Report :: TIPR Assignment - III

Put your name here

Put the date here

1 Configuration

- Python code is written for Python 3. It may not work with Python 2.
- The code should be executed from the src folder, otherwise relative paths may not work as expected.
- Pass the configuration as a string list. That is, argument should be –filter-config '[8 8 4 4]' (**with** quotes).

2 Part 1:- Fashion MNIST

2.1 Task 1:- Test Model with different number of layers

Tested model with following architectures:

- conv-5-5-32-conv-3-3-64-FCN-500-10
- conv-5-5-32-conv-5-5-32-conv-3-3-64-FCN-384-192-10
- conv-5-5-64-conv-3-3-64-FCN-384-192-10

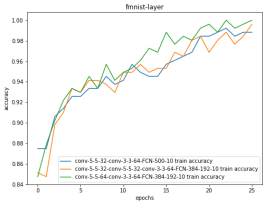

Figure 1: Accuracy

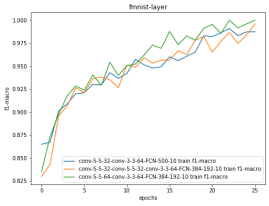

Figure 2: F1-Micro

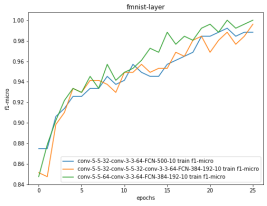

Figure 3: F1-Macro

Each plot shows the metric on Y-axis and epochs on X-axis. Different lines in plot corresponds to different models.

2.2 Task 2:- Test Model with different number of neurons and layers

Tested model with following configurations:

- conv-5-5-64-conv-3-3-64-FCN-512-384-10
- conv-7-7-32-conv-5-5-64-FCN-384-192-10
- conv-7-7-64-conv-3-3-64-FCN-512-192-10

Finally fixed and fine-tuned the 784-50-30-10 model as it took less time for training and had comparable performance.

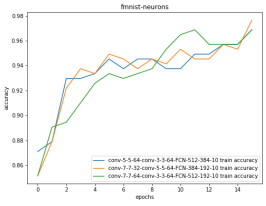

Figure 4: Accuracy

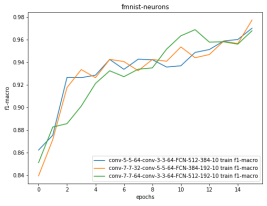

Figure 5: F1-Micro

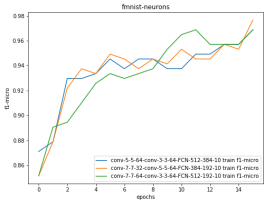

Figure 6: F1-Macro

Each plot shows the metric on Y-axis and epochs on X-axis. Different lines in plot corresponds to different models.

2.3 Task 3:- Test Model with different activation functions

Tested model with relu, tanh, sigmoid and swish activations functions. The performance of relu and swish were almost same and decided to use swish in the final model.

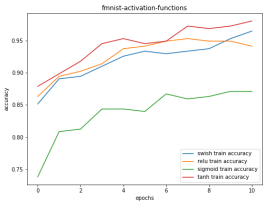

Figure 7: Accuracy

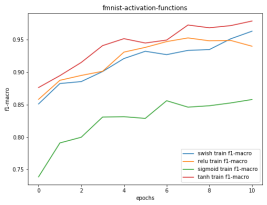

Figure 8: F1-Micro

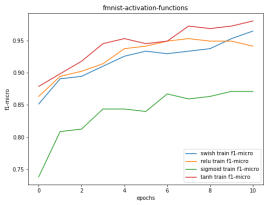

Figure 9: F1-Macro

Each plot shows the metric on Y-axis and epochs on X-axis. Different lines in plot corresponds to different models.

2.4 Task 4

After observing above results and trying out various initialization techniques following architecture was finalized:

1. Conv1: filter=7x7x1x64, activation=swish
2. MaxPool1: kernel=2x2, strides=3x3
3. Conv2: filter=3x3x64x64, activation=swish
4. MaxPool2: kernel=2x2, strides=3x3
5. FCN1: output=512, activation=swish
6. FCN2: output=192, activation=swish
7. Final output using softmax.

2.5 Task 5: Semi-supervised learning

In this task model (selected in Task 4) was trained on 10-50% of the data and embedding for rest of the images were extracted from the final layer just before the output layer of the network. The images were then clustered according to these embedding into 15 clusters and each cluster was assigned a class label depending on the highest number of images into that cluster belonging to same class.

