Abstract

The main purpose of this project is to develop a convolutional neural network which categorizes images from the CIFAR-10 dataset [1] using the famous AlexNet architecture. The project involved two the training of two architectures of this convolutional neural network where first its classical architecture was implemented, then a variant of it where some parameters are adjusted yielding improved performance of the classification of the images in the CIFAR-10 dataset using this neural network. In this report, the architectures used are detailed and the results of the performance that was obtained for each architecture are exposed and interpreted as well.

Introduction

A Convolutional neural network using the AlexNet architecture will be implemented in this project. It uses the data set CIFAR-10 instead of ImageNet due to the smaller quantity of data. In this project there will be variants of the architecture in order to improve the overfitting of basic implementation to achieve better performance. The performance was improved from 72% from the original architecture to 81.4% by the end of this project.

About the dataset

The dataset being used is CIFAR-10 is a labeled subset of 80 million tiny images collected in order to be used for machine learning and computer vision algorithms, it contains 60000 divided into five training batches and one test batch, each with 10000 images with low resolution of pixels (32 x 32).

The dataset contains 10 different classes including airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.

The following figure is a random display of some of the images in the dataset.

Figure 1: Sample images from different classes in the CIFAR10 dataset.

Methods

For this project Tensorflow and Keras were used. TensorFlow is an open-source machine learning platform that runs from end to end. It has a large, flexible ecosystem of tools, libraries, and community resources that allow academics to advance the state-of-the-art in machine learning and developers to quickly construct and deploy ML applications. Keras is a cognitive load-reduction API that delivers uniform and simple APIs, reduces the number of user activities required for typical use cases, and gives clear and actionable error signals.

About the architecture of the network

Architecture 1: Conventional AlexNet

AlexNet is a convolutional neural network architecture that competed and won in the ImageNet large scale visual recognition challenge in 2012, it achieved a 47.1% top-1 error and 28.2% top-5 error [2]. This architecture consists of eight different layers, five of them being convolutional and three fully-connected layers as shown in the following figure.

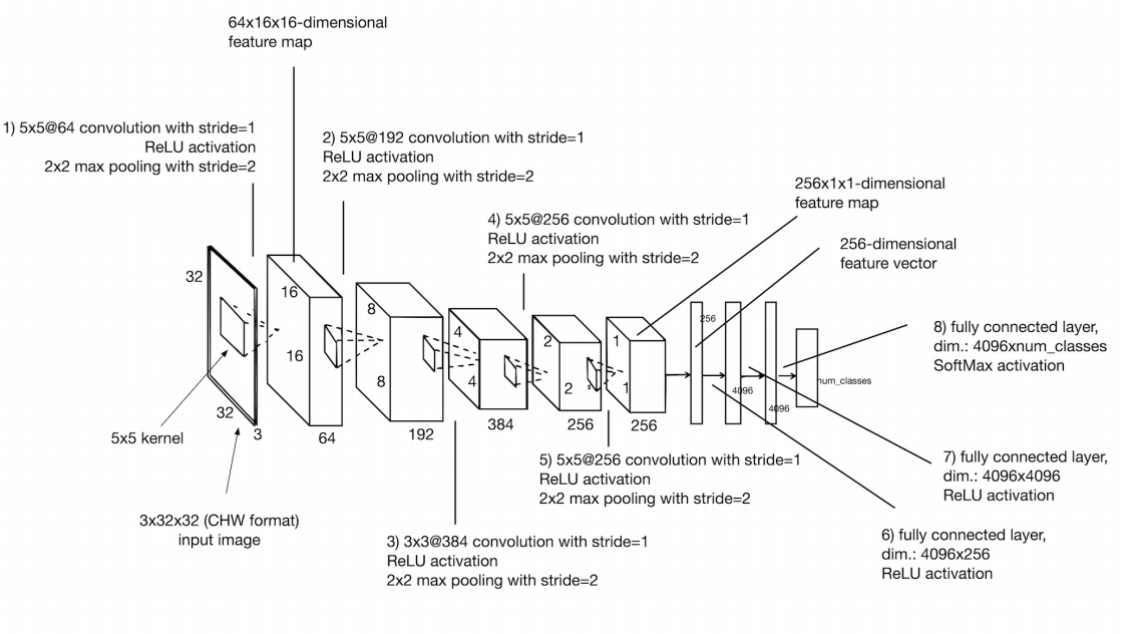

Figure 2: Conventional AlexNet (Same architecture that was used for imageNet).

