CS838: Deep Neural Networks

Lab 3: Convolutional Neural Network

Nikhil Nakhate and Felipe Gutierrez Barragan

1. Overview

In this report we present the results of a *convolutional neural network* applied to the classification of a subset of the *CIFAR-10* dataset. The code that we submitted for grading implements layers in the following order:
Input layer, Convolution layer, Maxpool layer, Fully connected layer and Output layer.
We have also implemented early stopping as described in this report.

2. Dataset

Our program takes a three input files corresponding to a trainset, tuneset and testset. Each instance of the dataset is a 32 X 32 X 4 image. Where 32 X 32 specifies the dimension of the image and 4 is the depth that specifies the number of channels namely – Red, Green, Blue and Greyscale.

3. Implementation Details

In this section we give a brief explanation of the implementation details of our neural network.

- ***Weight initialization:*** We initialize weights in our network by drawing them from a zero mean Gaussian distribution whose variance is given by:

  1 / fan-in

  Here fan-in is the number of neurons feeding into that particular neuron.
- ***Error function:*** We used mean squared error as the error function.
- ***Activation functions****:* We implemented Sigmoid and a ReLU as our activation functions.
  - Output layer: We have a Sigmoid activation function for the output layer
  - Conv layer: We have used a 5 X 5 filter for the convolution layer. The depth of the filter depends on the depth of the previous layer.
  - Maxpool layer: The Maxpool layer uses a 2 X 2 filter.
  - All the other layers apart from the output layer use the ReLU activation function in our architechture
- ***Early Stopping:*** Every 10 epochs we would evaluate the neural network on the tuning dataset. We refer to this action hereafter as the *tuning period.* Our program would then halt training if the tuning accuracy would not improve after a total of 10 tuning periods. If the accuracy improves in less than 10 tuning periods then we would restart the count of how many tuning periods had passed that the accuracy had not improved.

4. Experiments

We evaluated our neural network with various hyperparameter combinations. The following tables summarize the hyperparameter space that was explored. The hyperparameters were: activation function, learning rate (eta) and number of hidden units.

Activation function for hidden layers	
Sigmoid Activation	ReLU Activation

Table 1: Activation functions.

Learning Rates (eta)		
Eta = 0.01	*Eta = 0.05*	*Eta = 0.1*

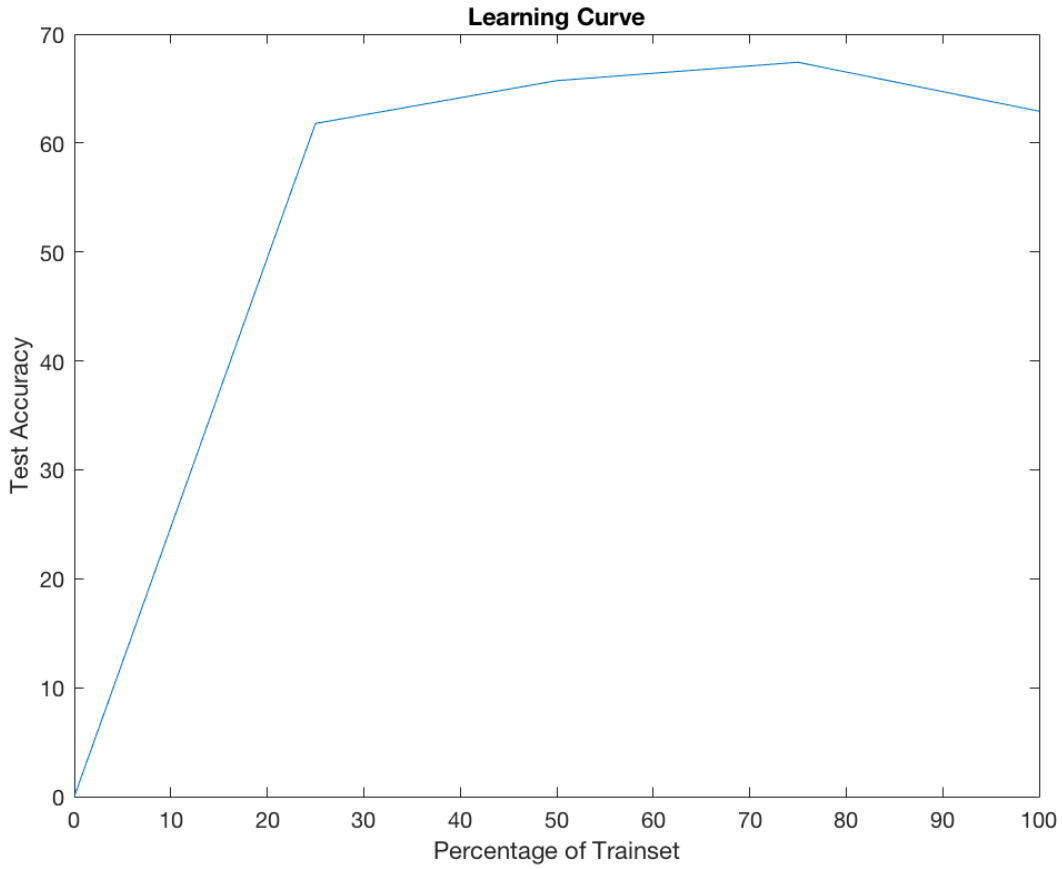
Table 2: Learning Rates.

