ADVANCED MACHINE LEARNING PROJECT- INDIVIDUAL REPORT

Suresh Balaiyan
COMP 6380
C3302159@UON.EDU.AU

Abstract

This report discusses the use of convolutional neural networks(CNN) for image classification. In Q1 the data set was downloaded from https://www.kaggle.com/alxmamaev/flowers-recognition, which has a total of 4242 images of 5 different flowers: daisy, tulip, rose, sunflower and dandelion. CNN model was trained to classify the multi class flower set. Various hyper parameters were experimented to find the best model such as learning rate, number of epochs ,batch size and drop out layer. The model template code used was downloaded from blackboard to build the model. In Q2 , the data set used to train , validate and test the CNN model was downloaded from https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia . The data set contains x ray images with and without pneumonia. The CNN model was trained to distinguish the x ray images based on the pneumonia criteria. The CNN Model architecture is built on experimentation of variations of various hyper parameters such as number of kernels/filters , size of kernels, learning rate of optimizers , max pooling dimensions, weight regularization-L2 and image augmentation . After training the models on the train dataset of 5216 images ,validated and tested , an optimum model based on test data set accuracy is obtained that can be used for future such image classifications. The python code used for this classifier was taken from the model template of code from blackboard which classifies dog/cat data set.

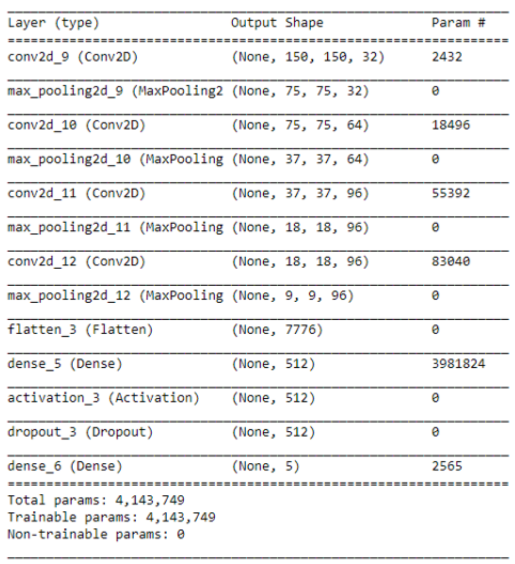
Introduction

One of the machine learning application is image classification or recognition that is been used to our daily lives on various industries such as medical diagnosis, gaming, automobile industry etc. **Supervised learning algorithms** such as convoluted neural networks (CNN) transforms such images into vectors of numeric array values for classification purposes to detect shapes, edges, curves, and objects as a whole. This process of detection is simply based on mathematical computations on matrices to learn such features of the images. In Q2 , the CNN model classifies binary types of images, however CNN's can be used to classify or predict multiple categorical of images like in Q1, that can be either coloured or grayscale images. Pixel values of such images are used to find dot products with kernels or filters , which is again matrices of integers between -1 and 1 to transform into more meaningful values that represent parts of an image. Each layer of the convolution layer progresses in learning the features of the images with the use of kernel matrices, non-linear activation functions(RELU) and max pooling to minimize the size of the matrices to the next layer of input by finding the maximum value within a given stride. Thus, at the end the convolutional process the images are flattened to form a single vector to obtain output using sigmoid for binary classification and soft max for multi class classification based on probability basis.[7]

Table of contents

Q1 Flowers Classification:

The CNN model was trained using 5 different flowers set that is been downloaded and labelled in different folders according to the category of the flowers and trained by varying the hyper parameters. The optimum model results are obtained by varying the epoch between 50 and 80, batch size(64 and 128) , learning rate with Adam optimizer (0.001 and 0.0001) and with or without drop out layer.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_9 (Conv2D)            (None, 150, 150, 32)      2432      
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 75, 75, 32)        0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 75, 75, 64)        18496     
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 37, 37, 64)        0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 37, 37, 96)        55392     
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 18, 18, 96)        0         
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 18, 18, 96)        83040     
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 9, 9, 96)          0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 7776)              0         
_________________________________________________________________
dense_5 (Dense)              (None, 512)               3981824   
_________________________________________________________________
activation_3 (Activation)    (None, 512)               0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_6 (Dense)              (None, 5)                 2565      
=================================================================
Total params: 4,143,749
Trainable params: 4,143,749
Non-trainable params: 0
_________________________________________________________________
```

Figure 1 – Summary of one of the models

CNN with dropout:

The CNN model was built based on 4 convolutional layers and one fully connected layer. The kernel sizes or number of neurons configured for each layer was 32,64,96,96 respectively with drop out layer.

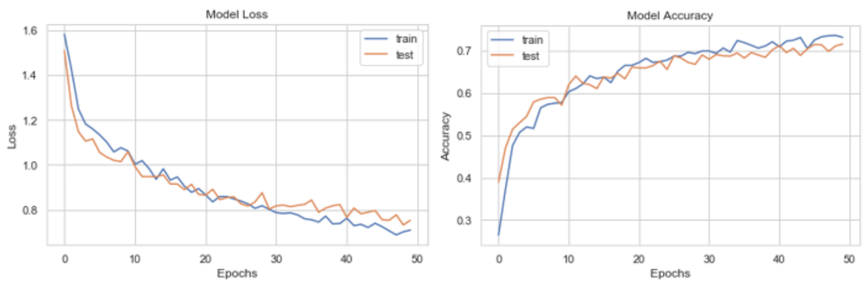


Figure 2 - Results of the CNN with 50 epochs, batch size of 128 and the learning rate of the Adam optimizer of 0.0001, with dropout layer.

In the above figure the train and test model performed efficiently with the error rate on the left diagram decreasing considerably as the training of epochs progresses and validation accuracy increases for both the train and test set as the epochs progresses. This model was built with a drop out layer, which decides the number of neurons selected randomly to be used in the convolution process. Typically, the dropout percentage selected is between 20% and 50 % of the total number of neurons [5].

CNN without dropout:

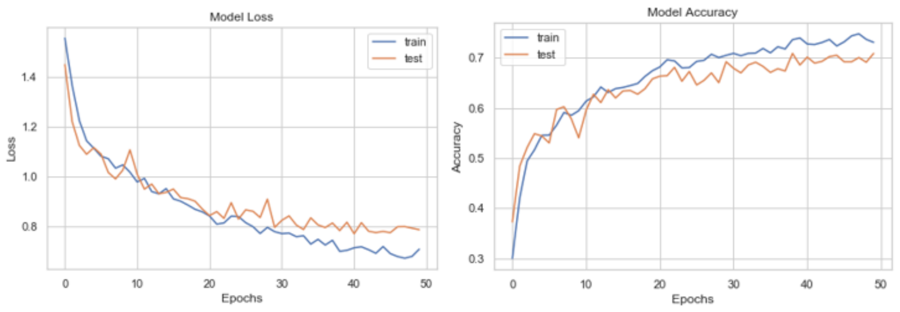


Figure 3 - Results of the CNN with 50 epochs, batch size of 128 and the learning rate of the Adam optimizer of 0.0001, without dropout layer.

In the above figure the model was trained with the same hyper parameters as the earlier model, but this model was having no drop out layer. As a result, the model overfitted , which is indicated by the distance between train and test model curves. Initially the loss or error is high for the training data then to the validation data , but then the train error reduces than the validation data , approximately after 10 epochs. Generally, the validation loss is higher than the train error but in the above diagram on the left this is not present in the initial epochs . This is because the drop out is applied only on train model and not on the test data set.[6]

Q2. X-Ray Image Classification

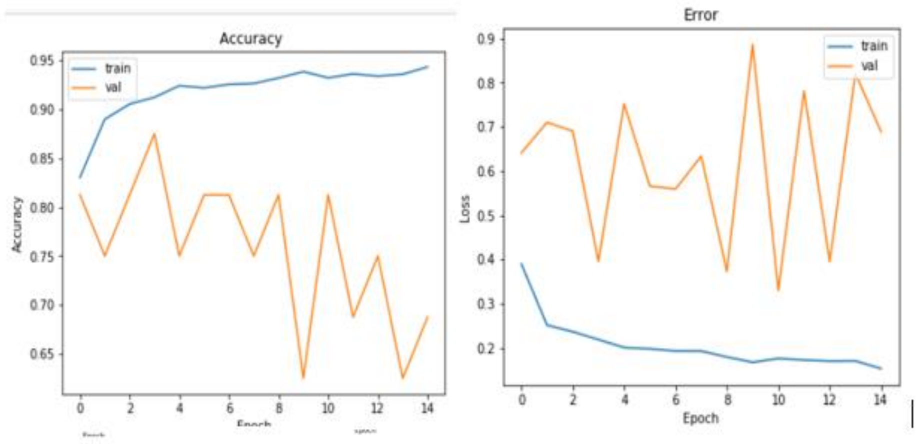
The CNN Model is trained on 5216 X ray images with and without pneumonia. As the size of the train dataset is reasonably high the model should be potentially free from overfitting because the model is well trained to learn the features of the images of both types. Various hyper parameters are varied to experiment and find optimum model that yield high accuracy rate on both validation and test data set such as :

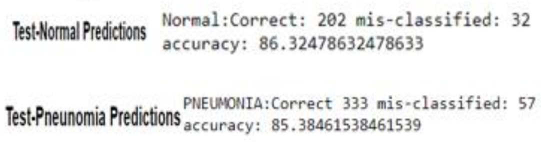
1. Number of kernels or filters.
2. Dimensions of each kernel.
3. Max pooling dimensions.
4. Learning rate of optimizers.
5. Number of neurons.
6. Number of epochs.
7. Dropout percentage.
8. Batch size per epoch.
9. Dimensions of the original image.
10. Image data augmentation.
11. Regularization of weights to avoid overfitting and minimize loss.

Experiment-1

We started to train the network with low batch size, as the result the loss function was plotted with high peaks and valleys. This CNN model was trained with 3 CONV2D and 1 fully connected layer with 32 and 64 neurons, respectively. The optimizer (ADAM) learning rate was set to 0.0001. The input images were resized to 256 * 256(width ,height) from the original dimensions of 2090 * 1858 to reduce the computational expenses such as matrix multiplication and find max in the max pooling function. **RELU** activation function used y = `max(0, x)` is added in between convolutional layer and max pooling to convert negative values into zero values , while positive values to remain same. Thus, by such measures the matrix size has been reduced to form a single vector at the output layer, where the sigmoid activation function classifies the output to one of the binary classes.