Architecture 1: Modified AlexNet

After training the AlexNet on the CIFAR10 dataset, different adjustments were done to improve the performance attained by the classical AlexNet Network. The adjustments are as follows:

Adding Dropout

In this task, the classical architecture is modified by adding dropout layers to reduce overfitting. This is done by using the Dropout2d function in Keras. A spatial dropout layer was implemented before each max-pooling layer. The probability was set at the beginning at p = 0.2 but later changed to get better performance as shown in the results. A dropout layer was placed following every fully connected layer using probability of p = 0.5 skipping the last layer [3].

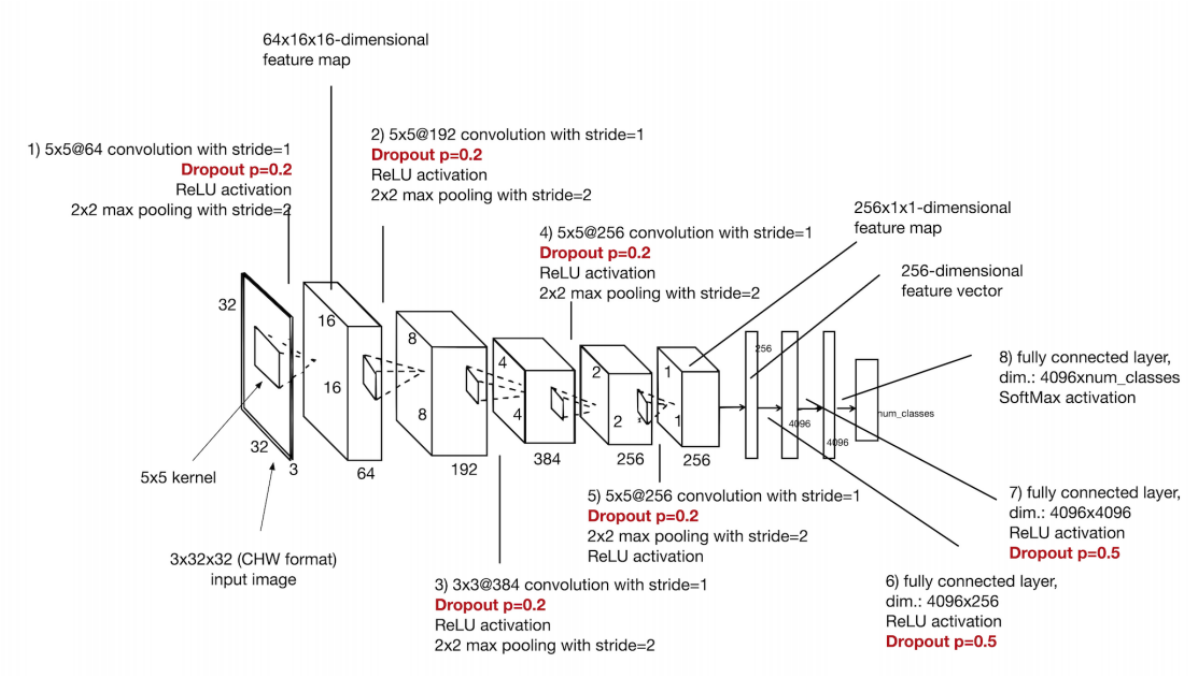

Figure 3: Modified AlexNet architecture by adding Dropout layers.

Adding Batch Normalization

Batch Normalization layer was added to increment the performance of the architecture. In this case a BatchNorm2D for convolutional layers and BatchNorm1D for fully connected layers were added [4].

