Deep Learning

Assignment 1

Task 1 Report

Abstract

To develop a connected neural network to process the three subclasses from the dataset CIFAR10, split the validation and test sets, train the developed model and finally to measure the accuracy by multi class classification.

Problem Statement

CIFAR10 is a public datasets with 10 classes and 6000 images each class. By default it is split into train and test datasets. The purpose of the task is to create a neural network, give the training set as the input which includes both the training and the validation sets. The network is hyper tuned to achieve maximum accuracy and finally evaluated against the test sets.

Proposed Solution

The purpose of the classification is to determine to which subset the given test image fits. Since three classes are selected, multiclass classification is implemented. In case of model design, the activation functions are relu for all the layers and softmax function for the last layer as this is multiclass classification. Finally the optimizer used is sgd.

Multiclass classification:

The purpose of this classification model is to determine the solution of high class processing such as image processing with multiple classes. The given data belongs to either of the multiple classes. In this case three classes are obtained from the CIFAR10 data.

Model design:

Activation = relu

RELU function is used for activation. The purpose of this is to obtain the maximum value when the given input is greater than zero and to substitute zero when the given input is negative. RELU activation gives better performance than other activation functions such as sigmoid and tanh in the hidden layers.

Activation = softmax

The last layer uses softmax as the activation function. Softmax layers determine multiclass probabilities better and therefore are used here. It converts the output values obtained from the hidden layers to probabilities between zeros to one. This probability value is used to determine the class of the given image.

Optimizer = sgd

Stochastic Gradient Descent is the optimizer used. It selects a few samples called 'batches' from the dataset and reads them at every iteration randomly. By this way it is easy to tune the model more accurately. This optimizer is more suitable for huge datasets.

Implementation Details

Implemented using anaconda and Jupyter.

Implementation issues and processes includes:

1. Since the datasets were huge, it was necessary to increase the number of hidden layers to give better results. It was also necessary to increase the number of iterations through the training set to make the model better.
2. The task was implemented in Jupyter notebook. It was found that Jupyter stored the values of the models and at each hyper tuning the kernel was restarted.
3. The datasets were found to be huge and thus processing the data with so many epochs and hidden layers were difficult.
4. The process of running such an entire data on the local CPU was difficult.

Evaluation metrics

Initially the validation data obtained from the train data was ran across the model and finally the test data was ran across the model which was tuned with the test label.

Out of 15000 train data 12500 were considered as train set and 2500 were taken as validation data.

The accuracy obtained were as follows:

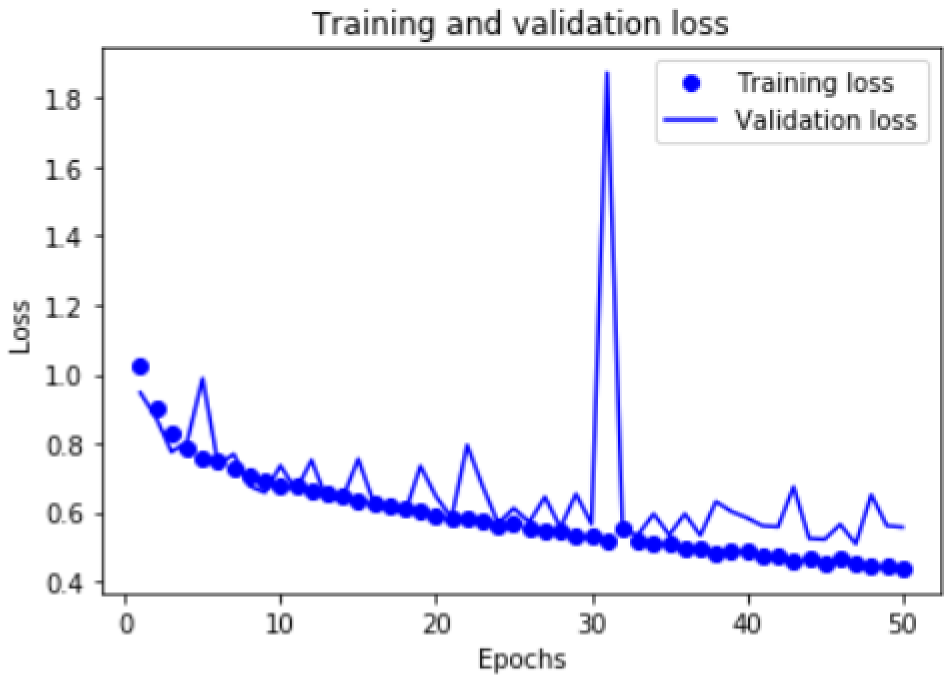
Validation set accuracy at each epoch:

```
Train on 12500 samples, validate on 2500 samples
Epoch 1/50
12500/12500 [==============================] - 2s 161us/step - loss: 1.0229 -
accuracy: 0.5050 - val_loss: 0.9465 - val_accuracy: 0.6324
Epoch 2/50
12500/12500 [==============================] - 2s 121us/step - loss: 0.8999 -
accuracy: 0.6242 - val_loss: 0.8752 - val_accuracy: 0.6076
Epoch 3/50
12500/12500 [==============================] - 1s 117us/step - loss: 0.8280 -
accuracy: 0.6532 - val_loss: 0.7749 - val_accuracy: 0.6732
Epoch 4/50
12500/12500 [==============================] - 1s 113us/step - loss: 0.7838 -
accuracy: 0.6732 - val_loss: 0.8042 - val_accuracy: 0.6456
Epoch 5/50
12500/12500 [==============================] - 1s 112us/step - loss: 0.7552 -
accuracy: 0.6874 - val_loss: 0.9881 - val_accuracy: 0.5276
Epoch 6/50
```

```
12500/12500 [==============================] - 1s 114us/step - loss: 0.7498 -
accuracy: 0.6899 - val_loss: 0.7370 - val_accuracy: 0.6968
```

…………………………………………………………

The training loss and the validation loss across the epochs were plotted as follows:



Observations from the graph:

The loss of both the classes training and validation reduce from the initial iteration thereby making the model learn by the time the iterations get over.

Since the two lines go at the same pace, the process of learning takes at a better pace.

However at certain places the training loss outcurves the validation loss meaning that the process pf overfitting takes place. This could be due to repeated learning of same data.

Test set accuracy at the epoch:

3000 test samples of the three classes were taken and the result obtained from the model were as follows:

```
3000/3000 [==============================] - 0s 93us/step
[0.5187073181470235, 0.7940000295639038]
```

Finally the accuracy of the test sample is obtained to be 80%. This accuracy is obtained after man y hyper tunings by

1. Changing the number of layers of the model
2. Changing the epochs
3. Changing the batch size
4. Choosing the correct optimizer.

References:

1. Keras code given by the professor.
2. https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/