Calendar

Description automatically generated 

Rahul Kumar

COMP534 - Applied Artificial Intelligence

IMAGE CLASSIFICATION

Table of Contents

[Abstract 2](#_Toc102940679)

[Introduction 2](#_Toc102940680)

[Evaluation 3](#_Toc102940681)

[Model Architecture and Implementation 3](#_Toc102940682)

[Alexnet 3](#_Toc102940683)

[ResNet50 4](#_Toc102940684)

[Improved ResNet50 5](#_Toc102940685)

[Model Result and statistics 6](#_Toc102940686)

[Conclusion 8](#_Toc102940687)

[References 8](#_Toc102940688)

# Abstract

Machine learning models seek to copy the senses of humans and animals’ ability to classify different objects and natural images. The mirroring effect is done using Convolutional Neural Networks (CNN). They do this by applying weights to small regions of pixels of an image instead of each individual pixel. Through training the CNN classifier updates these weights using backpropagation to improve the classification accuracy on the training set. The high-level features learned by neural networks that contribute to the success are still not fully understood.

# Introduction

The outbreak of Covid-19 on 11 March 2020 was recognised by the World Health Organisation (WHO) as a global pandemic. Covid-19 is a strain of coronavirus know as severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The symptoms of Covid-19 are like flu and cold like viruses that include, headaches, sore throat, high fever, loss of sense of smell or taste, shortness of breath, exhaustion, aching body, blocked nose, diarrhoea and vomiting, some of these symptoms are also find in pneumonia.

Our goal here is to Implement Convolution neural network (CNN) to differentiate the chest-X-ray images of healthy patients, Covid-19 patients and Pneumonia patients by using Alexnet and Resnet50 CNN.

The Convolutional Neural Network (CNN) we will be using is AlexNet and ResNet. AlexNet provides a simple approach and is not a complicated architecture to use. AlexNet was the first CNN to win an image recognition contest and a good starting point to understand the use of CNN’s and image recognition to pick up some good practices on model implementation techniques. AlexNet is also a very efficient computing resource utilisation, data augmentation to make the data more varied by generating image translations and horizontal reflections and dropout which, randomly turns neurons off. These prevent overfitting within neural networks. AlexNet consists of 8 layers, 5 convolutional and 3 fully connected. An advantage AlexNet had is the fact it uses Rectified Linear Units (ReLU) instead of the tanh function. The advantage of this is that it decreased the training time required when using ReLU.

ResNet is a deep CNN with multiple layers to recognise and classify images. We used the ResNet50 variant of the architecture. This model has 48 convolutional layers including one MaxPool and one Average Pool layer totalling 50 layers as the name of the variant suggests. We decided to choose ResNet over VGGNet due to the greater number of layers. This will produce greater training error percentages. It also allows for the addition of shortcut connections and the usage of residual functions. This allows the stacked layers to reduce their training errors.

In this report we will compare the performance of Alexnet and Resnet50 and later we will improve the Resnet CNN and compare the result of improved version with its initial version.

Here we have used jupyter notebook in anaconda to implement our code. There have been several libraries used in this assignment. TensorFlow has been used for the implementation and training of the model. Keras has been used to implement the neural network to run on both CPU’s and GPU’s. Matplotlib has been used to illustrate the charts and images. Cv2 has been used for the image processing, tracking and detection of the images. OS to merge, normalise and retrieve the path names in python. NumPy to convert each image into NumPy array matrix form and Time.

We used a generator to translate the values in each batch so that their mean value was 0 and their standard deviation value was 1. This facilitated the model training by normalising the input distribution.

The generator also transformed single channel X-ray images into three-channel format by repeating the values in the image across all channels. We will want this because the pre-trained model that we'll use requires three-channel inputs. For AlexNet we converted each image in grey scale into 227x227 from 1000x1000. For ResNet50 we converted each image in grey scale into 224x224 from 1000x1000.

The given dataset is organized into 2 folders (train, test) and both train and test contain 3 subfolders (COVID19, PNEUMONIA, NORMAL). Dataset contains total 6432 x-ray images and test data have 20% of total images. The test dataset is used to evaluate the model. Our model should perform well with unseen data and generalise well. If our model can classify the unseen data well, that means that the network is not underfitting or overfitting and it generalises well. Our training data was split further into subsets of training and validation data. 80% of the training subset was used for training and 20% for validation. When implementing our training data, we should see the accuracy increase as the iterations increase and the loss decrease as the number of iterations decreases. If these two actions are synchronised, then the data is good enough to be cross validated with the test data to classify the model.

