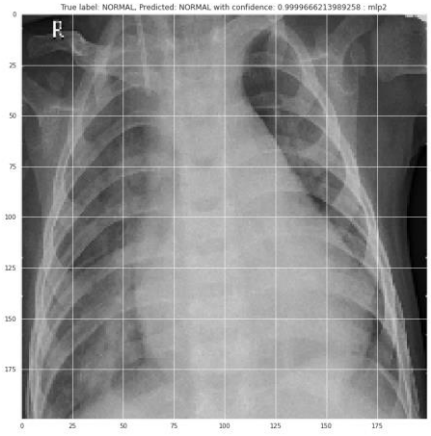

True label: NORMAL, Predicted: NORMAL with confidence: 0.9922472238540649 : mlp2

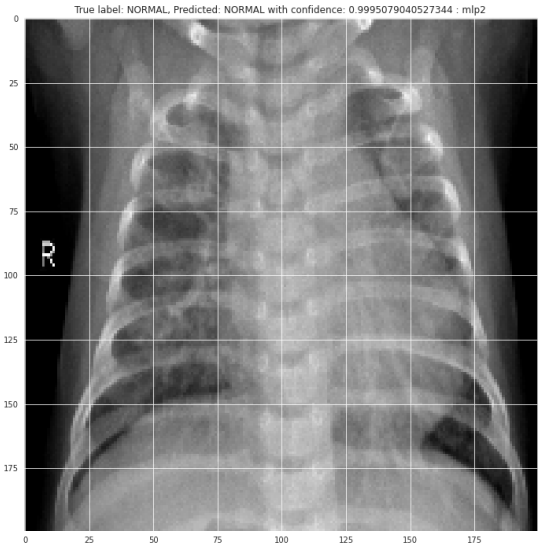

True label: NORMAL, Predicted: NORMAL with confidence: 0.9995367527008057 : mlp2

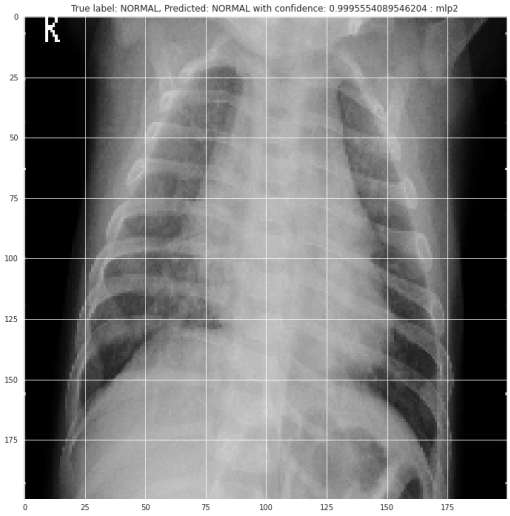

True label: PNEUMONIA, Predicted: NORMAL with confidence: 0.5683887004852295 : mlp2


True label: NORMAL, Predicted: NORMAL with confidence: 0.9976796507835388 : mlp2


True label: NORMAL, Predicted: NORMAL with confidence: 0.9995952248573303 : mlp2

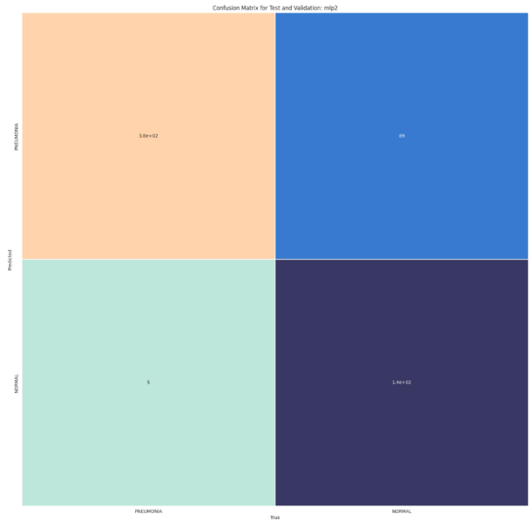

True label: NORMAL, Predicted: NORMAL with confidence: 0.9999666213989258 : mlp2


True label: NORMAL, Predicted: NORMAL with confidence: 0.9995079040527344 : mlp2



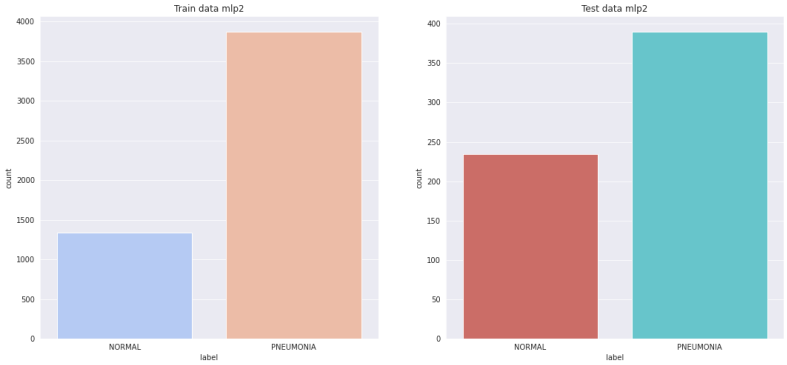

The 10 sampled test X-Rays predict and compare with some confidence the label.

Confusion matrix



The confusion matrix for true-to-true comparison for false/true positives/negatives for the classification.

Train Test Split statistics




Skewed data in terms of pneumonia inputs labelled higher.

Precision and Recall (per class and Overall) Statistics:

```
Overall Precision Accuracy for model mlp2: 0.7596153616905212
Overall Recall Accuracy for model mlp2: 0.7596153616905212
              precision    recall  f1-score   support

   PNEUMONIA       0.95      0.38      0.54       234
      NORMAL       0.73      0.99      0.84       390

    accuracy                           0.76       624
   macro avg       0.84      0.68      0.69       624
weighted avg       0.81      0.76      0.73       624
```

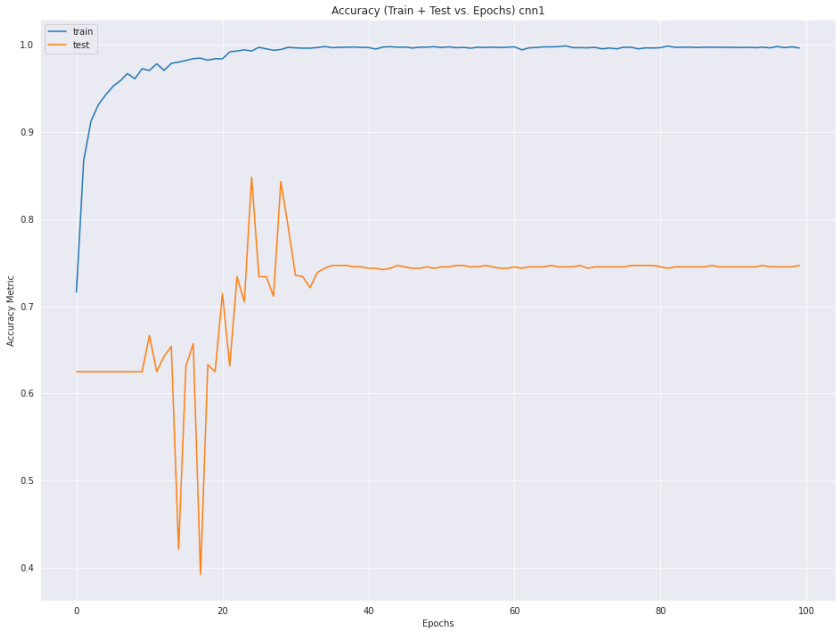
```
Test loss for model mlp2: 94.86842751502991%
Test accuracy for model mlp2: 75.96153616905212%
MLP Error for model mlp2: 24.038463830947876
```

Binary classification statistics and metrics tell a lot about a net.

