Project Report

Problem

Train a CNN to get >.75 accuracy on the test set of CIFAR-10.

Design and implementation

1. Model architecture

Convolutional layer: batch norm, relu, maxpool

Second convolutional layer: batch norm, relu, maxpool

Third convolutional layer: batch norm, relu, maxpool

Flatten conv output for FNN input

FNN 1 (512): batch norm, relu

FNN 2 (256, 512): batch norm, relu

FNN 3 (128, 256): batch norm, relu

FNN 4 (64, 128): batch norm, relu

FNN 5 (10, 64): softmax

1.1 Methods

Convolutional layers. Maxpool. Relu. Image flipping and cropping. Adding more FNN params. Image flipping and cropping helped the model generalize better. Adding more parameters to the fully-connected layers improved training loss without slowing it down much (from making the parameters all factors of two to increase efficiency of gpu/cpu usage). Learning rate decay improved at first, but decreased performance for longer training runs. Vectorization of batches for convolution and maxpool improved efficiency and decreased ram. Surprisingly, the optimal base learning rate I found to be 0.01, which is much higher than I would have thought.

2. Evaluation

I measured how fast the loss was dropping, and if it started to stagnate. Many times the loss might fail to drop as fast as I thought it should, or would get “stuck” and fail to drop after many epochs. A second thing I would check would be the changes to the test accuracy vs. the training loss. With some versions, the training cost was continuing to drop, but the test accuracy was not improving, or sometimes getting slightly worse. Then I would have to take measures to avoid overfitting.

3. Ablation study

Dropout

Goal: Introduced to mitigate overfitting.

Result: Training loss stagnated, and test accuracy plateaued quickly.

Summary: Dropout resulted in both loss getting stuck and failing to drop.

Flatten

Goal: Feature vectorization.

Result: Limited-to-no model performance.

Summary: Did not significantly improve training loss, test accuracy, or speed.

Learning Rate Decay

Goal: Adjusting the learning rate over epochs aimed at improving convergence.

Result: Initially accelerated achieving 75% accuracy but plateaued before reaching 80%. With extended epochs, the learning rate became excessively low, hindering further improvement.

Summary: Moderate learning rate decay (step decay) was beneficial initially but required precise tuning; overly aggressive decay impaired long-term training performance. With more time, I probably could have altered the rate of decay to improve this, since my mentality is that the learning rate was too high after a certain number of epochs, but then had decreased too much later on, resulting in extremely slow improvement.

Gradient Descent with Momentum

Goal: Intended to speed convergence.

Result: Did not substantially improve model performance.

Summary: Gradient descent with momentum did not create noticeable improvement, so I removed it.

Data Augmentation

Goal: Improve model generalization.

Result: Training and test accuracies improved uniformly; significantly reduced overfitting.

Summary: Data augmentation consistently enhanced the model's generalization. My finding was that training loss had dropped significantly, but test accuracy had not, resulting in overfitting. My solution to fix this was to add random cropping and flipping to the data, which then resulted in much better generalization, where the improvement in training and test accuracy occurred at roughly the same rate.

Increased Parameters in FCNs

Goal: Bring down training loss by increasing the number of parameters in fully-connected layers.