# Evaluation

Here we used Keras to implement the AlexNet and ResNet50 CNN’s. Below are our initial findings, some statistics and model description.

## Model Architecture and Implementation

### Alexnet

Here we have implemented AlexNet from scratch through the utilization of keras sequential API. Below is the architecture of AlexNet.

* The convolutional layer, this is the dot product between two sets of elements. The operation works on the filtering and array of the layers.
* Batch normalisation layer is an additional layer that acts on the inputs from the previous layer.
* MaxPooling layer, outputs average oof the pixels within the field of the kernel and ensures the maximum pixel value within the field is the output.
* Flatten layer, flattens the image data into a one-dimensional array.
* Dense layer has an arbitrary number of neurons each being a perceptron.
* Activation function adds non-linearity to the network and have greater representation to solve function.
* Rectified Linear Unit Activation Function (ReLU) ensures negative values of neurons are changed to zero and positive values unchanged, this creates the output for the current layer and the input for the next.
* Softmax Activation Function is the probability distribution of the occurrence of a class represented by a vector.
* Dropout reduces the number of interconnecting neurons by dropping one at each training set randomly.

Diagram, engineering drawing

Description automatically generated

The code snapshot of above implementation is shown below:

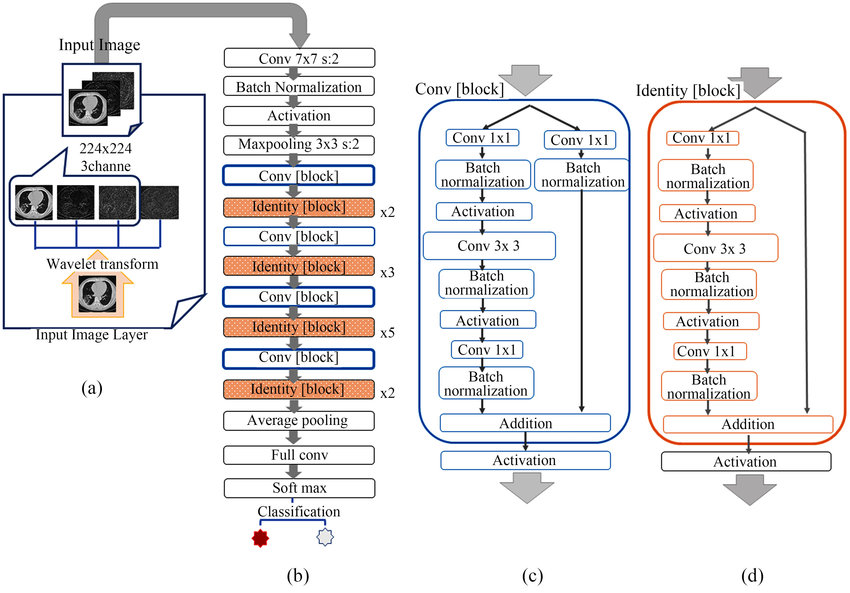
Text

Description automatically generated

### ResNet50

For the implementation of base model of ResNet50 we used kers.application api. Below is the architecture of the ResNet network.

* Convolution layer with a kernel size of 7 \* 7 and 64 different kernels all with a stride of size 2 giving us 1 layer.
* MaxPooling with also a stride size of 2.
* In the next convolution there is a 1 \* 1,64 kernel following this a 3 \* 3,64 kernel and at last a 1 \* 1,256 kernel, these three layers are repeated in total 3 time so giving us 9 layers in this step.
* Then a kernel of 1 \* 1,128, then a kernel of 3 \* 3,128 and finally a kernel of 1 \* 1,512. This is repeated 4 time to give us 12 layers.
* Then a kernal of 1 \* 1,256 and two more kernels with 3 \* 3,256 and 1 \* 1,1024. This is repeated 6 time with a total of 18 layers.
* A 1 \* 1,512 kernel with two more of 3 \* 3,512 and 1 \* 1,2048 and this was repeated 3 times giving us a total of 9 layers.
* Average pool, with a fully connected layer that contains 1000 nodes and a softmax function at the end. This also gives us 1 layer.



### Improved ResNet50

To improve the ResNEt50 we used the following steps:

* Modified the data argumentation shown below, our validation loss decreases over the iterations. The reason for this is that ResNet50 expects specific pre-processing operations

Data Augmentation for initial version of ResNet50

Text

Description automatically generated

Data Augmentation for improved version of ResNet50

Text

Description automatically generated

* Change the optimizer to NAdam so the model will not just stick to finding the local minimum but the global minimum.
* Added extra layers to our base model i.e., a flatten layer, a dense layer, a dropout layer and another dense layer as shown below:
* We added an extra dense layer with 3 output and with activation function as SoftMax. Since we need the probability of each class as output. The neural network will classify the class based of the highest probability of that class.