For the Deep CNN:

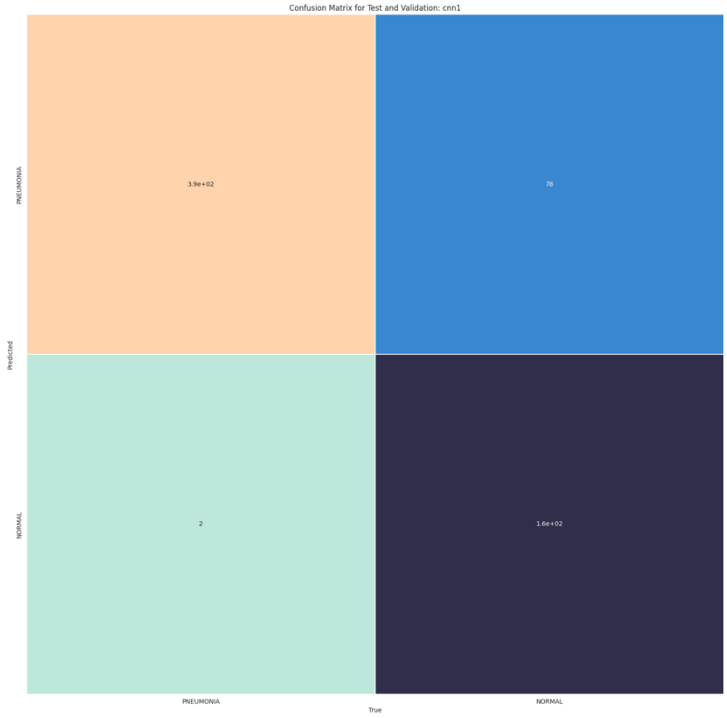
The following are the results:

The accuracy metric per epoch for training and testing



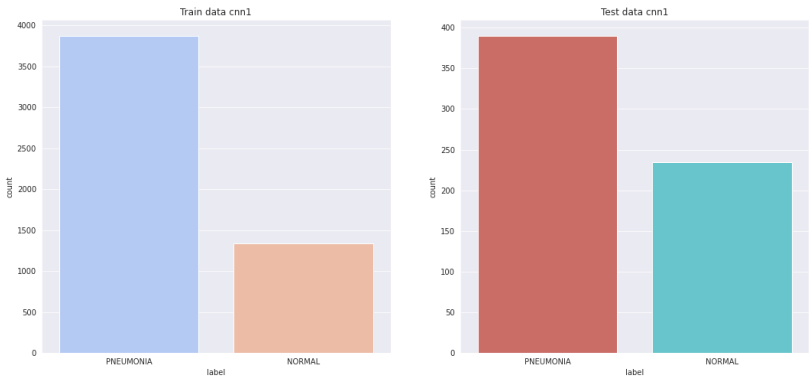

Depicts the trend of the accuracy of training and testing data across the learning era.

Confusion Matrix:



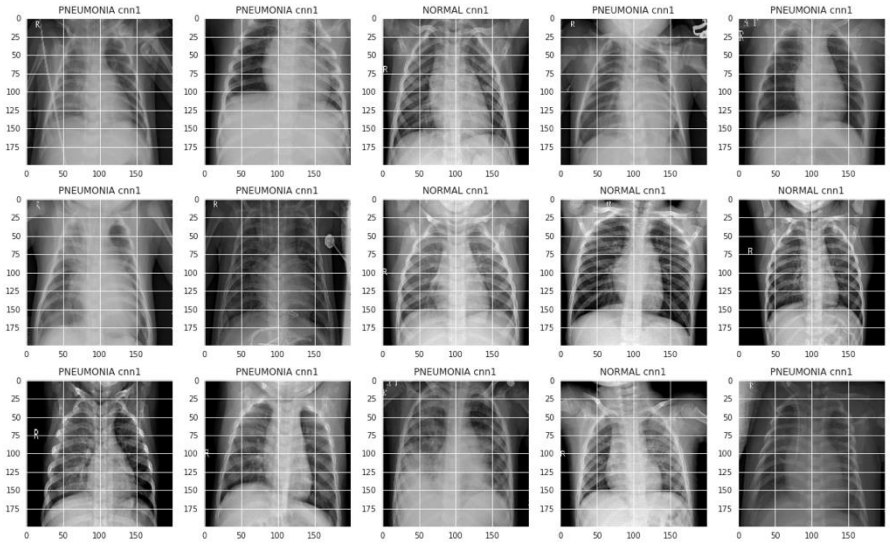

Depicts the confusion distributions of true-true and false-false combinations of data in testing.

Trian -test split statistics



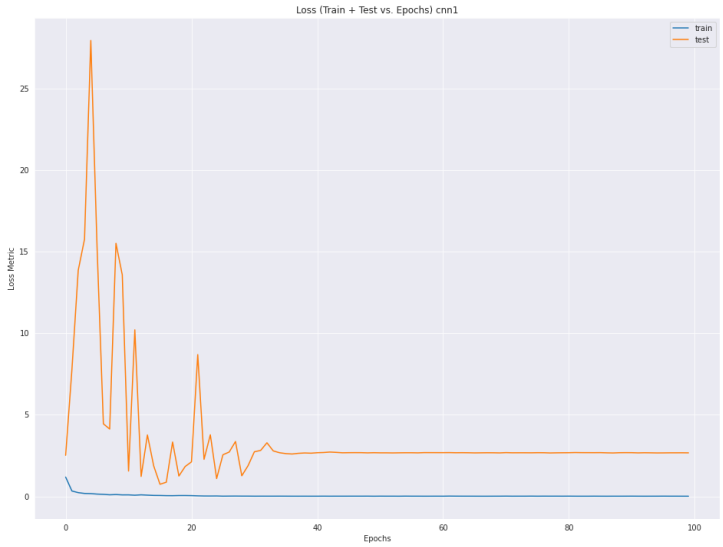
For the Deep sample, about 75-80% of the data in training and testing is that of Pneumonia status

Overview of sample data and classification



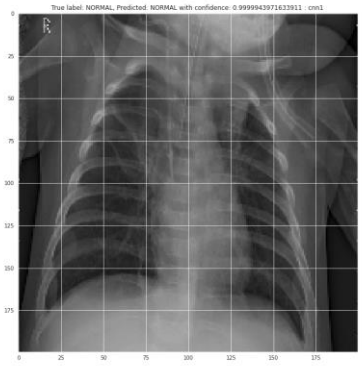

Depicts a general view of the data involved, and the types of images.