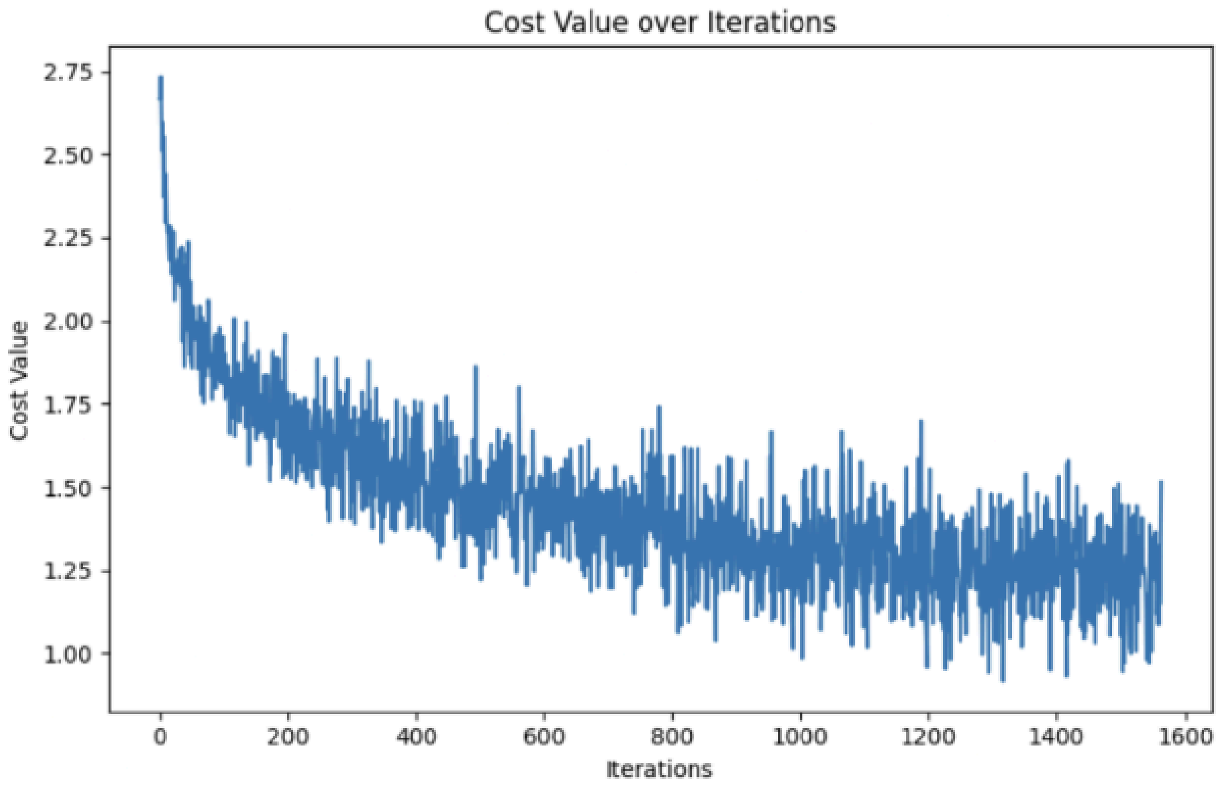
Result: Improved training loss without significant slowdowns; optimized by aligning layer sizes to computational hardware.

Summary: Increasing FCN parameters effectively improved model performance and efficiency.

Final model configuration

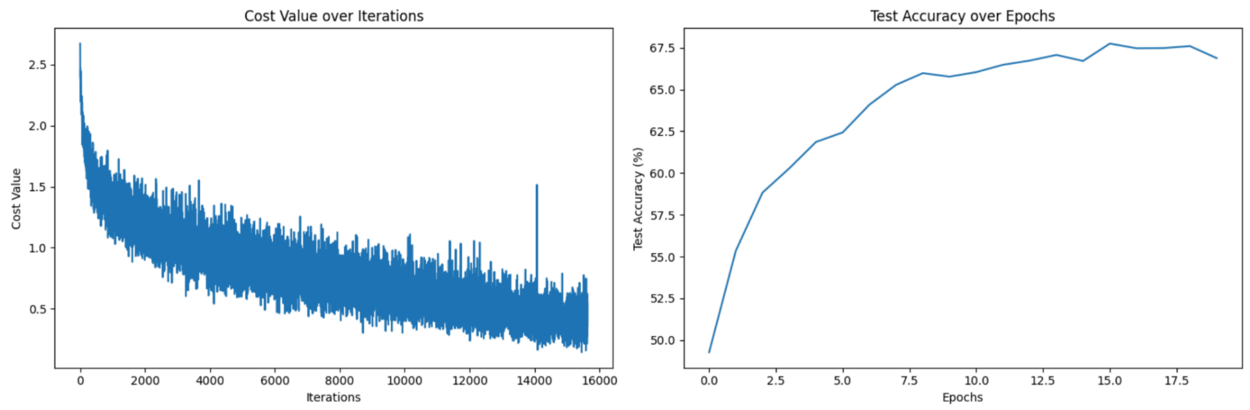
The final optimal model excluded dropout and learning rate decay, used a moderate increase in parameter count for FCNs, and incorporated data augmentation (random cropping and flipping). Final test accuracy exceeded 80%, with training and test accuracy consistently similar, indicating reduced overfitting and improved generalization.

