Applied AI

Third Assignment

By

Sanjay Rana

201536014

Max Barnett

201348780

Rishabh Jaiswal

201540604

a. Introduction

Project goal: - Write a python program to train and test pretrained models for a multiclass classification problem. Please see the Jupiter notebook “image_classification.ipynb” for that, which can be run on Google Colab.

i. **Libraries used:**

- Matplotlib: used for data visualization and to plot the curve for the best network.
- NumPy: used for carrying out some mathematical operations.
- Seaborn: used for plotting Pearson correlation matrix to know the correlation of each feature with the price.
- Sklearn: used for importing classification report, recall score, f1 score, accuracy score, confusion matrix
- TensorFlow: used for importing Keras, Sequential, Dense, Adam, KerasRegressor, ResNet50, VGG16, ImageDataGenerator
- OS: for managing directories and paths
- Warnings: used to ignore the warnings.

ii. **Data analysis process:**

For analyzing the data provided we have used several visualization techniques, firstly, random samples from the data where plotted to allow us to visualize what type of data we are dealing with. From these examples we can see the images are all in greyscale, it's also evident that the images are all different dimensions and contain a wide variation of pixel intensity. Therefore, the images need to be resized, this is achieved by introducing the function '**ImageDataGenerator**' to the image processing, this function allows us to undertake data augmentation to our image data. This is required as CNN works better with large amounts of image data. The function contains a range of techniques to enhance the data such as resizing, normalization, random shifting, shear intensity etc. It also converts our single-channel grey- scale X-ray images to a three-channel format by repeating image’s value over all channels (this is necessary because the pretrained model requires three-channel inputs). Normalization is set to 'True' so the mean size of all samples is centered to 0, the image size is divided by its standard deviation and the target size of the image is set to 320x320. Despite a ResNet50 model expecting 224x224 image resolution, a study on chest x-ray images by Sabottke and Spieler [1] suggested that ResNet50 models achieve better AUC scores up to 320x320, with scores plateauing after this size. This is perhaps because for x-ray images more clarity in the image is necessary for the model to identify the correct label.

Other parameters such as 'shear intensity', 'zoom range' and the 'height and width shift range' are all set to constant values. Shear intensity, equal to 0.1, means the images are randomly distorted along an axis, this means the computer's perception of the images is altered, in real life humans do not view images perfectly so shearing the image up to 10% allows the model too see the image from a human perspective. Zoom range, set to 0.1, randomly zooms the image in and adds new pixel values around the image. Again, this helps to form new and different examples to train datasets, examples that the model may see in the real world, the more the sufficient the dataset then the more robust the network will be. Finally, the height and width shift range are set to 0.1, so the images are randomly shifted along the x and y axis by 10% as they act as a noise for our network. These small augmentation techniques have helped to make a robust dataset that allows the model to generalize well by learning more parameters. For example if we flip and rotate the image, the filters in the convolution layers will be able to learn more features from these generated images like edge detection.

Another parameter to note is the batch size, for training the models, batch size is set to 8 as this was found as the optimal value for our models in terms of accuracy and training time. For testing, batch size is set to 1 because in a real scenario we would process images one at a time. Finally, the pixel intensity is centered with a mean of zero and divided by its standard deviation, this means all of the images in the dataset are transformed to the same scale so the images are better suited for training a CNN.

iii. **Pretrained models selected:**

Resnet50 & VGG16 are the most popular deep convolutional neural networks. For ResNet50 the top-5 accuracy on the 'ImageNet' dataset is 96.2%and for VGG it is 93% [Detection of COVID-19 Based on Chest X-rays Using Deep Learning]. Many studies have tried to find COVID-19 infections in CXR images by using different DL methods and ResNet and VGG has one of the best accuracies amongst all.