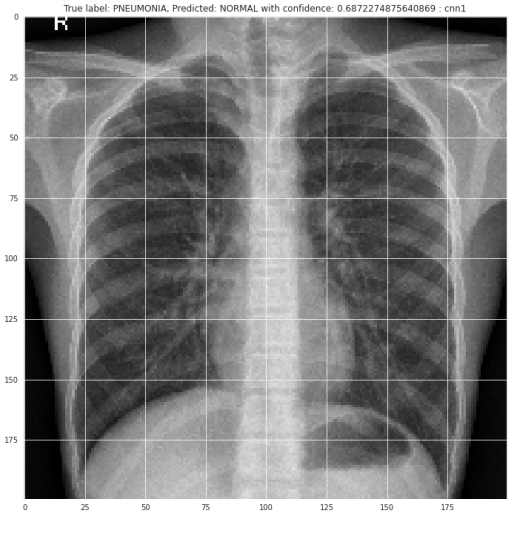
Loss metric plot

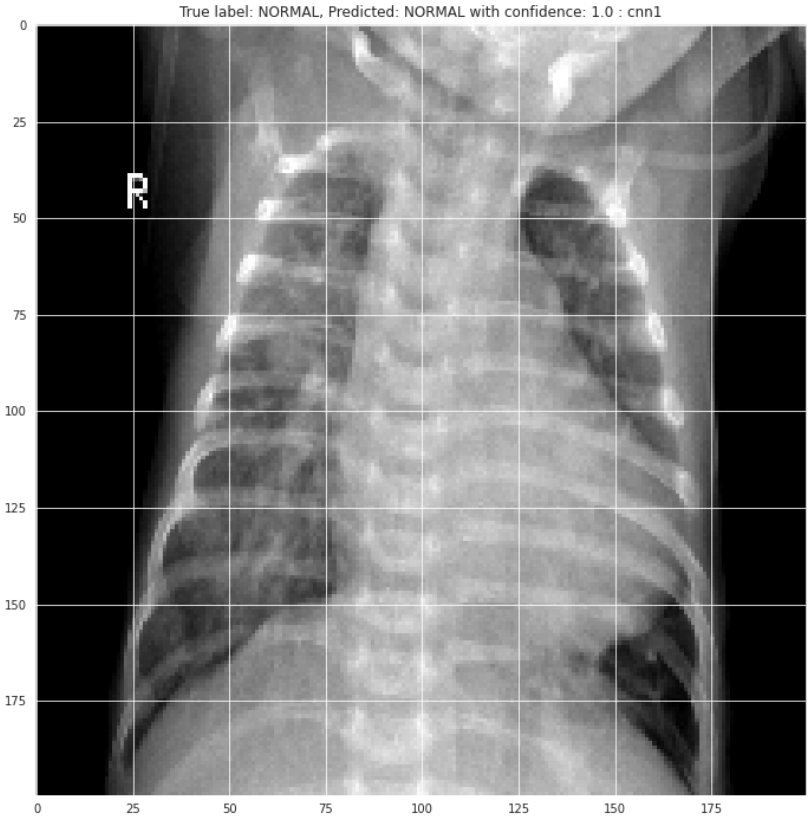


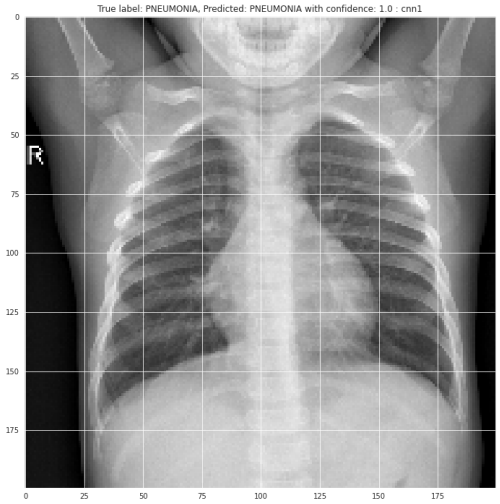

Depicts the per epoch loss for training and testing data

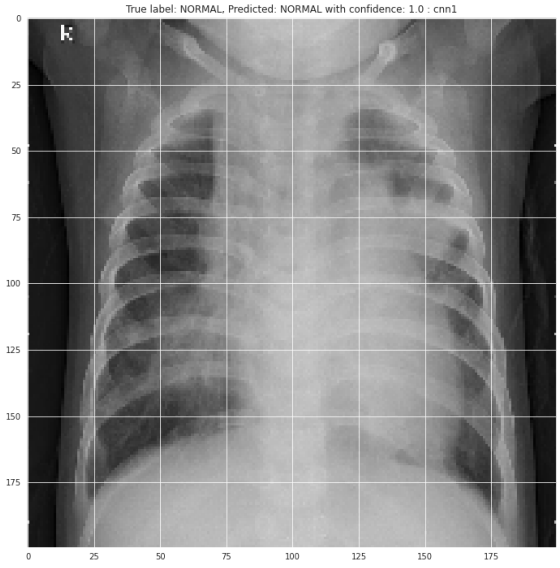
10 samples, random, for predictive accuracy on classification with Deep CNN

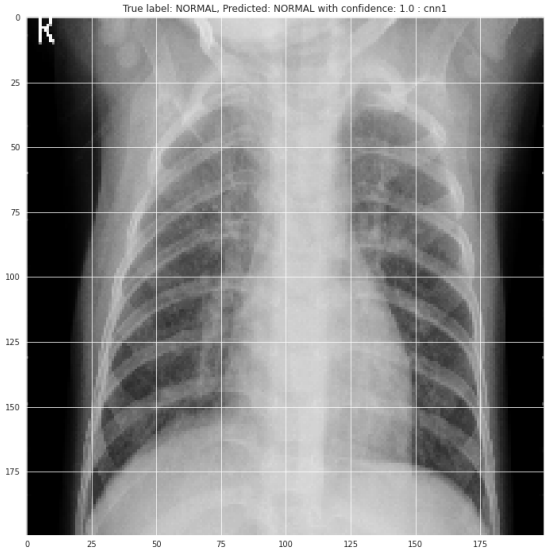

True label: NORMAL, Predicted: NORMAL with confidence: 0.9999943971633911 : cnn1

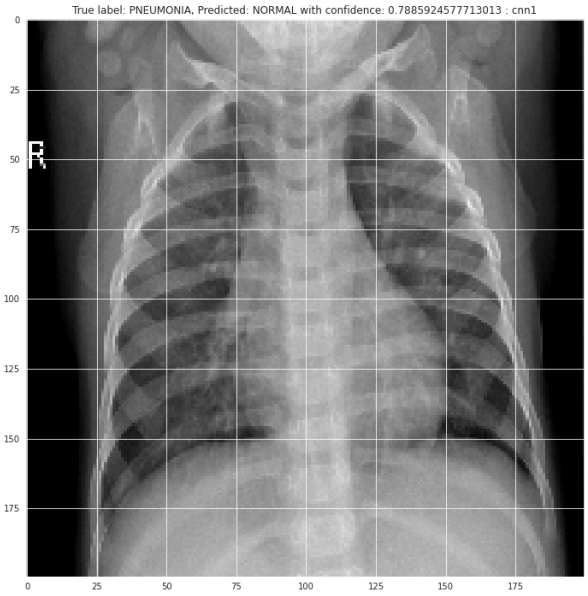

True label: PNEUMONIA, Predicted: NORMAL with confidence: 0.6872274875640869 : cnn1

