Individual Report
By Matteo Bucalossi

Introduction

For the final project, we decided to tackle a social problem that could solved using deep learning. Thus, we decided to work on a dataset of X-Ray images that could be classified by a computer vision model and thus enhance and expedite the clinical process dealing with COVID patients, especially in crisis times such as a pandemic.
The dataset is available on Kaggle and it was prepared and cleaned by Joseph Paul Cohen at the University of Montreal. They built a public open dataset of chest X-ray and CT images of patients positive or suspected of COVID-19 and similar pneumonias, as collected from public sources as well as hospitals and physicians.

One of the problems during 202 was the delay in getting test results back, especially in the early stages of the pandemic and with polymerase chain reaction (PCR) tests, as several days may be required before doctors could provide a reliable diagnosis for COVID-19.
Thus, an X-ray of a patient can provide a result in the matters of seconds, and we can use deep learning models such as convolutional neural networks (CNN) classify images, and thus improve prognostic predictions in triage phases and help ease pressure on hospitals and make quarantine recommendations more effective to slow the spread.

Work distribution

Patrick and I spread the work fairly equally and well worked together on this project. We both researched models for computer vision tasks, and I trusted Patrick's experience in vision to try out to solve the task with VGG16. Patrick set up the base architecture and downloaded the data on Collab for us to train upon; we then both worked together on fine-tuning hyperparameter and try different things to gain better results from the training. Given the use of Google Collab, we were able to both use GPU and both try different models and compare our results to finally produce a final best model to present.
For the qualitative part, I set up the LaTeX layout for the presentation, and we then both provided input to the slides and split the writing of the report equally.

Results

We implemented and trained our model on GPU within Google Collab, which took less than one hour overall – one of the advantages of transfer learning. As we trained the model, the early stopping was able to stop the training process at the 88^{th} epoch as showed below.

```
Epoch 82/100
11/11 [==============================] - 16s 1s/step - loss: 0.2625 - accuracy: 0.8996 - val_loss: 0.2740 - val_accuracy: 0.8958

Epoch 00082: ReduceLROnPlateau reducing learning rate to 1.6000001778593287e-06.
Epoch 83/100
11/11 [==============================] - 17s 1s/step - loss: 0.2549 - accuracy: 0.9120 - val_loss: 0.2740 - val_accuracy: 0.8958
Epoch 84/100
11/11 [==============================] - 17s 1s/step - loss: 0.2588 - accuracy: 0.9124 - val_loss: 0.2740 - val_accuracy: 0.8958
Epoch 85/100
11/11 [==============================] - 16s 1s/step - loss: 0.2648 - accuracy: 0.8912 - val_loss: 0.2740 - val_accuracy: 0.8958
Epoch 86/100
11/11 [==============================] - 17s 1s/step - loss: 0.2552 - accuracy: 0.9180 - val_loss: 0.2740 - val_accuracy: 0.8958
Epoch 87/100
11/11 [==============================] - 16s 1s/step - loss: 0.2706 - accuracy: 0.9022 - val_loss: 0.2740 - val_accuracy: 0.8958

Epoch 00087: ReduceLROnPlateau reducing learning rate to 3.200000264769187e-07.
Epoch 88/100
11/11 [==============================] - 16s 1s/step - loss: 0.2719 - accuracy: 0.8917 - val_loss: 0.2740 - val_accuracy: 0.8958
```

