EE782 : Advanced Machine learning assignment 1 report

Viraj Nadkarni (16D070013)

September 2019

1 Fully Connected Neural Network

To implement a fully connected neural network, I made a class which used several layers which were themselves represented as separate classes. Each class (which represented a layer) has member functions that allow one to calculate the output of the layer given the input (forward pass) and computes the derivative of the loss with respect to weights, biases (if any are present) and the inputs of the layer (used for backpropagation).

1.1 Classes defined

The classes used in the main code have been defined in the code "*layers.py*". The comments in the code describe the working of each class. This code includes classes to implement :

- Linear network layer (*class layer*)
- ReLU layer (*class relu*)
- SoftMax layer (*class softmax*)
- Loss computation layer (*class crossentropy*)
- Fully connected neural network (*class fcnn*)

The class *fcnn* has member functions that allow us to :

- Initialize all the constituent layers of the network by giving the number of neurons in each layer as input.
- Train the FCNN on given training data by dividing the data into batches.
- Validate/test the FCNN on given testing data.

1.2 Main code

The main code is present in the file "*mainF.py*".

The class *fcnn* defined in the previous section has been instantiated in this file and trained on the CIFAR-10 dataset. The code stops training after five consecutive epochs have validation accuracies that are within 1% of each other.

1.3 Data loading in the main code

Since we were required to train the dataset on 2 and 5 classes, I had to modify the existing CIFAR-10 data format to convert images of the particular classes I was interested in into *.npy* files and then load these directly in the first few lines of the main code. To do this, I had to write a seperate code to be ran only once (*loadDataForCifar.py*). Since doing this over the whole dataset took a lot of time I could not include it in my main code (otherwise it would have ran every time I run the main code).

What this code does is the following :

- Convert the 10 subdatasets of CIFAR-10 and combine them into one.
- Convert the combined dataset into a numpy array of dimension $10 \times 5000 \times 3072$.
- A similar thing is done for the testing dataset (converted to $10 \times 1000 \times 3072$).
- In the above arrays, the first index corresponds to the class(0 - 9), the second index corresponds to an image in that class, and the third index corresponds to a pixel in that image.
- The output files are named : "*trainingSeries.npy*" and "*testSeries.npy*".

1.4 Results

This section describes the results obtained when I ran *mainF.py* so as to train a FCNN to classify 2 and 5 classes of the CIFAR-10 dataset.

1.4.1 For 2 classes :

Parameters used for obtaining the results in this section are :

- Layer 1 (input layer) size : 3072
- Layer 2 (hidden layer) size : 70
- Layer 3 (hidden layer) size : 50
- Layer 4 (output layer) size : 2
- Batch size : 10
- Learning rate : 0.01

The following output was obtained after running the code on the classes of automobiles and aeroplanes.

```
Number of weights - 218640
Epoch 0
validation accuracy 77.55
Epoch 1
validation accuracy 80.6
Epoch 2
validation accuracy 81.95
Epoch 3
validation accuracy 83.6
Epoch 4
validation accuracy 84.15
Epoch 5
validation accuracy 84.1
```

```
Epoch 6
validation accuracy 84.35
Epoch 7
validation accuracy 84.6
final accuracy 86.85
```

Therefore, the fully connected neural network obtained an accuracy of 86.85% after training over the two classes.

1.4.2 For 5 classes :

Parameters used for obtaining the results in this section are :

- Layer 1 (input layer) size : 3072
- Layer 2 (hidden layer) size : 70
- Layer 3 (hidden layer) size : 50
- Layer 4 (output layer) size : 2
- Batch size : 25
- Learning rate : 0.01

The following output was obtained after running the code. The five classes chosen for training were aeroplane, automobile, bird, cat and deer.

```
Number of weights - 218790
Epoch 0
validation accuracy 35.32
Epoch 1
validation accuracy 45.4
Epoch 2
validation accuracy 49.48
Epoch 3
validation accuracy 51.46
Epoch 4
validation accuracy 54.06
Epoch 5
validation accuracy 55.68
Epoch 6
validation accuracy 56.78
Epoch 7
validation accuracy 56.38
Epoch 8
validation accuracy 57.3
Epoch 9
validation accuracy 57.78
Epoch 10
validation accuracy 58.5
Epoch 11
validation accuracy 58.66
Epoch 12
validation accuracy 59.84
Epoch 13
validation accuracy 60.26
```

```
Epoch 14
validation accuracy 60.36
Epoch 15
validation accuracy 60.0
Epoch 16
validation accuracy 60.44
Epoch 17
validation accuracy 60.44
Epoch 18
validation accuracy 60.66
final accuracy 61.22
```

Therefore, the fully connected neural network obtained an accuracy of 61.22% after training over the five classes.

2 Convolutional Neural Network

2.1 Main Code

The code for training and testing the convolutional neural network is present in the file "*convNN.py*".