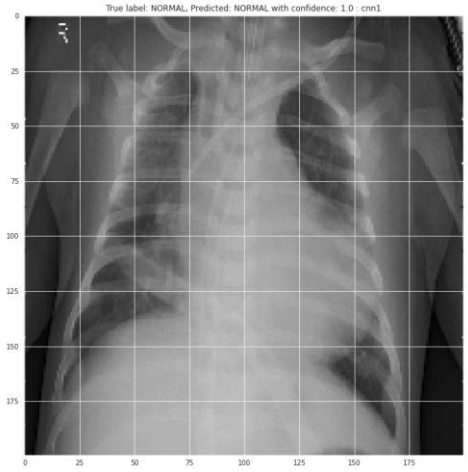

True label: NORMAL, Predicted: NORMAL with confidence: 1.0 : cnn1

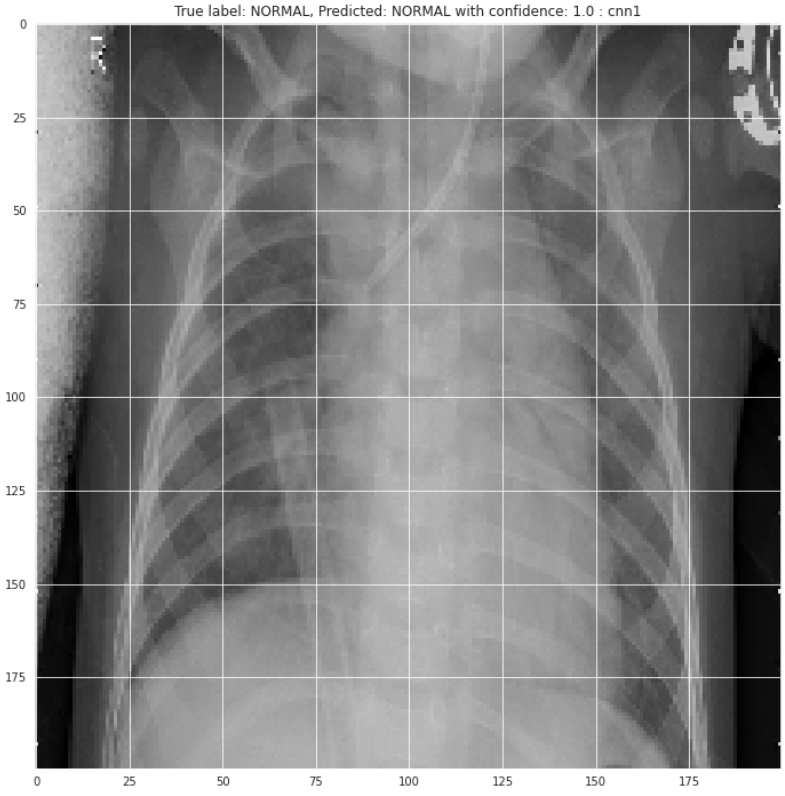

True label: PNEUMONIA, Predicted: PNEUMONIA with confidence: 1.0 : cnn1
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

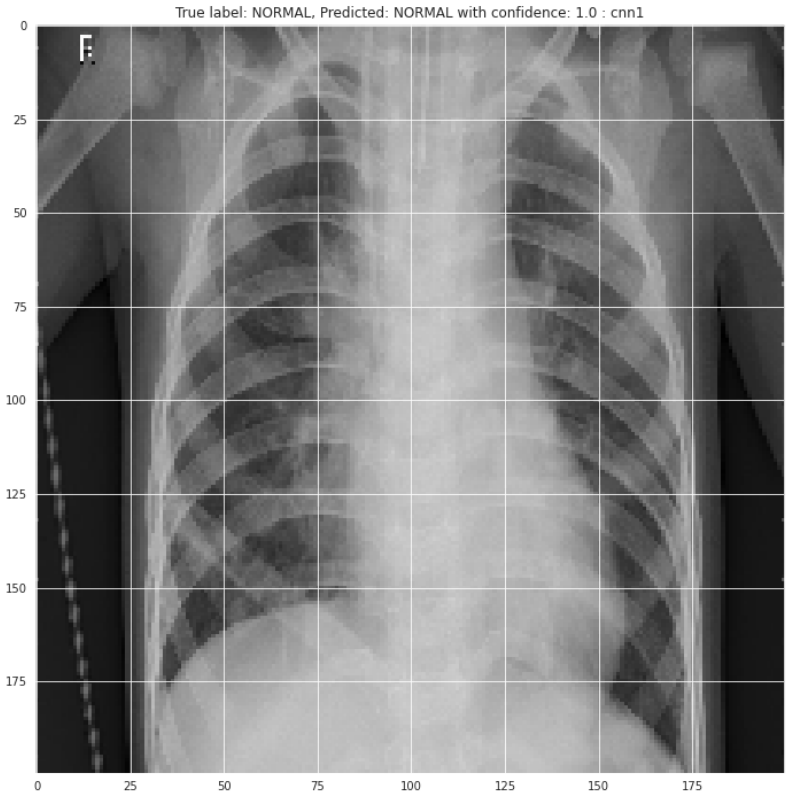

True label: NORMAL, Predicted: NORMAL with confidence: 1.0 : cnn1
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


True label: NORMAL, Predicted: NORMAL with confidence: 1.0 : cnn1
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


True label: PNEUMONIA, Predicted: NORMAL with confidence: 0.7885924577713013 : cnn1
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


True label: NORMAL, Predicted: NORMAL with confidence: 1.0 : cnn1
0
25
50
75
100
125
150
175


True label: NORMAL, Predicted: NORMAL with confidence: 1.0 : cnn1
0
25
50
75
100
125
150
175




Depicts a good mix of testing data that has been predicted with corresponding confidence in the same