Table

Description automatically generated

## Model Result and statistics

In this section we will discuss the result and statistics of base model of AlexNet, ResNet50 and improved version of ResNet50.

Below table compares our finding for Alexnet and initial ResNet50 model:

|  |  |  |
| --- | --- | --- |
| Statistic | AlexNet | ResNet50 |
| Training Loss | 0.0459 | 2.6157 |
| Training Accuracy | 0.9840 | 0.5312 |
| Validation Loss | 0.0859 | 998.6192 |
| Validation Accuracy | 0.9728 | 0.1875 |
| Evaluation | 0.0429 | 810.7047 |
| Evaluation Loss | 0.9868 | 0.2461 |

Table 1

We are considering validation loss of a model as a main measuring criterion. Considering this and looking at table 1, we can say that initial model of ResNET50 performs badly. Later, we tried to improve this model to perform better to classify Images from our dataset. On other hand the AlexNet does a better job.

From Figure 1 and 2, we can observe that the model accuracy increases, and model loss decreases as epoch increases for AlexNet base model. This confirms that the model is overfitting or underfitting and it’s generalising well. From figure 3 and 4, we can observe that the model accuracy for validation data remains increases and decreases for each epoch till epoch 10 and there is a sharp spike at epoch 8. Model loss for validation data kind of flatten out after 2nd epoch for ResNet50 initial model.

Chart, line chart

Description automatically generated Chart, line chart, histogram

Description automatically generated

Fig 1 Fig 2

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Fig 3 Fig 4

Below table compares our finding for initial ResNet50 and improved ResNet50 model:

|  |  |  |
| --- | --- | --- |
| Statistic | ResNet50 | ResNet50 Improved |
| Training Loss | 2.6157 | 0.9860 |
| Training Accuracy | 0.5312 | 0.6719 |
| Validation Loss | 998.6192 | 0.9889 |
| Validation Accuracy | 0.1875 | 0.6250 |
| Evaluation | 810.7047 | 0.9748 |
| Evaluation Loss | 0.2461 | 0.6638 |

Table 2

From Table 2 we can observe that the Initial model of ResNet50 performs badly as explained above. Whereas the improved ResNet50 performs much better. Figures 5 and 6 are showing model accuracy and model loss for the improved ResNet50 compared to the initial ResNet50 shown above.

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Fig 5 Fig 6

From above figures we can observe that validation loss decreases, and the validation accuracy increases for improved ResNet50 as the number of iterations increase.

# Conclusion

The challenges of this project were first selecting which architectures to use. We decided upon AlexNet for its simplicity and ease to use. At first, we decided to use VGGNet as a comparison architecture and found this difficult to use and manipulate. When reviewing our options, we discovered that ResNet was a much better architecture due to the extra layers and like AlexNet much easier to use. We used the ResNet50 variant of the architecture.

Once our architectures where chosen we ran into computational difficulties when building the AlexNet model from scratch. The computer used did not have the capacity to run the model and caused a lot of errors. A result of this was to use a library to run the ResNet50 architecture rather than build it from scratch.

Once our models were done, we then had to split them into two separates .ipynb files to run them. When we first implemented the

Throughout this assignment, Rahul worked on the code, analysis and readme file and James worked on the report and we both sat together and finalised the report. The two of us then brought it all together and cooperated when decisions needed to be made.

# References

* <https://ieeexplore.ieee.org/document/9687944>
* <https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951>
* <https://snap.stanford.edu/class/cs224w-2017/projects/cs224w-47-final.pdf>
* <https://www.researchgate.net/figure/Outline-of-ResNet-50-architecture-a-A-3-channel-image-input-layer-The-LL-LH-and-HH_fig3_343233188>
* <https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637>
* <https://www.tensorflow.org/api_docs/python/tf/keras/Model#evaluate>
* <https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator>
* <https://stackoverflow.com/questions/61230762/with-resnet50-the-validation-accuracy-and-loss-is-not-changing>
* <https://towardsdatascience.com/implementing-alexnet-cnn-architecture-using-tensorflow-2-0-and-keras-2113e090ad98>
* <https://keras.io/guides/sequential_model/>
* <https://www.kaggle.com/datasets/prashant268/chest-xray-covid19-pneumonia>
* <https://medium.com/edureka/convolutional-neural-network-3f2c5b9c4778>