“Vanilla” model with 2 CNN, 3 FNN (256, 128, 10):



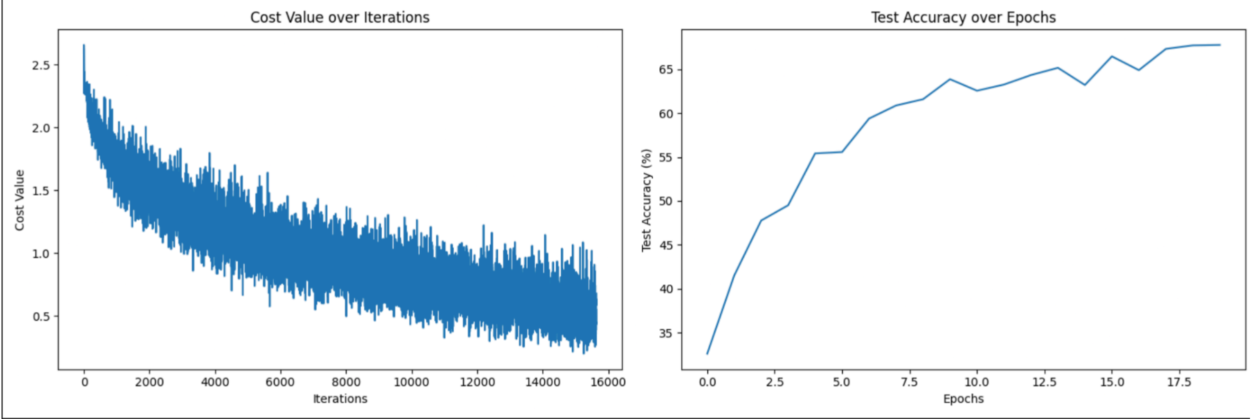
Model with significantly more FCN parameters:

```
        1024, 1024, 896, 896,

        768, 768, 640, 640,

        512, 512, 448, 448,

        384, 384, 320, 320,
        256, 256, 192, 192,

        160, 160, 128, 128,

        96, 96, 64, 64,

        32, 32,

        10
```



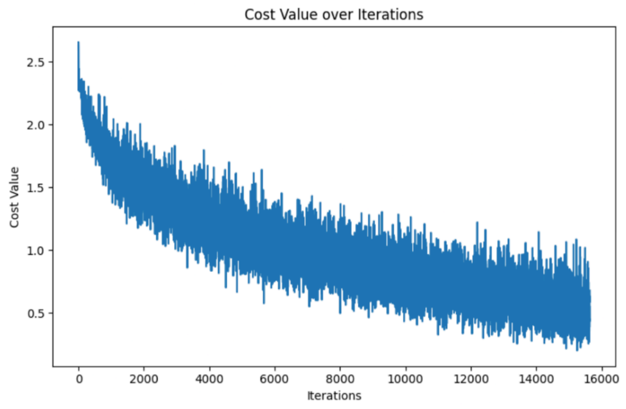
Model with tweaked parameter count:

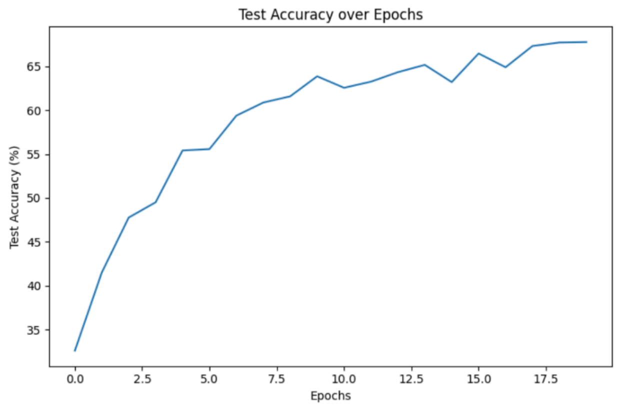
```
        2048, 2048,

        1024, 1024,

        512, 512,

        256, 256,

        128, 128,

        64, 64,

        32, 32,

        10
```



Model with moderately increased FCN parameters:

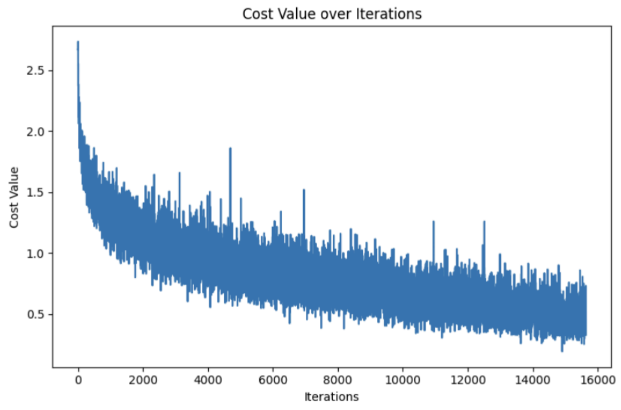
```
        512,

        256,

        128,

        10
```

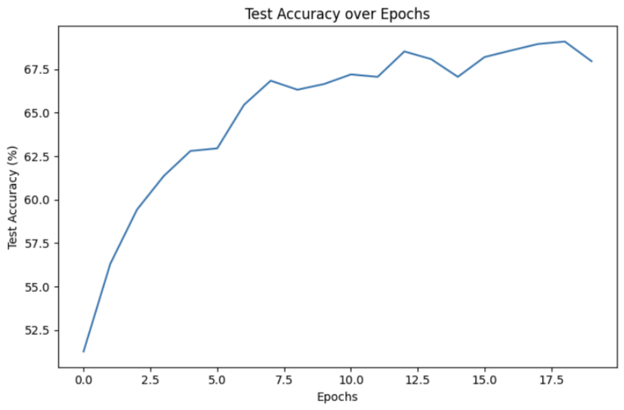




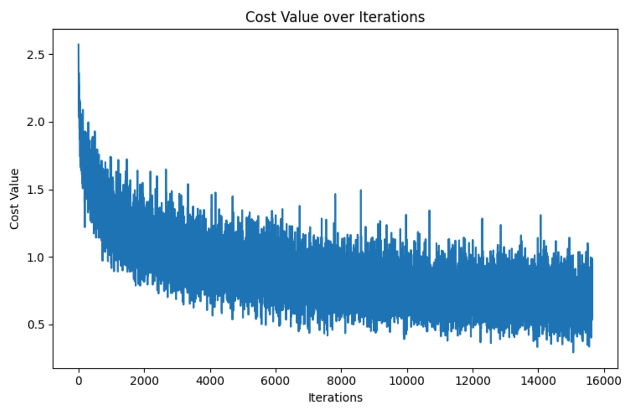
Model with slightly adjusted parameters (same params as final configuration):

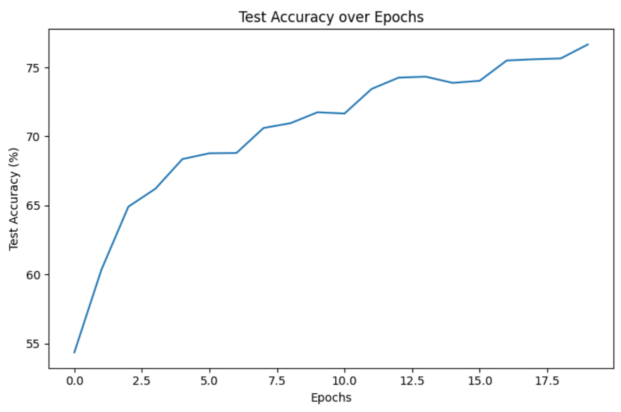
```
        512,

        256,

        128,

       64,

        10
```