Metric statistics for a Deep CNN

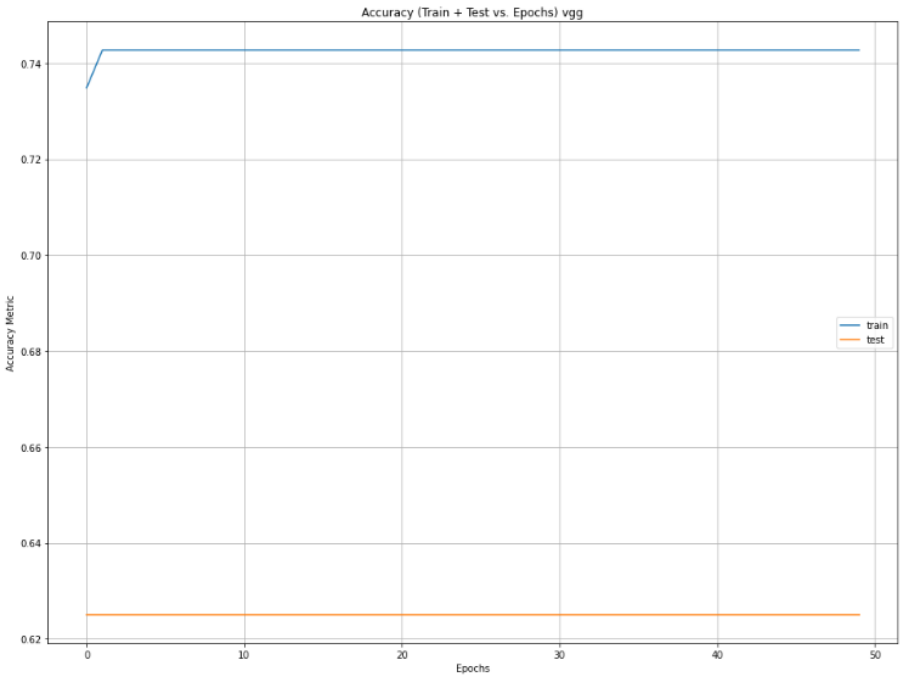
```
Test loss for model cnn1: 2.666163444519043%
Test accuracy for model cnn1: 74.67948794364929%
MLP Error for model cnn1: 25.320512056350708
20/20 [==============================] - 1s 5ms/step
Overall Precision Accuracy for model cnn1: 0.7467948794364929
Overall Recall Accuracy for model cnn1: 0.7467948794364929
              precision    recall  f1-score   support

   PNEUMONIA       0.97      0.33      0.50       234
      NORMAL       0.71      0.99      0.83       390

    accuracy                           0.75       624
   macro avg       0.84      0.66      0.66       624
weighted avg       0.81      0.75      0.71       624
```

Deep CNN precision, recall, and f-1 score, along with general loss and accuracy.

## For the VGG Neural Network:

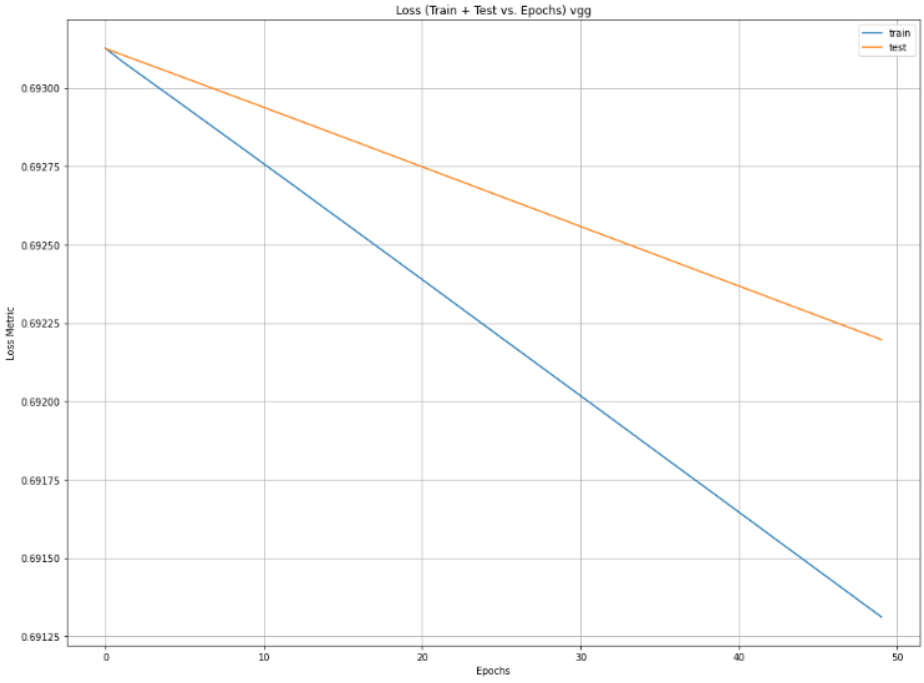
The following were the observed results after evaluating the VGG Net:

The Accuracy Metric per epoch



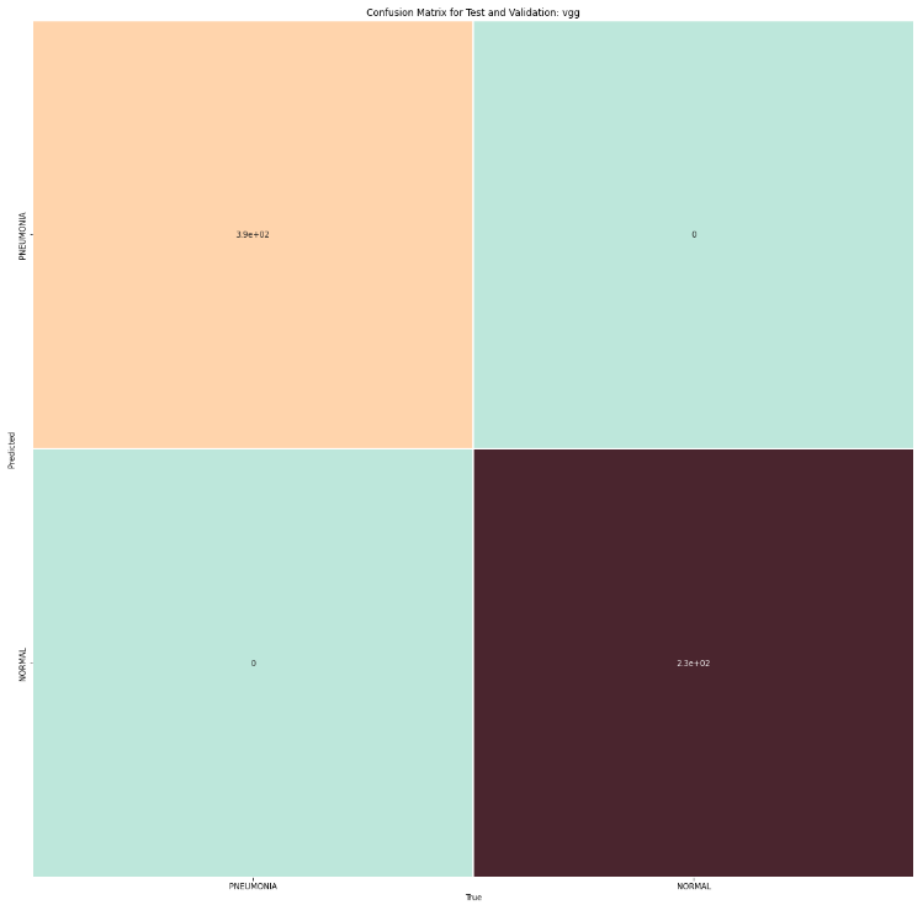

The above shows the accuracy trend for both training and testing data across the whole learning period