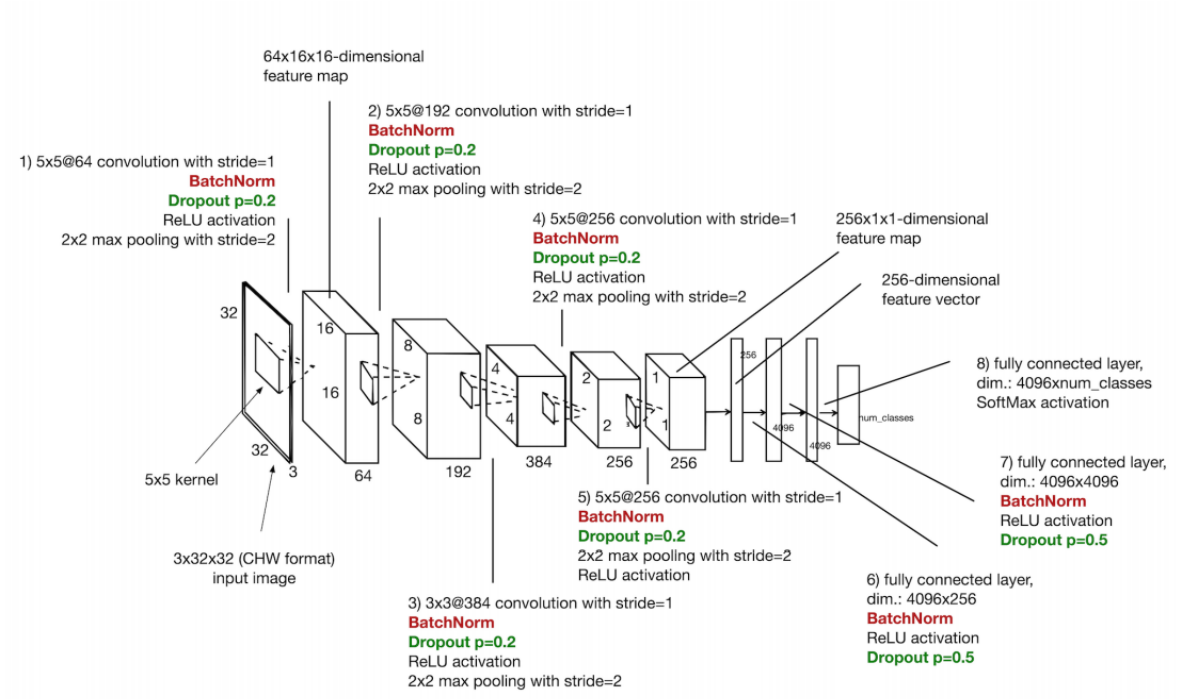

Figure 4: Modified AlexNet architecture by adding BatchNorm layers.

Removing max pooling layers

Max-pooling layers were removed and the fully-connected layers by convolutional layers were replaced. While removing the layers in order to keep the number of elements equal, the stride of convolutional layers were increased from 1 to 2.

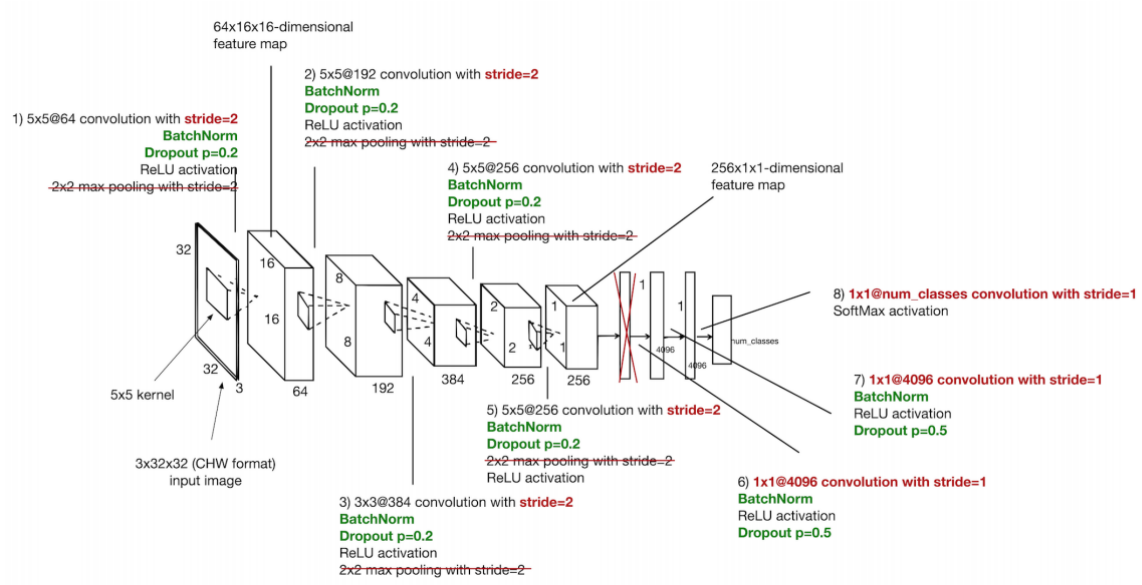

Figure 5: All convolutional approach by removing maxpooling.

