Paige Rosynek
CS 3450 021
Dr. Sebastian Berisha
04.13.2023

Lab 04 - Tuning Autograd Loop for Fashion-MNIST & CIFAR-10

I. Fashion-MNIST Experiments

For each of the following experiments run on the Fashion-MNIST dataset, I used the batch script shown below:

```
#!/bin/bash

#SBATCH --partition=teaching
#SBATCH --gpus=1
#SBATCH --cpus-per-gpu=8
singularity run --nv -B /data:/data -B /scratch:/scratch /data/cs3450/pytorch20.11.3.sif python run_fmnist.py
```

Therefore, for each of the experiments run below 1 NVIDIA T4 GPU with 8 CPUS was used. In addition, each experiment was run on a network with only a single hidden layer, however the number of nodes in each layer is listed in the table. The input layer has 784 nodes and the output layer has 10 nodes (one for each category). Also note that the number after the experiment number is the batch job number used for debugging purposes.

Experiment 1:

For this experiment, there were 256 nodes in the hidden layer. This value I guessed using intuition; I thought that since the network has only one hidden layer, then the hidden layer should most likely have a few hundred neurons in order to create a decent model. I randomly chose a batch size of 10 and 10 epochs without regularization in order to create a model for comparison.

Experiment 2:

In this experiment, I only changed the regularization term to 0.001. The motivation for this is to see how regularization affects the performance of the same model from experiment 1 (which was already performing well).

Experiment 3:

For this experiment I increased the batch size from 10 to 32. I did this to try and reduce any overfitting and to speed up the training time.

Experiment 4:

For this experiment I decided to decrease the batch size and increase the number of epochs to train for. I used a batch size of 8 and 15 epochs. Since the model performed about the same for a batch size of 10 and 32, I was curious to see how a smaller batch size paired with a longer training time would affect the results especially in regards to overfitting.

Experiment 5:

For this experiment I decided to decrease the batch size and increase the number of epochs to train for. I used a batch size of 20 and 20 epochs. I noticed that increasing the number of epochs in the last experiment increased the accuracy on the test set without too much overfitting, so I decided to increase the number of epochs again. However, in order to somewhat prevent overfitting, I also increased the batch size.

Experiment 6:

In this experiment, I decided that I wanted to see how changing the number of neurons in the hidden layer affected the model performance. Again, since the network is very small I wanted enough neurons so that the model was not simply guessing. Therefore, I chose to use 128 neurons in the hidden layer for this experiment. In addition, reducing the number of nodes in the hidden layer decreases the number of model parameters, which should also increase training time. In order to create a model for comparison, I used a batch size of 10 and trained for 10 epochs without any regularization.

Experiment 7:

In this experiment, I increased the number of epochs to 15 in order to see if training for longer would improve the model.

Experiment 8:

Based on the results from experiment 7, I introduced regularization to the model. I used a regularization constant of 0.001. This should help mitigate overfitting to some extent.

Experiment 9:

At this point, I wanted to change the number of hidden neurons one more time out of curiosity. I picked a value between the previous two, and I ended up picking 360 hidden neurons. Then I trained a base model with a batch size of 10 for 10 epochs and no regularization.

Experiment 10:

For this experiment, I introduce regularization and increase the number of epochs. Increasing the number of epochs should give the model more time to better learn the classification task and produce a better accuracy.

II. Experiment Results: Fashion-MNIST

Experiment #	Number Nodes Per Layer	Learning Rate	Batch Size	Epochs Trained	Regularization Constant	Training Accuracy	Training Loss	Test Accuracy
1 - 112259	784 -> 256 -> 10	0.01	10	10	0	89.482%	0.291	87.59%
2 - 112260	784 -> 256 -> 10	0.01	10	10	0.001	88.873%	0.314	86.92%
3 - 112261	784 -> 256 -> 10	0.01	32	10	0.001	89.237%	0.9547	87.21%
4 - 112262	784 -> 256 -> 10	0.01	8	15	0.001	90.730%	0.206	87.89%
5 - 112265	784 -> 256 -> 10	0.01	20	20	0.001	91.005%	0.508	88.21%