The Loss Metric per epoch



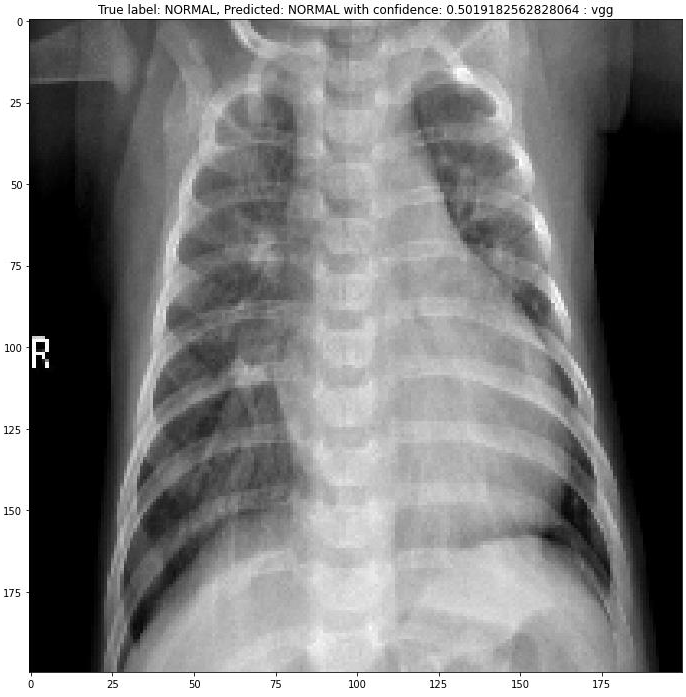

The above shows the loss trend for both training and testing data across the whole learning period

Confusion Matrix

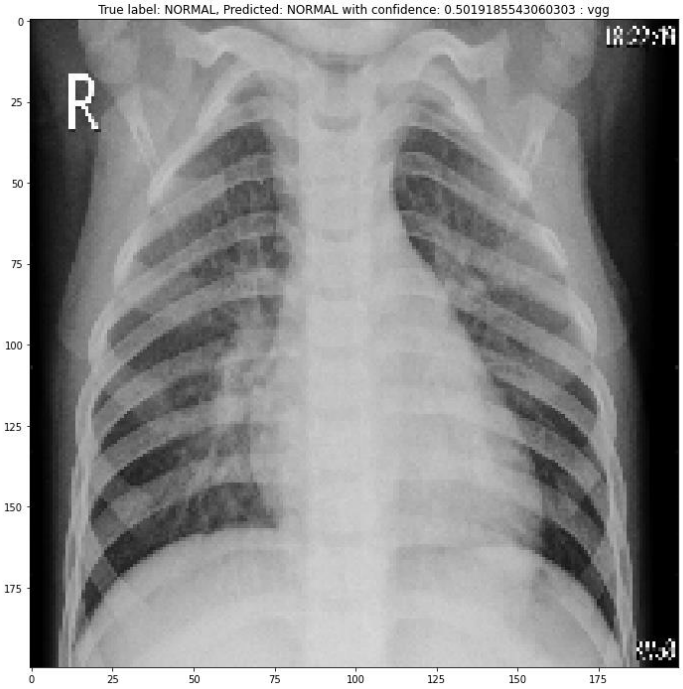



10 samples, random, for predictive accuracy on classification with VGG Net


True label: PNEUMONIA, Predicted: NORMAL with confidence: 0.5019184947013855 : vgg


True label: NORMAL, Predicted: NORMAL with confidence: 0.5019182562828064 : vgg


True label: NORMAL, Predicted: NORMAL with confidence: 0.501916766166687 : vgg
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

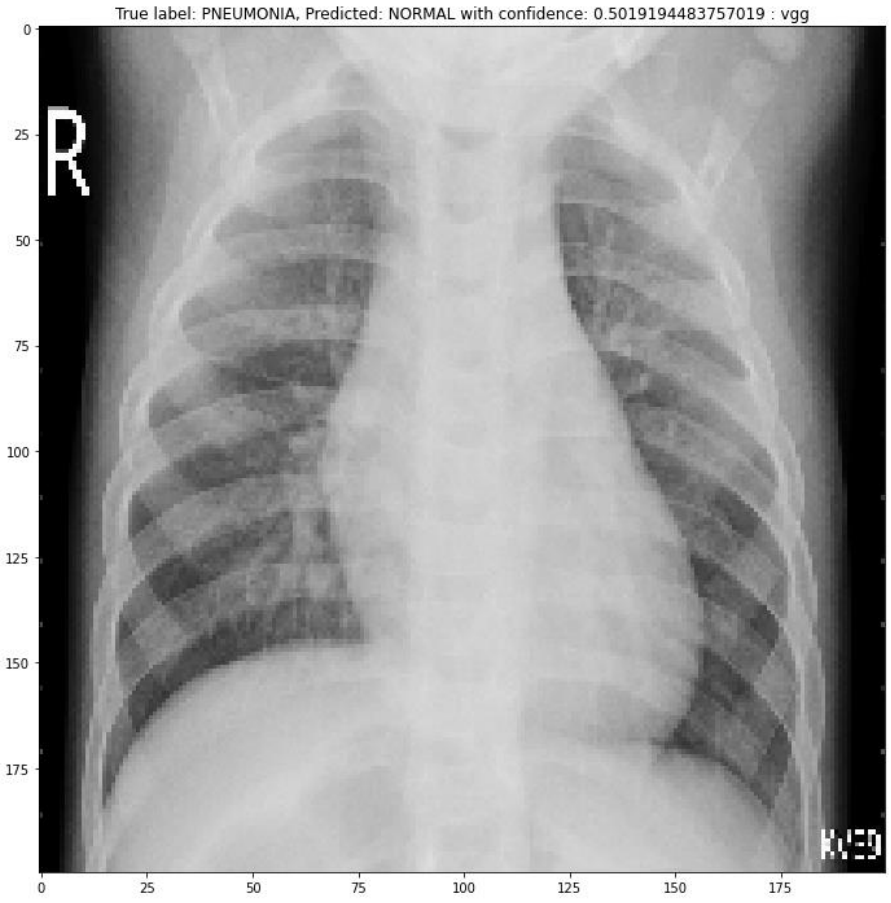

True label: NORMAL, Predicted: NORMAL with confidence: 0.5019185543060303 : vgg
R
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


True label: NORMAL, Predicted: NORMAL with confidence: 0.5019184350967407 : vgg
0
25
50
75
100
125
150
175


True label: PNEUMONIA, Predicted: NORMAL with confidence: 0.5019175410270691 : vgg
0
25
50
75
100
125
150
175


True label: NORMAL, Predicted: NORMAL with confidence: 0.501914918422699 : vgg
0
25
50
75
100
125
150
175


True label: NORMAL, Predicted: NORMAL with confidence: 0.5019164085388184 : vgg
0
25
50
75
100
125
150
175


Overall, the images above show the depiction of some of the test data with a corresponding confidence value.

The Overview of the VGG Net

```
Test loss for model vgg: 0.6921969652175903%
Test accuracy for model vgg: 62.5%
MLP Error for model vgg: 37.5
20/20 [==============================] - 3s 137ms/step
Overall Precision Accuracy for model vgg: 0.625
Overall Recall Accuracy for model vgg: 0.625
              precision    recall  f1-score   support

   PNEUMONIA       0.00      0.00      0.00       234
      NORMAL       0.62      1.00      0.77       390

    accuracy                           0.62       624
   macro avg       0.31      0.50      0.38       624
weighted avg       0.39      0.62      0.48       624

20/20 [==============================] - 3s 138ms/step
20/20 [==============================] - 3s 140ms/step
20/20 [==============================] - 3s 142ms/step
20/20 [==============================] - 3s 144ms/step
20/20 [==============================] - 3s 145ms/step
20/20 [==============================] - 3s 147ms/step
20/20 [==============================] - 3s 146ms/step
20/20 [==============================] - 3s 144ms/step
20/20 [==============================] - 3s 143ms/step
20/20 [==============================] - 3s 140ms/step
\VGG Model 1 Generated.
```

These statistics are important to understanding the intrinsic behavior of the network.

