Deep Learning Lab 3 Report

Shusrith S

PES1UG22AM155

February 2, 2025

1 Introduction

The MNIST (Modified National Institute of Standards and Technology) dataset is a widely used collection of 70,000 grayscale images of handwritten digits, each of size 28x28 pixels, representing the digits 0 through 9. It is divided into a training set of 60,000 images and a test set of 10,000 images, making it a standard benchmark for evaluating machine learning models, particularly for image classification tasks.

2 Architecture and Framework used

A feed-forward neural network is used to classify these images and identify the digit represented in the image. A tanh activation function is used in the model with the Adam optimizer. The TensorFlow library is used for its extensive support with various initialization techniques built-in with the layers.

The dataset is loaded from TensorFlow and used directly with TensorFlow's MODEL.FIT() method. A modular approach is used throughout, utilizing a single class and modifying the members of the class for each regularization and initialization technique.

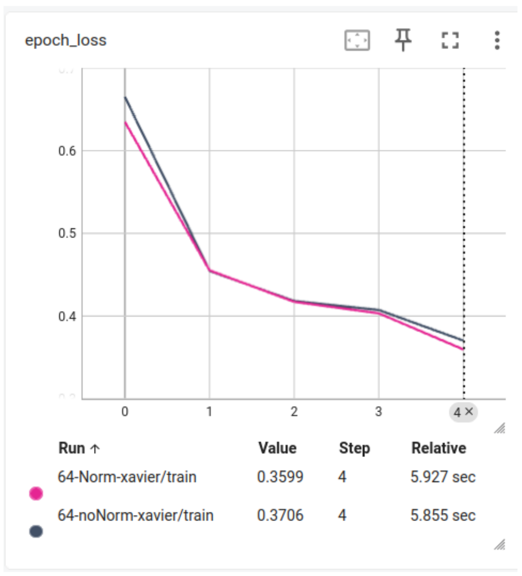
Since comparison plots are needed in this report, TensorBoard has been used extensively. It keeps track of training and validation loss during the training process and creates interactive plots and graphs. It also allows comparison between various runs, a key objective of this report.

All training was done on an Nvidia RTX 4050 laptop GPU. The learning rate used is 0.001 and the loss being used is **sparse categorical cross entropy**, because of the softmax activation applied to the final layer, considering only the positive class for loss.

It must be noted that since all the models are trained for only 5 epochs, the difference in train and val loss is very low, almost equal in some cases. The low number of training epochs means that the model hasn't trained enough for the batch normalization to show its effectiveness and as a result, the outputs seen are not fully indicative of how useful batch normalization can be. When the number of epochs are so low, it could also result in a few batches having outlier data points resulting in unexpected spikes and random variation in the loss curve. To see the full effect of the techniques used, the model should be trained for longer.

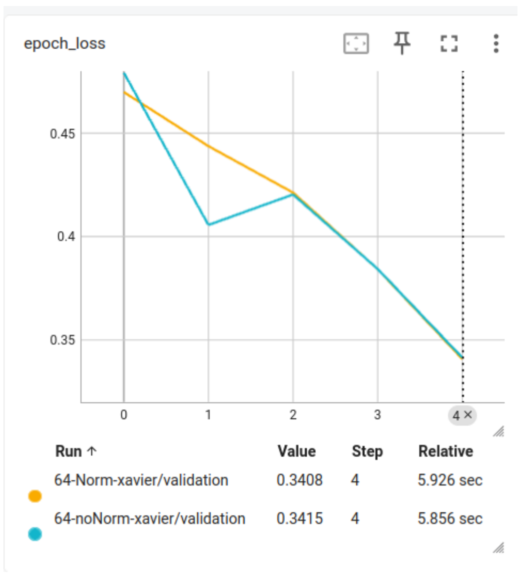
3 Xavier Weight Initialization

3.1 Batch Size 64



	Run ↑	Value	Step	Relative
●	64-Norm-xavier/train	0.3599	4	5.927 sec
●	64-noNorm-xavier/train	0.3706	4	5.855 sec

