Mayank Gupta
SR No.- 17112
M.Tech AI

Assignment-1 Report

For running the code on test data, run main.py file. Plots of loss and confusion matrix are provided in the images folder. Code used for training this network is provided in file train.ipynb. Training can also be done by running train_multi_layer.py and train_conv_layer.py files. Trained models are provided in the models folder.

Multi-Layered Perceptron

1) Modelled this problem as a classification problem with 10 categorical outputs and flattened the image for input.

2) Used neural network with 3 hidden layers with 500, 200, 60 neurons respectively. All layers use relu nonlinearity as activation. I have also used batchnorm layer after each fully connected layer.

3) For training the network I used Adam optimizer with a learning rate of 0.01. I am using a step-scheduler for decreasing the learning rate after a few iterations. Used weight decay of 0.001 to prevent overfitting and I am using cross-entropy as loss function.

4) I am using 20 epoch and batch size of 32, and I am shuffling the data in each epoch.

5) I am using accuracy on validation set for stopping criterion, and to set the hyperparameters of the network.

6) I am also normalizing the image to have 0.5 mean and 0.5 variance before passing it to the network.

Result:- I am getting 91.11 percent accuracy on the test set.

Analysis:

1) First I used a smaller neural network with two hidden layers, but the loss was not decreasing after a certain point. Then I tried to increase the size of the network but still accuracy(on validation set) was around 84 percent. After that I increased the size and used 4 hidden layers with a lot of nodes, but accuracy was not great and the model was overfitting, so I added dropout to the layers to decrease overfitting. After that I used batch-normalization, which decreased the accuracy, so I removed dropout and accuracy of model increased. Then I reduced the size of the network and accuracy did not change much. This gave me my final choice of network.

2) In this case the model was very sensitive to the hyperparameters and required few random initialization to get good results.

3) Shirt class is hardest to classify for this network. Pullover class is second hardest.

4) Some common points-

Learning rate matters a lot. So I tried a wide range to get faster convergence and good results.

Batch-normalization layers helped a lot. They also have regularizing effect.

Dropout and batch-normalization don't work well together.

Using relu as activation and Adam optimizer, convergence is faster than other methods.

Convolution-Neural Network:

Methods and hyperparameters used:

1) Modelled this problem as a classification problem with 10 categorical outputs.

2) Used neural network with 4 convolutional layers and 2 fully connected layers. I am also using batch-normalization and max pooling layers after convolutional layers. Used 5*5 filter for first convolution layers, and 3*3 for subsequent layers. Used 2*2 max-pooling layer with stride of 2 and 3*3 max-pooling layer with stride 3.

3) For training the network I used Adam optimizer with constant learning rate of 0.005. I am using a step-scheduler for decreasing the learning rate after a few iterations.

4) I am using 20 epoch and batch size of 32, and I am shuffling the data in each epoch.

5) I am using accuracy on validation set for stopping criterion, and to set the hyperparameters of the network.

6) I am also normalizing the image to have 0.5 mean and 0.5 variance before passing it to the network.

Result: I am getting accuracy of 92.8 on test set using this model.

Analysis:

Some findings when I was experimenting with hyperparameters:

1) First I used a smaller neural network with 3 convolutional layers and 2 hidden layers, but the loss was not decreasing after a certain point. Then I tried to increase the size of the network by adding another fully connected layer but still accuracy(on validation set) was around 85 percent. After that I increased the number of neurons in the fully-connected layers and increased the channels in convolutional layers, also used dropout in fully-connected layers. I got accuracy of around 87 percent using this. After that I used batch-normalization, removed dropout and one fully connected layer, and added another convolutional layer, this gave better performance than previous network. So my final neural network had 4 convolution layers and 2 fully connected layers

2) Had to do many random restarts to get good accuracy.

3) In this network also shirt class is hardest to classify for this network. Pullover class is second hardest.

4) Even though CNN has fewer parameters than MLP, it still takes longer for CNN to run 1 epoch on my laptop.

4) Some common points-

Learning rate matters a lot. So I tried a wide range to get faster convergence and good results.

Batch-normalization layers helped a lot. They also have regularizing effect.

Dropout and batch-normalization don't work well together.

Using relu as activation and Adam optimizer, convergence is faster than other methods.