Introduction to Deep Learning

Name: **Due Date:** Apr. 8, 2021
Programming Assignment: Number 3

Problem Description

The CIFAR -10 dataset is a collection of images of objects comprising airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks. The dataset is a popular dataset for classification problems in machine learning. Each of the image data is a RGB 32×32 pixel. 50000 training dataset and 5000 testing dataset.

Model Description

To classify the images into their classes, convolutional neural network is used. The network structure is as follows:

- An input layer with shape [Batch-size, 32, 32, 3]
- A Convolution layer with filter size [5, 5, 32] and an output shape of [Batch-size, 28, 28, 32] and a ReLU activation function.
- A Max pooling layer with pool size of (2, 2) and stride [1, 2, 2, 1]. The output shape is [Batch-size, 14, 14, 32].
- A second Convolution layer with filter size [5, 5, 32] and an output shape of [Batch-size, 10, 10, 32] and a ReLU activation function.
- A second Max pooling layer with pool size of (2, 2) and stride [1, 2, 2, 1]. The output shape is [Batch-size, 5, 5, 32].
- A flatten layer to vectorize the output of the second pooling layer. The output shape is [576, 1].
- A fully connected layer of 10 nodes. With a softmax activation function.

A forward pass of this model is implemented by feeding the image data into the network through the input layer. The convolution layers were implemented using tf.keras.layers.Conv2D() and the filter size, padding and activation arguments are passed along.
The pooling layers were implemented using tf.keras.layers.MaxPool2D(), the flattened layer was implemented using tf.keras.layers.Flatten() while the fully connected layer was implemented using tf.keras.layers.Dense().

Results

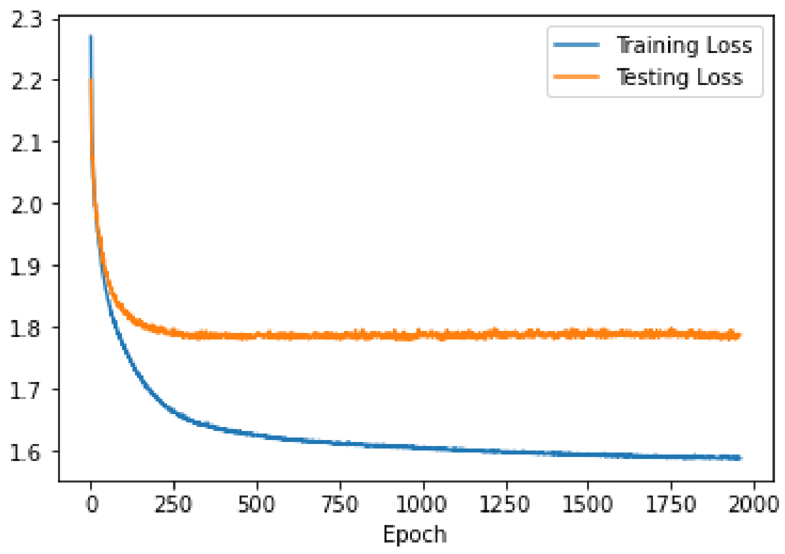

Figure 1: Training and Testing loss

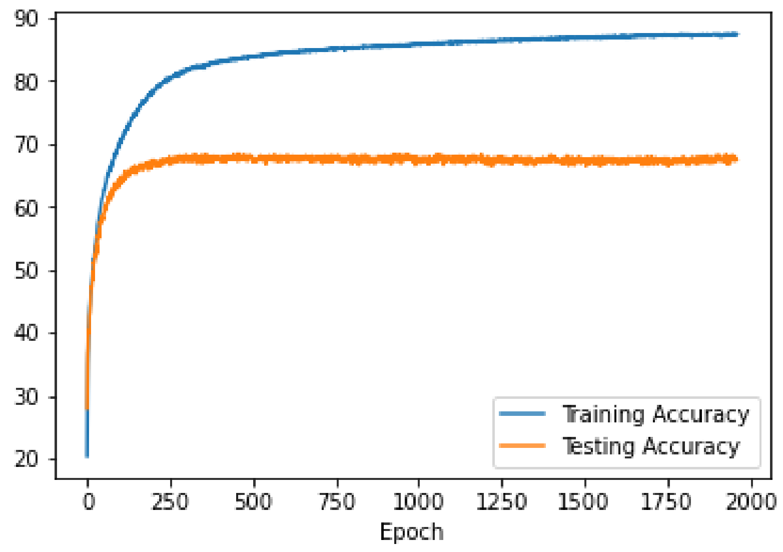

Figure 2: Training and Testing Accuracy

Classification Errors

Classification Errors	
Image Class	Error
Airplane	0.27439
Automobile	0.17131
Bird	0.40890
Cat	0.50193
Deer	0.38031
Dog	0.46654
Frog	0.22374
Horse	0.28810
Ship	0.24060
Truck	0.17850
Overall	0.31440