Figure 1: Batch size 64 train loss



	Run ↑	Value	Step	Relative
●	64-Norm-xavier/validation	0.3408	4	5.926 sec
●	64-noNorm-xavier/validation	0.3415	4	5.856 sec

Figure 2: Batch size 64 val loss

Loss	Loss Value
Without batch norm	0.3706
With batch norm	0.3599

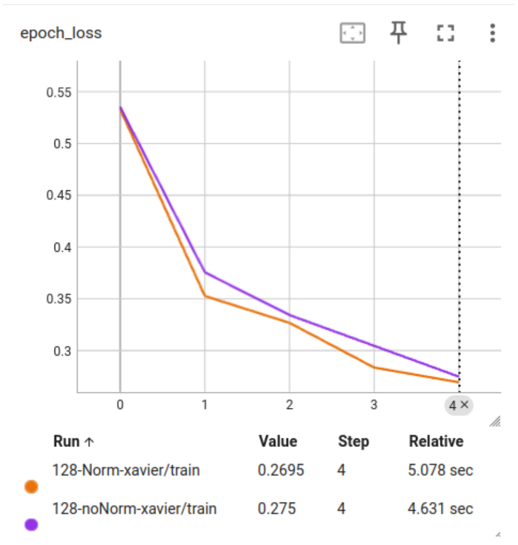
Table 1: Train Loss

Loss	Loss Value
Without batch norm	0.3415
With batch norm	0.3409

Table 2: Val loss

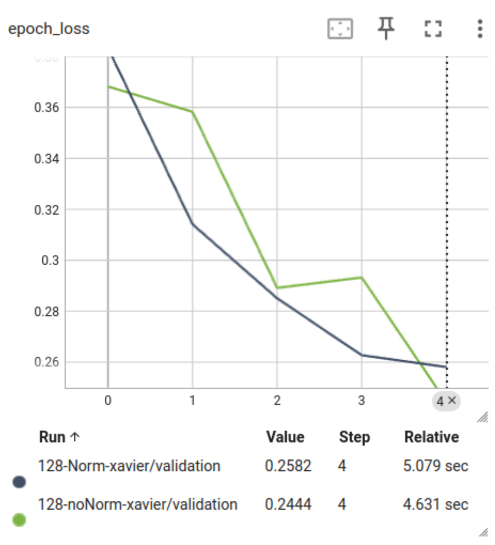
- A batch size of 64 was used, trained once with batch norm and once without batch norm
- We can see that in both cases, the loss is lower when batch norm is used.
- The difference with and without batch norm is low and one of the possible reasons could be the low batch size. When the batch size is so low, the statistics used to normalize may not be accurate enough and hence the batch norm does not prove to be too effective. The low number of epochs could also be the reason we don't get to see the influence of the batch norm yet.

3.2 Batch Size 128



	Run ↑	Value	Step	Relative
●	128-Norm-xavier/train	0.2695	4	5.078 sec
●	128-noNorm-xavier/train	0.275	4	4.631 sec

Figure 3: Batch size 128 train loss



	Run ↑	Value	Step	Relative
●	128-Norm-xavier/validation	0.2582	4	5.079 sec
●	128-noNorm-xavier/validation	0.2444	4	4.631 sec

