Class Challenge: Report

Task 1

I. *Architecture:*

The architecture I chose was a (CNN) Convolutional Neural Network due to its focus and success with image recognition and processing. I chose to start with the VGG-16 pre trained model that has been reported to achieve a 92.7% top-5 test accuracy in ImageNet and it works well with our input size. My model did not include the 3 fully-connected layers at the top of the network [`include_top = False`], the weights set were pre-trained ImageNet data [`weights = "imagenet"`], and the input shape was [`(224, 224, 3)`]. I decided to have a sequential model because it would be a simple approach that allowed me to add and customize layers easily. I will add below the layers of my sequential model and explain my thought process with each implementation:

`Flatten()` :
To flatten the input data to a format suitable for the convolutional layer without affecting the batch size.

`Dense (256, activation = 'relu')` :
Densely-connected Neural Network layer. I initially chose 256 as the dimensionality of the output space because we were shown the following when initially opening Task 1, and it worked well for me:

Layer (type)	Output Shape
dense_feature (Dense)	(None, 256)

Since Task 1's template showed the above output shape, I saw it as a hint to the output dimension I should have. I also chose the (ReLu) activation function for the density layer because it is computationally efficient and well fitted for a CNN. From my research, it seemed the most suitable for our middle layer and a non-binary result.

`Dropout (0.2)` :
I applied a dropout layer to drop 0.2 of the input units so overfitting can be avoided. I chose this rate through trial and error and by researching the best dropout rates.

`Dense (1, activation = 'sigmoid')` :
I added a second densely-connected Neural Network layer. I chose one as the dimensionality of the output space because we were shown the following when initially opening Task 1:

Layer (type)	Output Shape
dense_1 (Dense)	(None, 1)

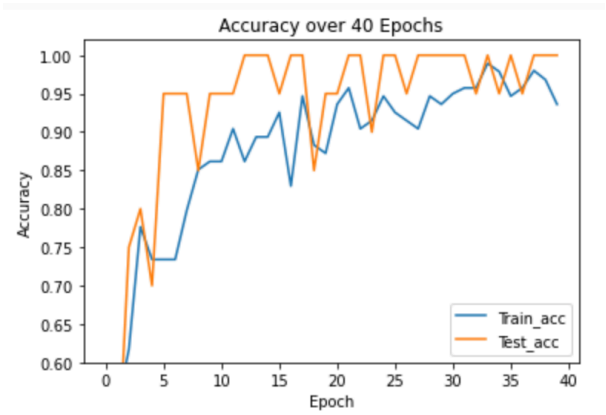
Since Task 1's template showed the above output shape, I saw it as a tip to the output dimension I should have. I chose the (Sigmoid) logistic function for the density layer because it has a fitting output to show whether the data reported covid or not covid with a binary result.

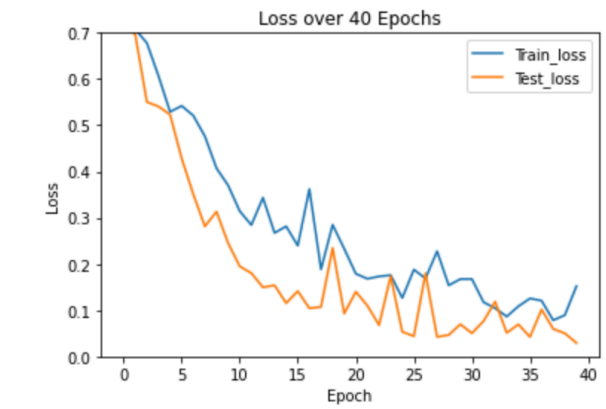
II. Training & Regularization:

I chose the gradient descent (with momentum) optimizer because it is the best optimizer for binary classification, and it is a standard optimizer to use for neural networks. I then set its learning rate to the `LEARNING_RATE` we were given for the load image data. For the loss function, I chose the binary cross-entropy loss between true labels and predicted labels. It is optimal with two label classes (0 and 1) suitable for our two label classes (not covid and covid). These were both implemented in `model.compile()`. For the metrics to be evaluated by the model during training and testing, I chose 'accuracy' to represent the model's correctness. Task 1 was also using 40 epochs to complete 40 training runs.