Adding Image Augmentation

Image augmentation is a technique for creating more training data from current training samples by "augmenting" them with a series of random modifications that result in believable-looking images.

Results

Task 1

Optimizer = RMSprop(learning_rate = 0.001)

Optimizer = RMSprop(learning_rate = 0.001)						
Trial	**Training loss**	**Training accuracy**	**Validation loss**	**Validation accuracy**	**Test loss**	**Test accuracy**
1	**0.1360**	**0.9750**	**2.2922**	**0.7339**	**2.395475**	**0.7353**
2	**0.0994**	**0.9779**	**3.7392**	**0.7376**	**4.013545**	**0.7289**
Optimizer = 'adam'						
Trial	**Training loss**	**Training accuracy**	**Validation loss**	**Validation accuracy**	**Test loss**	**Test accuracy**
1	**0.0369**	**0.9890**	**1.8722**	**0.7127**	**1.897395**	**0.7113**
2	**0.0343**	**0.9890**	**1.7727**	**0.7178**	**1.851101**	**0.712**

Table 1: Accuracies and losses of Conventional AlexNet with different optimizers.

Optimizer = RMSprop(learning_rate = 0.001)

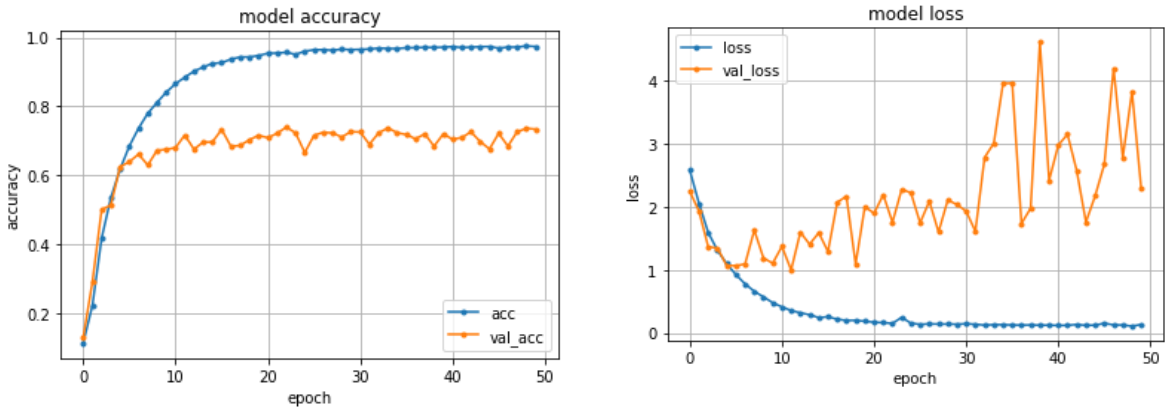

Figure 6: Graphs of Model accuracy and loss of Conventional AlexNet with RMSprop optimizer.

Optimizer = 'adam'

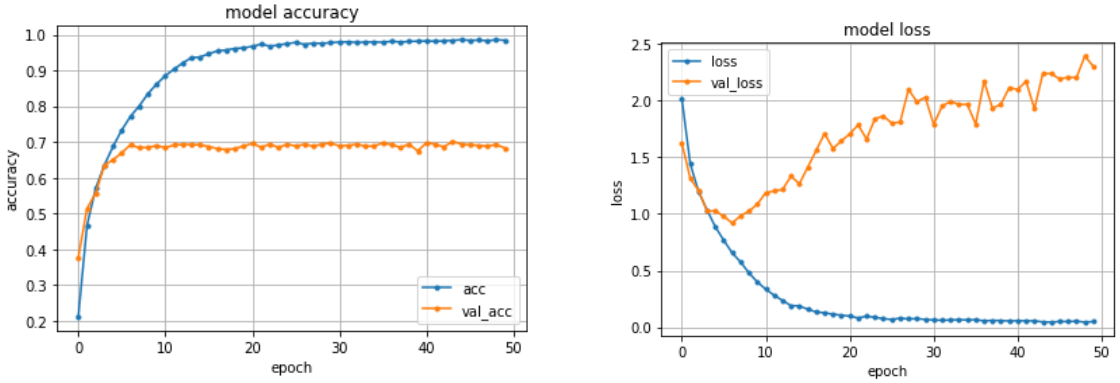

Figure 7: Graphs of Model accuracy and loss of Conventional AlexNet with adam optimizer.

