Automating Pneumonia Detection in X-Ray Images Using Deep Learning

K. Nguyen, I. Ostaptchenko, G. Pingitore - April 26, 2020 - WSU IE7860

Ken Nguyen
College of Engineering
Wayne State University
gm4354@wayne.edu

Igor Ostaptchenko
College of Engineering
Wayne State University
igor_ost@wayne.edu

Gabriel Pingitore
College of Engineering
Wayne State University
gabrielpingitore@wayne.edu

Abstract

The goal of this project was to develop a Convolutional Neural Network (CNN) able to correctly classify and automate the detection of pneumonia in x-ray images.

Introduction

The medical industry has been increasing focus on automating various medical operations, particularly in diagnostics. Time is an important and finite resource for both patients and medical staff. Patients with serious illness get worse the longer they have to wait for a diagnosis. Medical staff feel the pressure to work faster and are likely to come up with inaccurate results. (Francis). Therefore, different automated methods are being developed to properly diagnose patients quickly, accurately, and essentially automatically. One such way is through machine learning.

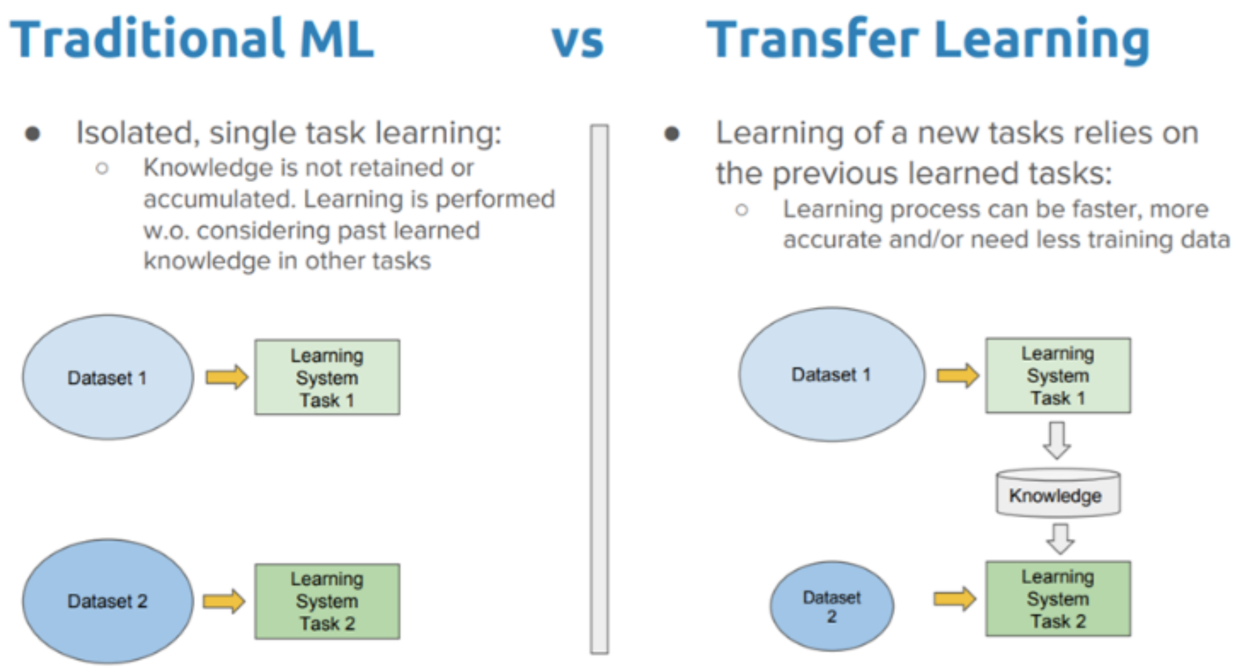
Currently, the world is in a state of lock down due to COVID-19 or Coronavirus. According to the World Health Organization, "Around 1 out of every 6 people who gets COVID-19 becomes seriously ill and develops difficulty breathing."(World Health). As a group, we have concerns about the virus and wanted to do our part to attempt to help out by having an alternative method to identify the virus if testing is not available. We decided that since many of the infected have lung issues, that we would build a classifier that could positively identify the virus. However, the issue we ran into was that we could not find enough scans of patient lungs with COVID to work with. We then started to explore the possibility of diagnosing lung issues that would be similar to COVID-19.

We decided to focus on diagnosing pneumonia on x-ray images since a model that could be trained by x-ray images of lungs for classification should be able to be trained to detect other lung conditions once enough data became available.

Methodology

"Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image. For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults." (Deshpande). There are many ways to accomplish image classification with the two most popular being Convolution Neural Networks (CNN) and Autoencoders. In this project, we decided to focus on these two methods.

Due to our data being so sparse, we focused building two separate CNNs using a different pre-trained network, also known as transfer learning, for each and building an autoencoder for our final network. "Transfer learning, where information from one dataset is used to inform a model on another,


Traditional ML
vs
Transfer Learning
Isolated, single task learning:
Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
Learning of a new tasks relies on the previous learned tasks:
Learning process can be faster, more accurate and/or need less training data
Dataset 1
Learning System Task 1
Dataset 2
Learning System Task 2
Dataset 1
Learning System Task 1
Knowledge
Dataset 2
Learning System Task 2

Traditional Learning vs Transfer Learning can be an effective tool for bridging sparse data while preserving the contextual differences in the underlying measurements." (Hutchinson). (Sarkar)

Many different image classification problems have been solved with transfer learning. "Transfer learning is a popular method in computer vision because it allows us to build accurate models in a timesaving way (Rawat & Wang 2017). With transfer learning, instead of starting the learning process from scratch, you start from patterns that have been learned when solving a different problem. This way you leverage previous learnings and avoid starting from scratch." (Marcelino) This allowed us to have networks trained on many different images with weights assigned accordingly so we could have increased results from our model.

We used the basic network configuration for each model as our baseline to see where improvements could be made.

The tuning for transfer learning has three different options: Train the entire model, Freeze the convolutional base, and Train specific layers and leave others alone. We ran different versions of these with one version training layers on top of the base model, with the weights of the pre-trained network not being adjusted. We also ran a network where we trained a small number of the top layers weights to attempt to learn the proper weights of our data while freezing the lower layers of the model to keep the original network weights.

We also included callbacks with early stopping on our final models. Too many epochs can lead to overfitting of the data and conversely, too few can lead to under-fitting. We implement early stopping that will allow us to have enough epochs while allowing us to stop should our performance degrade. In addition, we trained our model to retain the best weights during the training with the Model Checkpoint function in order to optimize the network

Additionally, we build an auto-encoder which is a type of feedforward neural network where the input is the same as the output. Our auto-encoder was built with the intention of compressing the images, finding the most relevant features, and using those features for feature classification. However, we ended up having issues with the implementation of the model and were not able to get this model running correctly. Due to the fact that we couldn't get it running, we were not able to implement any tuning of the model.

Scoring

The scoring metrics chosen for model performance evaluation were Accuracy, Precision, Recall, and F1 Score. Accuracy is a measure of how often our model is correct. It's given by the equation:

$$Accuracy = \frac{truepositives + truenegatives}{totalexamples}$$

Precision is of the times the model predicts positive, how often is it correct? It is given by the equation:

$$Precision = \frac{truepositives}{truepositives + falsepositives}$$

Recall helps to explain how sensitive the model is to identifying positive classes when the cost of false negatives is high. It is given by the equation:

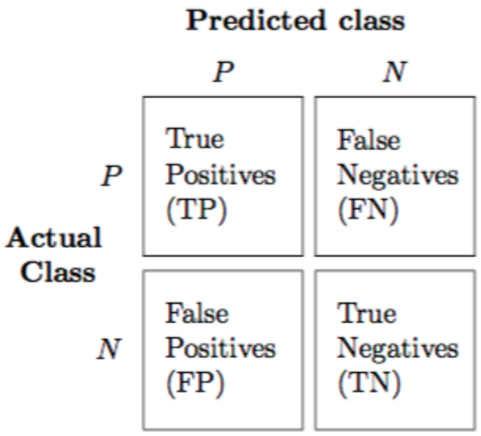
$$Recall = \frac{truepositives}{truepositives + falsenegatives}$$

F1 is an overall measure of a model's accuracy that combines precision and recall. It is given by the equation:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

"A good F1 score means that you have low false positives and low false negatives, so you're correctly identifying real threats and you are not disturbed by false alarms. An F1 score is considered perfect when it's 1, while the model is a total failure when it's 0." (Nicholson).