- **Resnet50:**
  ResNet-50 is a convolutional neural network that is 50 layers deep. It works by adding extra layers to deep NN after a certain point starts increasing the training error. And hence ResNets was proposed. ResNets adjust the input layers to increase the performance and solves the problem of

vanishing gradient and higher training error due by adding extra layers. It works by stacking many residual blocks (layers) to build much deeper convolutional networks. Adding the residual network to the deep NN does not affect the ability of the neural networks to do as well as the simpler network (without residual network) and also learns new information thereby boosting the performance because identity function is easy for residual blocks to learn because of this skip connection. [3]

- **VGG16:**
  VGGNet consists of 16 convolutional layers. The only preprocessing that is done is subtracting each pixel from the mean RGB value calculated on the training set. VGG utilizes filters with a very narrow receptive field 3x3 (the smallest dimension that spans in all directions) to send the image through a stack of convolutional layers. The small-size convolution filters allow VGG to have a large number of weight layers; of course, more layers lead to improved performance. Five max-pooling layers follow some of the conv. layers and do spatial pooling (not all the conv. layers are followed by max-pooling). Max-pooling is done with stride 2 over a 2x2 pixel window. The soft-max layer is the final layer. In all networks, the completely connected levels are configured the same way. All hidden layers are equipped with the 'relu' function. The small-size convolution filters allow VGG to have a large number of weight layers; of course, more layers lead to improved performance.

iv. The training and testing process:

Before training the models, our data is split into training and validation data with a ratio of 80:20 respectively. This split size allows sufficient amount of data for training, which requires the largest amount of data, while also keeping a diverse dataset for validating the training results. Without the addition of the validation dataset we would be unable to confirm that the results of the training process can be applied to unseen data.

The process of training our models consisted of optimizing the parameters. The optimizer 'Adam' was selected as the most suitable optimizer because it led to faster convergence in the training results, it also decreased the amount of noise. This aligns with our main objectives in training which was to smoothen the curve and avoid overfitting. It reduces loss and improves accuracy by modifying the attributes of the neural network such as weights and learning rate. We also didn't have to worry about learning rate.

For the loss function, 'categorical crossentropy' is selected because it is most suitable for problems with multi-class classification where labels are one-hot encoded [5], it is also appropriate in combination with the 'SoftMax' activation function where probabilities are produced for each class. Activation function determines if the neuron's input to the network is important or not in the prediction process when more than two class labels are required. It is used to predict multinomial probability distribution. SoftMax is used as the activation function in our use case because we have 3 classes.

Weights are used to initialize the weights of the layers. We have chosen ImageNet as it will use the pre-trained weights from ImageNet so as to get better accuracy rather than randomly initializing the weights.

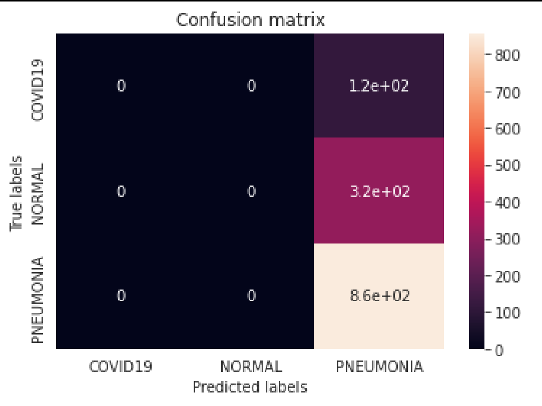
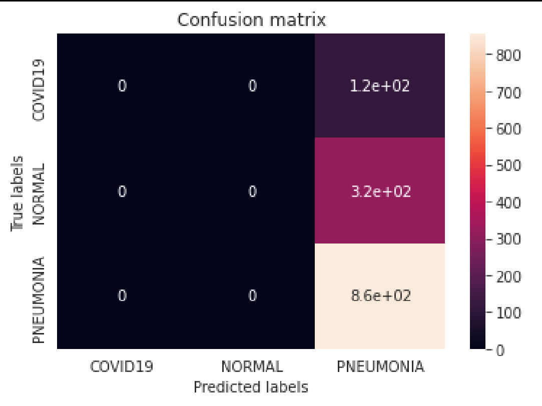
Pooling in neural networks is used to reduce variance and processing complexity. There are three types of pooling based on the type of value selected from the batch: max pooling, min pooling, average pooling. Since image generator has converted our single channel X ray images from grayscale to RGB we wanted to smooth out the image. Also, there are no sharp-edged features to learn in our dataset. So average pooling seems perfect for our use case.

