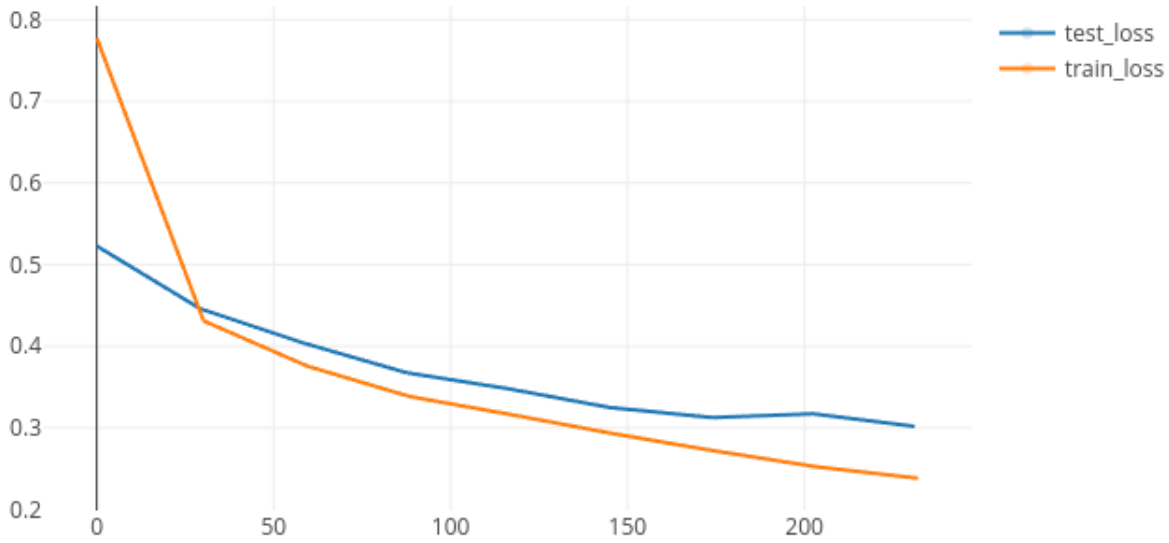
SysDL Assignment #3

Ajay S Joshi(CS15B047)
CS6886 SysDL
IITM

May 17, 2020

1 Basic model on Fashion-MNIST



Final Accuracies:

- Train: 91.14
- Test: 89.32

2 Hyperparameter exploration

Hyperparam values:

1. Filter sizes for conv layers: 3 x 3, 5 x 5

2. Channel sizes for conv layers: 8, 12, 16
3. FC layer size: 50, 100
4. Batch size: 64, 128, 256, 512
5. Learning rate: 0.1, 0.01, 0.001
6. Momentum: 0.9, 0.99

2.1 Top configurations

Parameters

batch_size	128	256	256
conv1_channels	12	16	12
conv1_kernel	3	5	5
conv2_channels	16	16	12
conv2_kernel	5	3	3
fc1_size	100	50	100
lr	0.01	0.01	0.01
max_epochs	50	40	40
momentum	0.9	0.9	0.9
weight_init	xavier_normal	xavier_uniform	xavier_uniform
workers	8	8	8

Metrics

test_accuracy	90.1	90.01	89.97
test_loss	0.291	0.294	0.286
train_accuracy	92.92	92.46	92.47

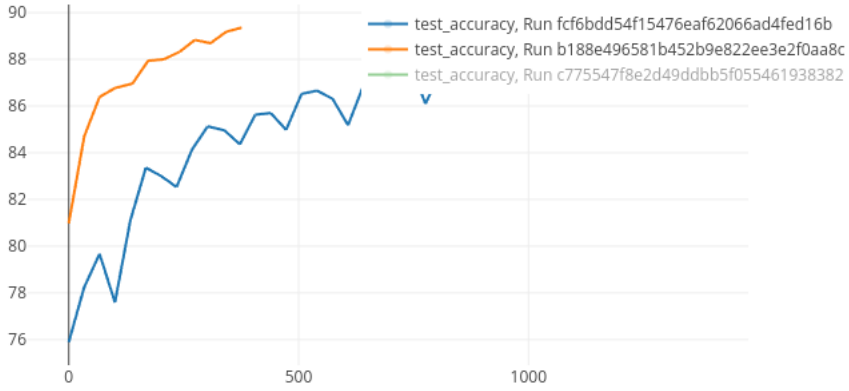
2.2 Performance of hyperparameters

- Models of medium sizes were the best performers, as smaller models underfit and larger ones overfit the data
- Xavier initilaization performed well over a initialization with a plain normal distribution, due to the theoretical grounding of the Xavier initialization
- Higher batch size and relatively high learning rate of 0.01 worked well as well as could be trained faster, as the stability and number of updates is well-balanced for this combination of batch size and learning rate.