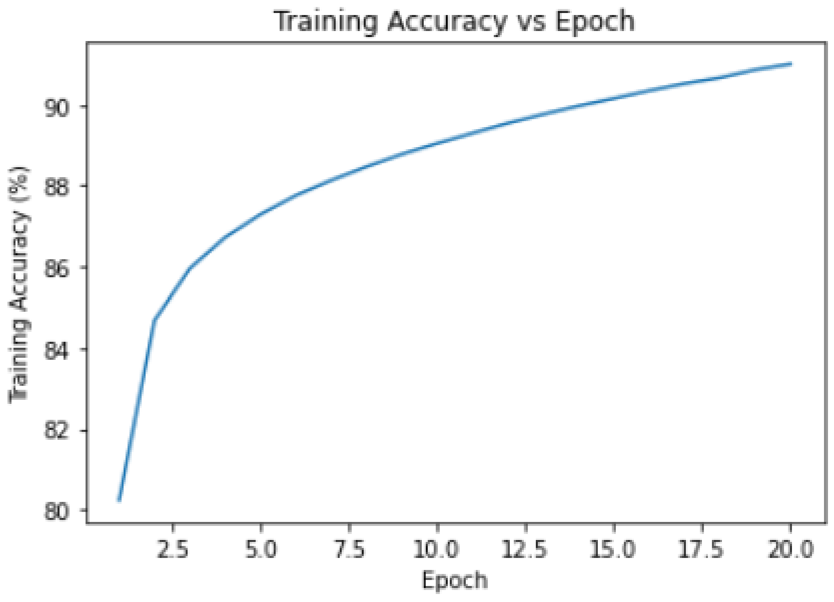
6 - 112268	784 -> 128 -> 10	0.01	10	10	0	88.889%	0.309	87.02%
7 - 112269	784 -> 128 -> 10	0.01	10	15	0	90.290%	0.270	87.75%
8 - 112270	784 -> 128 -> 10	0.01	10	15	0.001	89.212	0.305	87.17%
9 - 112272	784-> 360 ->10	0.01	10	10	0	89.938%	0.279	87.56%
10 - 112274	784-> 360 ->10	0.01	10	15	0.001	89.778%	0.290	87.62%

Experiment #	Number of Parameters	Estimated GPU Memory	Estimated Training Time	Approximate Epoch: Batch Ratio	Actual GPU Memory Usage	Actual Training Time
1	203,530	407060 KB? (did not specify)	x	1:1	945 MiB	92.33 s
2	203,530	945 MiB	92 s	1:1	945 MiB	91.23 s
3	203,530	945 MiB	30 s	1:3	945MiB	31.13 s
4	203,530	945 MiB	180s	2:1	945MiB	171.99 s
5	203,530	945 MiB	92 s	1:1	949MiB	93.97 s
6	101,770	203,540 KB?	x	1:1	947MiB	91.14 s

7	101,770	947MiB	120 s	1.5:1	947MiB	135.86 s
8	101,770	947MiB	120 s	1.5:1	947MiB	134.51 s
9	286,210	572420 KB?	91 s	1:1	941MiB	89.74 s
10	286,210	941MiB	120 s	1.5:1	941MiB	141.20 s

Final Model Training

The model that produced the best training accuracy was the model in Experiment 5. The network has 256 hidden neurons and was trained using a batch size of 20 for 20 epochs with 0.001 regularization. This model was able to achieve a training accuracy of about 91% and a test accuracy of 88.2%. Since the test accuracy is slightly greater than the test set accuracy, it should be noted that this model could be overfit. However, with further hyperparameter tuning I think an even better test accuracy can be achieved.



III. CIFAR-10 Experiments

For each of the following experiments run on the CIFAR-10 dataset, I used the batch script shown below:

```
#!/bin/bash

#SBATCH --partition=teaching
#SBATCH --gpus=1
#SBATCH --cpus-per-gpu=8
singularity run --nv -B /data:/data -B /scratch:/scratch /data/cs3450/pytorch20.11.3.sif python run_cifar.py
```

Therefore, for each of the experiments run below 1 NVIDIA T4 GPU with 8 CPUS was used. In addition, each experiment was run on a network with only a single hidden layer, however the number of nodes in each layer is listed in the table. The input layer has 3072 nodes and the output layer has 10 nodes (one for each category). Also note that the number after the experiment number is the batch job number used for debugging purposes.

IV. Experiment Results: CIFAR-10

Experiment #	**Number Nodes Per Layer**	**Learning Rate**	**Batch Size**	**Epochs Trained**	**Regulariz -ation Constant**	**Training Accuracy**	**Training Loss**	**Test Accuracy**
1 - 112278	3072 -> 256 -> 10	0.01	50	20	0	57.146%	6.047	47.77%
2 - 112279	3072 -> 256 -> 10	0.01	50	50	0	68.766%	4.485	48.44%
3 - 112281	3072 -> 256 -> 10	0.01	10	20	0	59.060%	1.169	48.78%
4 - 112284	3072 -> 256 -> 10	0.01	15	20	0.001	56.902%	1.838	47.58%