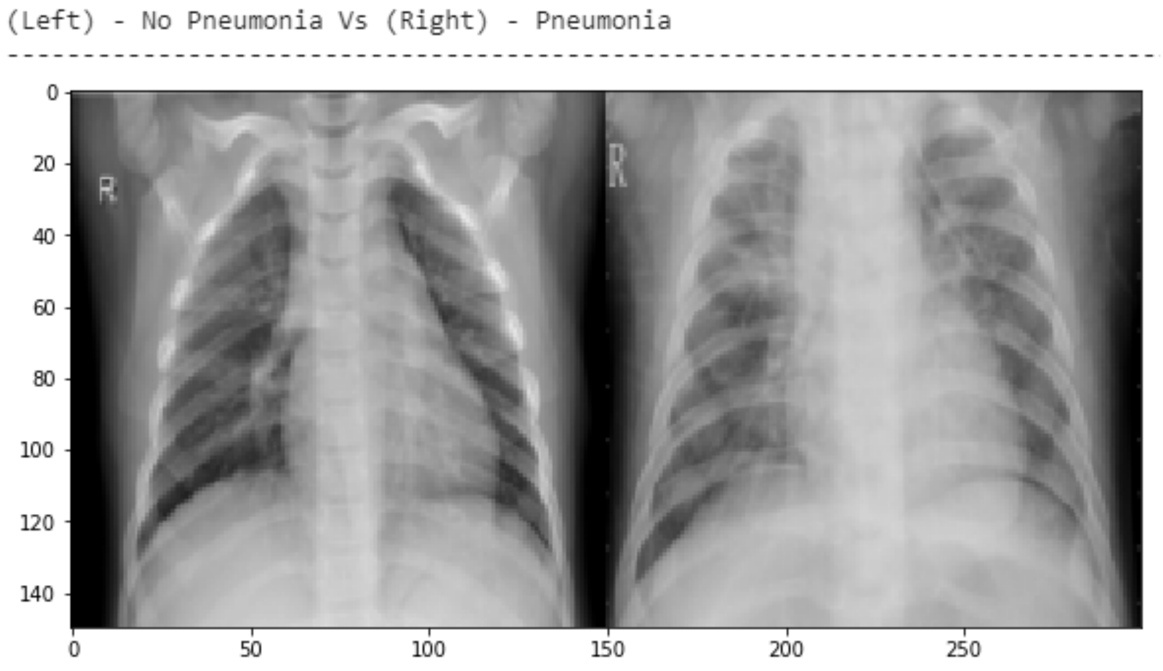
In general, we want our model to perform to reduce false negatives. Put another way, we are okay with indicating there is pneumonia when there actually isn't as further tests will disprove this. However, we are not okay with saying a patient is healthy when they are not as this would increase the patient's chance for death. Additionally, we used a confusion matrix to help visualize our results. An example can be seen here.



		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Data

The base data used for training and testing the models was procured from Mendeley Data (Kermany), an open source data company who focus on providing data for increasing the pace of scientific discovery by verifying or generating new findings on data. The data is image data separated into testing and training data sets with images being classified as having pneumonia or not having pneumonia. An example can be seen here.



```
(Left) - No Pneumonia Vs (Right) - Pneumonia
--------------------------------------------------------------------
```

Figure 1 - Images of lung

without pneumonia and with pneumonia.

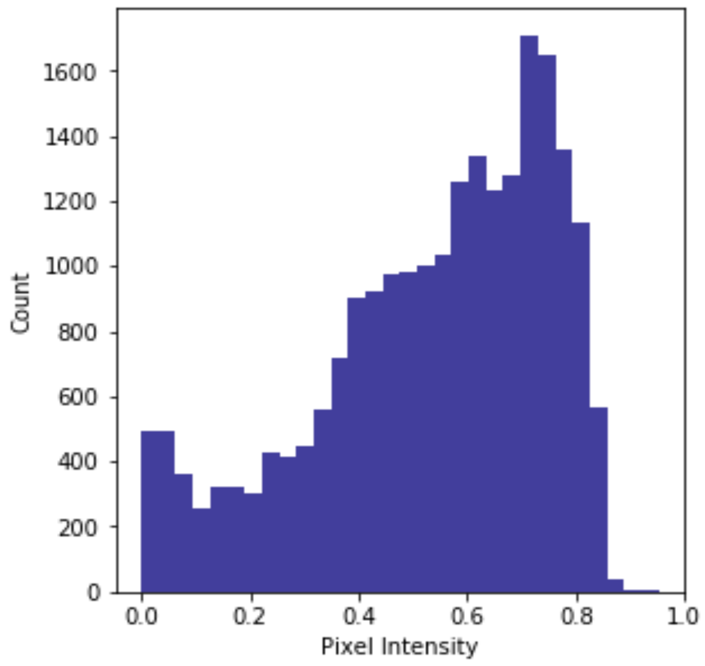
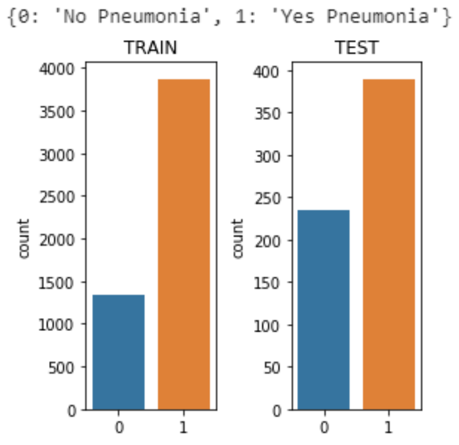


Figure 2 - Histogram showing the pixel intensity of the different images. As can be seen, much of our data has high pixel intensity in much of the image. Our models will learn to distinguish the variation.



```
{0: 'No Pneumonia', 1: 'Yes Pneumonia'}
```

Figure 3 - Breakdown of Train and Test datasets showing breakdown of lungs without pneumonia and with pneumonia.

Here we can see there is a large amount of data that would indicate the patient has pneumonia compared to those which do not. This may lead to our model to be biased towards the patient having pneumonia when in fact they do not. This will be solved by under sampling which is a technique used to adjust the class distribution of a dataset. We chose to under sample the data due to the fact we did not want duplicate images included so that all examples would be unique to train on. We will bring the lower represented class representation to be on par with the dominant using `RandomUnderSampler` and encode the response variable into binary two category encoded.

```
ros = RandomUnderSampler(sampling_strategy='auto')
X_trainRos, Y_trainRos = ros.fit_sample(X_trainFlat, Y_train)
X_testRos, Y_testRos = ros.fit_sample(X_testFlat, Y_test)
# Encode labels to hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
Y_trainRosHot = to_categorical(Y_trainRos, num_classes = 2)
Y_testRosHot = to_categorical(Y_testRos, num_classes = 2)
```

Once we complete this, we have the following results.

```
{0: 'No Pneumonia', 1: 'Yes Pneumonia'}
```

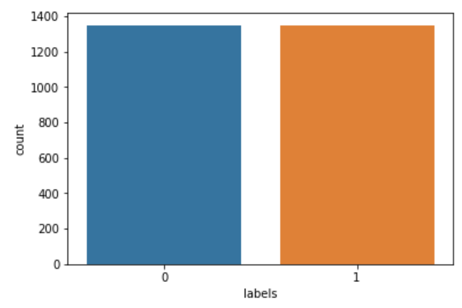


Figure 4 - Breakdown of Train and Test datasets showing breakdown of lungs without pneumonia and with pneumonia after under sampling

In order to get better performance, we also compressed the images. Additionally, we had to reshape the data in order to make it workable for our machine learning, downscale the images to match the dimensions for the small image available so our models could properly ingest and learn from them, and also resized images so they would have equal shapes.

Modeling

Helper Functions

We started building the model by creating helper functions for visualizing model results such as the learning curve and confusion matrix as well as other features. We took advantage of several Keras features for our models. EarlyStopping allowed us to stop training the model when the validation accuracy of the model stopped improving. ModelCheckpoint allowed us to save the model after each epoch. ReduceLROnPlateau automatically reduces our learning rate when our validation accuracy stopped improving with the hope of further fine-tuning model weights. All of these functions help refine the models as the data is passed through in order to have better generalization.

Baseline Models

Due to the size of our dataset, we decided to use two different pre-built models; namely, VGG16 and Google's Inception V3.

"VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition"." (Hassan). The model is trained on the ImageNet dataset which has over 14 million images with 1,000 classes. The architecture of the network can be seen below.

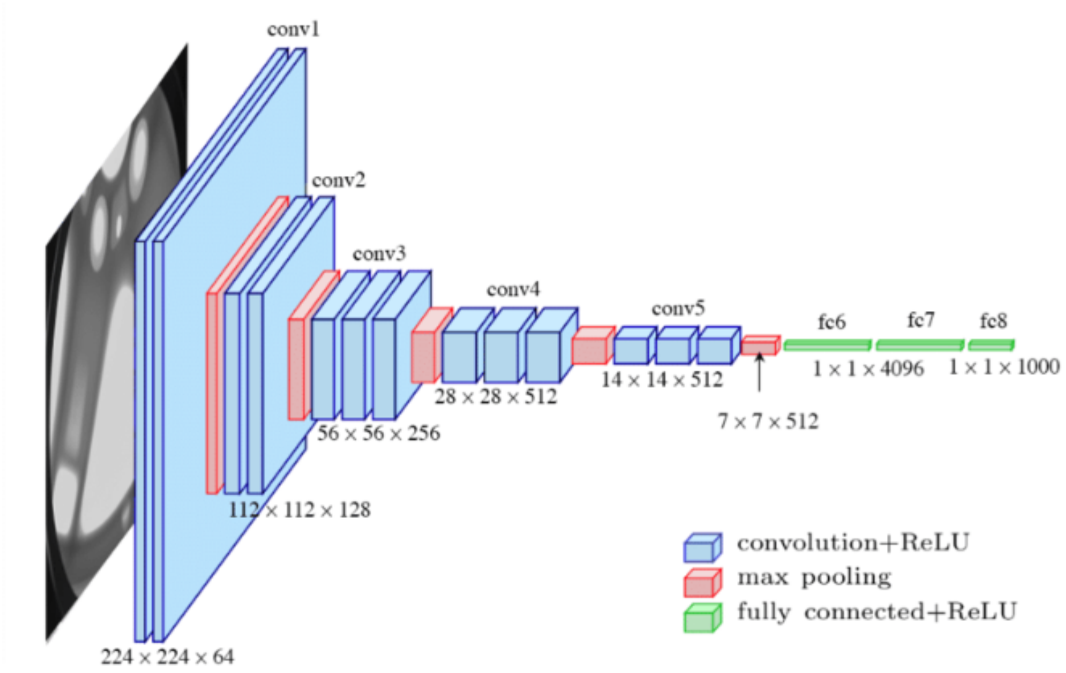


Figure 5 – VGG16 Architecture

This network is trained and already has predefined weights associated with each layer of the network. This helped us in that the network already has been pre-trained to classify images over a number of different classes. The ”Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights.” (Thakur)