Figure 4: Batch size 128 val loss

Loss	Loss Value
Without batch norm	0.2750
With batch norm	0.2695

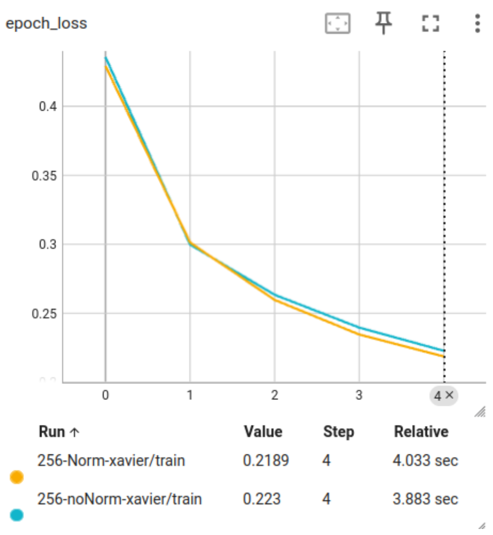
Table 3: Train Loss

Loss	Loss Value
Without batch norm	0.2444
With batch norm	0.2582

Table 4: Val loss

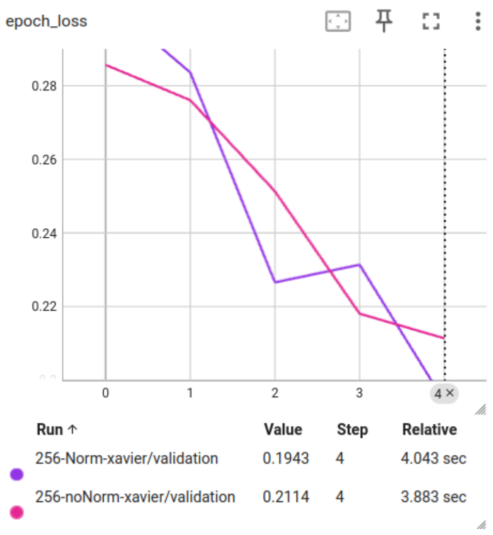
- A batch size of 128 is used, and trained once with batch norm and once without it.
- In the train loss curve, we can see that the loss is lower when batch norm is used. In the val loss curve, even though the loss is slightly higher, it could be as a result of larger variations that occurs in the first few epochs, which settle down later as the model trains for longer.
- Another possible reason is that Xavier normal initialization already draws initial weights from a normal distribution, hence, it takes longer to show improvement in comparison to when batch norm is not used.

3.3 Batch Size 256



Run ↑	Value	Step	Relative
● 256-Norm-xavier/train	0.2189	4	4.033 sec
● 256-noNorm-xavier/train	0.223	4	3.883 sec

Figure 5: Batch size 256 train loss



Run ↑	Value	Step	Relative
● 256-Norm-xavier/validation	0.1943	4	4.043 sec
● 256-noNorm-xavier/validation	0.2114	4	3.883 sec