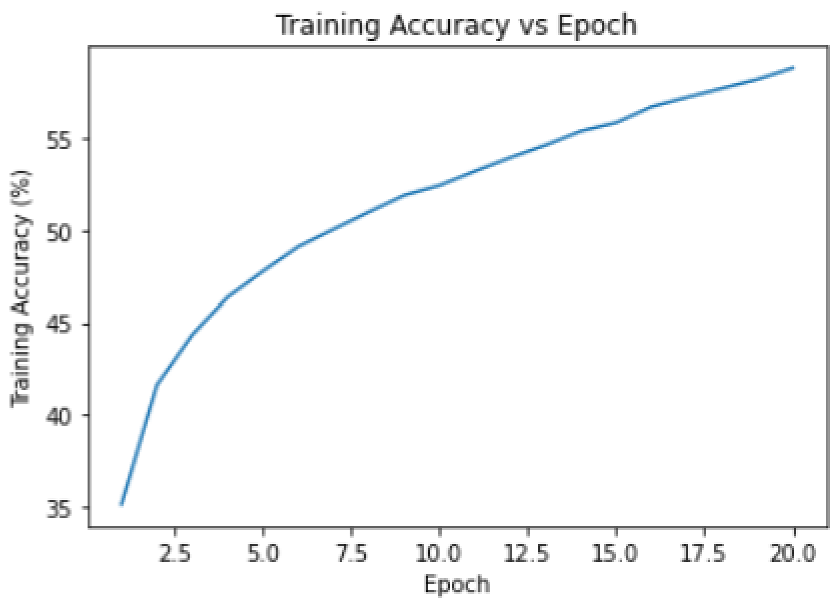
5 - 112285	3072 -> 256 -> 10	0.015	10	20	0.001	54.700%	1.278	47.75%
6 - 112288	3072 -> 512 -> 10	0.01	25	10	0	54.21%	3.253	47.99%
7 - 112290	3072 -> 512 -> 10	0.01	128	50	0	62.860%	13.626	48.70%
8 - 112292	3072 -> 512 -> 10	0.01	128	20	0.001	53.636%	16.869	47.73%
9 - 112294	3072 -> 512 -> 10	0.01	25	20	0.001	60.480%	2.828	48.49%
10 - 112296	3072 -> 512 -> 10	0.02	10	20	0.001	54.560%	1.284	46.67%

Experiment #	**Model Parameters**	**Estimated GPU Memory**	**Estimated Training Time**	**Approximate Epoch: Batch Ratio**	**Actual GPU Memory Usage**	**Actual Training Time**
1	789,258	1578516 KB?	x	2:5	1347 MiB	34.01 s
2	789,258	13467 MiB	80 s	1:1	1347 MiB	79.32 s
3	789,258	13467 MiB	160 s	2:1	1351 MiB	156.99 s
4	789,258	13467 MiB	90 s	2:1.5	1351 MiB	107.15 s
5	789,258	13467 MiB	160 s	2:1	1351 MiB	158.65 s
6	1,578,506	3,157,012 KB?	x	1:2.5	1367 MiB	33.66 s

7	1,578,506	1367 MiB	x	1:0.5	1367 MiB	32.48 s
8	1,578,506	1367 MiB	15 s	1:0.25	1367 MiB	14.35 s
9	1,578,506	1367 MiB	60 s	2:2.5	1367 MiB	63.34 s
10	1,578,506	1367 MiB	120 s	2:1	1367 MiB	163.37 s

Final Model Training

The model that produced the best training accuracy was the model in Experiment 3. The network has 256 hidden neurons and was trained using a batch size of 10 for 20 epochs with no regularization. This model was able to achieve a training accuracy of 59.06% and a test accuracy of 48.78%. Since the test accuracy is much greater than the test set accuracy, it should be noted that this model is most likely overfit and will not generalize well to unseen data. This is already somewhat evident in the low testing accuracy of 48.78%. The addition of more layers would improve the test accuracy for the dataset overall.



V. Discussion

Throughout all the experiments on both the Fashion-MNIST and CIFAR-10 datasets, I observed some interesting relationships between various hyperparameters and network architecture with the accuracy of the overall model. One thing I noticed is that the GPU memory usage is most likely dependent on the number of model parameters or otherwise size of the network. For all the experiments, I found that networks that had the same number of model parameters, or more specifically in this case, hidden neurons, used almost the exact same amount of GPU memory. Additionally, I was able to estimate the runtime of a network given the runtime of a single base model by using the ratio of the number of epochs to the batch size for the model. For instance, if a model with a epoch to batch size ratio of 1:1 had a run time of 20s, then a model with a 2:1 ratio would approximately have a runtime of 40s and a model with a 1:2 ratio would have a runtime of about 10s. It can also be observed that introducing regularization often decreases the train and test accuracy of the model and increases the loss. However, I learned that tuning the regularization term is as important as tuning the rest of the hyperparameters and tuning it may or may not improve the model.

Note, the lab document did not specify an introduction so I omitted this. In addition, I did not fully understand how to estimate the initial GPU memory usage as described in the lab, so I just doubled the model parameters and assumed the unit was KB. Also, I do not understand why the loss for some of the results is > 1.