b. Evaluation

i. Results of pre-trained models without training on current dataset:

Table 1 - Results of testing pre-trained models

	ResNet50				**VGG16**			
Classification Report		Precision	Recall	F1-score		Precision	Recall	F1-score
	0	0.00	0.00	0.00	0	0.00	0.00	0.00
	1	0.00	0.00	0.00	1	0.00	0.00	0.00
	2	0.66	1.00	0.80	2	0.66	1.00	0.80
		accuracy		0.66		accuracy		0.66

Confusion Matrix		

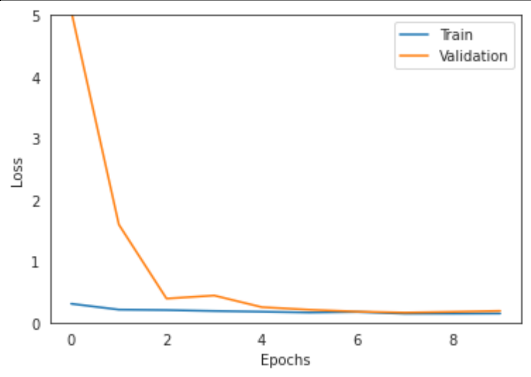
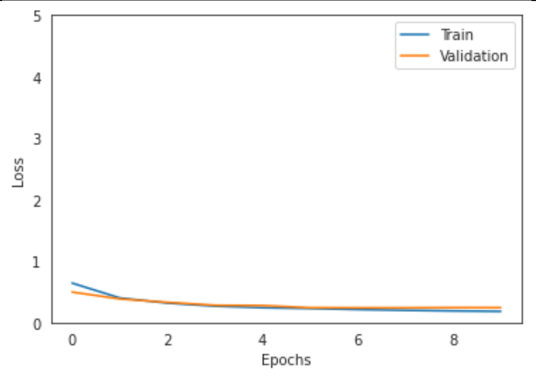
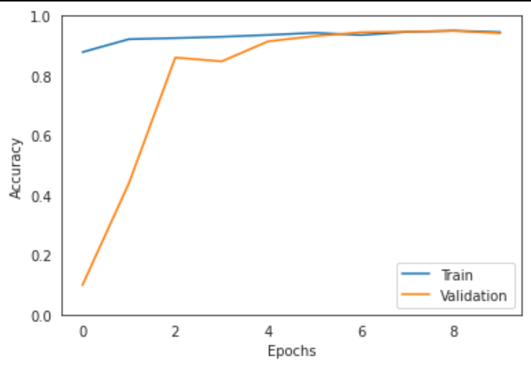
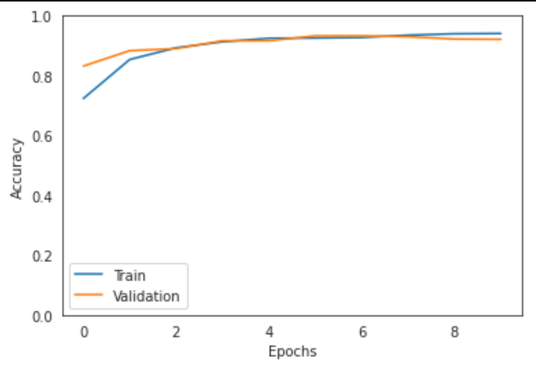
From the results it is clear that without previously fitting the models to our training data, the predictions of the Resnet50 and VGG16 model are not great, with both models producing the same results as they are aiming to optimize the loss. Subsequently, it realizes that because the class labels are unbalanced and skewed towards 'pneumonia', the outputs should likely be pneumonia to achieve good results. Therefore, it always predicts the same class. We have observed this anomaly on a few occasions, one time the ResNet50 model predicted all labels as 'normal', this is clearly down to the pre-trained model not being trained on our data, instead it is trained on the 'ImageNet' dataset. The dataset contains over 10 million images, the dataset does not contain x-ray scans or images of lungs instead it contains RGB images of random objects, therefore initially both models are unable to understand which symptoms or section of lung can diagnose Covid-19/pneumonia.

ii. Pre-trained models with single top layer and trained on current dataset:

For the improved network we decided to optimize both the VGG16 and ResNet50 networks that where investigated previously. As discussed previously, both models resulted in the same results, following this, both ResNet50 and VGG16 are both fitted to the data to understand further how suitable these models are for the problem. Once this is complete, further optimization will take place by adding additional layers and altering slightly some parameters to the best trained model.