```
1.Epochs=15 , batch size= 16,Image dimensions 256 * 256, Kernel 32(3,3) 32(3,3)
and 64(3,3) ,Max pool(2,2)
```





Test-Normal Predictions

```
Normal:Correct: 202 mis-classified: 32
accuracy: 86.32478632478633
```

Test-Pneunomia Predictions

```
PNEUMONIA:Correct 333 mis-classified: 57
accuracy: 85.38461538461539
```

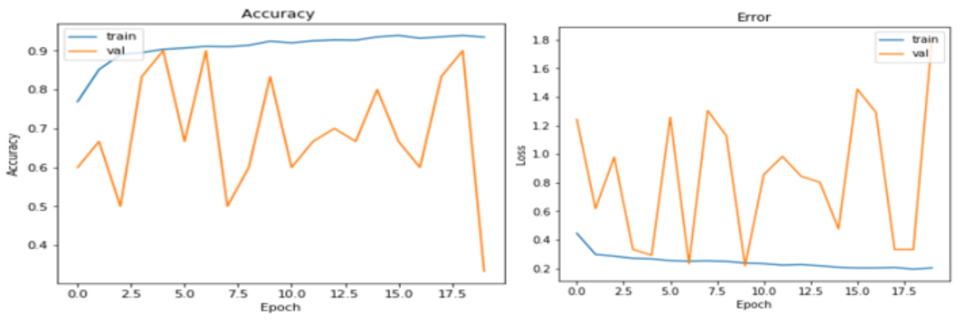
*Figure-4 shows the accuracy and error rate for both train and validate data set with image size 256 * 256 with number of epochs=15.*

Result: We were able to achieve more than 80% accuracy on both the class labels in the test dataset. Thus, the trained model has a reasonably successful classification result. From the above figure even though the error rate is high , as the total number of parameters was considerably reduced due to number of kernel layers and sizes , the model predicted reasonably on the test data set. [1]

Experiment -2

In this training model we varied the above model(1)by reducing the input batch size from 16 to 10 and increased epochs to 20 thus the number of steps per epoch increased for training and thus the model was trained well than the above model. However not decreasing the learning rate to compensate the reduction of the batch size has resulted in high variance of error rate in the validation data set.