First layer filters

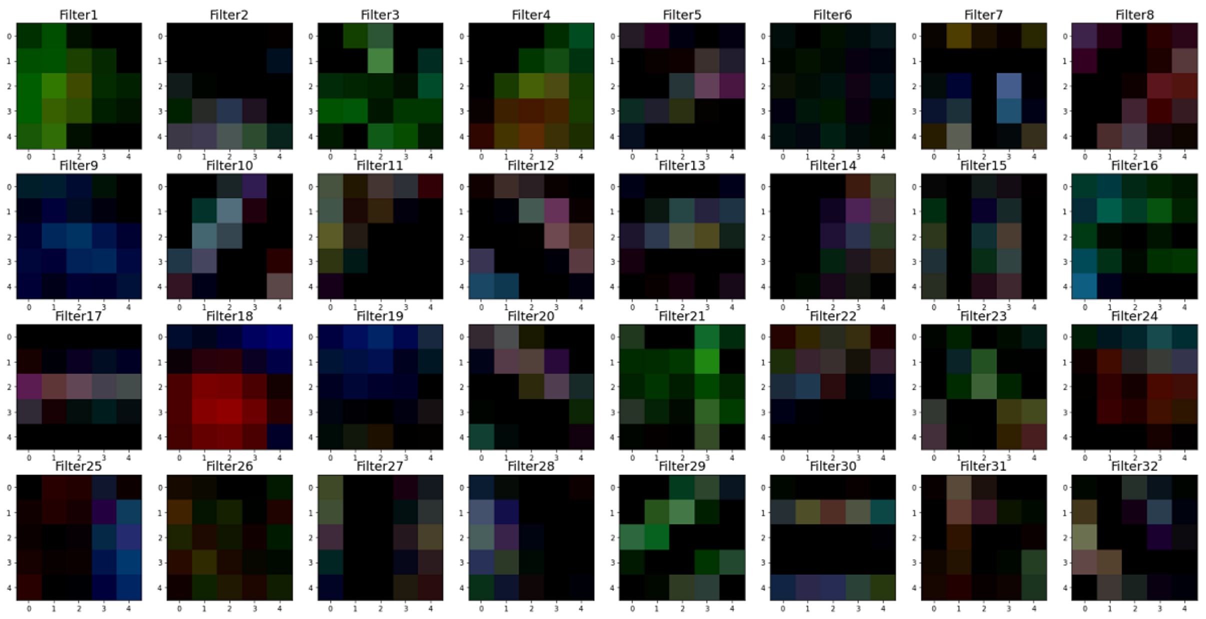

Figure 3: First layer filters

Experimental setting

- Python 3.8 was used for coding.
- Training and testing data were imported and processed using the hints provided. The data were normalized by dividing with 255.0.
- Training and testing labels were used to obtain one of k encoding.
- Tensorflow version 2.4.0 was to build the model class by subclassing tf.keras.Model.
- the model architecture was set during the class initialization using the constructor __init__() and the forward propagation is implemented using the call function.
- The gradients of the weights were computed using tensorflow's tf.GradientTape() function. The loss function used is the cross entropy; tf.nn.softmax_cross_entropy_with_logits()
- Adam optimizer with learning rate $\eta = 0.0001$ was initially used for training. The optimizer was also used to apply the gradients to the model's trainable variables.
- Model is initialized by calling the model class.
- Training data is shuffled and batches are generated using tf.data.Dataset.from_tensor_slices().batch() function.
- Training was carried out by calling model's train model function.
- Although training loss was calculated during each weight update, the training and testing losses and accuracy were recorded after each epoch.

Hyper-parameters Used

- batch size = 500, No of batches = 50000/500 = 100
- Epoch = 2000
- Number of iterations: No of batches $\times$ Epoch = 100 $\times$ 2000 = 200000
- learning rate $\eta = 0.001$

Discussion

Initially, using the learning rate above yielded a test accuracy of 67.45. When 0.001 was used, there was a slight improvement. Then a variable learning rate was used. After reaching test accuracy of 67%, the rate is then multiplied by 0.9 after each epoch and the test accuracy increased up to **68.56%**. This improved model was used to generate the classification errors and filter plots. At learning rates above 0.01, training and testing accuracy remained at about 10%. Between 0.01 and 0.005, the accuracies obtained were less than 65%.