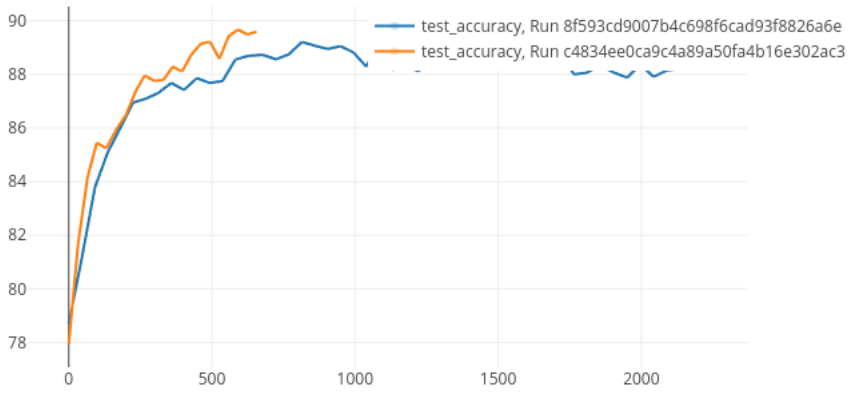
2.3 Accuracy v/s Epochs plots

- High learning rate: Comparison between learning rates of 0.001 and 0.1 reveals that a high learning rate leads to unstable updates and thus is not able to learn at all.

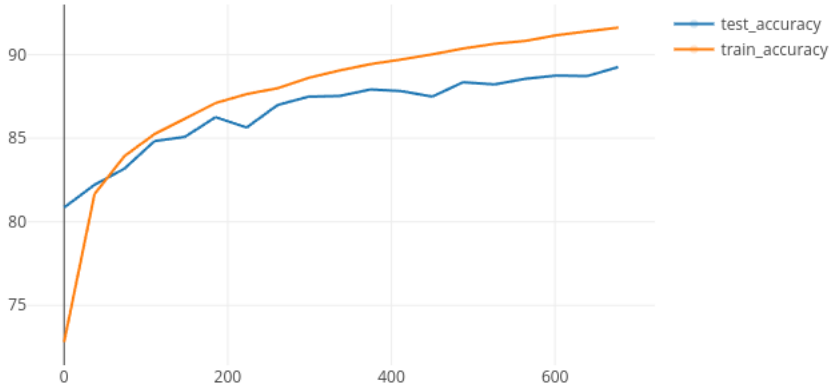
- Small learning rate: Comparison between learning rates of 0.01 and 0.001 reveals that a smaller learning can achieve similar accuracy, but takes longer training time than a higher learning rate of 0.01



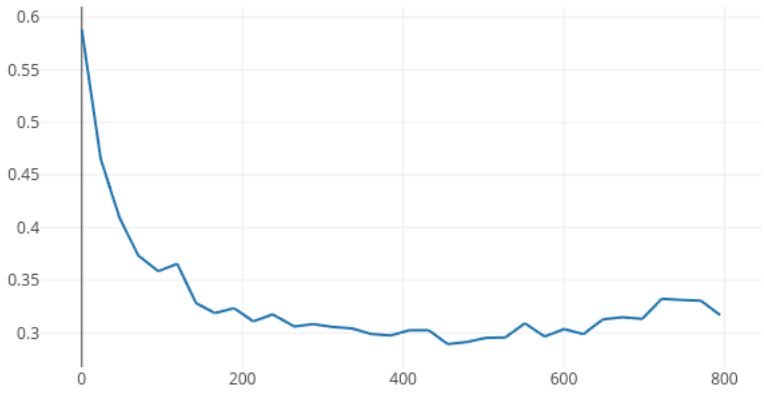
- High Momentum: Comparison between high momentum of 0.99 and a relatively lower momentum of 0.9 shows that there is no significant difference in performance, but take longer training time, possibly due to excessive momentum



- Low batch size:



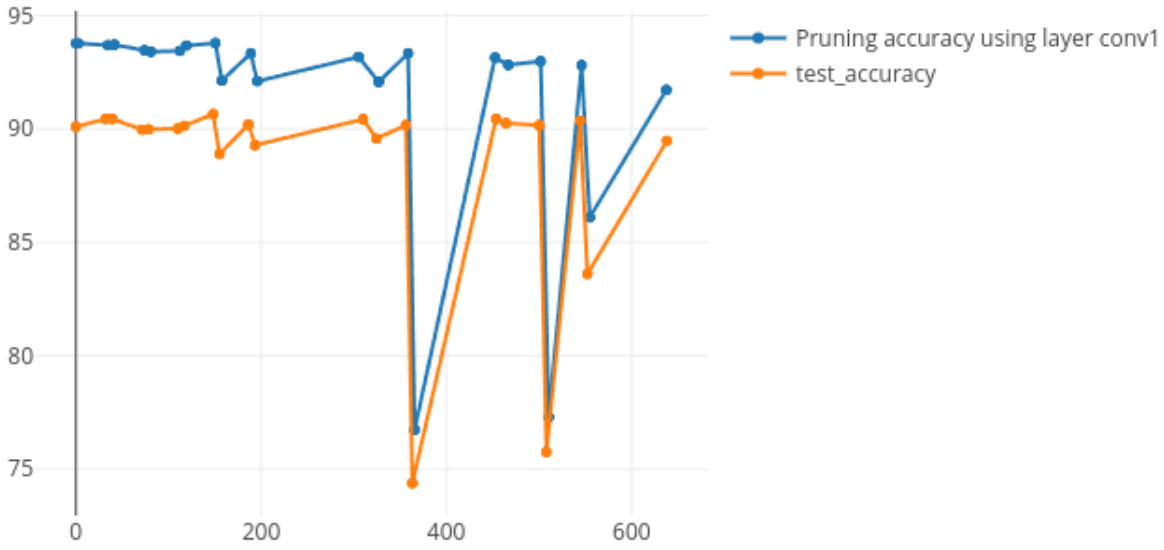
- Overfitting: High capacity model starts overfitting after some epochs of training, which is evident from the loss v/s epochs plot for the model