Note that the training ant testing sample histogram (bar plot), and the sample images of inputs are omitted here to avoid repetition.

## *Observations:*

The observations for the DCNN and CNN models (both) entail complex metrics, and typical trend observations, along with actual image validations. We can see that:

## For the CNN model 1:

We can see that there is a high accuracy in tests, about 78% even after 50 epochs, and 8 layers of CNN (in theory). The training data is about 97% accurate, but the non-plateauing of the dataset at Early Stopping indicates that there may be a diminishing returns optimization left beyond the timesteps employed.

The predicted data shows about 70-80% of train and test data (split into 80-20 ratio of train-test) is pneumonia labelled, reflected in the confusion matrix of test values. Same goes for normal labels, as the confidence in predictions is near 100% for both values on average. The true and false for a predicted label adds up to the correct proportions, with about 99.8% of true to true and false to false assignment.

In the 10 random samples, 9 or 10 of the 10 images are correctly predicted, with a high confidence per sample itself. This varies uniformly but depicts from the test the high ratio of true-to-true learnt labels.

The learning rate decreases exponentially, and the accuracy for training and testing rises logarithmically per epoch. The loss metric shows an exponential drop across epochs from the given values of 1.5 to exceptionally low ones.

Exact values are in the plots and provided png files of the plots. Additionally, high precision and recall shows that binary classification is learnt well within an adaptable memory context for the system, problem solving-wise.

There is a low error in general, along with per class and overall f-scores and metric recalls and validation scheme scores being extremely high.

## For the CNN model 2:

We can see that there is a high accuracy in tests, about 74% even after 100 epochs, and 12 layers of CNN (in theory). The training data is about 92% accurate, but the non-plateauing of the dataset at Learning Rate plateauing after 40 epochs indicates that there may be a diminishing returns optimization left beyond the timesteps employed, next to 0.

The predicted data shows about 70-80% of train and test data (split into 80-20 ratio of train-test) is pneumonia labelled, reflected in the confusion matrix of test values. Same goes for normal labels, as the confidence in predictions is near 100% for both values on average. The true and false for a predicted label adds up to the correct proportions, with about 99.8% of true to true and false to false assignment.

In the 10 random samples, 9 or 10 of the 10 images are correctly predicted, with a high confidence per sample itself. This varies uniformly but depicts from the test the high ratio of true-to-true learnt labels.

The learning rate decreases exponentially, and the accuracy for training and testing rises logarithmically per epoch. The loss metric shows an exponential drop across epochs from the given values of 1.5 to exceptionally low ones.

Exact values are in the plots and provided png files of the plots. Additionally, high precision and recall shows that binary classification is learnt well within an adaptable memory context for the system, problem solving-wise.

There is a low error in general, along with per class and overall f-scores and metric recalls and validation scheme scores being extremely high.

## ***For both CNN models:***

*A special note for both CNN models is that categorical cross entropy to compute loss is used instead of binary to expand the chances of a generalized cross entropy loss comparison across specific 2 cases of labels.*

The results are remarkably similar in terms of precision, recall, accuracy, and MLP loss, (along with general loss), even after change of epochs by a factor of double, and not early stopping, but plateauing the learning rate from an initial of 0.001. The loss and accuracy trends for training and testing are remarkably similar, but slightly different, with CNN 1 better are training accuracy, and CNN2 better at testing accuracy. In addition, the confusion matrix provides for a similar true prediction to true labels split, of a high record.

The chief difference can be observed slightly in the metric value classification report of the macro and precision to recall statistics, and in the 10 random samples of binary classification of X-rays and their confidences of predictions. This shows the key difference is minute, but present for predicting NORMAL labels with marginal increase in probability for CNN2 and for PNEUMONIA labels with marginal increase in probability for CNN1.

Tensor implementations for both CNN's were used with increasing kernel size for optimal computational cost to complexity ratio, and in addition to that, the key idea conceptually here was to ensure feature detection and classification (binary, for Pneumonia or not) based on that parameter set. This can be extended through pre-trained models that the code generates and stores as a hdf5 file format, by using the file as a loaded model, and training it in the same training schema already setup, corresponding to each of the CNNs. In future exploration, it may be interesting to see the benefit to cost ratio of training a pre-trained model on this dataset and seeing potential metric improvements.

## For the Deep CNN Model:

We can see for a similar sample of input images and that of the label splits across training and testing, that the model has an exceptionally high training accuracy, of 99.75% (at peak), which

stagnates to around 75% for training data. This is without early stopping, to let the Deep network train with unrestricted hyperparameters.

A more consolidated trend for random samples is seen here, where 0.6 or less (60% or less) of confidence in a prediction is a clear demarcation for a false label, while anything else is a correct correlation, about 9 out of 10 times for the same model. With the same levels of training and testing split though, the precision, recall and f1 scores are the same for a variety of factors within the layer related comparison of the CNN 1 and 2 and the Deep CNN. Chiefly, the Deep CNN having convolutional layers, which is void in the CNN's (dense instead for layers), provide an aggregation of precision scores for the 2 labels.

This is reflected in the confusion matrix as well, where the expected trend of true-true correlations for the NORMAL and PENUMONIA labels are seen, though with about 95% of each classification adhering, and the rest being a false positive (or negative) relation. These predictive measures are still valid in an active application situation, due to the high performance of the system.

In general, since the classification was binary, and that the images were complex in resolution and features, the models performed well overall, with varying pros and cons at their complexities and architectures.

**Comments on the timing metrics of each of the networks in general:** Overall, there is an increase in training time, for the same constant data loading time onto the networks train set layer 1 (input layer, taking 3-4 minutes each). The increase is observed with increasing connections within the hidden layers of the network, hence from CNN 1 to CNN2 to Deep CNN. The overall train time ranges thusly, from 10 minutes to 20 minutes. Generating the data on predictions from the model takes 2 more minutes after. The accelerated GPU of the Google

Colaboratory (or Colab) system helps in most of these processes, with testing itself taking 1-2 minutes on heavy datasets such as the one used for this project.

## For the VGG Neural Network:

To begin, we can see that there is a decent training accuracy score of around 74% using 50 epochs in the 16 layer deep neural network. However, it is quite evident that the testing data has not improved over the course of the learning period. This can be due to an incorrect implementation of preprocessing the data because the VGG model that the keras API provides requires a specific input for the model or the model requires more fine-tuning because of the nature of extensive networks such as this. We were able to attempt at a more keras-catered model (in the vggv2 notebook) with certain image size inputs and utilizing data augmentation for the preprocessing of the data but was unable to complete this because of time constraints and how long it took to compute each epoch (taking around 30-40 minutes for 50 epochs). Something that can be considered for future implementation is to use pre-trained VGG weights while also increasing the total number of epochs. Despite the stagnant testing data, we were fortunately able to visualize the decrease of the loss metric over the learning period.