Figure 6: Batch size 256 val loss

Loss	Loss Value
Without batch norm	0.2230
With batch norm	0.2189

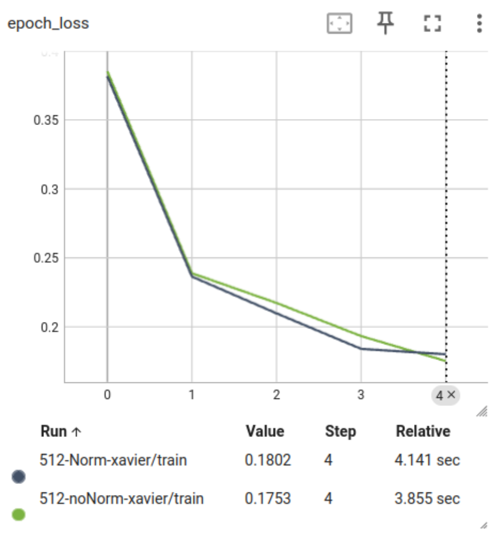
Table 5: Train Loss

Loss	Loss Value
Without batch norm	0.2114
With batch norm	0.1843

Table 6: Val loss

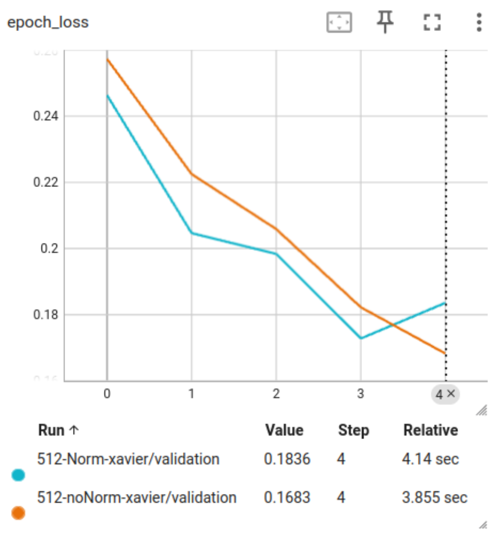
- Batch size of 256 is used, trained once with batch norm and once without batch norm,
- In this case we can clearly see the model trained with batch norm starts to have lower losses in comparison to when batch norm is not used.
- Both train and val loss curves show lower losses when batch norm is used,

3.4 Batch Size 512



	Run ↑	Value	Step	Relative
●	512-Norm-xavier/train	0.1802	4	4.141 sec
●	512-noNorm-xavier/train	0.1753	4	3.855 sec

Figure 7: Batch size 512 train loss



	Run ↑	Value	Step	Relative
●	512-Norm-xavier/validation	0.1836	4	4.14 sec
●	512-noNorm-xavier/validation	0.1683	4	3.855 sec

Figure 8: Batch size 512 val loss

Loss	Loss Value
Without batch norm	0.1753
With batch norm	0.1802

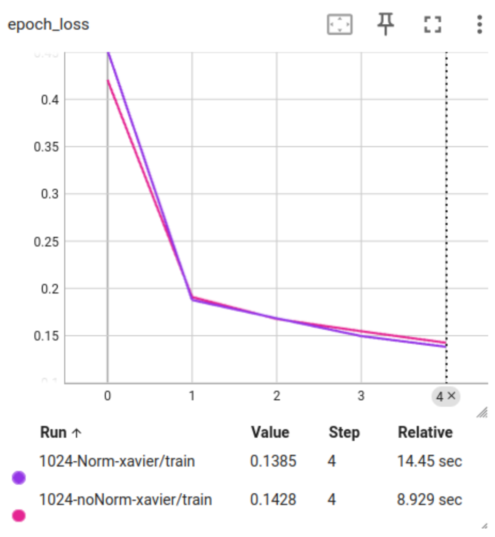
Table 7: Train Loss

Loss	Loss Value
Without batch norm	0.1683
With batch norm	0.1836

Table 8: Val loss

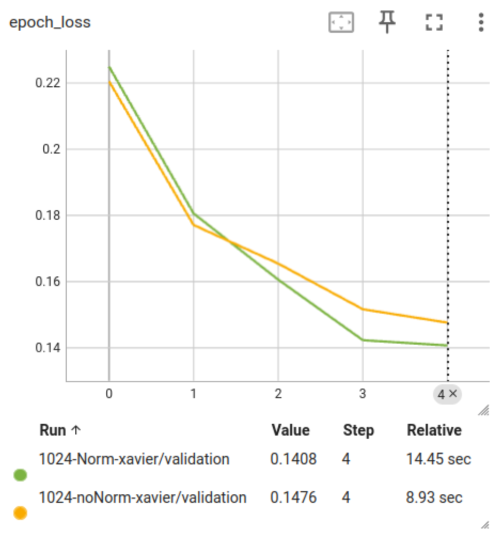
- Batch size of 512 was used and the model was trained once with batch norm and once without batch norm.
- In both cases we can see from the curve that when batch norm was used, the model tended to perform better for the first 4 epochs and has a slight increase in the last epoch.
- This could be because of some random outlier in the training samples that has not been encountered yet in the other batch size experiments so far.