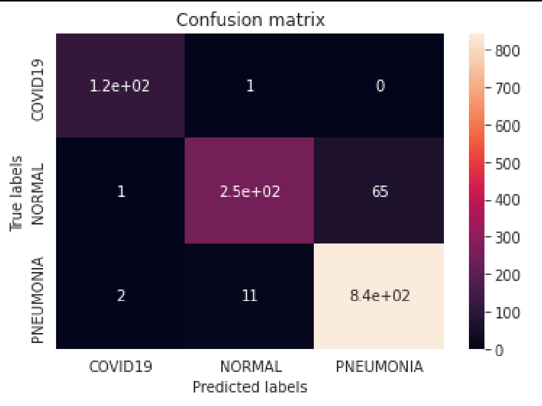
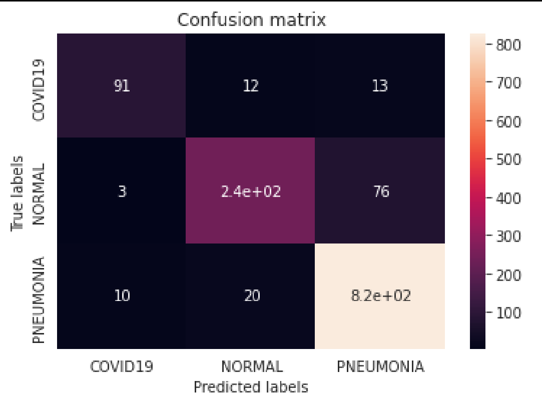
iii. Results of models:

Table 2 - Results of training and testing optimized pre-trained models

	ResNet50	**VGG16**
Training Loss		
Training Accuracy		

Classification Report

	ResNet50			VGG16		
	Precision	Recall	F1-score	Precision	Recall	F1-score
0	0.97	0.99	0.98	0.88	0.78	0.83
1	0.95	0.79	0.87	0.88	0.75	0.81
2	0.93	0.98	0.96	0.90	0.96	0.93
	accuracy		0.94	accuracy		0.90

Confusion Matrix		

Discussion of ResNet50:
From the above graph for ResNet50, you can see both training and validation accuracy has a gap at the beginning which may be because the training and test data may have an unrepresentative split (may have a little different behavior) but it starts increasing as the number of epochs increases and converge with each other at the later epochs which is our goal for the neural network. Both the training and validation loss have converged and plateau after the 7th epoch, 94% accuracy is achieved which is a very respectable score.

Discussion of VGG16:

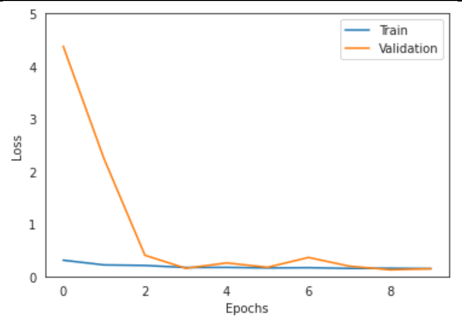
From the above graphs for VGG, you can see both training and validation accuracy increase steadily during training, while the training and validation loss curves decrease which is our goal primary during training. Furthermore, both validation curves are extremely close after 2 epochs of training, this means there is not too much overfitting occurring in the model. Initially, the validation set performed better than the training set, perhaps because the training data is harder to model than the validation data. But after training for long enough, the curves correlate strongly and the performance of the training set ends up beating that of the validation set, both curves converged and the validation curve has plateaued nicely as it doesn't improve after the 6th epoch. This proves the VGG16 model is a good fit for the problem as it has produced smooth curves and resulted in a test accuracy of 92% (the test accuracy is 2% lower than the accuracy achieved in training because the hyperparameters are tuned to the training/validation set).