Google’s Inception V3 model is the other model we decided to use. “Inception v3 is a widely-used image recognition model that can attain significant accuracy. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al. The model has a mixture of symmetric and asymmetric building blocks, including: convolutions, average pooling, max pooling, concats, dropouts,

fully connected layers. Loss is computed via Softmax." (Google). Below you can see the model architecture.
(Tsang)

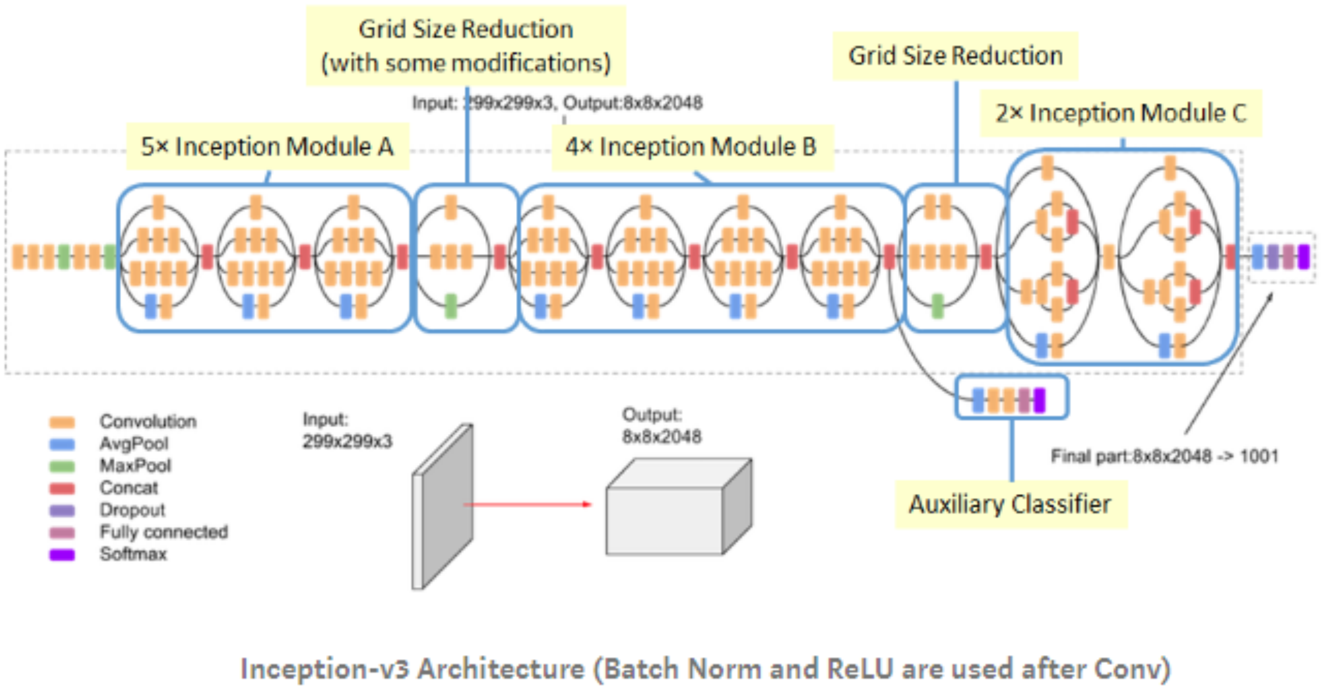


Figure 6 - Inception V3 Architecture.

The V3 model was also trained on the ImageNet dataset. This is perfect for comparison of the performance of the VGG16 and V3 since they have the same source of truth. The architecture and weights are the differentiators between the two. The base models used the pre-trained models' weights and didn't update the weights during the training.

The attempt was made to train the classifier using the convolutional neural network and convolutional autoencoder (Sharma). One of the application of the of autoencoders according to (Bok,Langr) is: "… as a *one-class classifier* (an anomaly-detection algorithm), where we can see the items in a reduced, more quickly searchable latent space to check for similarity with the target class." While the initial results were promising showing the final confusion matrix shows the signs of overfitting.

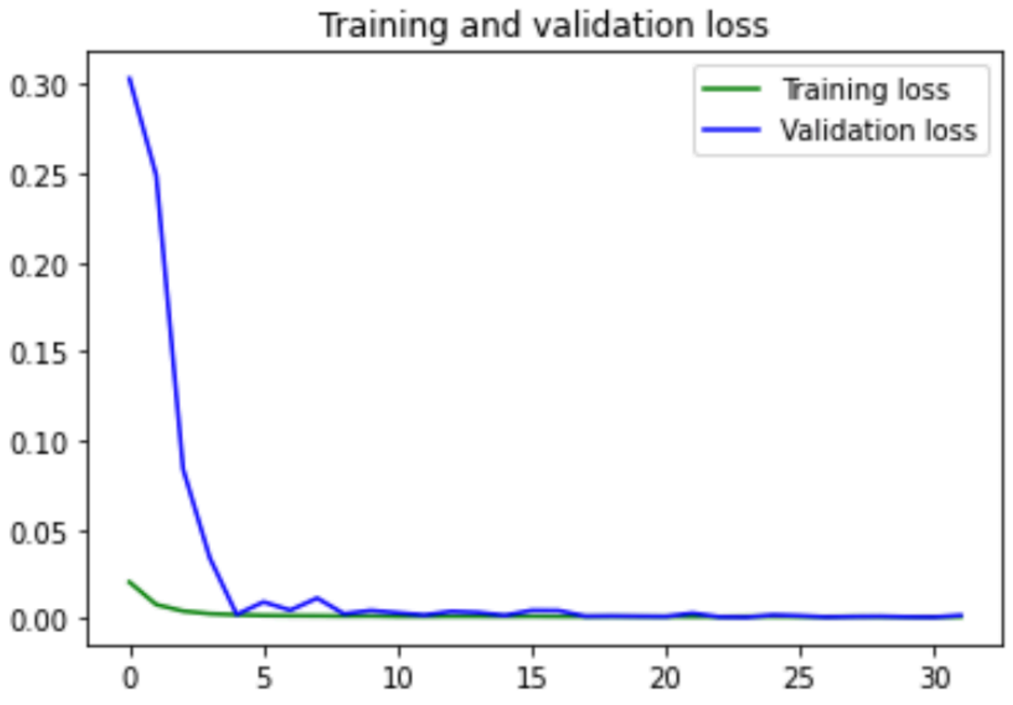


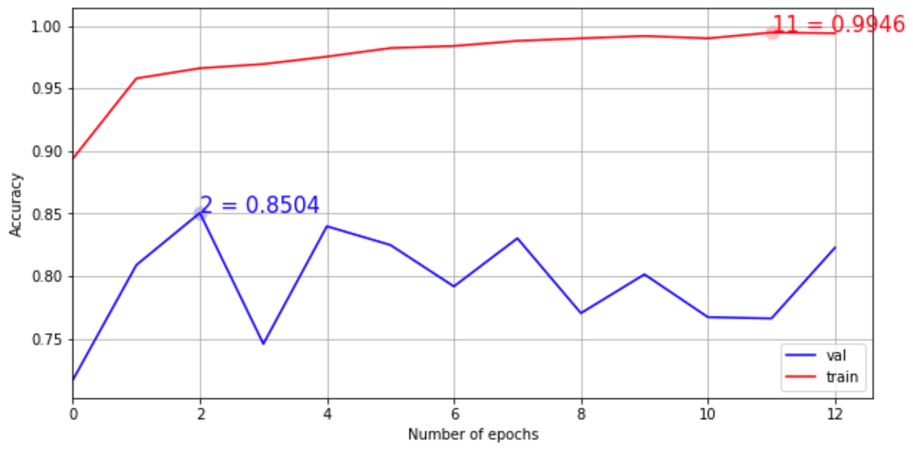
Figure 7 - Autoencoder training.

Fine Tuning

Once we trained the base models and got the results, we proceeded to attempt to tune the networks to improve performance. We focused on unfreezing layers as well as early stopping.

Unfreezing Top Layer

“One way to increase performance even further is to train (or "fine-tune") the weights of the top layers of the pre-trained model alongside the training of the classifier you added. The training process will force the weights to be tuned from generic feature maps to features associated specifically with the dataset.” (TensorFlow). The top layers of the neural networks have weights that greatly influence the resultant models. We “unfreezed” the top layers so that the model could learn the weights for our network while keeping the lower layers of the network frozen to keep the trained generalization of the prebuilt models. This provided the best of both worlds in a network that was trained on the new dataset while also keeping some of the power of the original network.



Early Stopping

According to Jason Brownlee, “A problem with training neural networks is in the choice of the number of training epochs to use. Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset.” (Brownlee). Early stopping is usually a good idea in most neural networks to help prevent overfitting. We found improvement in our model from it.

Results

Baseline Models

The results of our baseline models indicate that the **Inception V3** performed best overall versus the VGG16 as the number of false negatives was zero which is our target variable. However, we ran into issues of overfitting as evidenced by the wide divergence between the training and validation accuracy. Additionally, our model bias was quite high.

Figure 7 - Inception V3 Base Accuracy Results.