3.5 Batch Size 1024



	Run ↑	Value	Step	Relative
●	1024-Norm-xavier/train	0.1385	4	14.45 sec
●	1024-noNorm-xavier/train	0.1428	4	8.929 sec

Figure 9: Batch size 1024 train loss



	Run ↑	Value	Step	Relative
●	1024-Norm-xavier/validation	0.1408	4	14.45 sec
●	1024-noNorm-xavier/validation	0.1476	4	8.93 sec

Figure 10: Batch size 1024 val loss

Loss	Loss Value
Without batch norm	0.1428
With batch norm	0.1385

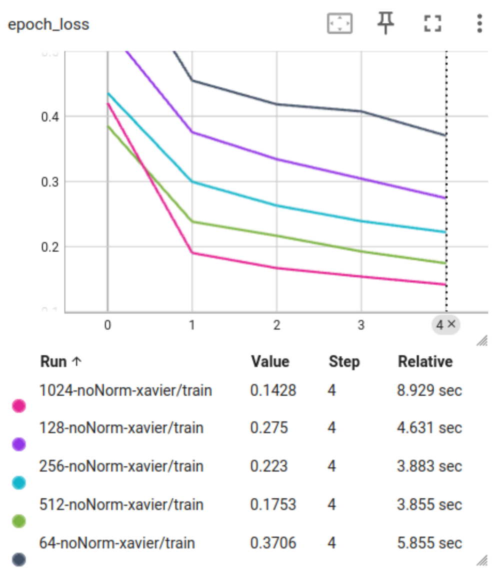
Table 9: Train Loss

Loss	Loss Value
Without batch norm	0.1476
With batch norm	0.1498

Table 10: Val loss

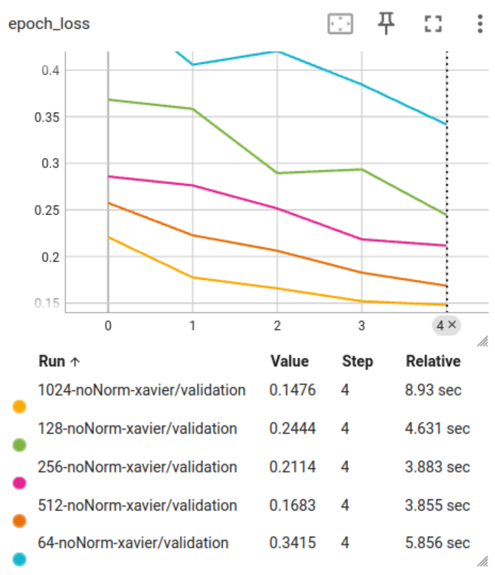
- Batch size of 1024 is used, the model is trained once with batch normalization and once without.
- A large batch size of 1024 yields the most stable results, with lower variation and spikes.
- It can be seen that utilizing batch norm uniformly reduces loss in both training and validation data.

3.6 Batch Size Comparison



Run ↑	Value	Step	Relative
● 1024-noNorm-xavier/train	0.1428	4	8.929 sec
● 128-noNorm-xavier/train	0.275	4	4.631 sec
● 256-noNorm-xavier/train	0.223	4	3.883 sec
● 512-noNorm-xavier/train	0.1753	4	3.855 sec
● 64-noNorm-xavier/train	0.3706	4	5.855 sec

Figure 11: Batch size 64 train loss



Run ↑	Value	Step	Relative
● 1024-noNorm-xavier/validation	0.1476	4	8.93 sec
● 128-noNorm-xavier/validation	0.2444	4	4.631 sec
● 256-noNorm-xavier/validation	0.2114	4	3.883 sec
● 512-noNorm-xavier/validation	0.1683	4	3.855 sec
● 64-noNorm-xavier/validation	0.3415	4	5.856 sec