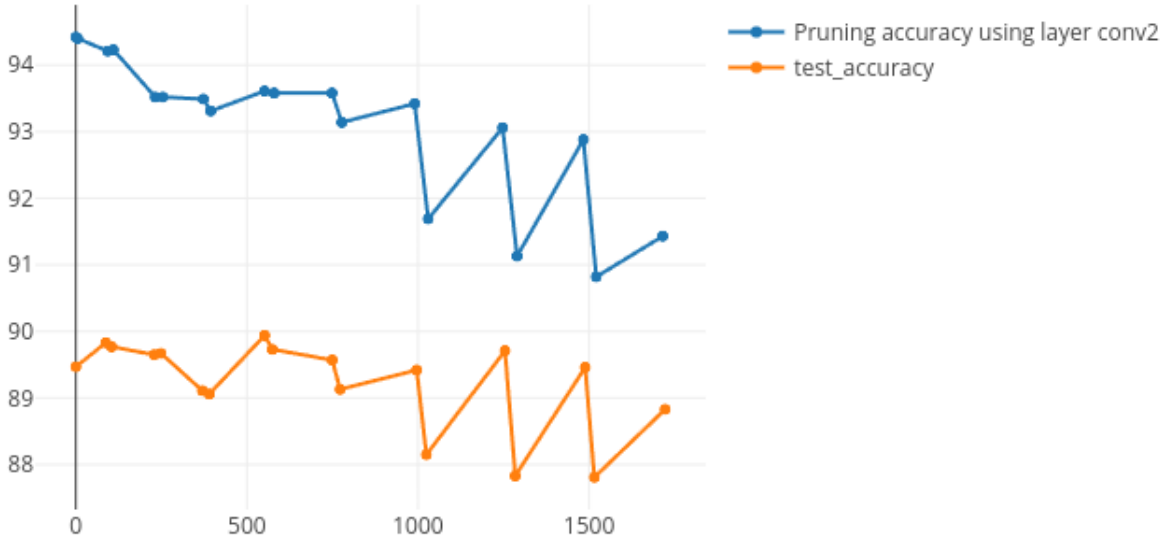


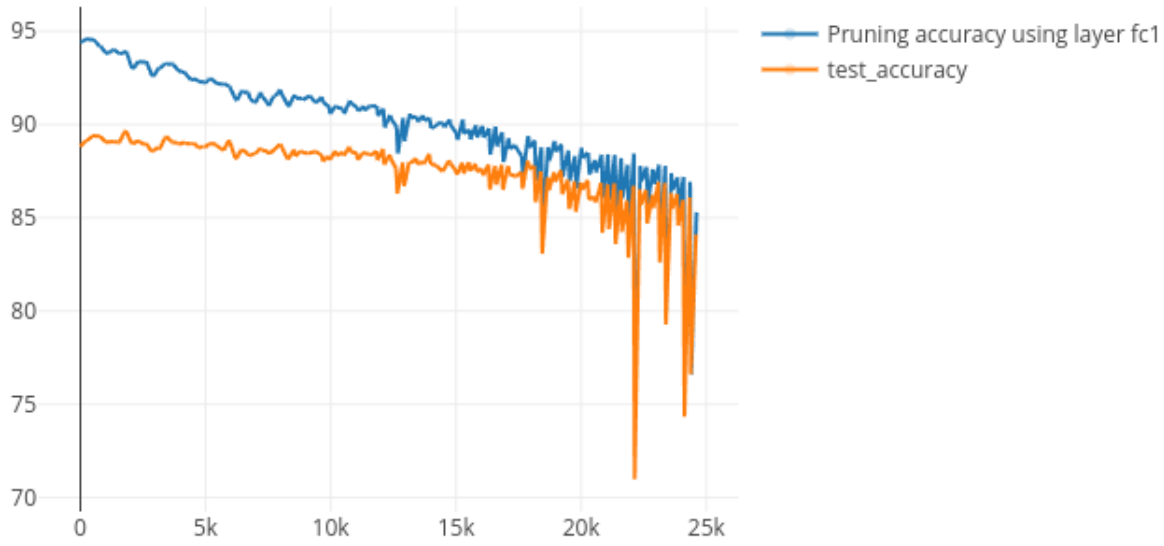
3 Pruning

- The heuristic used for pruning a channel is the **norm of the weight matrix** of the channel.
- 3 layers are chosen for pruning here: conv1, conv2 and fc1 layers. In the convolution layers, 1 channel is dropped in a step and for the fully connected layers, 1 neuron is dropped per pruning step.
- The output layer is not pruned as it leads to severe drop in accuracy.

The variation in validation and test accuracy in the process of successively pruning layers conv1, conv2 and fc respectively is shown below:


95
90
85
80
75
0
200
400
600
Pruning accuracy using layer conv1
test_accuracy


94
93
92
91
90
89
88
0
500
1000
1500
Pruning accuracy using layer conv2
test_accuracy



4 Accelerating Inference

The best performing model with the following configuration was chosen for pruning:

- Conv1 channels (c1): 12
- Conv1 filter size: 3 x 3
- Conv2 channels (c2): 16
- Conv1 filter size: 5 x 5
- Neurons in the hidden FC layer (f1): 100
- Neurons in the final layer: 10

During the course of pruning, except c1, c2 and f1, all other things remain constant. Thus a model can be represented as a tuple (c1, c2, f1).

- Analytical formula for the number of computations as a function of c1, c2, f1 is the addition of the following 4 quantities:

    1. Computations in conv1 layer: (26 x 26) x (1 x 3 x 3) x c1 + 2 x [(c1 x 26 x 26)] (∵ Conv computations + (Bias + ReLU))

2. Computations in conv2 layer: (22 x 22) x (c1 x 5 x 5) x c2 + 2 x [(c2 x 22 x 22)]
3. Computations in fc1 layer: f1 x (22 x 22 x c2) + 2 x f1
4. Computations in fc1 layer: f1 x 10

- The pruned model has the configuration: **(c1, c2, f1) = (1, 7, 72)** and gets an accuracy of **87.62 %** on the test dataset.
- The pruned produces a speedup of around **11x**, with the original model taking around **87-88 sec** for inference on the training data and the pruned model taking around **7-8 sec**.

5 Submission Format

1. Code is not in the form of notebooks as it has been executed using command line on Colab.
2. Installing mlflow on Colab allows the code to execute properly.
3. Instructions regarding running the code have been given in a README in the submission.