Optimizers are used to speed up large training data models, in the first case RMSprop does that by using exponential decay in the first step. In the case of adam (Adaptive Moment Estimation), it keeps records of the average of previous gradients, which decays exponentially. By looking at these results it can be said that the best optimizer is adam, this is why it is used in the forthcoming tasks.

There is overfitting because the training set increases accuracy till 100% and validation saturates around 70%.

Task 2

Dropout rate = 0.2

Trial	**Training loss**	**Training accuracy**	**Validation loss**	**Validation accuracy**	**Test loss**	**Test accuracy**
1	**0.4527**	**0.8578**	**0.7815**	**0.7454**	**0.796788**	**0.741599**
2	**0.4583**	**0.8537**	**0.7661**	**0.7547**	**0.789529**	**0.748399**

Table 2: Model accuracy and loss of AlexNet with Dropout rate of 0.2.

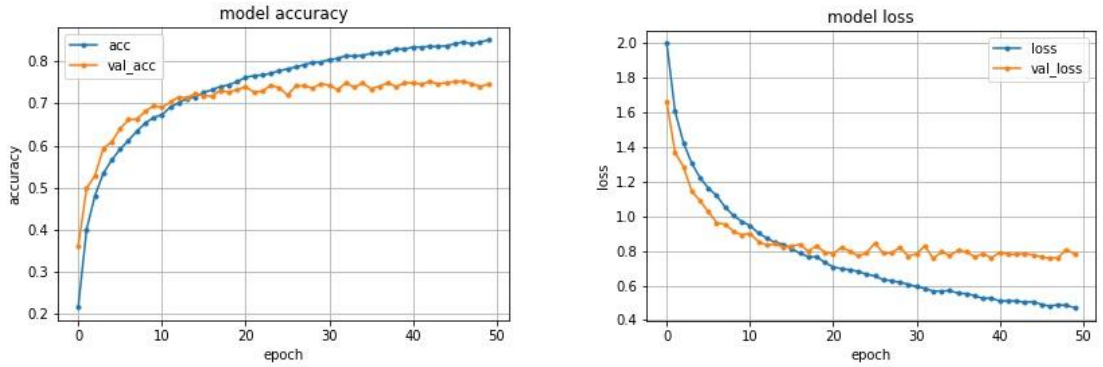

Figure 8:Graphs of Model accuracy and loss of AlexNet with Dropout rate of 0.2.

After adding dropout the accuracy has increased compared to previous case by roughly 4% decreasing the overfitting, because we are dropping 30% of the original neurons and the other 70% adapts and the weights of the neurons rescale to fit the gap.

Task 3

Trial	Training loss	Training accuracy	Validation loss	Validation accuracy	Test loss	Test accuracy
1	0.2310	0.9216	0.7625	0.8021	0.773	0.79689
2	0.2308	0.9232	0.7376	0.8023	0.761	0.79939

Table 3: Model accuracy and loss of AlexNet with BatchNormalization.

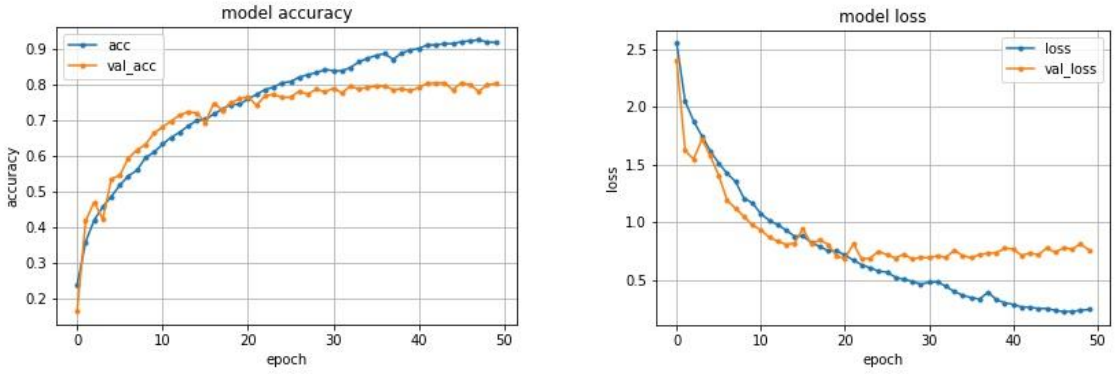

Figure 9: Model accuracy and loss of AlexNet with BatchNormalization.