Figure 12: Batch size 64 val loss

Batch size	Loss Value
64	0.3706
128	0.2750
256	0.2230
512	0.1753
1024	0.1428

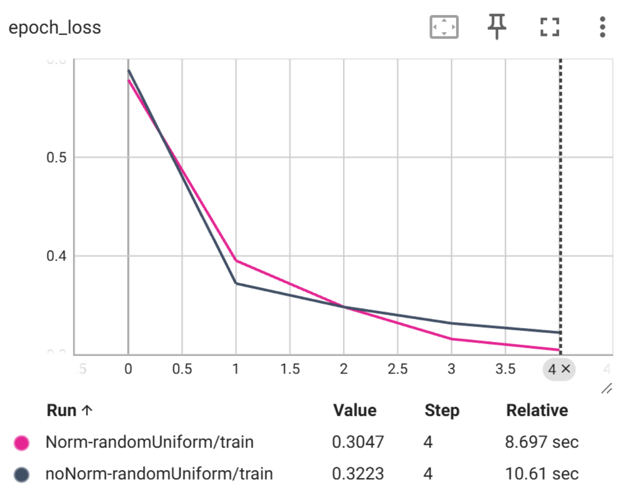
Table 11: Train Loss

Batch size	Loss Value
64	0.3415
128	0.2444
256	0.2114
512	0.1683
1024	0.1476

Table 12: Val loss

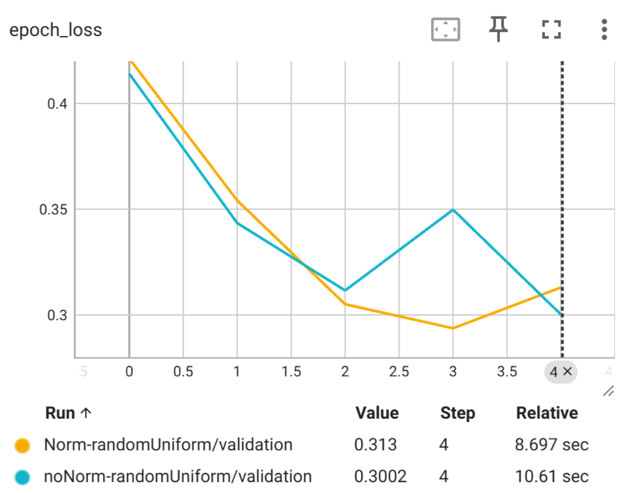
- Batch sizes of 64, 128, 256, 512, 1024 are used with Xavier normal weight initialization.
- The graphs shows that loss increases uniformly with increase in batch size, the curves show fewer spikes and are more stable in general.
- This is because as batch size increases, we get more stable gradients which are less noisy and less susceptible to the noise in the samples. Larger batch sizes yield gradients with lower variance which helps the model move in the right direction and converge faster. It also utilizes parallelization technology better leading to faster overall training.

4 Uniform initialization



Run ↑	Value	Step	Relative
● Norm-randomUniform/train	0.3047	4	8.697 sec
● noNorm-randomUniform/train	0.3223	4	10.61 sec

Figure 13: Batch size 128 train loss



Run ↑	Value	Step	Relative
● Norm-randomUniform/validation	0.313	4	8.697 sec
● noNorm-randomUniform/validation	0.3002	4	10.61 sec

Figure 14: Batch size 128 val loss

Loss	Loss Value
Without batch norm	0.3223
With batch norm	0.3047

Table 13: Train Loss

Loss	Loss Value
Without batch norm	0.3002
With batch norm	0.3130

Table 14: Val loss

- Uniform weight initialization is used, with a batch size of 128, trained once with batch norm and once without batch norm.
- In Both cases, the model that used batch norm tends to have a much smoother loss curve, with fewer spikes and even reduction in loss, compared to the model that did not use batch norm, which shows erratic and varying reduction in loss.
- We can see that despite using a sub optimal intiialization strategy like uniform normal along with tanh, when batch norm is used, it tends to remain stable because the outputs are normalized before going to the next layer. This enables faster convergence and better generatlization.