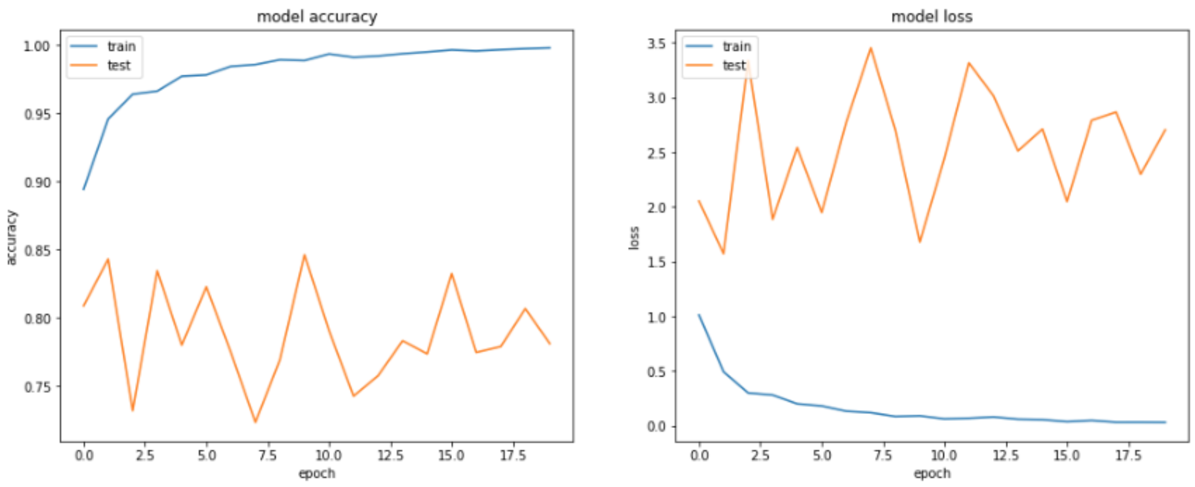
According to Bert Markgraf, "Bias is the error in estimates due to systematic mistakes that lead to consistently high or low results as compared to the actual values. The individual bias of an estimate known to be biased is the difference between the estimated and actual values." (Markgraf).

Fine Tuning

Once the baseline was established, we attempted to improve on these results with unfreezing the top layer and early stopping.

Unfreezing Top Layer

Unfortunately, when we ran the models with the top layers unfrozen, we saw a degradation of the models and their ability to generalize. On the VGG16 Unfrozen model, the best performing unfrozen model, we saw accuracy decrease and model loss increase. By altering the weights of the network



near the top, the models appeared to not be able to establish new weights properly for the network. This could be due to the sparseness of the dataset.
Figure 8 – VGG16 Unfrozen Accuracy Results.

Early Stopping

Early stopping is when the models started to show improvement. Results show VGG16 with early stopping is the best performing model based overall based on bias and the target of low false negatives. This model had a high degree of accuracy, near perfect accuracy, and a high f1-score which is a combination of precision and recall. Multiple combinations or early stopping were ran with a patience of 10 being the most generalized model we were able to create. Our model was trained on 2,698 samples and validated on 468 samples.

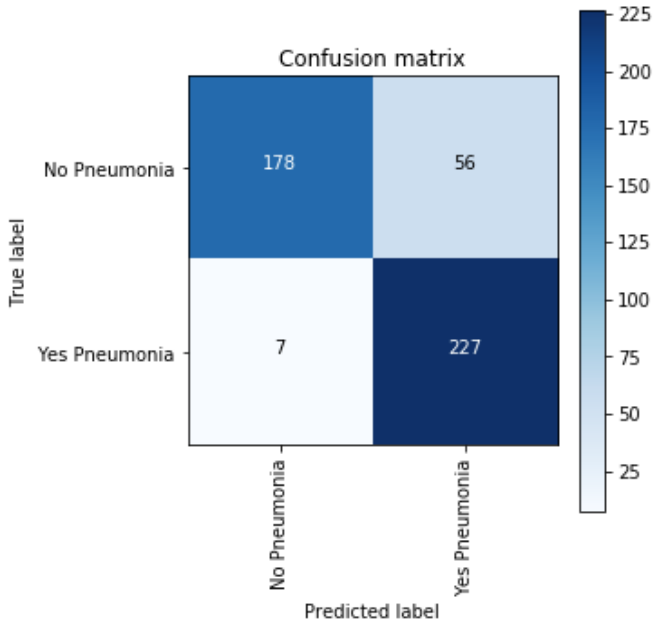


Figure 9 – VGG16 Early Stopping Confusion Matrix .

Final Overall Results

Model	False Negatives	True Negatives	False Positives	True Positives	Bias
VGG16 Base	2	141	93	232	20.30 %
InceptionV3 Base	0	142	92	234	19.66 %
Unfrozen VGG16	1	135	99	233	21.37 %

Unfrozen InceptionV3	1	131	103	233	22.22 %
VGG16 Early Stop	18	198	36	216	11.54 %
Inception V3 Early Stop	7	178	56	227	13.46 %
Unfrozen VGG16 Early Stop	16	194	40	218	11.97 %
Unfrozen Inception V3 Early Stop	13	185	49	221	13.25 %

Table 1 – Confusion Matrix Results – All Models with Bias

Here we can see the final results for all the individual models. The first four models did not implement early stopping while the last four did.

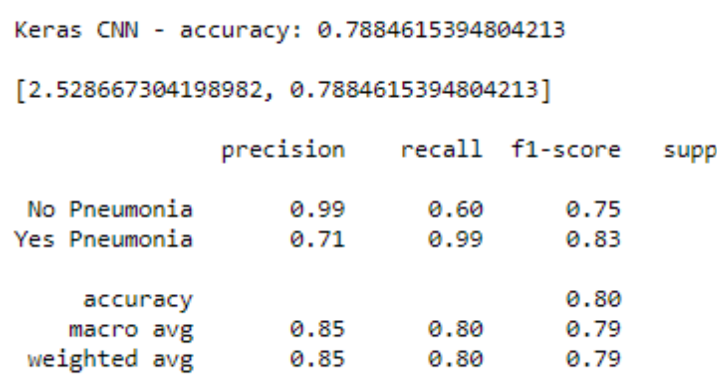
Conclusion

This research paper focused on image classification in order to properly classify cases of pneumonia vs no pneumonia. We used transfer learning to jump start our model training and augmented the training with unfreezing layers and early stopping. The baseline models with pre-established weights performed very well in general and were able to be improved on our data with early stopping. However, we reached a limit on the generalization of the network and the performance could potentially be improved with further hyperparameter tuning.

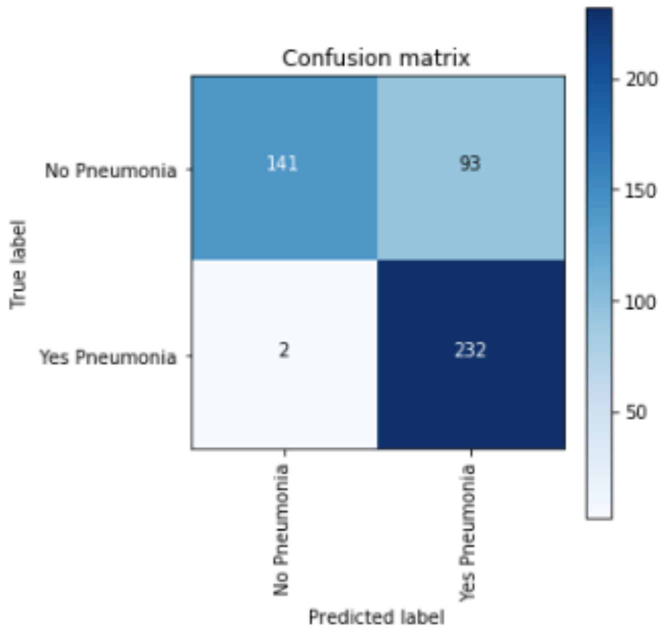
Our goal with this model was to provide a tool that could help the medical facilities reduce the time for diagnosis and also increase accuracy over humans doing the same work. We found that we were able to create a network that in general produces accurate results and of the false negatives that the model missed, these accounted for only for 3.84% of the total predictions.