The accuracy has increased due to batchNormalization. This operation is used to address the vanishing/exploding gradients problems by letting the model learn the optimal scale and mean of each of the layer's inputs.

Task 4

Trial	Training loss	Training accuracy	Validation loss	Validation accuracy	Test loss	Test accuracy
1	0.1916	0.9366	1.0033	0.7342	1.068962	0.7229
2	0.1830	0.9381	1.0603	0.7309	1.045284	0.7275

Table 4: Model accuracy and loss of AlexNet without Max-pooling.

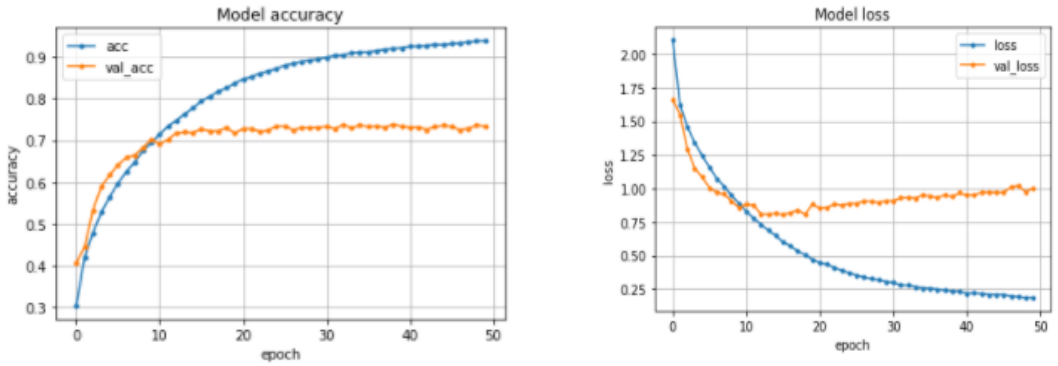

Figure 10: Graphs of Model accuracy and loss of AlexNet without Max-pooling.

The accuracy decreased due to the removal of Max-pooling, this function acts as a filter helping to keep the main features of the images increasing the accuracy.

Task 5

Trial	Training loss	Training accuracy	Validation loss	Validation accuracy	Test loss	Test accuracy
1	0.4820	0.8308	0.5154	0.8223	0.5379	0.81449

Table 5: Model accuracy and loss of AlexNet with Image Augmentation.

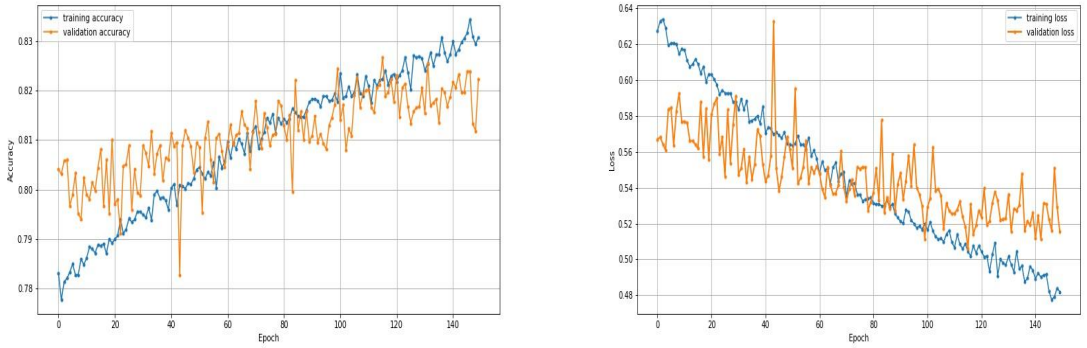

Figure 10: Graphs of Model accuracy and loss of AlexNet with Image augmentation.

Image augmentation increases the training set by transforming the pictures and creates more samples to be used, this increases the accuracy of the model.

Conclusion

In this project, AlexNet was implemented on the CIFAR10 dataset where the performance of the conventional network was investigated and some modifications on the original network architecture such as adding dropout, batchNormalization, data augmentation were added for enhancing the performance of the original architecture. The results of the original architecture and the modified architectures were assessed through the accuracy and loss metrics for the testing, the validation and the test datasets. This resulted in an improvement of almost 10%. This method can be applied to other CNN architectures in order to optimize them and get better performances.

Bibliography

[1] Extracted from: https://www.cs.toronto.edu/~kriz/cifar.html

[2] Extracted from: **https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951**

[3] Extracted from: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

[4] Extracted from: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/