```
Epoch=20,batch size=10,Image dimensions 256 * 256 ,kernel
32(3,3),32(3,3) and 64(3,3) , Max pool(2,2)
```



Test Normal Predictions
```
Normal:Correct: 189 mis-classified: 45
accuracy: 80.76923076923077
```

Test-Pneunomia Predictions
```
PNEUMONIA:Correct 365 mis-classified: 25
accuracy: 93.58974358974359
```

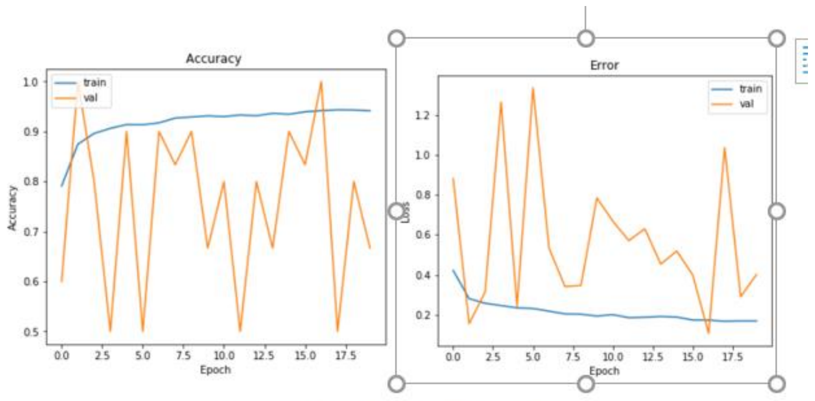
Figure-5 showing the variation from Train-1 model with accuracy and error rate for train and validation data set with batch size reduced from 16 to 10 and number of epochs increased from 15 to 20.

Result: We were able to achieve 81% accuracy on normal test data set, while 93.5% accuracy achieved on pneumonia data set. This imbalance in prediction accuracy on different class labels is due to the high imbalance in the presence of total number of samples of each class in train and test data sets. We have 1341 and 3875 normal and pneumonia class labels in train dataset. While 234 normal and 390 pneumonia class labels in test data set. [2]

Experiment -3

Variation from Train-2:Number of neurons are reduced for the first 2 kernel layers from 32 to 16 in the first 2 layers and from 64 to 32 in the 3 rd layer. Kernel size is reduced from(3,3) to (2,2) for the first 2 layers. Adding more layers and neurons will result in more training time, overfitting and vanishing or exploding vanishing gradient problem.[5]

```
Epoch=20,batch size=10,Image dimensions 256 * 256 ,kernel
16(2,2),16(2,2) and 32(3,3) , Max pool(2,2)
```



Test Normal Predictions
```
Normal:Correct: 234 mis-classified: 0
accuracy: 100.0
```

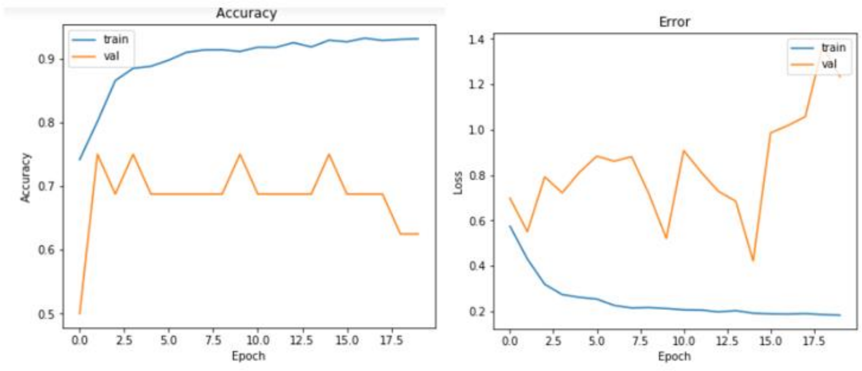
Test-Pneunomia Predictions
```
PNEUMONIA:Correct 78 mis-classified: 312
accuracy: 20.0
```

Figure 6 showing variation from Train-2 with accuracy and error rate for train and validation data set with number of neurons for kernels reduced from 32 to 16 for the first 2 layers and from 64 to 32 for 3 rd layer.

Result: Reducing the number of neurons and kernel matrix size has resulted the train model to overfit which is evidenced in the accuracy of the test data set. The accuracy on normal test data set is 100% while the model predicted poorly on the pneumonia test data set-20%. This inefficiency of prediction is due to the process of convolution that is affected with reduction of kernel filters and size such as `(N X N ) * (F X F)=(N-F+1) X (N-F+1)` where N is the dimension of image and F is the kernel dimension will result into an output that is fed into the next layer. Thus, the CONVD layer has difficulty in extracting the feature of the images and resulted in poor performance.[3]

Experiment -4 Variation from the above model (4) Max pool size resized from (2,2) to (3,3).,image dimension resized to 150,150

