Report ISE 244

Problem Definition :

Imagine that you are sitting in an automatic car and you can see a pedestrian crossing the road, but how will you make the system understand that?
Imagine a person has a very small tumor in his stomach, but it is so small that it's invisible to the human eye. So how to detect such tumors?
The answer to these questions is the implementation of applications that help to detect and classify the object. Automatic cars can utilize this technique to see if there is some obstacle ahead or not. Medical practitioners can use this application to detect undetectable tumors. Artificial Neural Networks are trained so that they can process the input image and then detect what is the object in front of them.

Project Objectives:

My project is based on the research study that was conducted in[1]. In this study, the authors aimed to implement a Neural Network framework that can detect and classify objects. Not only that, these applications should outperform human beings in this task. In this paper, authors have utilized CIFAR 10 dataset to train and evaluate the performance of their model.
In my project also I have utilized the same dataset but my Neural Network Architecture is completely different from theirs.

The main objective of my study is to:

A) Utilize more than one Dense layer in the Neural Network Architecture in order to analyze how adding more dense layers affect the performance of Neural Network framework. In simple terms what is the effect of increasing network complexity on the performance of a model in this case.
B) Secondly I aim to work with both colored as well as grayscale images in order to compare classification accuracies for the two cases.

My Analysis as well as result is based on the above objectives

Analysis

Number of epochs =10	Grayscale image with network having multiple dense layers	GrayScale images network having with single dense layer	Colored images with network having multiple dense layers	Colored images with network having single dense layer
Training, Validation And Test Losses (Sparse Categorical Cross Entropy)	0.6049 0.6809 0.6794	0.6505 0.7747 0.7865	0.4734 0.6300 0.6421	0.5094 0.7200 0.7064
Training, Validation And Test Accuracies	79.07% 77.39% 77.64%	77.67% 74.83% 74.32%	83.4% 79.4% 79.3%	82.17% 77.24% 77.63%

The number of epochs are the same in all four cases for fair comparison. From the above table we can conclude that:

For grayscale images

1) Network having multiple dense layer gives slightly better accuracy
2) Network having multiple dense layers has slightly less loss compared to the other model.

For colored images

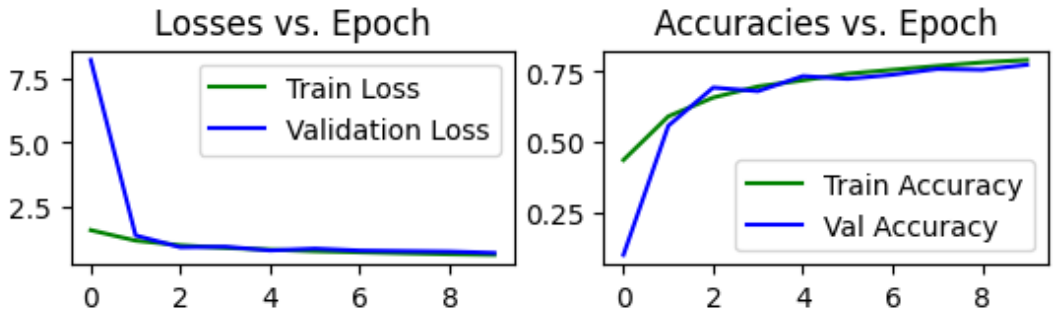
1) Network having multiple dense layer gives slightly better accuracy
2) Network having multiple dense layers has slightly less loss compared to the other model

So I can deduce that a network having multiple dense layers has slightly better performance. Also another deduction is that performance of both the networks is better with colored/RGB images as compared to grayscale images. This is the reason I believe that the authors in [1] choose to work with RGB images.

Results:

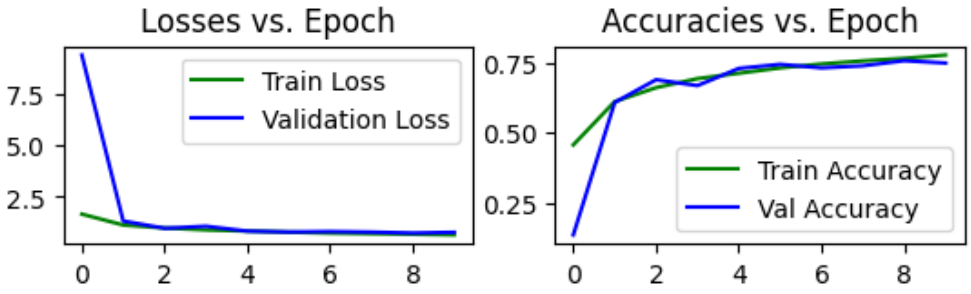
1) Grayscale with network having multiple dense layers

```
['test_loss', 'test_accuracy']
['loss', 'sparse_categorical_accuracy']
[0.6794183850288391, 0.7764000296592712]
```