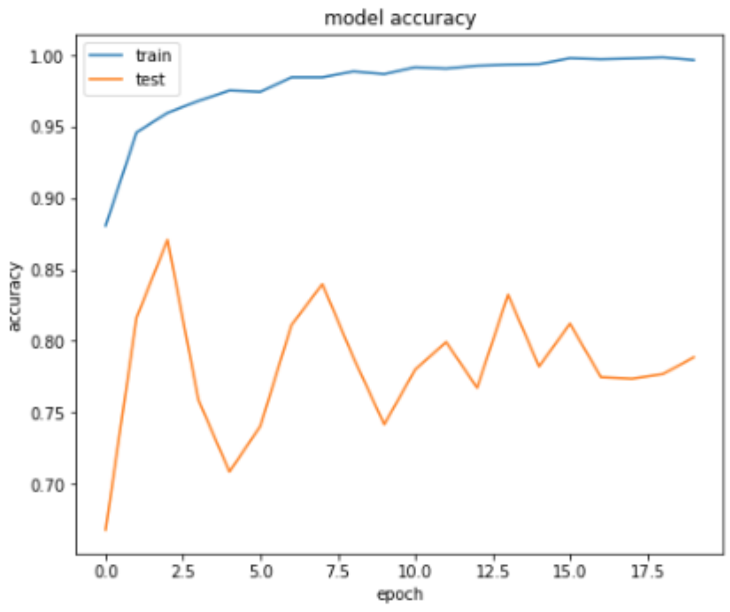
Appendix

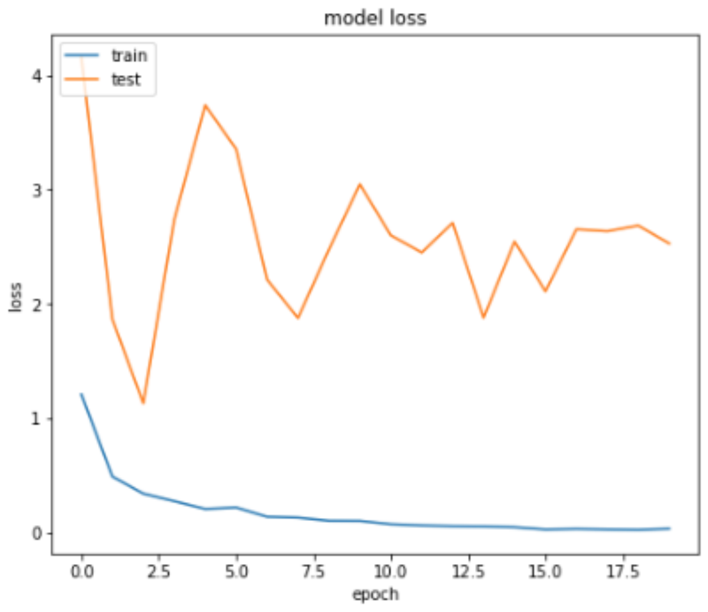
VGG16 Base – Full Results



```
Keras CNN - accuracy: 0.7884615394804213

[2.528667304198982, 0.7884615394804213]

               precision    recall  f1-score   supp

 No Pneumonia       0.99      0.60      0.75
Yes Pneumonia       0.71      0.99      0.83

     accuracy                           0.80
    macro avg       0.85      0.80      0.79
 weighted avg       0.85      0.80      0.79
```

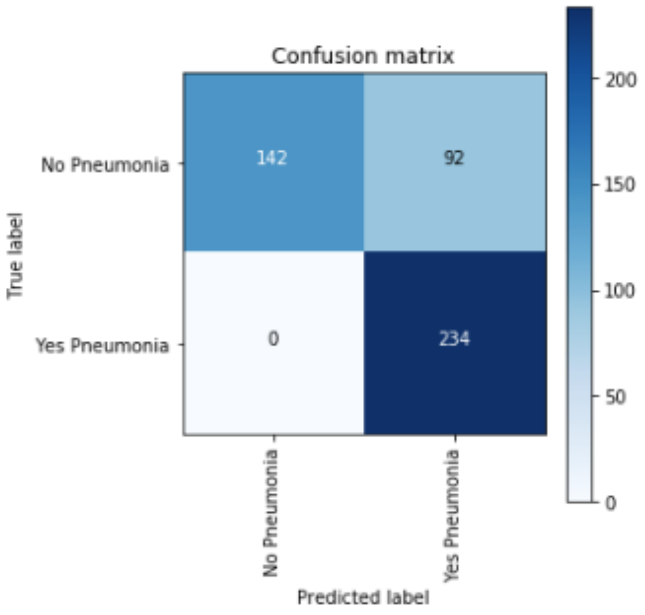


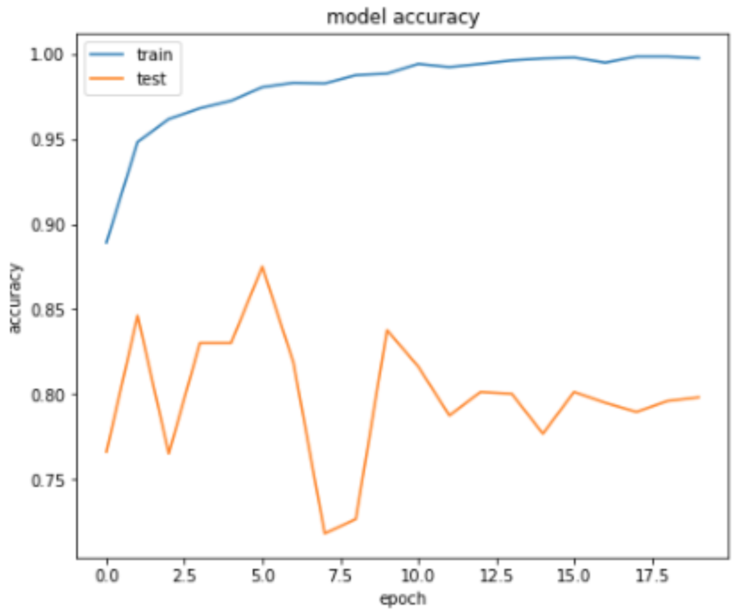


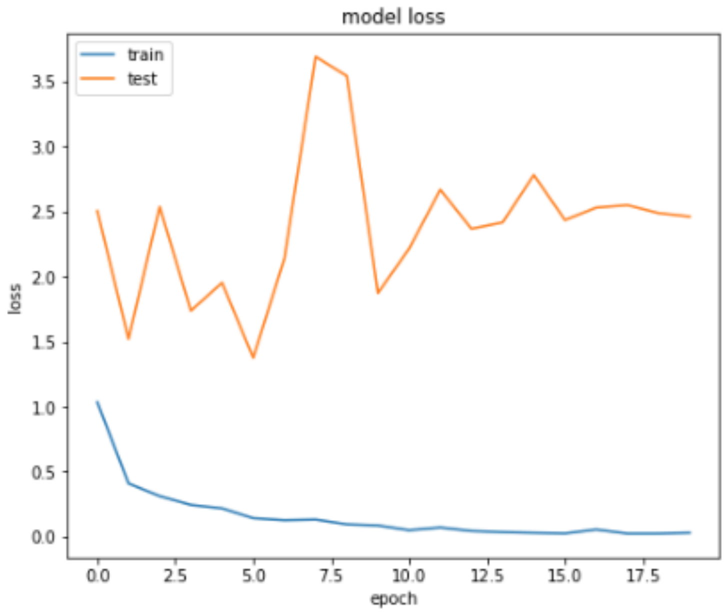


Inception V3 Base – Full Results

```
Keras CNN - accuracy: 0.7980769230769231

[2.4592587104273247, 0.7980769230769231]

              precision    recall  f1-score   support

 No Pneumonia       1.00      0.61      0.76       234
Yes Pneumonia       0.72      1.00      0.84       234

     accuracy                           0.80       468
    macro avg       0.86      0.80      0.80       468
 weighted avg       0.86      0.80      0.80       468
```