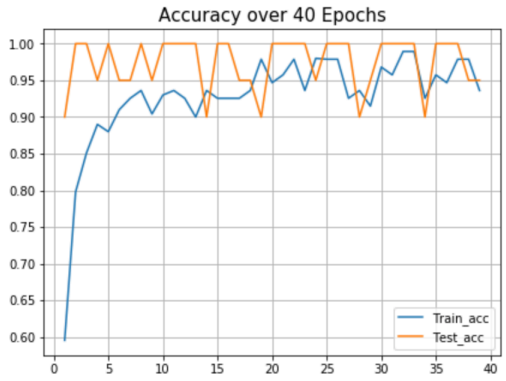
III. Accuracy & Loss:

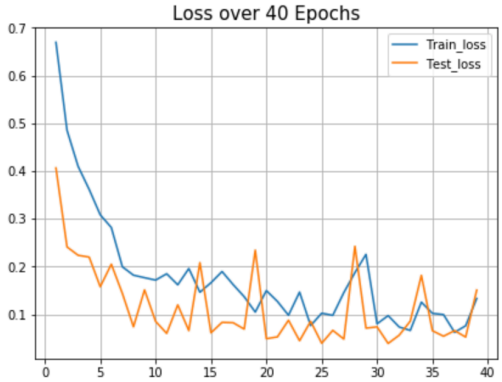
My Model's Accuracy & Loss





Task 1's Template Accuracy & Loss

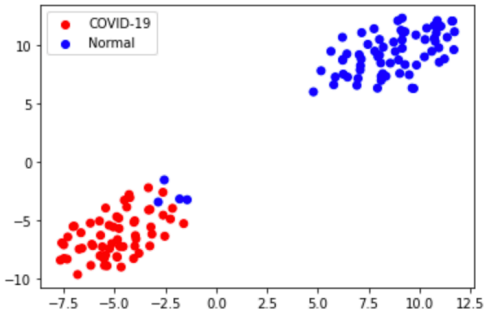




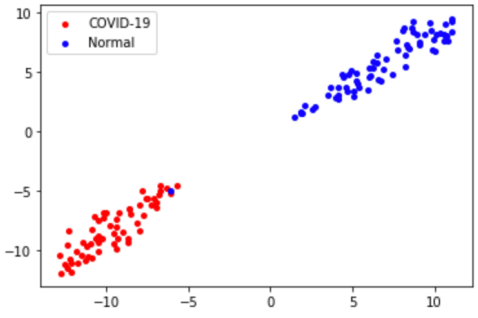
My accuracy was hitting 1.00, consistently similar to Task 1's template. My loss also seems to be in the same range as Task 1's template, around 0 and 0.3. By the end, my loss shows the lowest range from 0 to 0.2. The training and testing data also showed similar relationships; for example, my testing accuracy was mostly higher than my training accuracy, and for Task 1's template, it was the same relationship. As for the loss, my training loss was mostly higher than the testing loss, and that same relationship was shown in Task 1's template.

IV. t-SNE:

My Model's t-SNE



Task 1's Template t-SNE



My model's t-SNE was hitting the same range as Task 1's template. The COVID-19 and Normal classification were showing the same relationship as well. Despite a few outliers, the t-SNE shows a dimensionality reduction allowing us to visualize the data clusters and the different classification from COVID-19 to Normal.

Task 2

I. Architecture:

I chose a similar architecture as for Task 1 due to the same optimization factors. Task 1's report includes the break down of my implementations for each layer, which was also implemented with Task 2's model build. I did change some specifications of the density layer to be more well suited for Task 2:

`Dense (4, activation = 'softmax')` :

Densely-connected Neural Network layer. I chose four as the dimensionality of the output space because we were shown the following when initially opening Task 2:

Layer (type)	Output Shape
dense(Dense)	(None, 4)