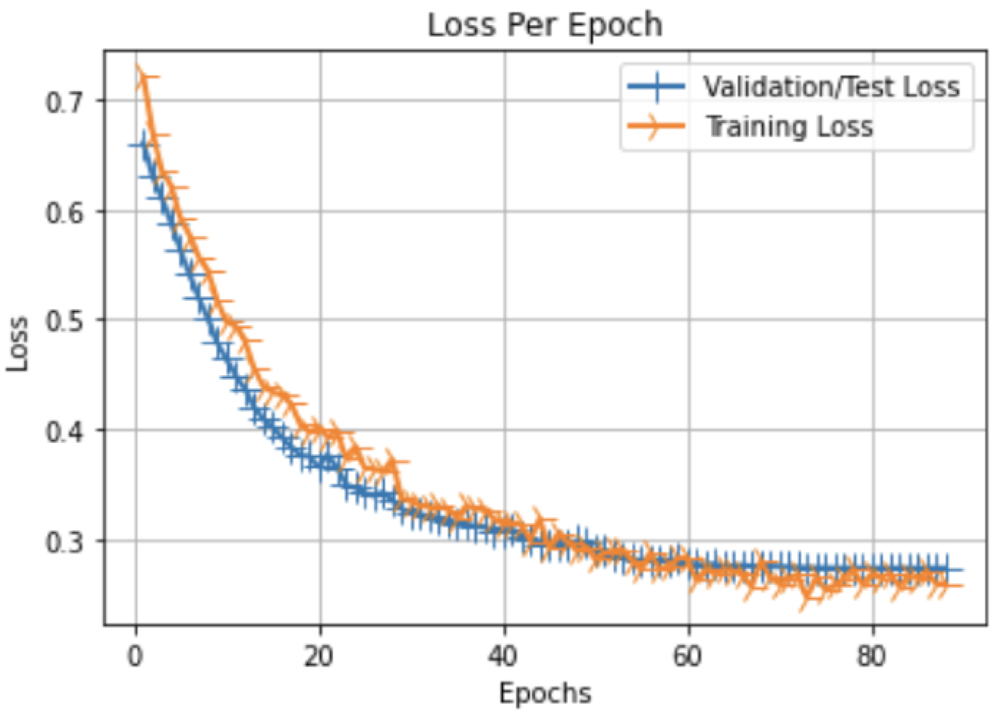
The model outputs confirmed that the learning rate was indeed reduced on the 59th epoch to 0.0002, on 70th epoch to 4, on the 77th to 8, on 82nd to 1.6, and finally on the 87th epoch to 3.2, this being the eventual value used to train the classifier.

After using the model to predict on the test set, we can see very strong results from the classification report by Keras.

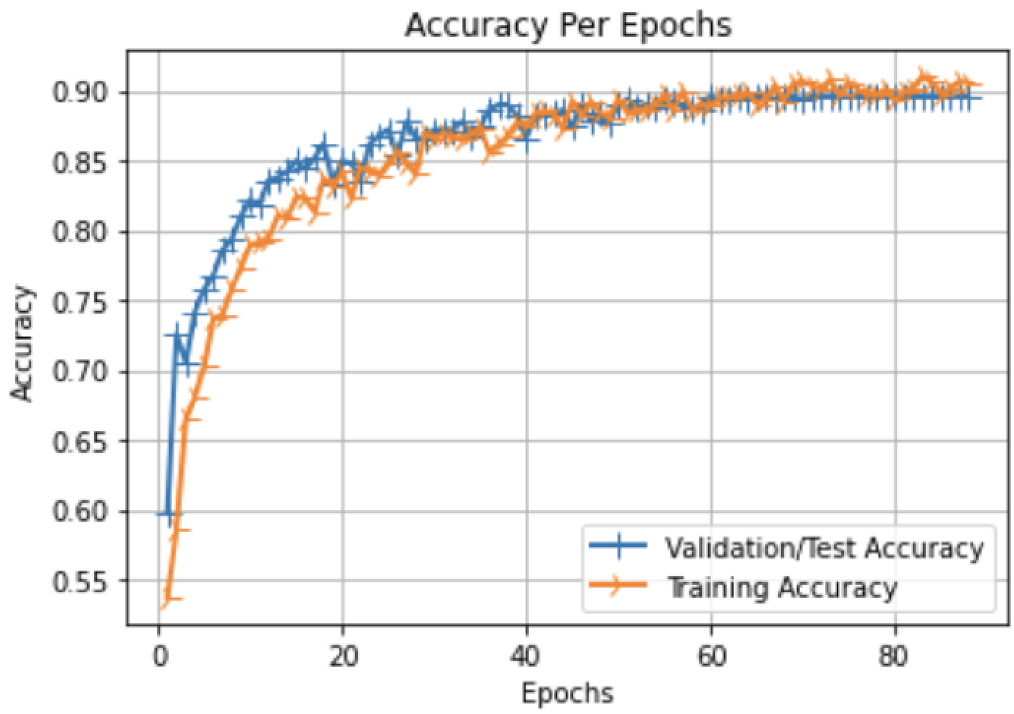
```
              precision    recall  f1-score

           0       0.89      0.91      0.90
           1       0.91      0.88      0.89

    accuracy                           0.90
   macro avg       0.90      0.90      0.90
weighted avg       0.90      0.90      0.90
```