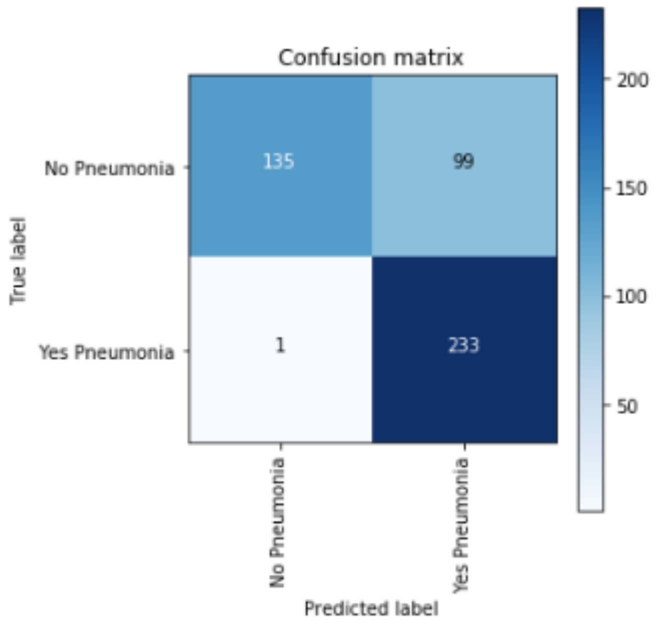


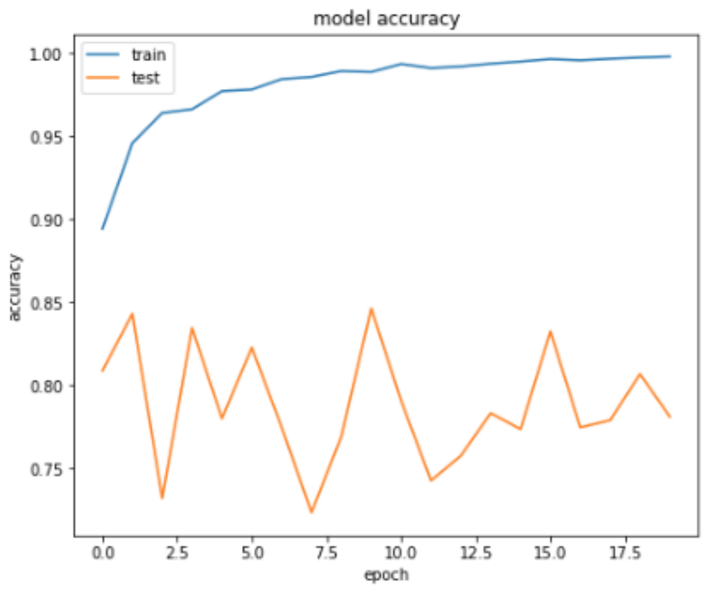


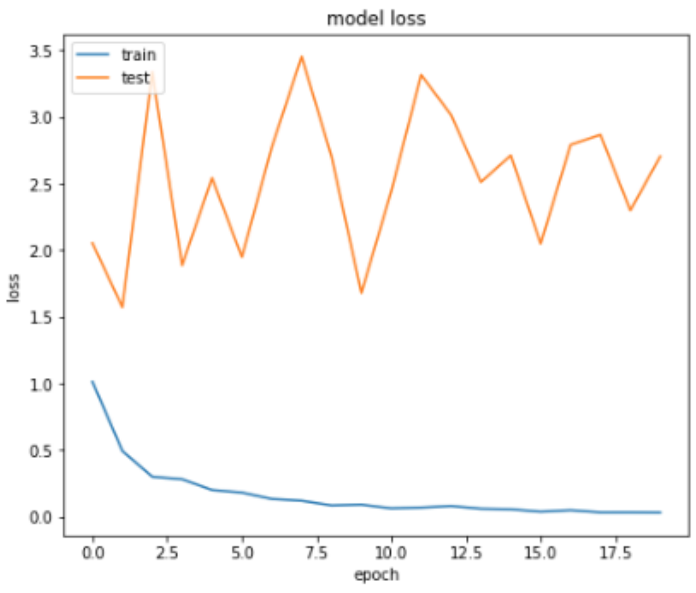


VGG16 Unfrozen – Full Results

```
Keras CNN - accuracy: 0.7809829070017889

[2.7052416460268827, 0.7809829070017889]

              precision    recall  f1-score   support

 No Pneumonia       0.99      0.58      0.73       234
Yes Pneumonia       0.70      1.00      0.82       234

     accuracy                           0.79       468
    macro avg       0.85      0.79      0.78       468
 weighted avg       0.85      0.79      0.78       468
```

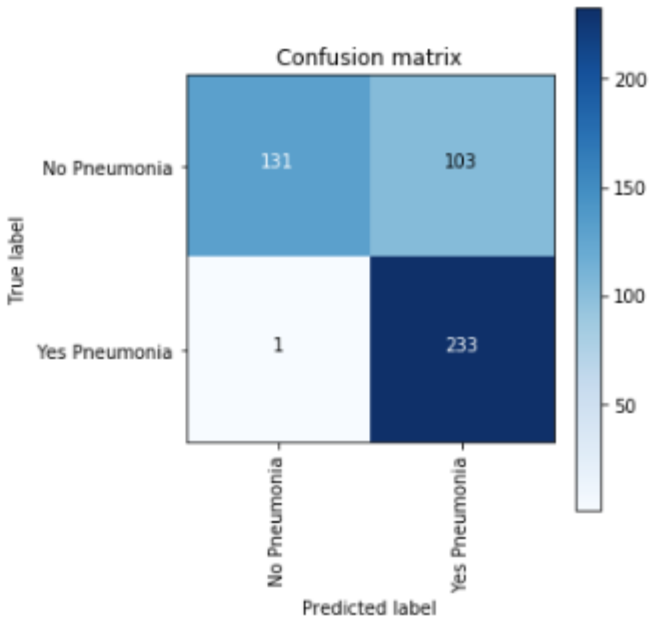


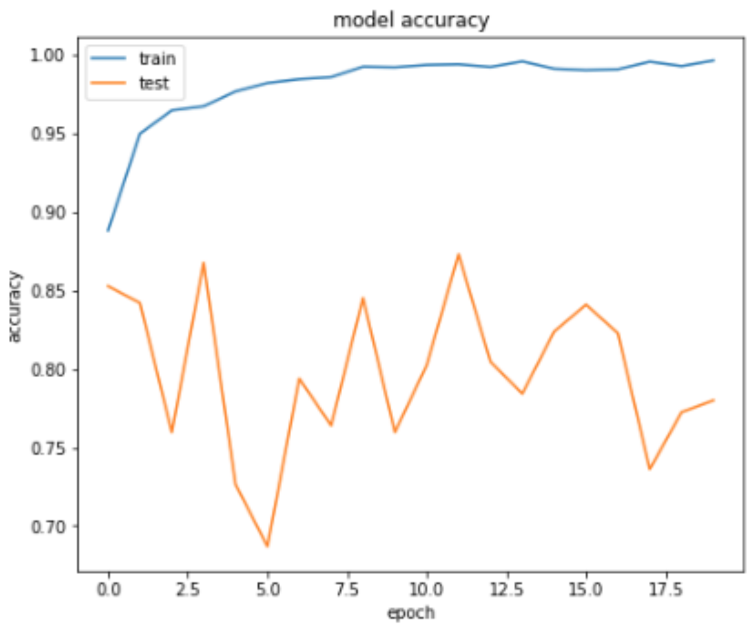


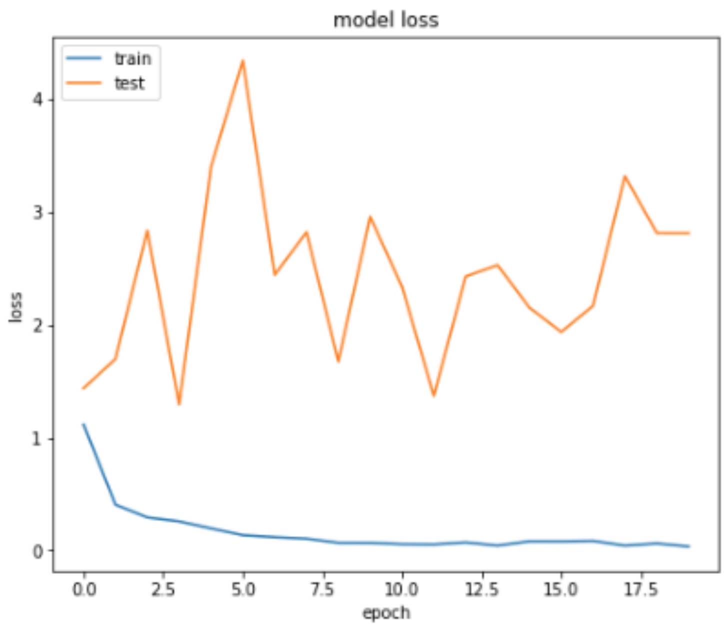


Inception V3 Unfrozen – Full Results

```
Keras CNN - accuracy: 0.7799145299145299

[2.8124447175905187, 0.7799145299145299]

               precision    recall  f1-score   support

 No Pneumonia       0.99      0.56      0.72       234
Yes Pneumonia       0.69      1.00      0.82       234

     accuracy                           0.78       468
    macro avg       0.84      0.78      0.77       468
 weighted avg       0.84      0.78      0.77       468
```