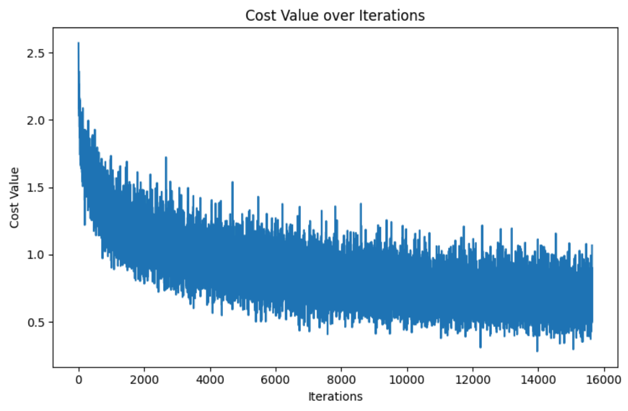


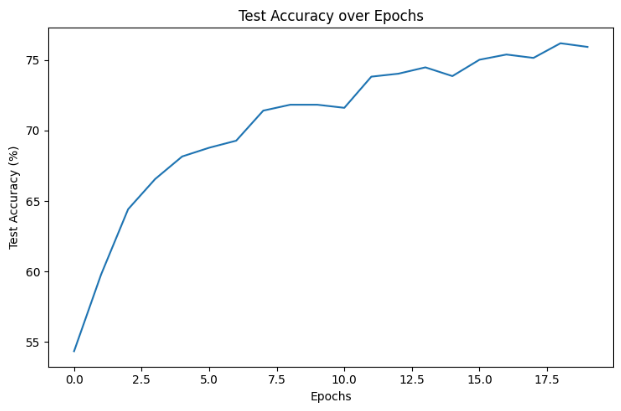
Model trained with random cropping, image flip, more params:



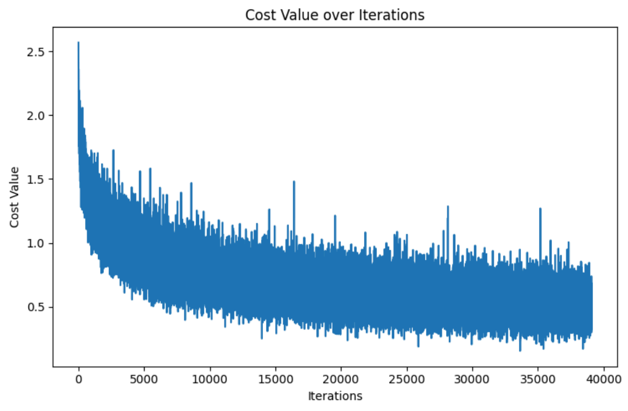


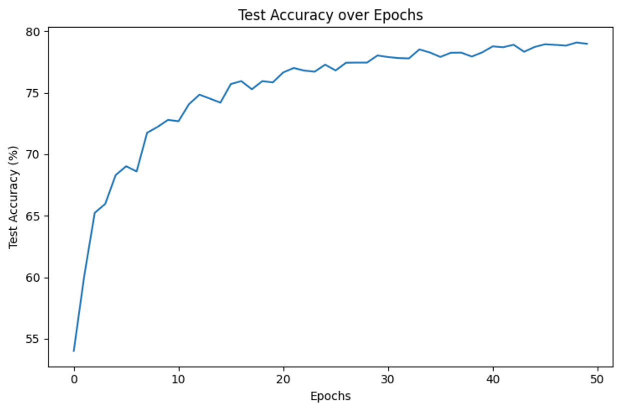
Model with learning rate decay (step decay) added:



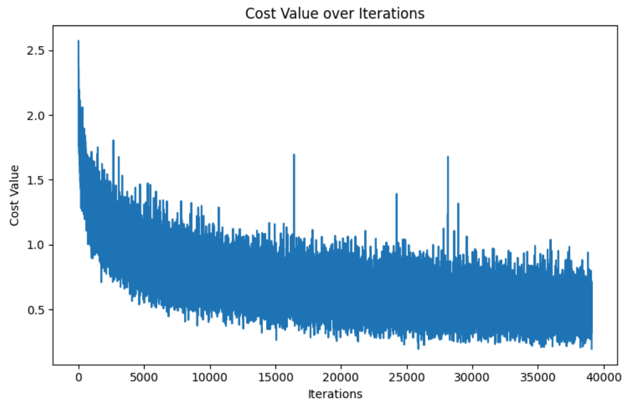


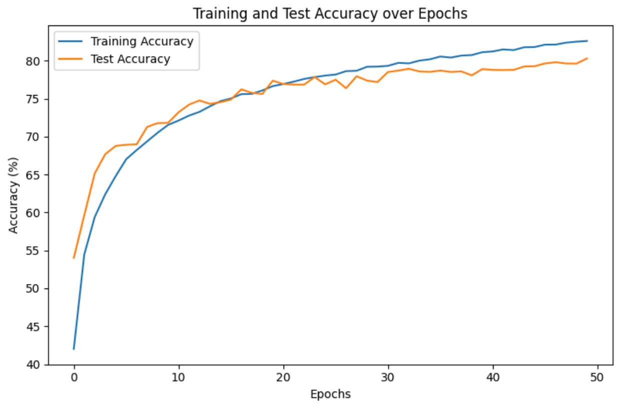
Model with learning rate decay (step decay) and more epochs:

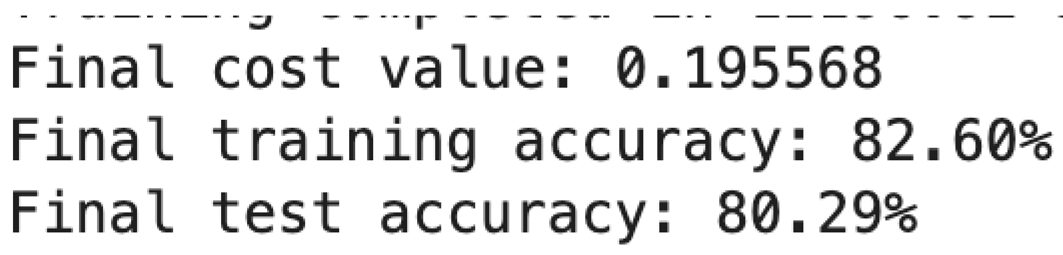




Final model with more epochs and learning rate decay removed, trained on A100 using Google Collab:

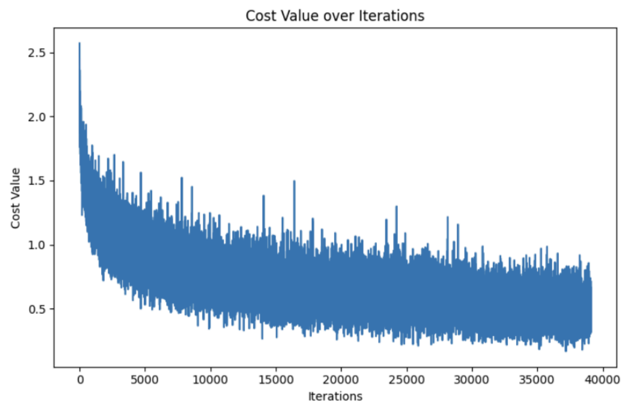


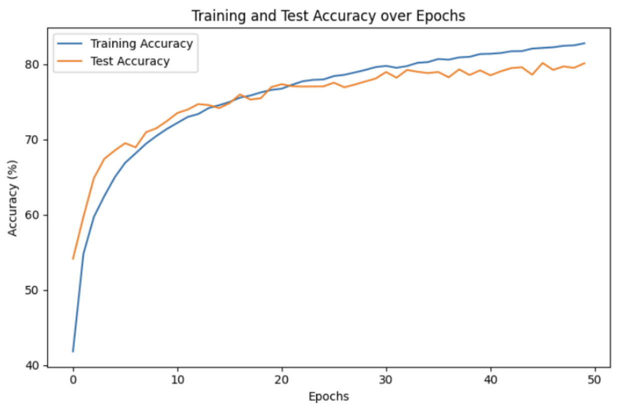


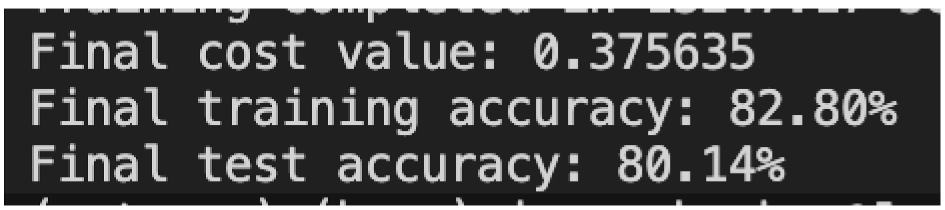


```
Final cost value: 0.195568
Final training accuracy: 82.60%
Final test accuracy: 80.29%
```

Final model trained using Macbook Pro Intel chip CPU:



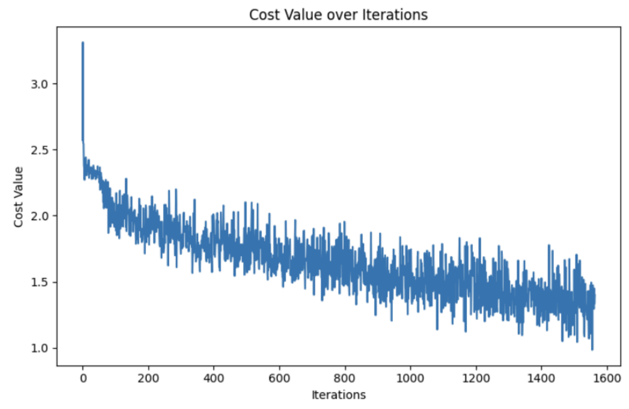


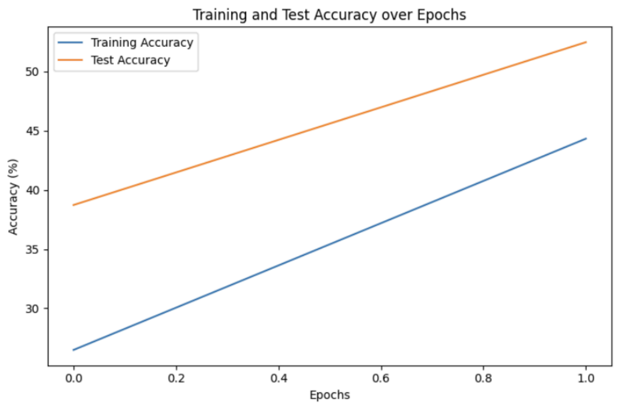


```
Final cost value: 0.375635
Final training accuracy: 82.80%
Final test accuracy: 80.14%
```

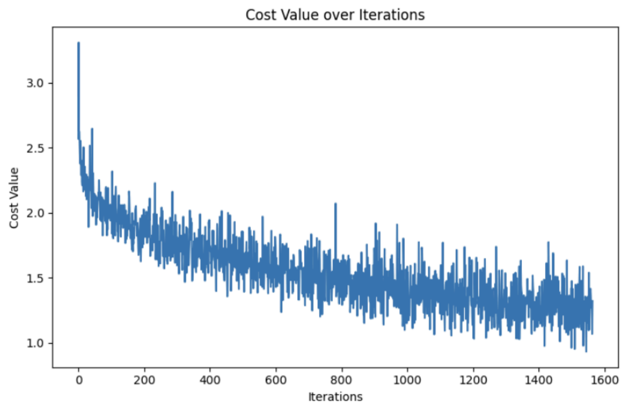
Other implementations (added separately to final configurations for shorter runs, and some older configs with longer training runs):

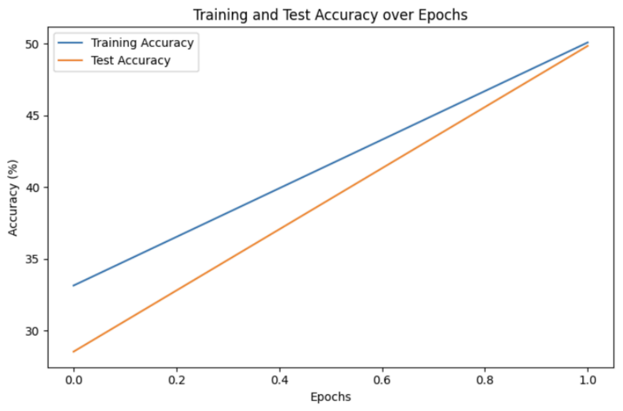
Adam (final config; beta1 = 0.9, beta2 = 0.999, epsilon = 1e-8):