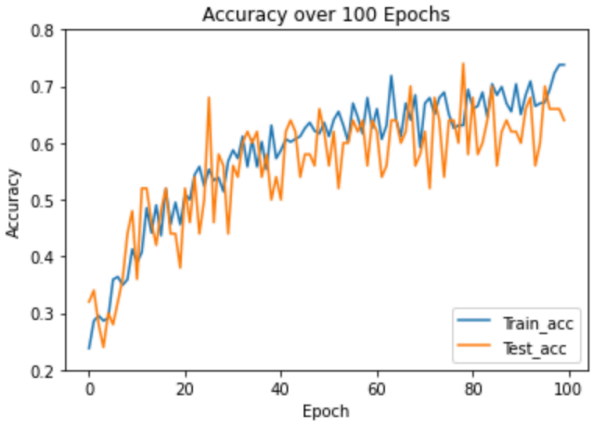
Since Task 2's template showed the above output shape, I saw it as another hint to the output dimension I should have. I chose the (softmax) activation for the density layer because it is well suited for multi-class classification. It is also trained commonly under categorical cross-entropy loss, which I would implement in my training model, so I knew this would best fit my plans.

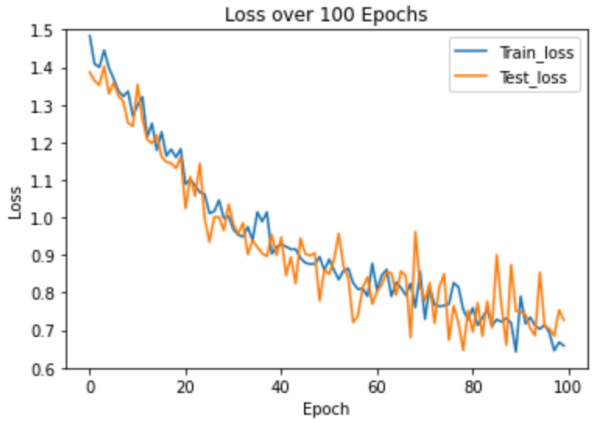
II. Training & Regularization:

I chose to implement a similar training model that I chose for Task 1. The main difference is that I chose a categorical cross-entropy loss function to compute the cross-entropy loss between the labels and predictions. It was better suited for multiple classification problems, which Task 2 consists of with our desired output categories of: normal, covid, pneumonia_bac, pneumonia_vir. Task 2 was also using 100 epochs to complete 100 training runs.

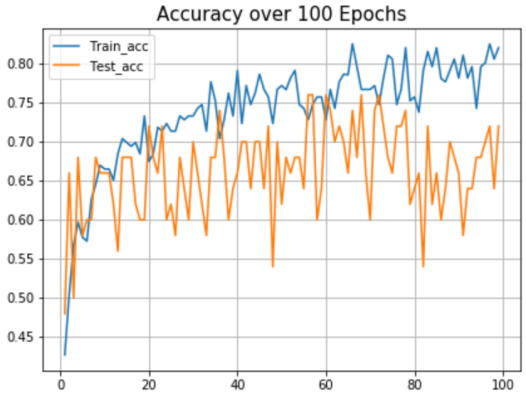
III. Accuracy & Loss:

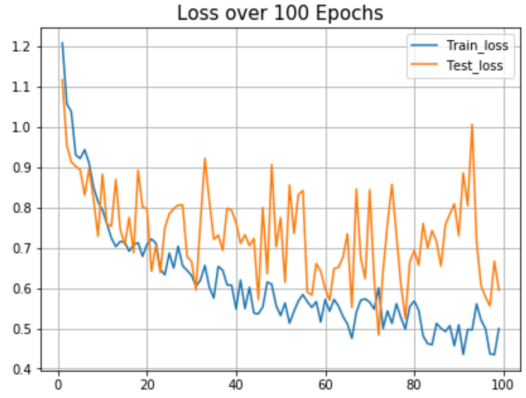
My Model's Accuracy & Loss





Task 2's Template Accuracy & Loss

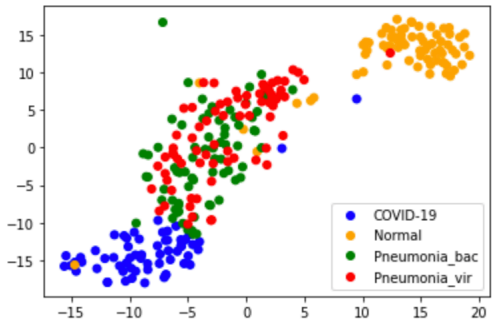




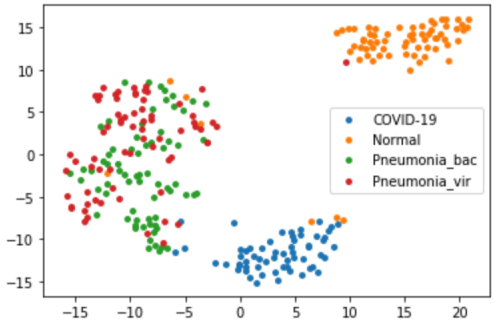
My accuracy was hitting 0.7 - 0.8, consistently high similar to Task 2's template, which was around 0.75. My loss also seems to be in the same range as Task 2's template, around 0.6 to 1.5. By the end, my loss shows the lowest range from 0.6 to 0.9. The model's success is apparent with the accuracy on a positive spike, and my loss over the 100 epochs had a downward trend.

IV. t-SNE

My Model's t-SNE



Task 2's Template t-SNE



My model's t-SNE was hitting the same sort of clusters as Task 2's template. COVID-19, Normal, Pneumonia_bac, and Pneumonia_vir classification were showing the same mixed relationship as well. Despite a couple differences, the t-SNE shows a dimensionality reduction allowing us to visualize the data clusters and the different classification from COVID-19 to Normal to Pneumonia_bac to Pneumonia_vir. The Normal data clusters seem the most distant from the other data points, same as Task 1's template, and there is also an apparent similarity as well with the Pneumonia_bac and Pneumonia_vir clustering together. You can see their similar relationships with dimensional mapping.

I. *Architecture Comparison (Task 1 vs. Task 2):*
The difference between the architecture I chose from Task 1 and Task 2 was the last output dimension and the last activation function.

For Task 1, I chose the sigmoid activation function, and for Task 2, I chose the softmax activation function. The change in the densely-connected Neural Network layer was because Task 1 was dealing with two (binary) labels: covid or not-covid, and Task was classifying more than two label options for a total of 4: normal, covid, pneumonia_bac, pneumonia_vir. I could not have implemented the sigmoid activation function to Task 2 because it does not work with binary classification due to its four labels. Softmax activation was a better fit for Task 2 because its optimal for multi-class classification.

There were also different output dimensions. For Task 1's second density layer, it had one output dimension, as for Task 2's second density layer, it had four output dimensions. This is due to the above explanation of the multi-class and binary classification difference between the two tasks.

The other difference between the two that I am not sure would be in architecture difference, but I just wanted to mention just in case would be the different epochs. Task 1 had 40 epochs, and Task 2 had 100 epochs. This served as the number of training trials per task. There was a lower epoch number on Task 1 because it had simpler data to manage with two labels. There was a higher epoch value for Task 2 because it was a more complex data set; it included four different labels (multi-class) and not just a binary classification like Task 1. Even with a more considerable amount of trials, it had a lower accuracy because of the testing data/classification.

Bibliography

https://neurohive.io/en/popular-networks/vgg16/

https://www.pyimagesearch.com/2019/10/28/3-ways-to-create-a-keras-model-with-tensorflow-2-0-sequential-functional-and-model-subclassing/

https://www.tensorflow.org/guide/keras/sequential_model#when_to_use_a_sequential_model

https://www.tensorflow.org/api_docs/python/tf/keras/Sequential

https://developer.ibm.com/technologies/artificial-intelligence/articles/cc-machine-learning-deep-learning-architectures/

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

https://missinglink.ai/guides/keras/using-keras-flatten-operation-cnn-models-code-examples/

https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/SGD

https://ruder.io/optimizing-gradient-descent/

https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy

https://www.tensorflow.org/api_docs/python/tf/keras/Model#compile

https://www.tensorflow.org/api_docs/python/tf/nn/softmax

https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy

https://towardsdatascience.com/implementing-t-sne-in-tensorflow-manual-back-prop-in-tf-b08c21ee8e06

https://mlexplained.com/2018/09/14/paper-dissected-visualizing-data-using-t-sne-explained/