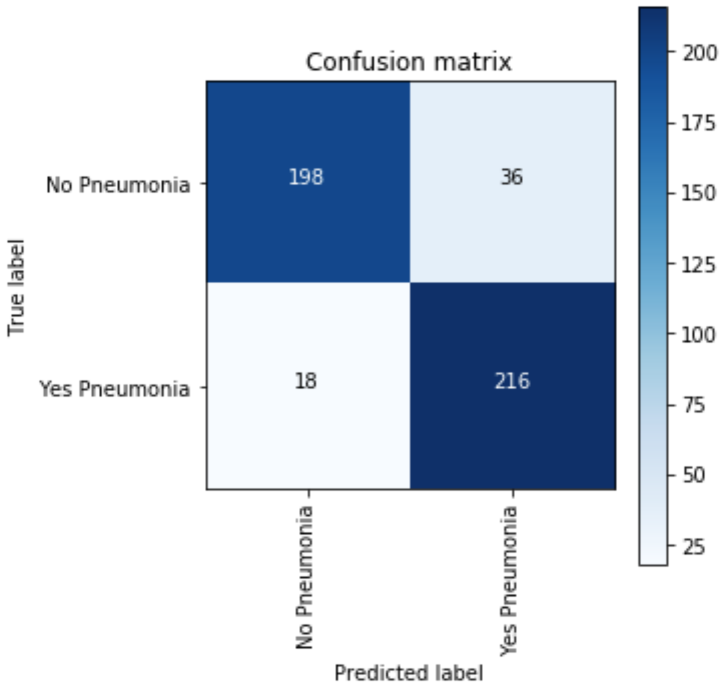


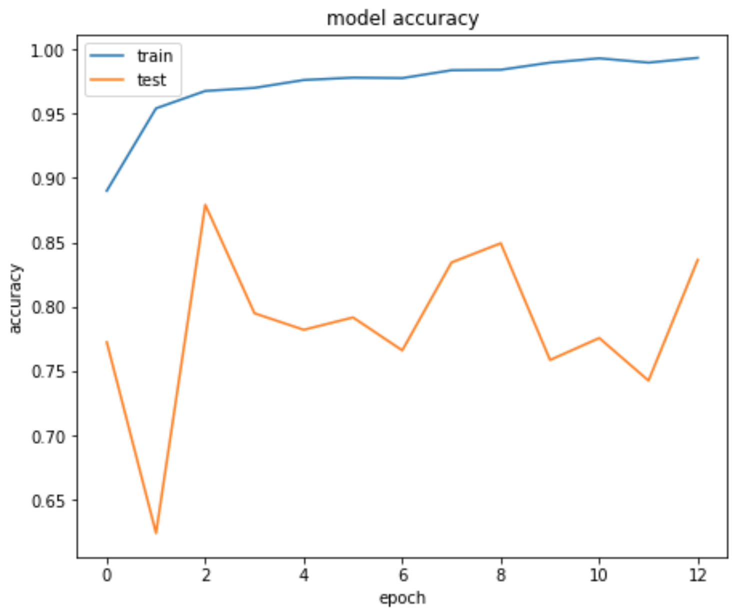


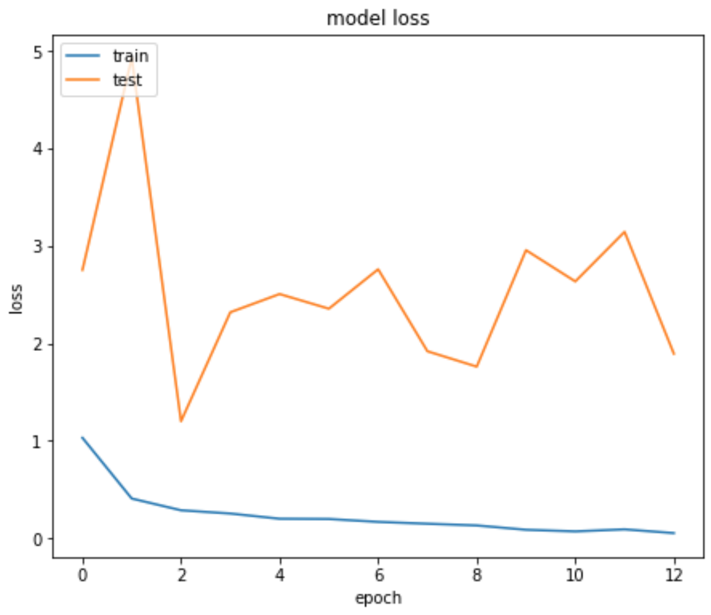


VGG16 Early Stopping – Full Results

```
Keras CNN - accuracy: 0.8792735052923871

[1.2014373605792275, 0.8792735052923871]

               precision    recall  f1-score   support

 No Pneumonia       0.92      0.85      0.88       234
Yes Pneumonia       0.86      0.92      0.89       234

     accuracy                           0.88       468
    macro avg       0.89      0.88      0.88       468
 weighted avg       0.89      0.88      0.88       468
```

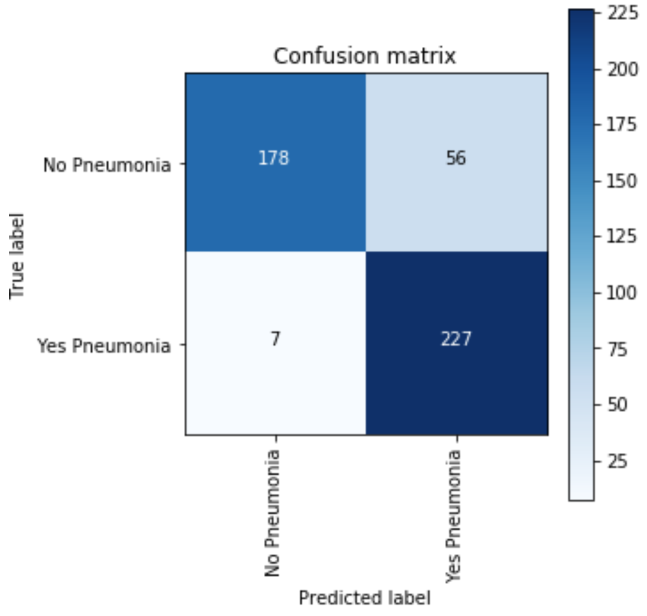


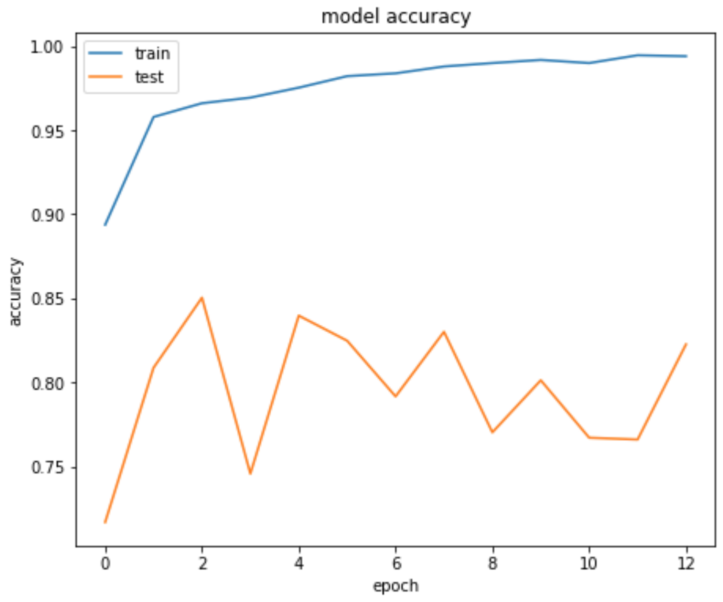


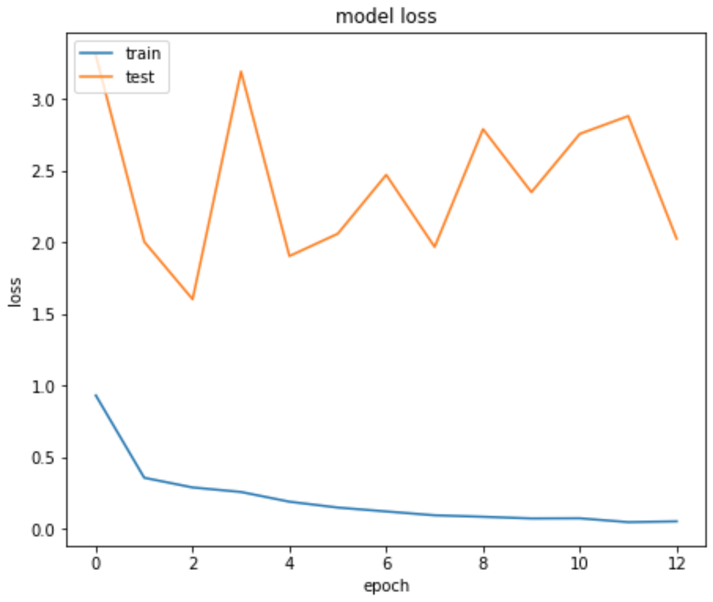


Inception V3 Early Stopping – Full Results

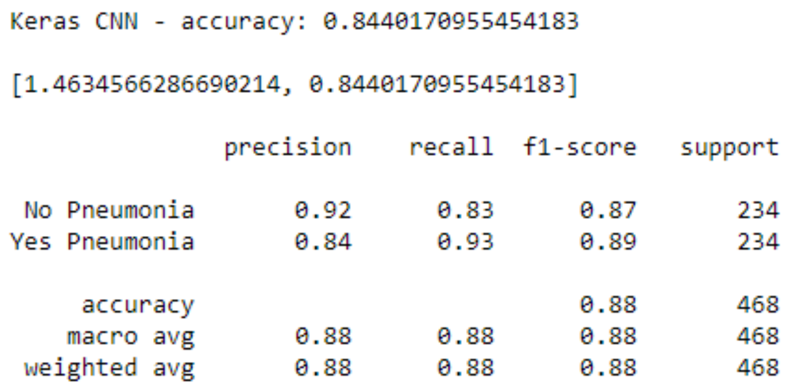
```
Keras CNN - accuracy: 0.8504273504273504

[1.6025327748809812, 0.8504273504273504]

               precision    recall  f1-score   support

 No Pneumonia       0.96      0.76      0.85       234
Yes Pneumonia       0.80      0.97      0.88       234

     accuracy                           0.87       468
    macro avg       0.88      0.87      0.86       468
 weighted avg       0.88      0.87      0.86       468
```



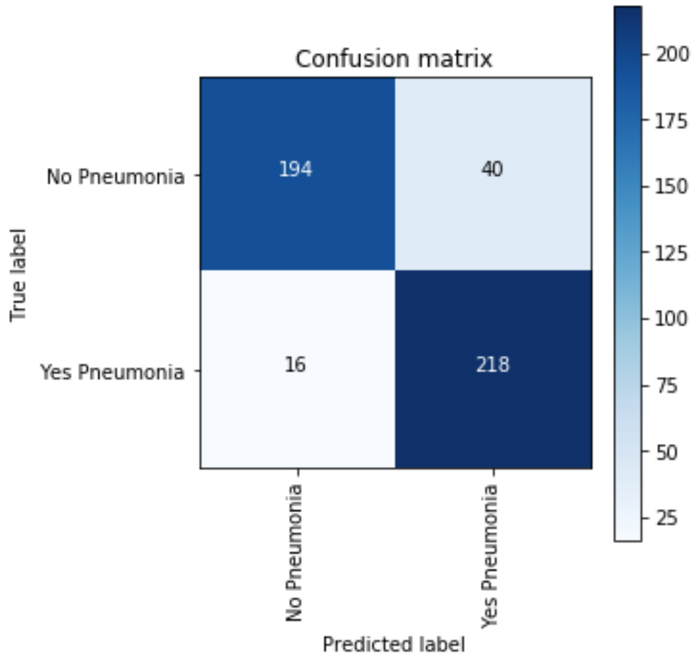




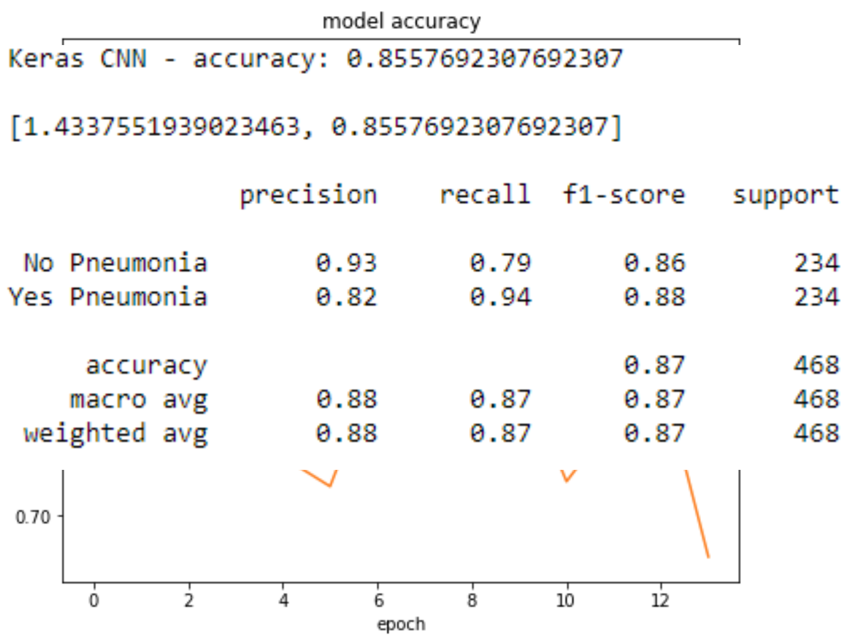
**VGG16 Early Stopping Unfrozen –
Full Results**



