Project – 02b Report

Aparna Cleetus (axc190011), Obuli Vignesh Rangasamy (oxr170630)

Method 3: TensorFlow using Keras API in Python with CNN

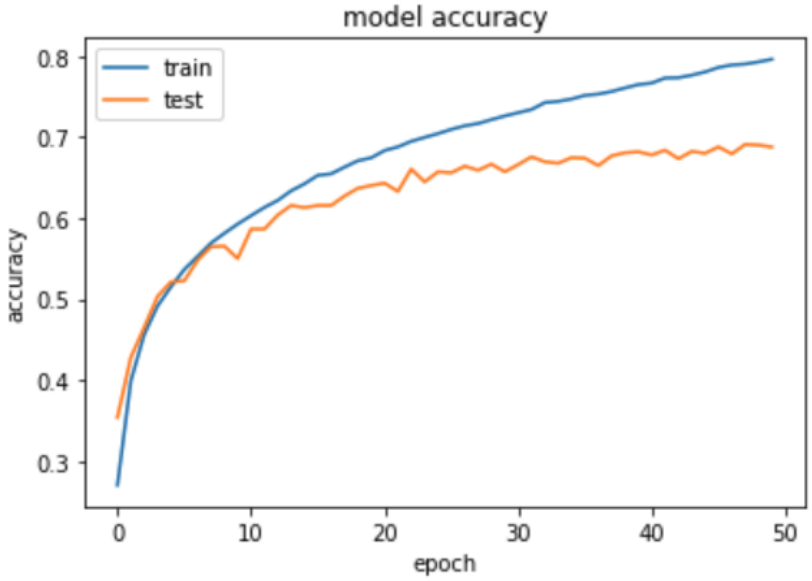
Here we are using the original CIFAR-10 – Object Recognition Image Dataset color images for model building and prediction. The dataset consists of 60000 32x32 color images in 10 classes with 6000 images per class. There are 50000 training images and 10000 test images.

Model	**Parameters**							**Training Accuracy**	**Testing Accuracy**
	Number of Convolution and Max pool Layers	Number of Dense Layers	Size of each Layer and Drop Out Ratio	Optimizer	Activation Function	Learning Rate	No. of Epochs		
1.	Conv2D: 4 MaxPool: 2	3	DropOut: 20% Input: 1024 1: 500 2: 500 3: 500 Output: 10	Adam	Relu	1e-5	20	**65%**	**40.13%**
2.	Conv2D: 4 MaxPool: 2	3	Conv2D (256, (3, 3) MaxPool (2,2) Conv2D (64, (3, 3)) MaxPool (2,2) Conv2D (64, (3, 3)) Conv2D (32, (3, 3)) Dense Layer (64) Dense Layer (32) Dense Layer (10)	Adam	Relu	0.0001	50	**74%**	**68.9%**
3.	Conv2D: 4 MaxPool: 2	3	Conv2D (256, (3, 3) MaxPool (2,2) Conv2D (64, (3, 3)) MaxPool (2,2) Conv2D (64, (3, 3)) Conv2D (32, (3, 3)) Dense Layer (64) Dense Layer (32) Dense Layer (10)	SGD	Relu	0.0001	50	**65%**	**58.9%**