```
Epoch=20,batch size=16,Image dimensions 150 * 150 ,kernel
32(3,3),32(3,3) and 64(3,3) , Max pool(3,3)
```



```
                        Normal:Correct: 217 mis-classified: 17
Test Normal Predictions accuracy: 92.73504273504274

                          PNEUMONIA:Correct 308 mis-classified: 82
Test-Pneunomia Predictions accuracy: 78.97435897435898
```

Figure-7 Variation from the above model (3) Max pool size reduced from (2,2) to (3,3), image dimension reduced to 150,150 from 256,256

Result: Pooling reduces the spatial size of the input matrix by converting a given kernel size into a single neuron by finding max of it. Here we increased the max pool size from 2,2 to 3,3 that reduces the output matrix pixels even smaller than the previous model and thus the feature extraction is much abstract and thus avoids overfitting and as a result the train model was slightly improved in terms of test data accuracy.

Experiment -5

Variation from above (4) model: Image size reduced from 256,256 to 22,22,batch size increased to 16 from 10 and epoch to 75.

We varied the above training model by adding weight decay or regularization-L2. Regularization penalizes large weights by adding a term (λ)to the loss function to avoid overfitting and minimise loss. [5]

$$\text{Loss + x where x} = \left(\sum_{j=1}^{n} \|W^{(f)}\|^2\right)\frac{\lambda}{2m} \qquad (1)$$

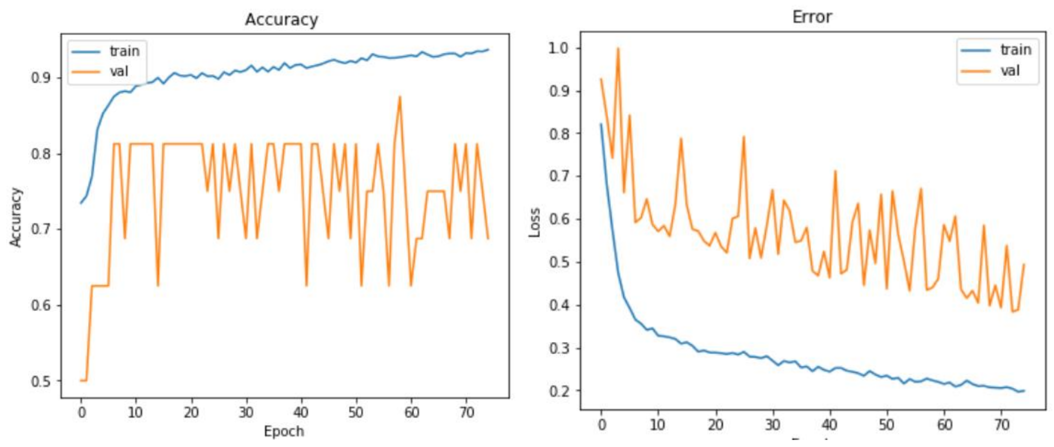
Where n = number of layers,

$W^{(f)}$ = weight matrix of the jth layer,

m= number of inputs.