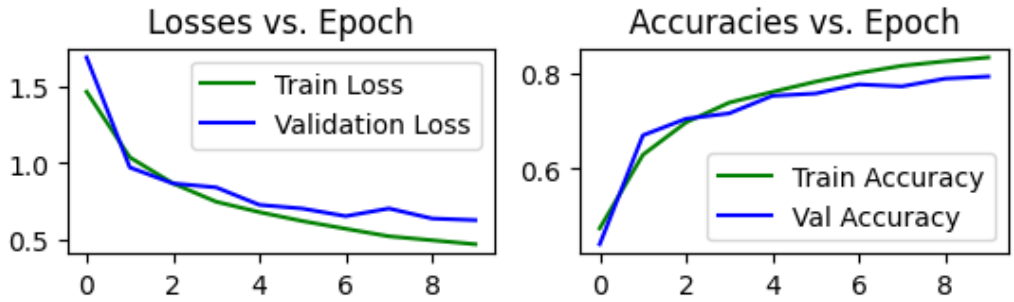
2) Grayscale with network having single dense layer

```
['test_loss', 'test_accuracy']
['loss', 'sparse_categorical_accuracy']
[0.7865287661552429, 0.7432000041007996]
```



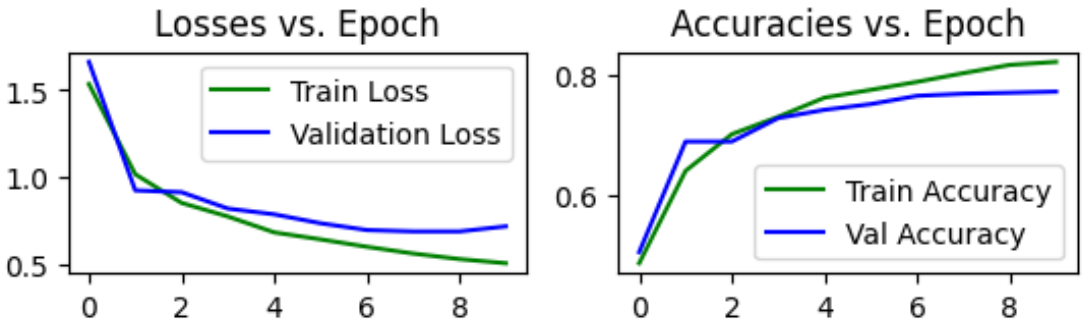
3) Colored with network having multiple dense layers

```
['test_loss', 'test_accuracy']
['loss', 'sparse_categorical_accuracy']
[0.642134964466095, 0.7929999828338623]
```



4) Colored images with network having single dense layer

```
['test_loss', 'test_accuracy']
['loss', 'sparse_categorical_accuracy']
[0.7063859105110168, 0.7763000130653381]
```



Discussion

The above study shows that Neural Networks models can be utilized to detect and classify objects. From the analysis section, we can deduce that all the cases give an accuracy of nearly 75- 80 percent, which is quite decent if we look at the complexity of the task that is being performed.
The above networks work well with both colored as well grayscale images.In reality, objects are colored, so Neural Network frameworks can do a good job in classifying those objects. Now looking at the questions asked in the problem definition section, we can say that a framework similar to the above can be utilized in both automatic cars as well as medical equipment to detect tumors.

Evaluation and Reflection

It was not only about coding and implementing a neural network framework that can detect and classify objects. In fact there was more to it than that. The first phase was to find a strong problem statement then the next step was to gather some research papers based on the selected problem statement . Going through those research papers helped me understand two scenarios: first one is before the application that can be utilized for visual recognition framework was implemented and one after such application was implemented.
Object recognition applications not only helped to solve a lot of problems , but also led to a huge advancement in the field of science and technology. These kinds of applications help to solve numerous problems.

Artifacts:

Dataset Information:

CIFAR 10 dataset consists of 60000 images out of which 50000 are used for training and validation and 10000 are used for evaluation.
The images are divided into 10 categories and they are, “airplane”, “automobile”, "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck".