The final accuracy achieved on validation set after taking 10-50% of training data in increments of 10% is as below:

Training instances	Accuracy
10%	88.5%
20%	87.2%
30%	89.4%
40%	88.9%
50%	87.3%

2.6 Task 6:- Cluster Projection

The graphs obtained from above configuration of training instances is as below. These graphs were generated by projecting 1000 randomly selected instances into 2D using t-SNE projection method.

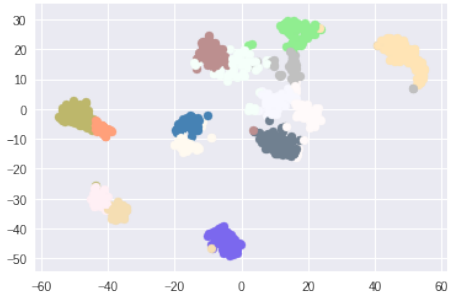

Figure 10: 10% training data clusters

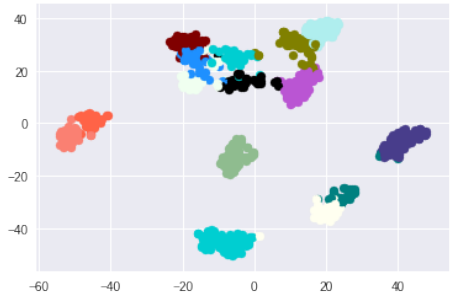

Figure 11: 20% training data clusters

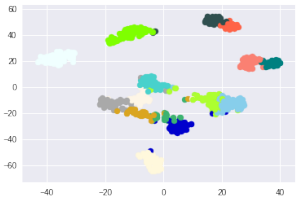

Figure 12: 30% training data clusters

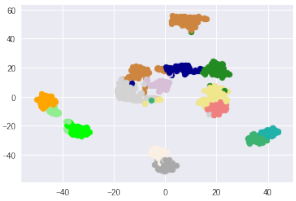

Figure 13: 40% training data clusters

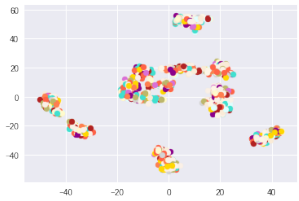

Figure 14: 50% training data clusters

The corresponding class distribution of points is as below:

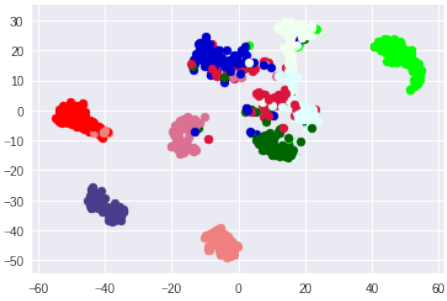

Figure 15: 10% training data classes

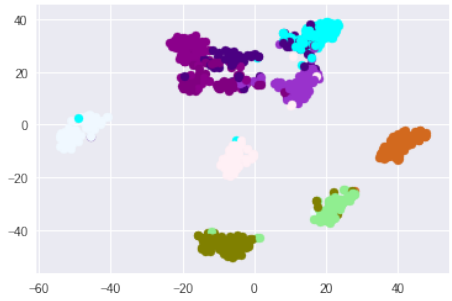

Figure 16: 20% training data classes

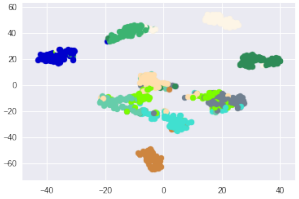

Figure 17: 30% training data classes

Figure 18: 40% training data classes

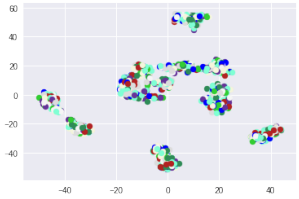

Figure 19: 50% training data classes

2.7 Task 7: MLP vs CNN

Trained model on a simple MLP consisting of 512 and 192 hidden layer nodes in two layers. The plots of various evaluation metrics are as follows:

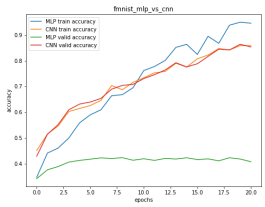

Figure 20: Accuracy

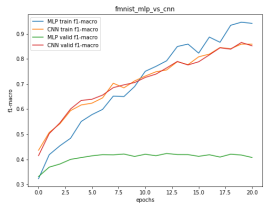

Figure 21: F1-Micro

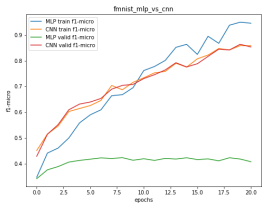

Figure 22: F1-Macro

3 Part 2:- CIFAR

3.1 Task 1:- Test Model with different number of layers

Tested model with following architectures:

- conv-3-3-64-fcn-100-10
- conv-5-5-32-conv-3-3-64-fcn-100-10
- conv-5-5-64-fcn-384-192-10

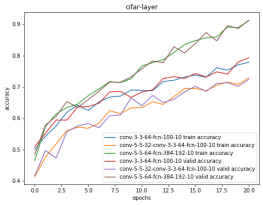

Figure 23: Accuracy

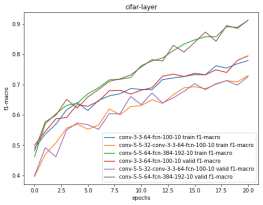

Figure 24: F1-Micro

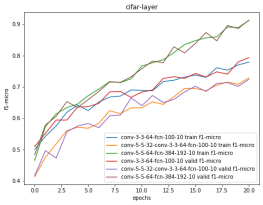

Figure 25: F1-Macro

Each plot shows the metric on Y-axis and epochs on X-axis. Different lines in plot corresponds to different models.

3.2 Task 2:- Test Model with different number of neurons and layers

Tested model with following configurations:

- conv-5-5-32-conv-3-3-64-fcn-300-100-10
- conv-5-5-32-conv-5-5-32-fcn-200-50-10
- conv-5-5-64-conv-3-3-64-fcn-100-50-10

Finally fixed and fine-tuned the 784-50-30-10 model as it took less time for training and had comparable performance.

Each plot shows the metric on Y-axis and epochs on X-axis. Different lines in plot corresponds to different models.

3.3 Task 3:- Test Model with different activation functions

Tested model with relu, tanh, sigmoid and swish activations functions. The performance of relu and swish were almost same and decided to use swish in the final model.

Each plot shows the metric on Y-axis and epochs on X-axis. Different lines in plot corresponds to different models.

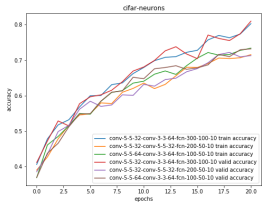

Figure 26: Accuracy

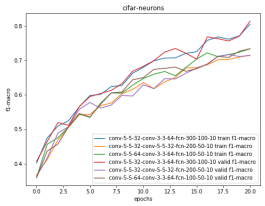

Figure 27: F1-Micro

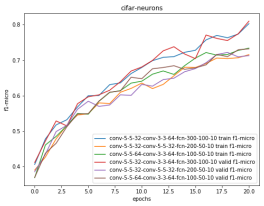

Figure 28: F1-Macro

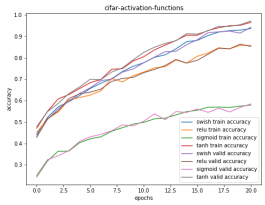

Figure 29: Accuracy

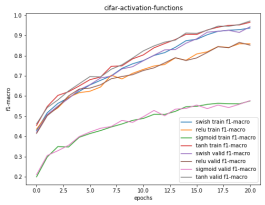

Figure 30: F1-Micro

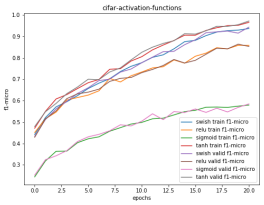

Figure 31: F1-Macro

3.4 Task 4

After observing above results and trying out various initialization techniques following architecture was finalized:

1. Conv1: filter=5x5x3x64, activation=relu
2. MaxPool1: kernel=2x2, strides=3x3
3. Conv2: filter=3x3x64x64, activation=relu
4. MaxPool2: kernel=2x2, strides=3x3
5. FCN1: output=384, activation=relu
6. FCN2: output=192, activation=relu
7. Final output using softmax.

3.5 Task 5: Semi-supervised learning

In this task model (selected in Task 4) was trained on 10-50% of the data and embedding for rest of the images were extracted from the final layer just before the output layer of the network. The images were then clustered according to these embedding into 15 clusters and each cluster was assigned a class label depending on the highest number of images into that cluster belonging to same class.