```
λ= regularization parameter.
```

```
Epoch=75,batch size=16,Image dimensions 22 * 22 ,kernel
32(3,3),32(3,3) and 64(3,3) , Max pool(2,2),Adam(lr=0.0001),
kernel_regularizer=l2(0.01)
```



```
Normal:Correct: 212 mis-classified: 22
accuracy: 90.5982905982906
```

```
PNEUMONIA:Correct 317 mis-classified: 73
accuracy: 81.28205128205128
```

Figure-8 Variation from above (4) model: Image size reduced from 150,150 to 22,22, batch size increased to 16 from 10 and epoch to 75 from 20, weight regularizer added.

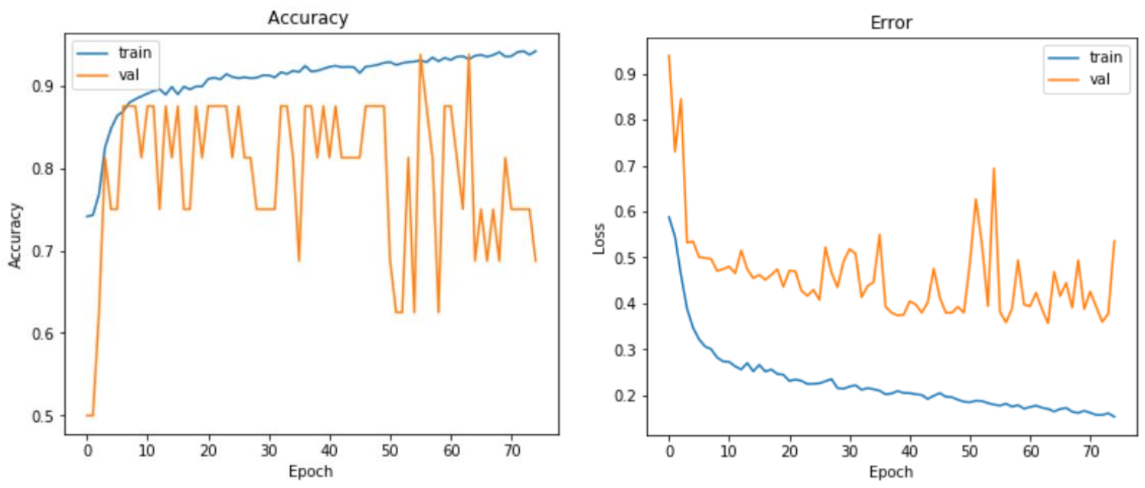
.

Result: If accuracy on test data set is a measurement of train model efficiency then it is recommended to train the model with high image size. However, in our case we used **transfer learning technique** of using layers and weights of previous models to train a variation of the model by downsizing the image size from 256,256 to 22,22. Thus, we reduce the number of parameters to learn from earlier model and computational resources. As we increased the **number of epochs** from 20 to 75 and batch size from 10 to 16,the model improved in learning the features of the image thus the error rate has reduced for the validation data set. [4]. In addition ,**weight regularizes** was added. The higher the value of the regularization , the lower the weights on the network. Thus, it enables to zero out the layers to avoid overfitting and generalize the model using train data to predict efficiently on the unknown data set(test data set). Kera's provides a weight regularization API that allows you to add a penalty for weight size to the loss function. Regularizes reduce training error. The weights connect the nodes between layers.[5]

Experiment -6

Image data augmentation was varied such as:

1. brightness range=[0.2,1.0]- 20% brightness applied to the image to make it darker.



```
Normal:Correct: 112 mis-classified: 122
accuracy: 47.863247863247864

PNEUMONIA:Correct 381 mis-classified: 9
accuracy: 97.6923076923077
```

Figure-9 Image data augmentation applied - brightness range=[0.2,1.0]-

Result: Brightness of the image has been reduced to 20% to make it darker , so that image data generator class can transform the images randomly and apply the convolutional process , as a result the model can generalize and improve the classification results on the test data set. How ever the transformation has adversely decreased the prediction accuracy.

Final output:

The saved model **experiment 5** to the disk has been loaded to test whether the model was able to predict the class of a single image. This experiment model has been chosen because of the lowering trend in the validation data set -error rate and consistent prediction accuracy on both the classes of the test data set.

```
img = image.load_img('E:/chest-xray-pneumonia/chest_xray/test/PNEUMONIA/person1_virus_6.jpeg' , target_size=(22,22))
```

```
 Image class pneumonia:  [1.]
```

From the above python coding we can see the test image of pneumonia class has been predicted correctly.