Network Architectures:

A) Model with multiple dense layers

1) Batch Normalization layer
2) Convolution Layer - 64 x 5 x 5 - Relu Activation
3) MaxPooling - 2x2
4) Dropout - 25 percent observation dropped

5) Batch Normalization layer
6) Convolution Layer - 128 x 5 x 5 - Relu Activation
7) MaxPooling - 2x2
8) Dropout - 25 percent observation dropped

9) Batch Normalization layer
10) Convolution Layer - 256 x 5 x 5 - Relu Activation
11) MaxPooling - 2x2

12) Flatten

13) Dense(256) - ReluActivation
14) Dense (512) - ReluActivation
15) Dense(10) - SoftMax Activation

B)Model with single dense layer

1) Batch Normalization layer
2) Convolution Layer - 64 x 5 x 5 - Relu Activation
3) MaxPooling - 2x2
4) Dropout - 25 percent observation dropped

5) Batch Normalization layer
6) Convolution Layer - 128 x 5 x 5 - Relu Activation
7) MaxPooling - 2x2
8) Dropout - 25 percent observation dropped

9) Batch Normalization layer
10) Convolution Layer - 256 x 5 x 5 - Relu Activation
11) MaxPooling - 2x2

12) Flatten

13) Dense(10) - SoftMax Activation

Explanation of Layers

1) Batch Normalization:

Batch Normalization layers are utilized to speed up the process of training. But they should be utilized carefully because sometimes speeding up the training process can prevent the model from learning some important information about the input data.
These layers recenter and rescale data between the layers.

2) Convolution layers:

Convolution layers are used to extract important information from the input data. The encoded information is utilized for tasks such as reconstruction of images, classification of image , etc. As the encoded information has lower dimensionality compared to input data it helps to save a lot of time and resources.

3) Dense Layer:

Dense Layers are used to classify the encoded information.

4) Relu Activation

It is defined as

```
if f(X) > 0:
   then return X
else:
   return 0
```

5) SoftMax Activation:

It converts an input vector into a vector of probabilities. This is utilized at the end because we need class probability distribution for every image in the dataset. This helps in classifying The objects.

6) MaxPooling :

MaxPooling layers are utilized for downsampling tasks. It computes the maximum values over patches of feature map. In this case patches will be 2x2 dimensional.

7) Dropout:

Dropout layer is used to nullify the effect of a certain percentage of neurons. This help to avoid overfitting.

Image Normalization

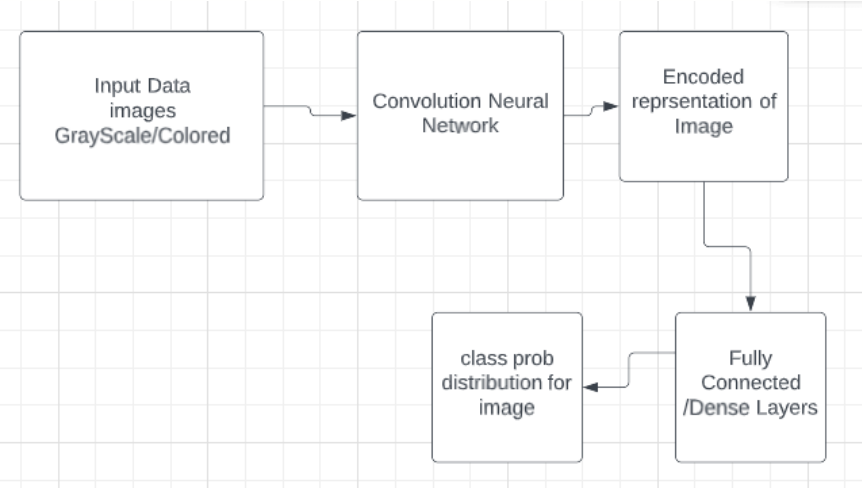
The image pixels are between 0 and 255. These values are so large that if directly passed through neural networks it can give nan loss. To avoid this the images are normalized by dividing all the pixels by 255.

In case of grayscale conversion images are multiplied by [0.229,0.587,0.114]
In this case higher weightage is assigned to the green color component because green is lighter than blue and red.
This conversion formula proved to be more accurate as compared to using rgb2gray inbuilt function.

Each image pixel is represented using 3 dimensions (R,G,B). Then white pixel is represented as (255,255,255) and black is represented as (0,0,0) and gray is represented using (127,127,127).
In order to convert colored pixel (x1,y1,z1) to grayscale we perform the following operation:
0.229*(x1) + 0.587 *(y1) + 0.114*(z1)

Decimals are rounded off to the nearest integer.

Flow diagram:



1) Input is passed through CNN for feature extraction
2) Extracted images are given as input to fully connected layer
3) The output of final layer is class probability distribution for input

Appendix

The plot_prediciton function in my code and splitting of validation and train function is based on a repository in [2].

References:

[1]T. Ho-Phuoc, “CIFAR10 to Compare Visual Recognition Performance between Deep Neural Networks and Humans” *in arXiv,* Nov. 2018.

[2] https://github.com/santanu13/CIFAR_10_Classification_TPU.git