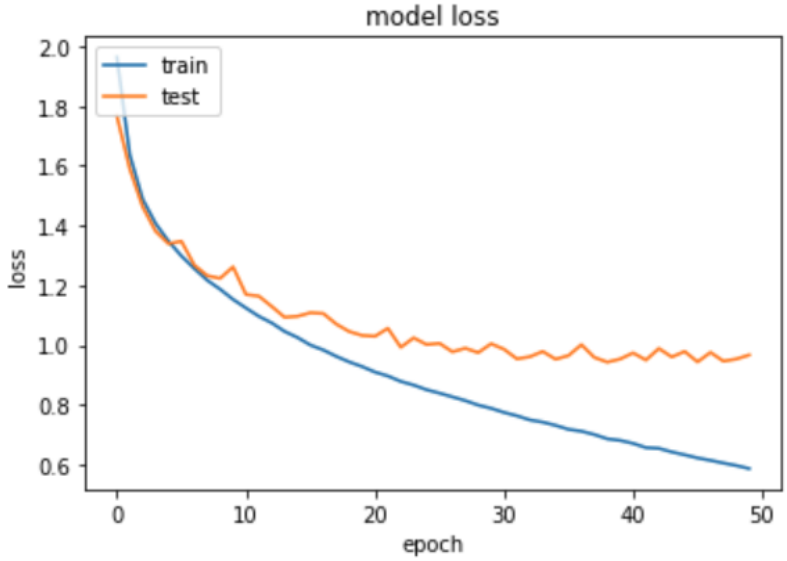
Model	Parameters							Training Accuracy	Testing Accuracy
	Number of Convolution and Max pool Layers	Number of Dense Layers	Size of each Layer and Drop Out Ratio	Optimizer	Activation Function	Learning Rate	No. of Epochs		
4.	Conv2D: 4 MaxPool: 2	4	Conv2D (256, (3, 3) MaxPool (2,2) Conv2D (64, (3, 3)) MaxPool (2,2) Conv2D (64, (3, 3)) Conv2D (32, (3, 3)) Dense Layer (64) Dense Layer (40) Dense Layer (32) Dense Layer (10)	Adam	Relu	0.001	50	**90.79%**	**65.03%**
5.	Conv2D: 3 MaxPool: 3	3	Conv2D (128, (3, 3) MaxPool (2,2) Conv2D (64, (3, 3)) MaxPool (2,2) Conv2D (32, (3, 3)) MaxPool (2,2) Dense Layer (128) Dense Layer (64) Dense Layer (10)	Adam	Relu	0.0001	50	**75.36%**	**67.63%**
6.	Conv2D: 3 MaxPool: 3	3	Conv2D (256, (3, 3) MaxPool (2,2) Conv2D (64, (3, 3)) MaxPool (2,2) Conv2D (64, (3, 3)) MaxPool (2,2) Dense Layer (64) Dense Layer (32) Dense Layer (10)	Adam	Relu	0.0001	50	**79.58%**	**69.44%**
7.	Conv2D: 2 MaxPool: 2	3	Conv2D (256, (3, 3) MaxPool (2,2) Conv2D (64, (3, 3)) MaxPool (2,2) Dense Layer (64) Dense Layer (32) Dense Layer (10)	Adam	Relu	0.0001	50	**83.50%**	**67.88%**
8.	Conv2D: 4 MaxPool: 2	3	Conv2D (32, (3, 3) MaxPool (2,2) Conv2D (32, (3, 3)) Conv2D (64, (3, 3)) MaxPool (2,2) Conv2D (64, (3, 3)) Dense Layer (64) Dense Layer (32) Dense Layer (10)	Adam	Relu	0.0001	50	**76.90%**	**68.34%**