Training instances	Accuracy
10%	39.7%
20%	42.3%
30%	48.5%
40%	49.7%
50%	58.4%

The final accuracy achieved on validation set after taking 10-50% of training data in increments of 10% is as below:

3.6 Task 6:- Cluster Projection

The graphs obtained from above configuration of training instances is as below. These graphs were generated by projecting 1000 randomly selected instances into 2D using t-SNE projection method.

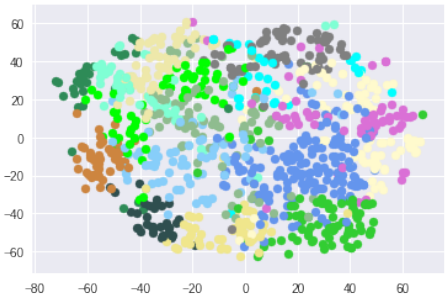

Figure 32: 10% training data clusters

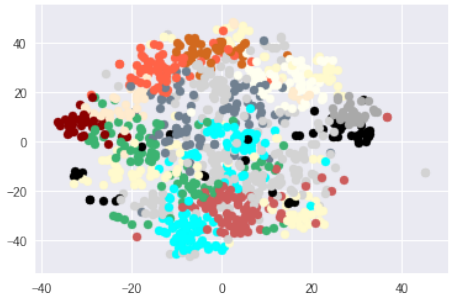

Figure 33: 20% training data clusters

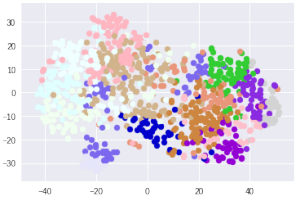

Figure 34: 30% training data clusters

Figure 35: 40% training data clusters

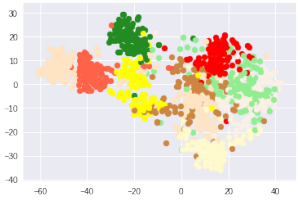

Figure 36: 50% training data clusters

The corresponding class distribution of points is as below:

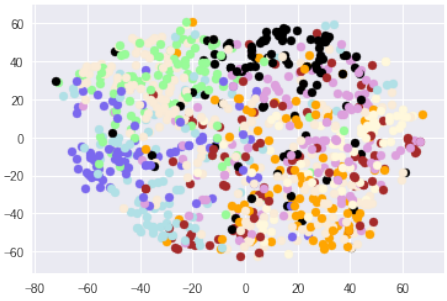

Figure 37: 10% training data classes

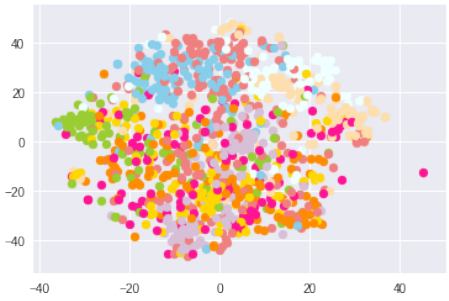

Figure 38: 20% training data classes

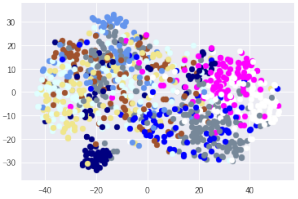

Figure 39: 30% training data classes

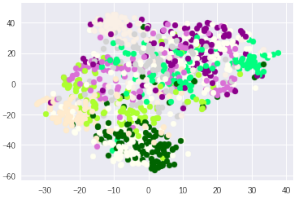

Figure 40: 40% training data classes

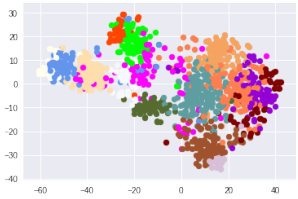

Figure 41: 50% training data classes

3.7 Task 7: MLP vs CNN

Trained model on a simple MLP consisting of 512 and 192 hidden layer nodes in two layers. The plots of various evaluation metrics are as follows:

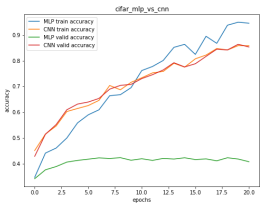

Figure 42: Accuracy

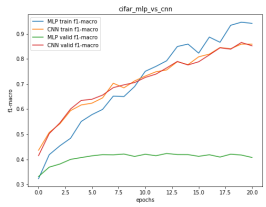

Figure 43: F1-Micro

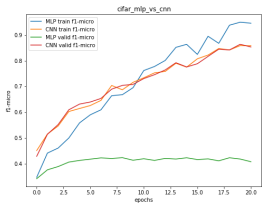

Figure 44: F1-Macro