```
img = image.load_img('E:/chest-xray-pneumonia/chest_xray/chest_xray/test/NORMAL/IM-0010-0001.jpeg' , target_size=(22,22))
```

```
Image class normal:  [0.]
```

From the above python coding we can see the test image of normal class has been predicted correctly.[8]

Novel Corona Virus 2019 Data

We studied COVID 19 using the data set downloaded from https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset. This data set consists of total confirmed cases that includes all the regions and the countries. Time series prediction was implemented using multi-layer perceptron for number of cases per day wise. New data sets were generated for only 6 countries out of the confirmed cases for all the countries over the world, such as Australia, Japan, Italy, U.S, Brazil, and Chile.

The time distribution of cases varies for the countries, with some countries are in decline and others are increasing towards high pandemic. From below histogram we can see that the Australia data distribution is in normal distribution, while Brazil cases increases exponentially.

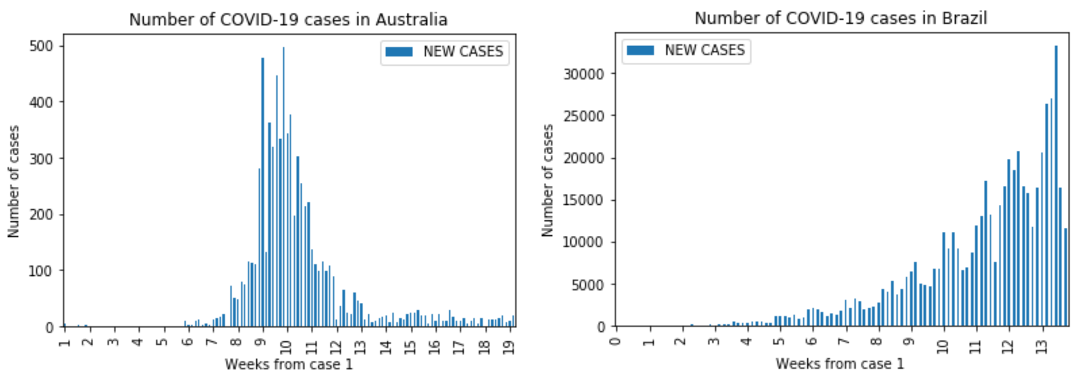


Figure 10 - Histograms with the number of new cases of COVID-a per day from day 1 in Australia and Brazil.

We trained the model using data for the last given number of days(X), so that the model can predict the number of new cases for the following day. Experimentation was conducted with X=3 and X=10. We split the data of a country into train and test data set. After training the model ,test was conducted on the following day data of the time series in the model. Figure 11 showing the training result for the Australia with X=3 and X=10.

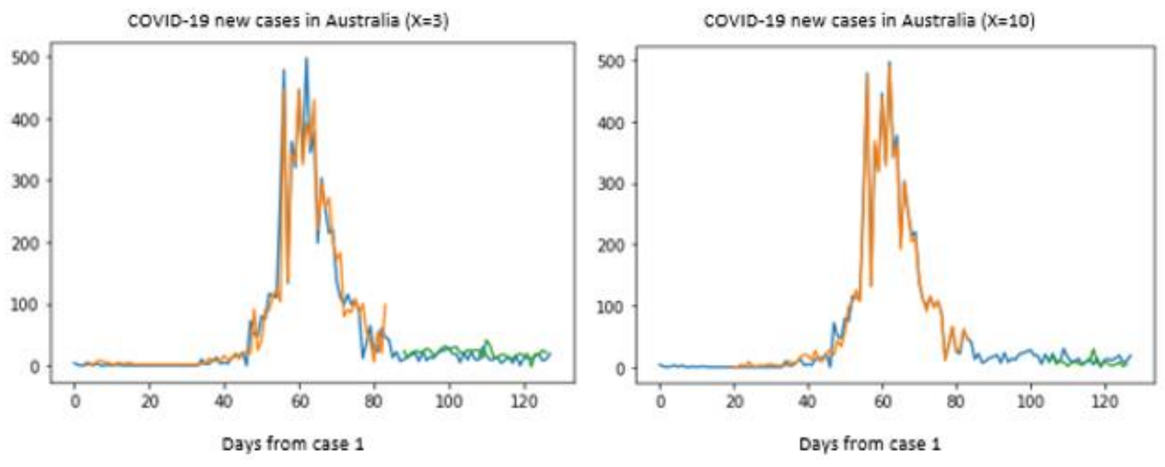


Figure 11 – Showing the prediction of new cases for Australia with X=3 and 10.

In the second experiment, we used to train the data of a country and predict the number of new cases in the other country. Australia data set has been used to train the model and predict new cases in Italy.

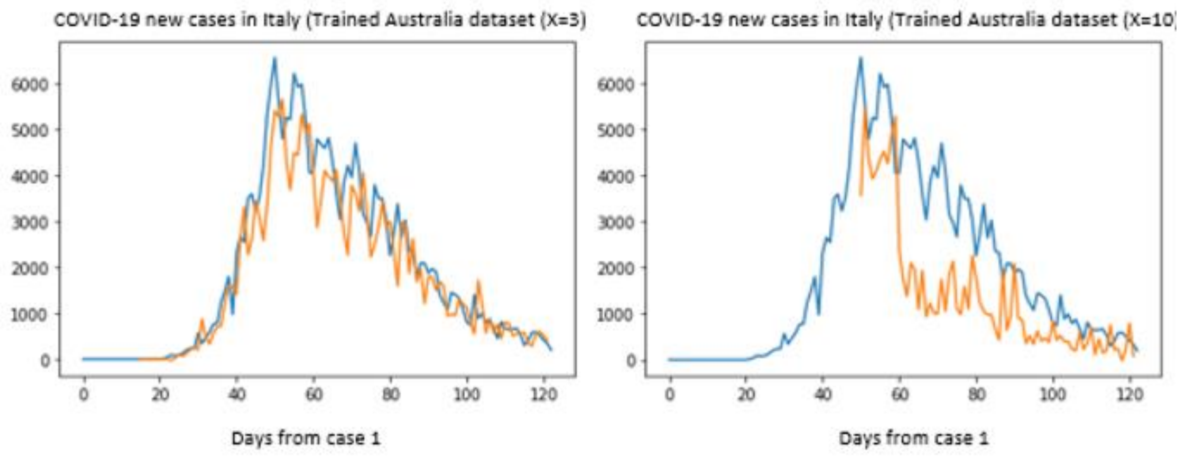


Figure 12 – Showing the prediction of new cases for Italy based on the training, using Australia data set