Plots of Model 6, which has an accuracy of 69.44%

Accuracy vs. Epoch

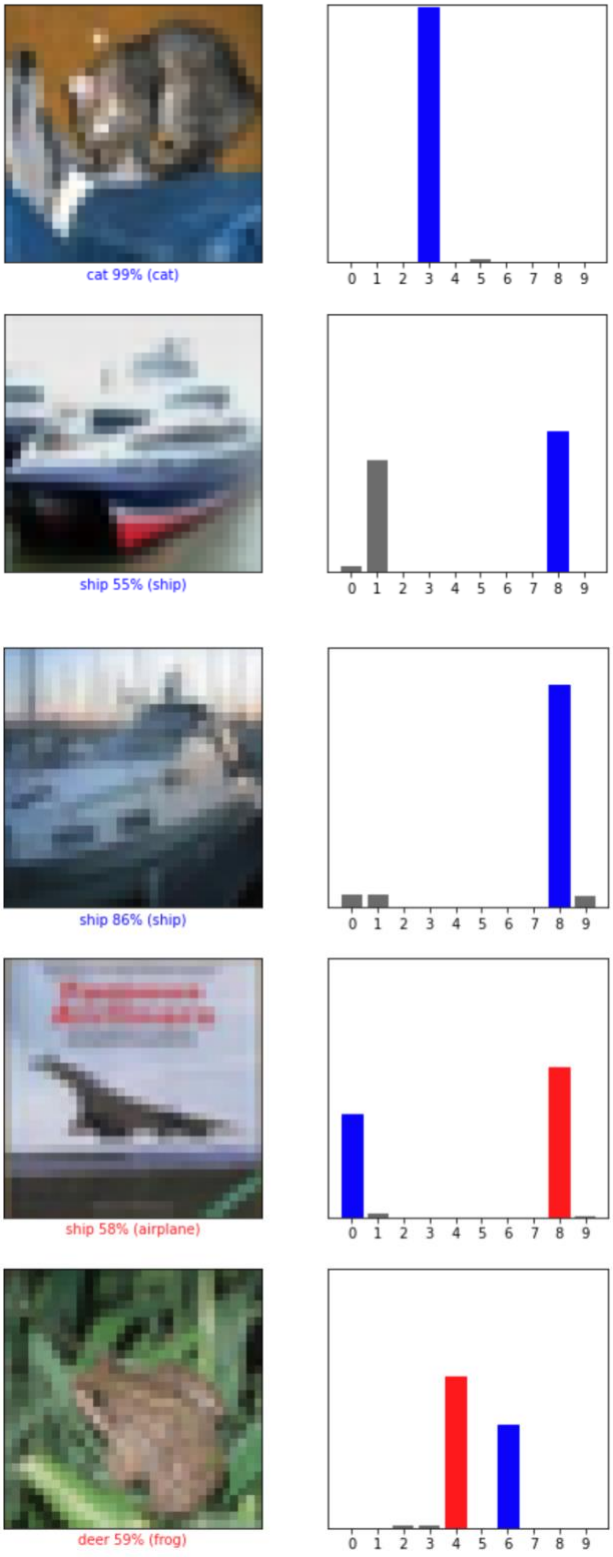


Loss vs. Epoch



Prediction results of Model 6, which has an accuracy of 69.44%

Predicted label with confidence% (True Label) **Prediction probability of different label classes**



Summary

Method 1: R based, non-TensorFlow technique

In this method we used a grey-scale version of the CIFAR-10 data for image recognition using H20 based API. The original color image of size 32x32x3 has been converted into a grey scale image of 32x32x1 to improve the prediction accuracy in R and for faster training times. The dataset consists of 60000 32x32 grey images in 10 classes with 6000 images per class.

After training 4 models on various parameters the best results that we could come up with was 45.67% test accuracy. This model had 3 hidden layers each with 750 neurons, dropout ratio was set to 20%, activation function used was Relu, with a learning rate of 1e-5 and we ran it for 30 epochs.

Method 2: TensorFlow using Keras API in Python

Here we are using the original CIFAR-10 – Object CIFAR-10 – Object Recognition Image Dataset color images for model building and prediction. The dataset consists of 60000 32x32 colour images in 10 classes with 6000 images per class. There are 50000 training images and 10000 test images.

Here again we trained multiple models with multiple parameters and we found that the best model had a test accuracy of 52.1% while the training accuracy went upto 95.8% after running 50 epochs. The model had 8 hidden layers with 20% dropout on 3 hidden layers. We used Relu activation function with learning rate set to 0.0001 and used adam optimizer. The batch size was set to 32. This performed better than the R based method since we used the 3-channel dataset with a deeper network.

Method 3: TensorFlow using Keras API in Python with CNN

Here, again, we are using the same dataset used in method 2.

We trained multiple models with different number of convolution and max pool layers. It was clear that these models with convolution layers performed better than the other two methods mentioned above because of adding convolution/max pooling layers before the dense layers.

After multiple experiments, we got **69.44%** accuracy on the test data set for the model 6 which has 3 sets of convolutional and max pool layers - 1st conv2d layer of size 256*3*3 with maxpool of 2*2, followed by 2nd convolutional layer of size 64*3*3 with maxpool of 2*2 and the last convolutional layer of size 64*3*3 with maxpool of 2*2. Following this there is 3 dense fully connected layers that takes the flatten input from the preceding convolutional and maxpool layers. For this model all the activation functions were Relu. An Adam optimizer with learning rate set to 0.0001 was used.

We noticed that by adding convolutional layers the number of trainable parameters were significantly reduced and the perform of models, in terms of training accuracy, was consistently better as compared to the fully connected dense network from Method 1 & 2. This is due to the fact that weights are shared in convolutional layers as compared to fully connected layers. This sharing of weights not only helps in improving the prediction accuracy especially on image datasets, but also reduces the number of trainable parameters, hence reducing the training time.