CSE 575 Statistical Machine Learning

Project Part 3

Saianirud Reddy Nellore | snellore@asu.edu | ASU ID: 1219495602

The given dataset, SVHN, consists of images with digits taken from the house plates. It contains a training set of size 73257 and a testing set of size 26032. The resolution of each input image is $32 * 32$ and consists of 3 channels (RGB). The dataset belongs to 10 classes.

The data is given in the form of a dictionary. So, we loaded the data using SciPy and stored it in the following variables.

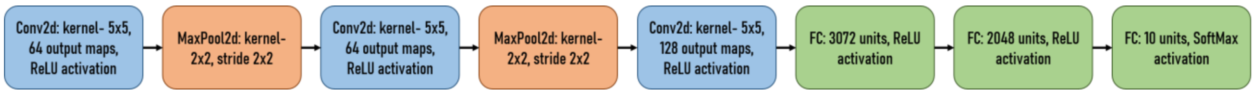
trX– samples in the training dataset

trY – labels of corresponding samples in the training dataset

tsX – samples in the testing dataset

tsY – labels of corresponding samples in the testing dataset

Once the data is loaded, we need to normalize the input data and get it in the range $0 - 1$ (divide by 255 which is the largest value), and encode the labels using 1-hot vector encoding. Then build the model using the CNN architecture given below.


Conv2d: kernel- 5x5, 64 output maps, ReLU activation
MaxPool2d: kernel- 2x2, stride 2x2
Conv2d: kernel- 5x5, 64 output maps, ReLU activation
MaxPool2d: kernel- 2x2, stride 2x2
Conv2d: kernel- 5x5, 128 output maps, ReLU activation
FC: 3072 units, ReLU activation
FC: 2048 units, ReLU activation
FC: 10 units, SoftMax activation

1. Convolutional layer with 64 output feature maps, ReLU activation, $5 * 5$ kernel size and stride $1 * 1$.
2. Max pooling with $2 * 2$ kernel size and stride $2 * 2$.
3. Convolutional layer with 64 output feature maps, ReLU activation, $5 * 5$ kernel size and stride $1 * 1$.
4. Max pooling with $2 * 2$ kernel size and stride $2 * 2$.
5. Convolutional layer with 128 output feature maps, ReLU activation, $5 * 5$ kernel size and stride $1 * 1$.
6. Flatten the output obtained by the last convolutional layer and then pass it to the fully connected layer.
7. Fully connected layer with 3072 nodes (output size), ReLU activation.
8. Fully connected layer with 2048 nodes, ReLU activation.
9. Fully connected layer with 10 nodes (corresponding to 10 classes), SoftMax activation.

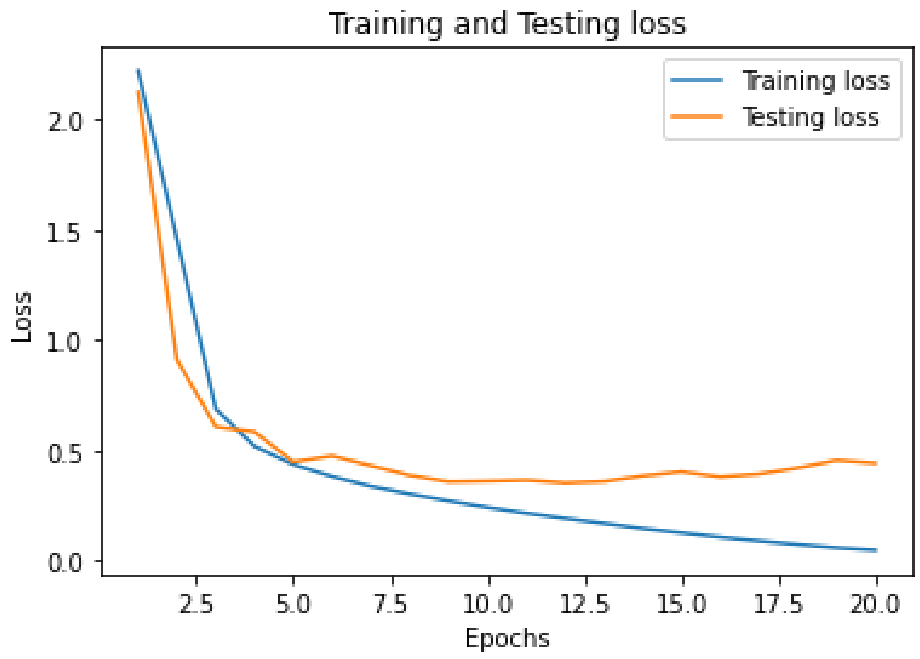
Once the model is built, train the CNN on training dataset using Stochastic Gradient Descent(SGD) optimizer and then test it on testing dataset. Categorical Cross Entropy is used as the loss function.