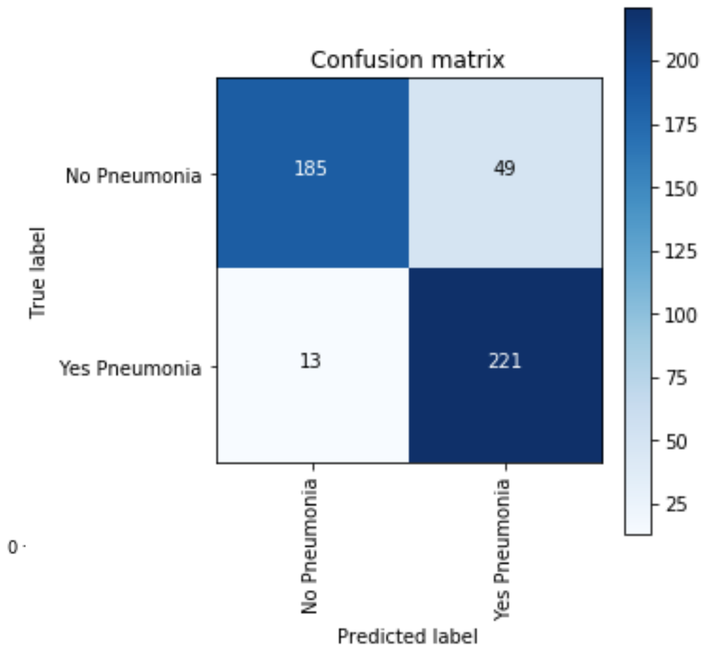
```
Keras CNN - accuracy: 0.8440170955454183

[1.4634566286690214, 0.8440170955454183]

               precision    recall  f1-score   support

 No Pneumonia       0.92      0.83      0.87       234
Yes Pneumonia       0.84      0.93      0.89       234

     accuracy                           0.88       468
    macro avg       0.88      0.88      0.88       468
 weighted avg       0.88      0.88      0.88       468
```

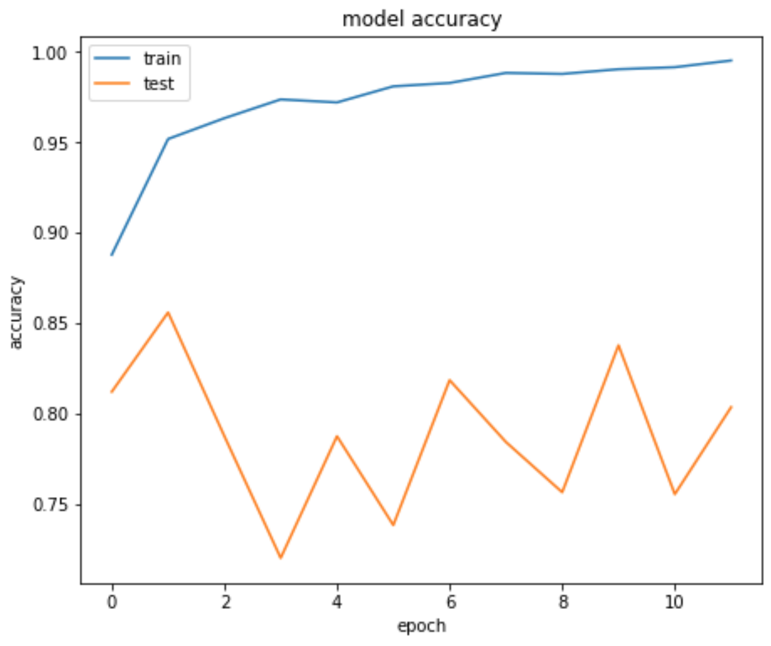


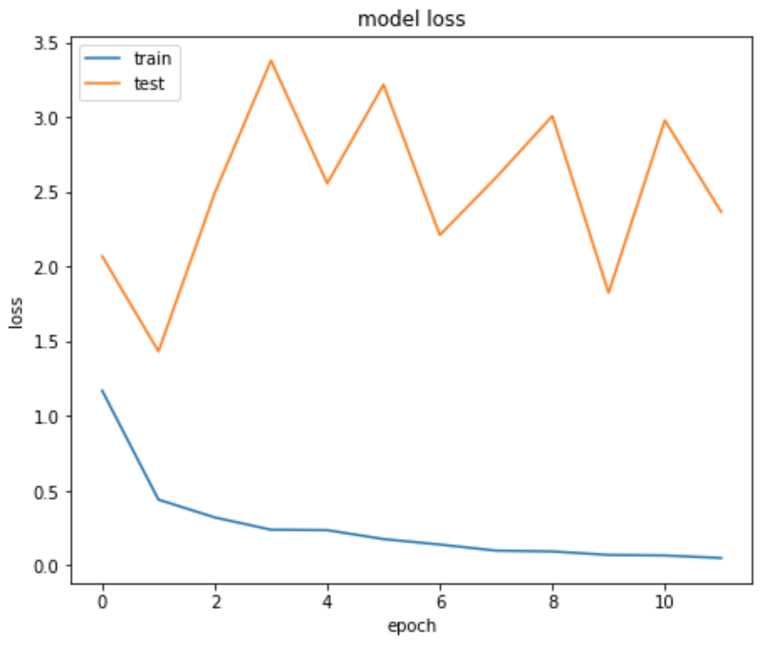
Inception V3 Early Stopping Unfrozen – Full Results



```
Keras CNN - accuracy: 0.8557692307692307

[1.4337551939023463, 0.8557692307692307]

              precision    recall  f1-score   support

 No Pneumonia       0.93      0.79      0.86       234
Yes Pneumonia       0.82      0.94      0.88       234

     accuracy                           0.87       468
    macro avg       0.88      0.87      0.87       468
 weighted avg       0.88      0.87      0.87       468
```







References

Bok, Vladimir and Langr, Jakub. (October 3, 2019). *GANs in Action* ISBN:9781617295560

Brownlee, Jason. (October 3, 2019). *Use Early Stopping to Halt the Training of Neural Networks At the Right Time.* Retrieved from https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Deshpande, Adit. (July 20, 2016). A Beginner's Guide To Understanding Convolutional Neural Networks. Retrieved from https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Francis, Sam. (June 17, 2017). *The advantages of automation in medical diagnostics.* Retrieved from https://roboticsandautomationnews.com/2017/06/17/the-advantages-of-automation-in-medical-diagnostics/12963/

Google. (April 14, 2020). *Running Inception on Cloud TPU.* Retrieved from https://cloud.google.com/tpu/docs/tutorials/inception

Hassan, Muneeb. (November 18, 2018). VGG16 – *Convolutional Network for Classification and Detection.* Retrieved from https://neurohive.io/en/popular-networks/vgg16/

Hutchinson, Maxwell. (November 2, 2017). *Overcoming data scarcity with transfer learning.* Retrieved from https://arxiv.org/abs/1711.05099

Kermany, Daniel, Zhang, Kang, and Goldbaum, Michael. (January 6, 2018). Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification. Retrieved from https://data.mendeley.com/datasets/rscbjbr9sj/2

Marcelino, Pedro. (October 23, 2018). Transfer learning from pre-trained models. Retrieved from https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

Markgraf, Bert. (March 23, 2018). *How to Calculate Bias.* Retrieved from https://sciencing.com/how-to-calculate-bias-13710241.html

Nicholson, Chris. (No Date). *Evaluation Metrics for Machine Learning - Accuracy, Precision, Recall, and F1 Defined.* Retrieved from https://pathmind.com/wiki/accuracy-precision-recall-f1

Sarkar, Dipanjan. (November 14, 2018). *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning.* Retrieved from https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Sharma, Aditya. (July 20th, 2018). *Autoencoder as a Classifier using Fashion-MNIST Dataset.* Retrieved from https://www.datacamp.com/community/tutorials/autoencoder-classifier-python

TensorFlow. (April 17, 2020). *Transfer learning with a pretrained ConvNet.* Retrieved from https://www.tensorflow.org/tutorials/images/transfer_learning#fine_tuning

Thakur, Rohit. (August 6, 2019). *Step by step VGG16 implementation in Keras for beginners.* Retrieved from https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

Tsang, Sik-Ho. (September 10, 2018). *Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015.* Retrieved from https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

World Health Organization. (April 8, 2020). Q&A on coronaviruses (COVID-19). Retrieved from https://www.who.int/news-room/q-a-detail/q-a-coronaviruses#:~:text=symptom