Using accuracy as a metric for evaluating the performance of our model, we can see a 90% accuracy score; at the same time, the weighted F1 score is also 90%, yielding a fairly solid result for our goal of streamlining and improving COVID-19 patients triage processes.

We can also visualize the loss per epoch as well as accuracy score per epoch.



We can clearly see that the test loss significantly decreases after 20 epochs and continues to do gradually so until 70+ epochs. This confirms the early stopping output at 80+ epochs, as we can see no further improvement in performance with this model would be likely to occur.



Equally, the accuracy improves greatly up until 20 epochs, to then only gradually increase until 70+ epochs, confirming what we have noticed from the loss function trend.

Finally, we can get a summary of the model, which conforms to the VGG16 architecture displayed above, with output dense layers with customized neuron size for our task.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_6 (InputLayer)         [(None, 224, 224, 3)]     0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
average_pooling2d_5 (Average (None, 1, 1, 512)         0         
_________________________________________________________________
flatten_5 (Flatten)          (None, 512)               0         
_________________________________________________________________
dense_10 (Dense)             (None, 64)                32832     
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 130       
=================================================================
Total params: 14,747,650
Trainable params: 32,962
Non-trainable params: 14,714,688
```

Conclusions

As stated in the goals of our project, even a lower accuracy result would have improved the clinical processes at stake: our output of 90% provides a strong classifier able to quickly diagnose likely COVID-19 positive cases. As we have assumed, the integration of such tool within the healthcare system could greatly benefit the response to this pandemic, easing the pressure on hospitals and ER units, and provide the public with quicker test results as well as better and timely assistance to the most severe patients.

A future project for implementing this tool on the front line would be to integrate this model within TensorFlow Lite, as the framework offers the capability to run MobileNets on mobile devices. Thus, we could export this tool to a mobile app and allow physicians and nurses to have access to this quick diagnosis tool directly on their devices, greatly improving the clinical workflow in times of crisis.

Considering the pretrained section of the code and the parts we tweak within that code, we can estimate we copied from the internet 20% of the code and edited/tweaked the rest.

References

- Dataset: El-Shafai, Walid; Abd El-Samie, Fathi (2020), "Extensive COVID-19 X-Ray and CT Chest Images Dataset", Mendeley Data, V3, doi: 10.17632/8h65ywd2jr.3
- https://keras.io/api/callbacks/reduce_lr_on_plateau/
- https://keras.io/api/callbacks/early_stopping/
- https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a
- Joseph Paul Cohen and Paul Morrison and Lan Dao and Karsten Roth and Tim Q Duong and Marzyeh Ghassemi (2020) "COVID-19 Image Data Collection: Prospective Predictions Are the Future" arXiv:2006.11988
- Simonyan, K., Zisserman A. (2015) "Very Deep Convolutional Networks for Large-Scale Image Recognition" ICLR, arXiv:1409.1556
- S. Tabik, A. Gómez-Ríos, et al. (2020) "COVIDGR dataset and COVID-SDNet methodology for predicting COVID-19 based on Chest X-Ray images", arXiv:2006.01409