Taking $learning\ rate = 0.01, epochs = 20, batch\ size = 64$, the final classification accuracy is $\mathbf{90.5}\%$.

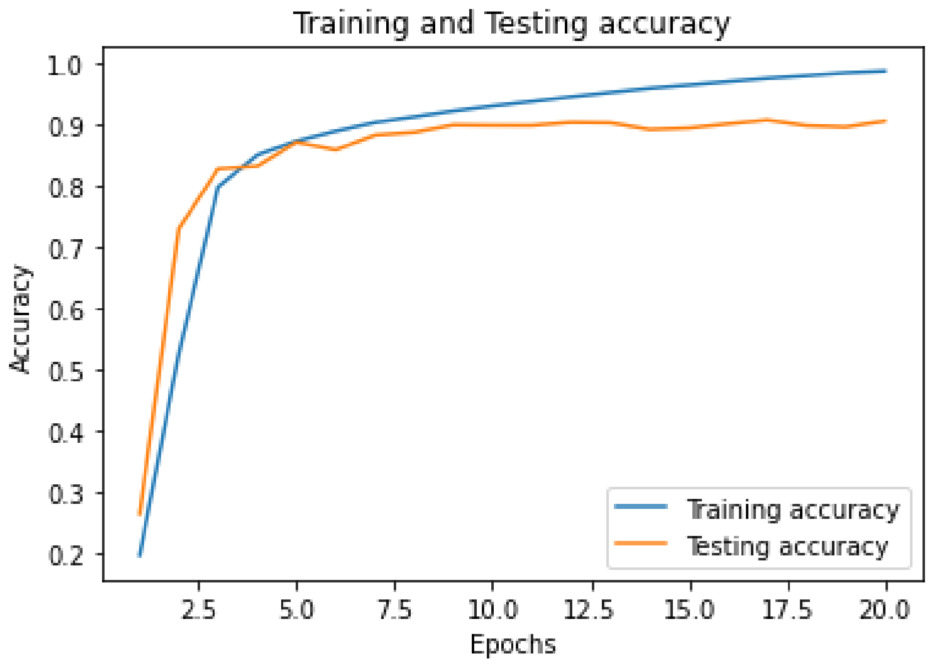
Training loss = 0.0446   Training accuracy = 98.9%

Testing loss = 0.4545   Testing Accuracy = 90.5%

Loss vs Epochs:



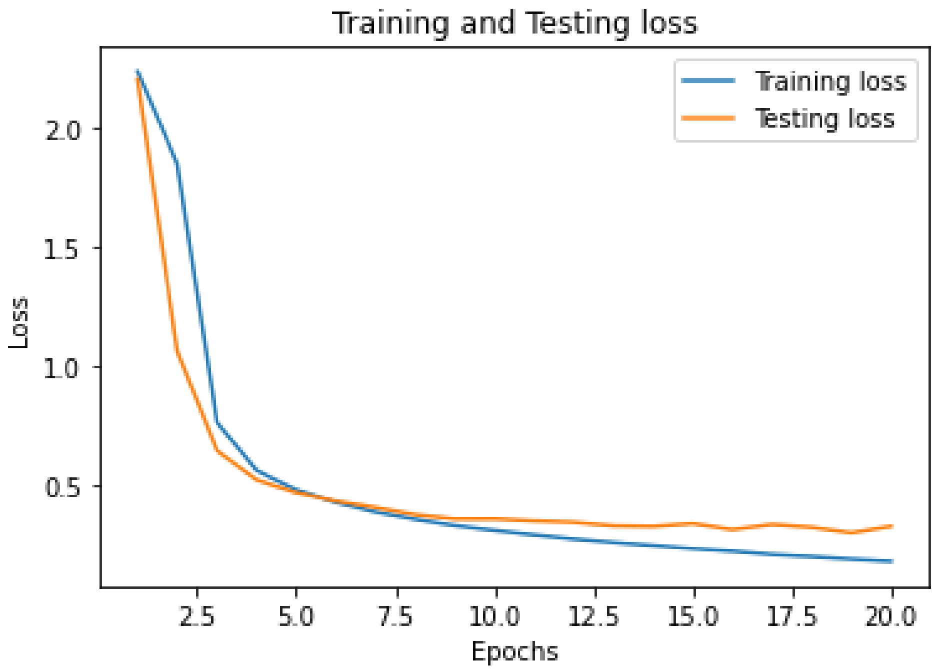
Accuracy vs Epochs:



Observations:

From the above graphs, we can see that the model overfits for the training dataset as the difference between the testing and training loss increases with increase in epochs. One way to alleviate overfitting problem is to add a drop out layer. The following is the loss graph obtained after adding a dropout layer with 30% drop rate using the same parameters mentioned above.

Training loss = 0.1714	Training accuracy = 95%
Testing loss = 0.3222	Testing Accuracy = 91.5%


Training and Testing loss
Training loss
Testing loss
Loss
Epochs