I used TensorFlow to build the CNN in this file. The final CNN consists of :

- Two convolutional layers with increasing number of features and decreasing filter size .
- One max pooling layer after each convolutional layer.
- Two fully connected hidden layer of neurons at the output.

2.2 Data loading in the code

I used the same code as before (*"loadDataForCifar.py"*) to create the files *"trainingSeries2.npy"* and *"trainingLabels2.npy"* from the CIFAR dataset that were used in training the CNN.

2.3 Results

This section describes the results obtained when I ran *convNN.py* so as to train a FCNN to classify 2 and 5 classes of the CIFAR-10 dataset using 50% and 25% of the number of weights used in the fully connected case.

2.3.1 For 50% of the number of weights , two classes :

Parameters used for obtaining the results in this section are :

- Layer 1 filter size : 10×10
- Number of features at the output of layer 1 : 20
- Layer 2 filter size : 5×5
- Number of features at the output of layer 2 : 70
- Layer 3 (fully connected) size : 15
- Batch size : 20

- Learning rate : 0.001

The following output was obtained after running the code on the classes of automobiles and aeroplanes.

```
number of weights : 108230
Epoch 0 - 49.799999594688416
Epoch 1 - 80.25000095367432
Epoch 2 - 76.95000171661377
Epoch 3 - 82.59999752044678
Epoch 4 - 84.95000004768372
Epoch 5 - 85.50000190734863
Epoch 6 - 84.79999899864197
Epoch 7 - 86.50000095367432
Epoch 8 - 85.54999828338623
Epoch 9 - 83.79999995231628
Epoch 10 - 83.95000100135803
Final accuracy - 0.8565
```

Therefore, the CNN obtained an accuracy of 85.65% after training over the two classes with 50% weights.

2.3.2 For 50% of the number of weights , five classes :

Parameters used for obtaining the results in this section are :

- Layer 1 filter size : 5×5
- Number of features at the output of layer 1 : 20
- Layer 2 filter size : 5×5
- Number of features at the output of layer 2 : 70
- Layer 3 (fully connected) size : 15
- Batch size : 100
- Learning rate : 0.0001

The five classes chosen for training were aeroplane, automobile, bird, cat and deer.

```
number of weights : 103775
Epoch 0 - 44.200000166893005
Epoch 1 - 47.999998927116394
Epoch 2 - 51.499998569488525
Epoch 3 - 51.899999380111694
Epoch 4 - 53.39999794960022
Epoch 5 - 54.60000038146973
Epoch 6 - 55.19999861717224
Epoch 7 - 56.40000104904175
Epoch 8 - 56.199997663497925
Epoch 9 - 56.00000023841858
Epoch 10 - 57.099997997283936
Epoch 11 - 56.90000057220459
Final accuracy - 0.5958
```

Therefore, the CNN obtained an accuracy of 59.58% after training over the five classes with 50% weights.

2.3.3 For 25% of the number of weights, two classes :

Parameters used for obtaining the results in this section are :

- Layer 1 filter size : 10×10
- Number of features at the output of layer 1 : 15
- Layer 2 filter size : 5×5
- Number of features at the output of layer 2 : 50
- Layer 3 (fully connected) size : 10
- Batch size : 20
- Learning rate : 0.001

The following output was obtained after running the code.

```
number of weights : 55270
Epoch 0 - 49.799999594688416
Epoch 1 - 75.3000020980835
Epoch 2 - 75.70000290870667
Epoch 3 - 76.30000114440918
Epoch 4 - 78.29999923706055
Epoch 5 - 80.29999732971191
Epoch 6 - 81.99999928474426
Epoch 7 - 80.9000015258789
Epoch 8 - 82.99999833106995
Epoch 9 - 83.89999866485596
Epoch 10 - 83.79999995231628
Final accuracy - 0.8555
```

Therefore, the CNN obtained an accuracy of 85.55% after training over the two classes with 25% of the number of weights used in the FCNN .

2.3.4 For 25% of the number of weights, five classes :

Parameters used for obtaining the results in this section are :

- Layer 1 filter size : 5×5
- Number of features at the output of layer 1 : 20
- Layer 2 filter size : 5×5
- Number of features at the output of layer 2 : 40
- Layer 3 (fully connected) size : 10
- Batch size : 100
- Learning rate : 0.0001

The following output was obtained after running the code.

```
number of weights : 47150
Epoch 0 - 51.20000243186951
Epoch 1 - 55.400002002716064
Epoch 2 - 57.099997997283936
Epoch 3 - 57.70000219345093
Epoch 4 - 58.20000171661377
Epoch 5 - 58.399999141693115
Epoch 6 - 59.700000286102295
Epoch 7 - 60.50000190734863
Epoch 8 - 60.79999804496765
Epoch 9 - 61.400002241134644
Epoch 10 - 62.199997901916504
Epoch 11 - 62.59999871253967
Final accuracy - 0.6518
```

Therefore, the CNN obtained an accuracy of 65.18% after training over the five classes with 25% of the number of weights used in the FCNN .