Number of Hidden Units for the fully connected layer		
HU = 100	*HU = 300*	*HU = 500*

Table 3: Number of hidden units for the fully connected layer

5. Results

Learning curve:



The learning curve that we see in the above figure demonstrates the Test accuracy as a function of the percentage of the trainset used for training the convolutional neural network. As we can see. The accuracy increases with the percentage of the trainset used, but then reduces as the trainset used approaches 100 percent. The configuration of the neural network that was used was as follows. The dimension of the layer is mentioned next to it:

- **Input layer- 32 X 32 X 4**
- **Convolution layer- 28 X 28 X 20**
- **Maxpool layer- 14 X 14 X 20**
- **Fully Connected layer- 1 X 1 X 300**
- **Output layer- 1 X 1 X 6**

Learning rate: 0.05

Single hidden layer:
This configuration of the neural network uses just one fully connected hidden layer.

Number of hidden units	Test Accuracy (%)
100	73.03
300	69.10
500	71.35

Table 4: Learning rate fixed at 0.01

Configuration 1: (Input-Convolution-Maxpool-Fully Connected-Output):

Learning rate	Test Accuracy (%)
0.01	73.59
0.05	62.92
0.10	10.67

Table 4: Number of hidden units fixed at 300

Configuration 2: (Input-Convolution-Maxpool-Convolution-Maxpool-Fully Connected-Output):

Learning rate	Number of hidden units	Test Accuracy (%)
0.01	300	70.22

Final Configuration:
Based upon the above results we reached the conclusion that the best accuracy is achieved using **Configuration 1 with learning rate 0.01.** The code that we have submitted has this configuration. The following is the architecture followed and the confusion matrix obtained for the same:

- **Input layer- 32 X 32 X 4**
- **Convolution layer- 28 X 28 X 20**
- **Maxpool layer- 14 X 14 X 20**
- **Fully Connected layer- 1 X 1 X 300**
- **Output layer- 1 X 1 X 6**

Learning rate: 0.01
Final Test Accuracy: 73.59%
Confusion Matrix:

Predicted labels

		Airplanes	Butterfly	Flower	Grand Piano	Starfish	Watch
Actual Labels	**Airplanes**	36	0	1	0	0	4
	Butterfly	3	7	3	0	4	1
	Flower	1	0	29	1	5	1
	Grand Piano	2	0	1	12	0	4
	Starfish	0	1	3	2	9	2
	Watch	2	2	3	0	1	38

The results are also submitted as text files.

The Key for the file names is as follows:

The files used for the learning curve are named as follows:
trainset_<percent of trainset used>_<learning rate digits after decimal place>_<number of hidden units>.txt

The rest of the results files are named as follows:
<hidden layer list>_<learning rate digits after decimal place>_<number of hidden units>.txt