In the 10 randomly selected samples, 4 out of the 10 samples were incorrectly predicted with a confidence value of about 50%. Not much else can be said about the VGG network from our results due to the stagnancy of the learning rate over the period of 50 epochs.

Overall, with proper implementation of the VGG neural network, we would have hopefully seen a better accuracy in depicting pneumonia cases in making use of its pre-trained ImageNet which consists of more than 14 million images belonging to 1000 different classes.

## *Conclusion:*

In conclusion we can see from the analysis of the 2 CNN models and the DCNN that for the metrics of loss, accuracy, f1, recall, and precision, with tallied random samples for confidences in predictions, that there is nominal increase in accuracy and variance in predictive sets for tested data, even when considering a binary classification schema. From the CNN with lesser layers, to the evolved version with higher layers, to a further deeper classification of a CNN architecture, the accuracy and precision increase, while recall rates, with a loss decrease, are varied but follow similar trends in the rise, and drop of expected metric points.

For classifying Pneumonia or Normal conditions from X-Ray images alone (compressed and resampled to 200px-200px, as the lowest resolution), it is possible, even with any of the CNN models devised here, to an accurate enough level (say 70%+). Of course, with modern technology, this is advised to be paired with human judgement, but the fact that a singular grayscale image can be used to give a sensible diagnosis (with the features being extracted to higher detail with increasing the "deepness" of the model), is an astonishing find.

The models all work effectively and are lightweight for future use and loading, and to predict any similarly sized data within the parameters of the given labels. To explicitly summarize the facts we find, a higher accuracy whilst having lower loss is observed with ~5% ranges from CNN1 and CNN2 to Deep CNN 1. The precision and recall values follow a similar pattern, though the exact processing of training data suggests a lot to do with those datapoints.

With the VGG Net, however, the trend seen is slightly different in that the network has a high rate of adaptability and learning on feature-rich domains, such as these tested here. We see around the same range of accuracy in sense of reported values, and in terms of metrics and their precisions and recalls, compared to Deep CNN one as well, considering the further "deepness" of this model with multiple convolutions occurring. The loss is annealed linearly, which is peculiar considering the nature of exponential loss decay and high retention in the DCNN. This model seems to work effectively, though with an extremely high compile and train time due to an extremely elevated level of complex calculations required, it may not be feasible to utilize given

marginal or no benefits in confidence of predictions (with increased variance in random variables of testing data of X-Rays).

Along with this, the issue seems to be in its varied complexities seeming to overfit and hence under analyze the nuances structural intricacies in each X-Ray relative to each other. Typically seeing the DCNN alongside a further deeper counterpart, we see the local minima and error reduction for this seems to be a challenge, due to the skewed classification report suggesting abnormal f1 and recall values (higher than expected, lower retention), but with apt (as expected considering the AlexNet) levels of precision in each class. Furthermore, it only makes sense to compare the VGG to the AlexNet due to the nature of both nets matching more (and it more interesting to see how they perform relative to each other, given this "problem set to work on").

## *Presentation summary:*

Summary of work is presented in the presentation video covering this report, the code itself, generated output on command line (training trends, tables, and computed metric datapoints), along with evaluation of plots and trends (or other visuals for data analysis, including testing image accuracy by random sampling). The presentation gives the overview of the problem statement, and explains the motivation, challenges, and other interesting technical aspects (and implementation related quips (or rather, unique aspects) conceptually for the project and its entailed goals.

There are 2 YouTube videos associated with this project, explaining in an overview format, the technical aspects of the implementation of each neural network, and a brief touch-up about the helper functions, results, and the problem to solve. 1 Video is a standalone video for all nets other than the VGG (with Arya Keni as the speaker), and the other video (with a 13 min length, and a link in description to a slightly speedup 10-minute version (and vice versa)) covered by Arya Keni and Sung Chang talks about the complete code implementation. *To clarify vgg.ipynb*

*highlights the complete code for the VGG, deep_cnn_ee456.ipynb highlights the complete code the AlexNet, and the cnn_ee456.ipynb highlights the complete code for the CNNs (both non-deep versions).*

# *Appendix:*

All the code is hosted on GitHub at this repository:
https://github.com/codersupreme99101/NeuralNetwork_FinalProject_FA22

It is open source. Result images are also contained in there, with a link to the original dataset (email me for access to dataset, as it is currently under request access mode). Only 3 references are made from reputable sources to add focus to the report's findings, and the inspiration and theory gained from research papers only as necessary.

**Links for Videos:**

Video Link for code run-through and explanation (not including VGG component, and detailed overview of other parts of the matrix operations behind layers, with Arya Keni only):
https://youtu.be/2lioEtFuvms

Video Link with full team (Sung Chang and Arya Keni) for overview of full codebase:
https://youtu.be/XoAedmABFWw (13 minutes, to account for an extra NN, with full team).

(On YouTube for most accessibility, link covering all details of technicality as needed).

## *Workload distribution:*

Old Workload Distribution (Set 1):

The work is evenly split with one member handling training, validation and initialization, and the member handling the training also handled the plot generation. Since the code is implemented in python, with existing python open-source global libraries, there are no logistical issues of support. Additionally, if there is time for a DCN implementation, the workload is split up the same way, though each team-mate worked on different tasks, if necessary, within these categories.

Arya Keni did the 2 CNN models from full training to analysis (validation, initialization, plot, and data+results+table generation), and Sung Chang did the same (training to analysis) for 1 DCN. Additionally, this report, the proposal, the model generation, and the dataset procurement (+workload pipeline setup on GitHub), was done by Arya Keni.

***Note: Revised Workload distribution (Set 2):***

*Arya Keni did all the work required for the project: handling training, validation and initialization, and the member handling the training also handled the plot generation. Arya Keni did the 2 CNN models from full training to analysis (validation, initialization, plot, and data+results+table generation) and then did the same (training to analysis) for 1 DCN (or DCNN or Deep CNN). Additionally, this report, the proposal, the model generation, and the dataset procurement (+workload pipeline setup on GitHub), was done by Arya Keni.*

*Sung Chang handled all aspects from training and testing by using functions of evaluation from the DCN and the same run sequence for the DCN (dataset preprocessing and so on), but applied to a custom VGG network, along with its unique training and testing. Analyzing the statistics and results for the same was done from the previous written helper functions by Arya but for the VGG and written on the report.*

# *References:*

Pneumonia Detection Using CNN based Feature Extraction: Varshani D, Thakral K, Agarwal L. Et. Al (2019 IEEE ICECCT), Jun. 2015

Pneumonia Detection in chest X-ray images using an ensemble of deep learning models: Kundu R, Das R, Geem Z (PLOS ONE), Sept. 2021

Pneumonia Detection on Chest X-Ray Images Using Ensemble of Deep Convolutional Neural Networks: Mabrouk A, Redondo R, Dahour A, Et. Al. (Applied Sciences MDPI), May 2022

Ensemble of CheXNet and VGG-19 Feature Extractor with Random Forest Classifier for Pediatric Pneumonia Detection: Nahida H, Md Mahmodul H, Md Mahfuz R, Et. Al (SN Computer Science), Nov. 2020