Comparing results of models:
We can deduce from the training that both ResNet50 and VGG16 have successfully been trained to predict Covid-19 and pneumonia from lung images. The VGG network has produced a smoother loss curve than the ResNet50 model but based on test accuracy and the confusion matrix the ResNet50 model wins. Overall, it has achieved 4% higher accuracy and better recall for the testing, as this problem contains data related to health, recall also has importance.

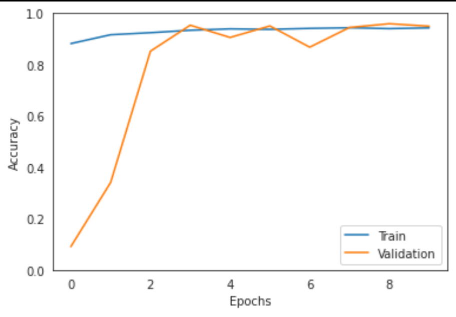
Overall, the results show that ResNet50 can outperform VGG16 for this task, this is down to the architecture of the 2 models. ResNet50 is a much deeper model than VGG16 with 34 extra layers, so the levels of features can be boosted due to the extra stacked layers. Referring to [6], when the depth of VGG16 model is increased, gradient vanishing problem can occur that leads to a higher training accuracy. To conclude, the ResNet50 model is better suited to the problem, we will further optimize the ResNet50 to achieve higher accuracy in the following section.

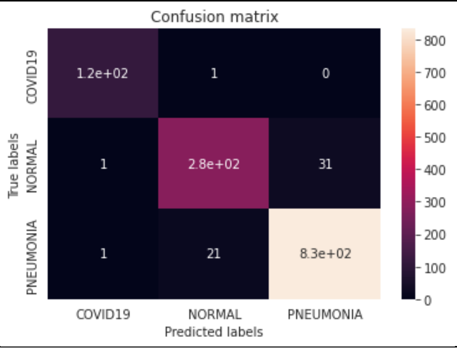
Proposed Resnet50 models:
Model 1.

- Added hidden dense layer with 16 neurons with 'relu' activation function
- Added dropout layer between hidden layer and output layer
- Dropout = 0.2

Training Curves	**Classificaiton/Confusion Matrix**
	

	Precision	Recall	F1-score
0	0.98	0.99	0.99
1	0.93	0.90	0.91
2	0.96	0.97	0.97
	accuracy		0.96

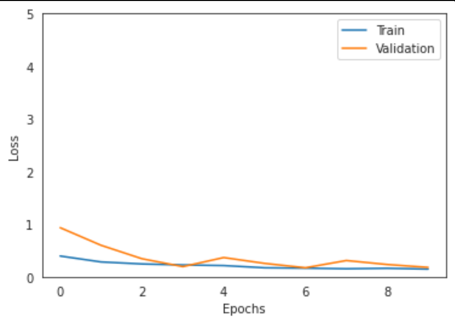
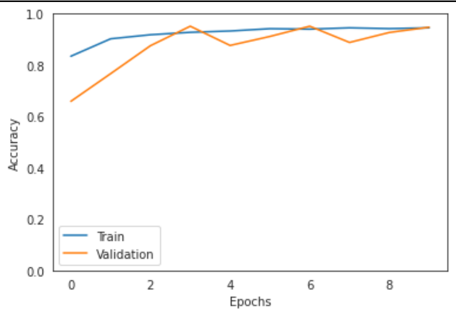
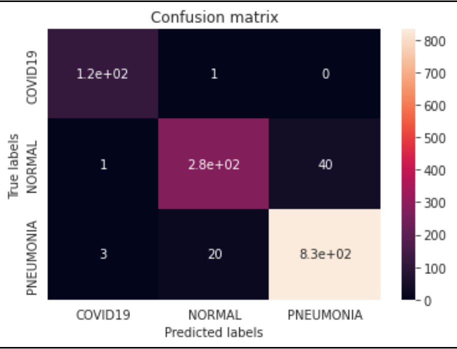




From the model with single top layer and trained on the dataset, we have added a dropout and dense layer in our proposed model. We observe the validation accuracy has dropped from more than 5 to less than 4.5. This can be due to the fact that the added dense layer is extracting more parameters or useful information for the previous network thus decreasing loss and improving accuracy. This can also attributed to dropout which is a regularization technique that helps prevent overfitting.