The analysis results are valid ,especially if the same countries data is used to train and predict the new cases of COVID-19. This analysis results are not only influenced by the past confirmed cases for a country , there are other factors that has impact on the new cases such as isolation level, smaller area for train and predict than a whole country, and number of active cases, government measures to stop the spread of COVID-19 and the demographic nature of the population etc. The given data sets are free from these influences and there could be high variance in the distribution of data across a given period. Therefore, the experimentation conducted is a pilot study only to make time prediction and an attempt to predict the new cases of COVID-19.

References

1. Cs231n.github.io. 2020. *Cs231n Convolutional Neural Networks For Visual Recognition*. [online] Available at: <https://cs231n.github.io/convolutional-networks/> [Accessed 1 June 2020].
2. Brownlee, J. (2020). Failure of Classification Accuracy for Imbalanced Class Distributions. Retrieved 1 June 2020, from https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/
3. Convolutional Neural Network. (2020). Retrieved 1 June 2020, from https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529
4. Boost your CNN image classifier performance with progressive resizing in Keras. (2020). Retrieved 2 June 2020, from https://towardsdatascience.com/boost-your-cnn-image-classifier-performance-with-progressive-resizing-in-keras-a7d96da06e20
5. Convolutional Neural Networks (CNNs) explained. (2020). Retrieved 4 June 2020, from https://deeplizard.com/learn/video/YRhxdVk_sIs

6. From raw images to real-time predictions with Deep Learning. (2020). Retrieved 6 June 2020, from https://towardsdatascience.com/from-raw-images-to-real-time-predictions-with-deep-learning-ddbbda1be0e4
7. network?, W. (2020). Why must a nonlinear activation function be used in a backpropagation neural network?. Retrieved 6 June 2020, from https://stackoverflow.com/questions/9782071/why-must-a-nonlinear-activation-function-be-used-in-a-backpropagation-neural-net
8. Python, O., Scatterday, D., Scatterday, D., Gupta, T., & Baars, S. (2020). One class classification using Keras and Python. Retrieved 8 June 2020, from https://stackoverflow.com/questions/57309958/one-class-classification-using-keras-and-python