Model 2.

- Reduced number of neurons in hidden layer to 8
- Keeping dropout = 0.2

Training Curves	**Classificaiton/Confusion Matrix**
	see classification report below
	

	Precision	Recall	F1-score
0	0.97	0.99	0.98
1	0.93	0.87	0.90
2	0.95	0.97	0.96
	accuracy		0.95

From the previous model, we changed the number of neurons to 8, and the curves became smooth. This may be because more no. of neurons was resulting in slight overfitting by making the network complex. We observed noise in the loss and accuracy curve also decreased, this may be because of dropout, since dropout randomly removed given amount of data, so it must have removed important features for 16 neuron ResNet50 thus resulting in more noise.

iv. Conclusion of proposed models

Finally, we can see the affect of added hidden layer and dropout layer. Proposed model 1 achieves a further 2% improvement on the test accuracy compared to the 1st trained model. With recall scores for covid19 and pneumonia of 99% and 97% respectively, F1 scores massively improving (showing a balance between exactness and completeness of the model), this model is extremely robust and suited for diagnosis within medicine. This shows the advancements of transfer learning as we have demonstrated the training of deep neural networks to classify x-ray images on Google Colab. However, our results show that it is computationally intensive to achieve state of the art accuracy so complete generalization can only be achieved with more investment into the study.

v. Future Works

There are a lot of parameters with which still we can play but because of constraints of time and training, we couldn't. For example, we can build models by adding convolution layers, skip connection, in combination with and without dropout layers, and its value etc. Along with this due to limitation of Goggle Colab's run time limit of 2 hours for GPU, we couldn't train our models for more than 10 epochs. With better resources this can also be explored.

c. Final conclusions

There are many challenges, during the development of this project. The toughest one is the training and testing of neural network model for getting the best parameters. It takes a very long to train a neural network and visualize the output of your plan, more so we had to retune training for many different parameters as we optimized our models. Additionally, we have used Google Colab for this assignment because taking advantage of Colabs' GPU can speed up the training time, but after a period of training, Colab restricted the use of the GPU meaning it was difficult to reproduce our results. It was also challenging to understand the stochastic nature of the models created, as seen with the ResNet50 model, we tried to add regularization to the model in hope of decreasing overfitting but this actually led to worse results. In the future, we should set a random seed from the beginning of the assignment to remove the randomness of the results. A lack of resources on VGG16 also reduced our development of the VGG model.

All the tasks were equally distributed amongst all the three members. Rishabh was mainly involved in data importing, pre-processing, collecting resources from blogs, research papers, books and YouTube. Sanjay was mainly involved building and training the models. Also contributed in critically analyzing the results of the models. Max was mainly involved in creating the report, collecting resources for selecting the initial parameters.

References

[1] Sabottke, C. and Spieler, B., 2020. The Effect of Image Resolution on Deep Learning in Radiography. *Radiology: Artificial Intelligence*, 2(1).

[2] Douglass, M., 2020. Book Review: Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow, 2nd edition by Aurélien Géron. *Physical and Engineering Sciences in Medicine*, 43(3).

[3] NG, A. 2022. *C4W2L03 Resnets*. [video] Available at: https://www.youtube.com/watch?v=ZILIbUvp5lk

[4] 2022. *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION*. [ebook] Available at: <https://arxiv.org/pdf/1409.1556.pdf>

[5] 2022. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. [ebook] Available at: https://arxiv.org/pdf/1412.6980.pdf

[6] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.