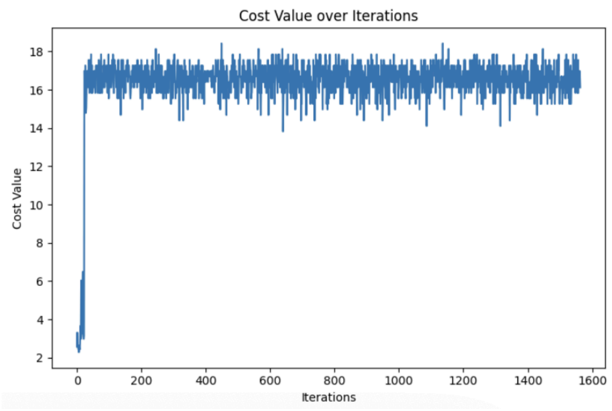


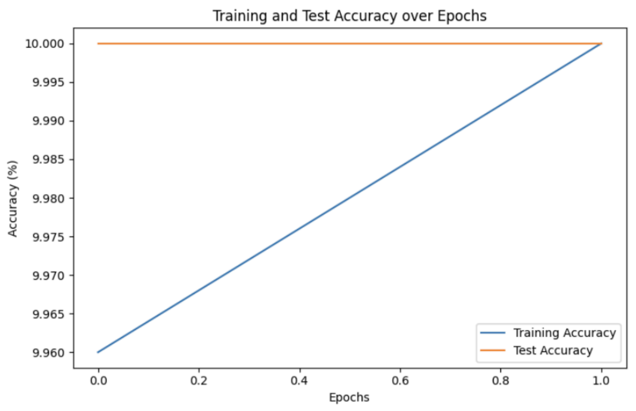
Adam (final config; beta1 = 0.09, beta2 = 0.999, epsilon = 1e-8):



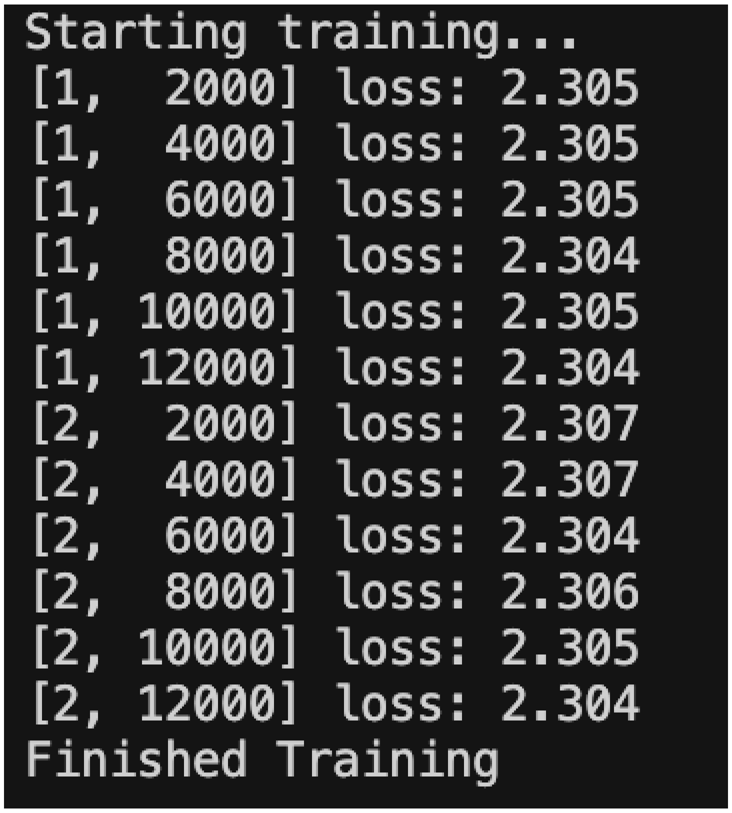


Adam (final config; beta1 = 0.9, beta2 = 0.099, epsilon = 1e-8):



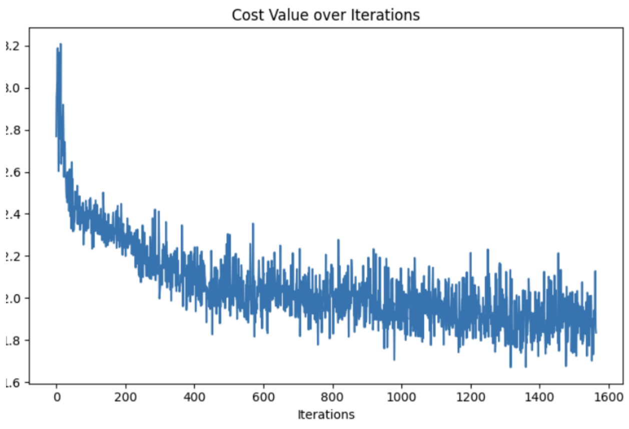


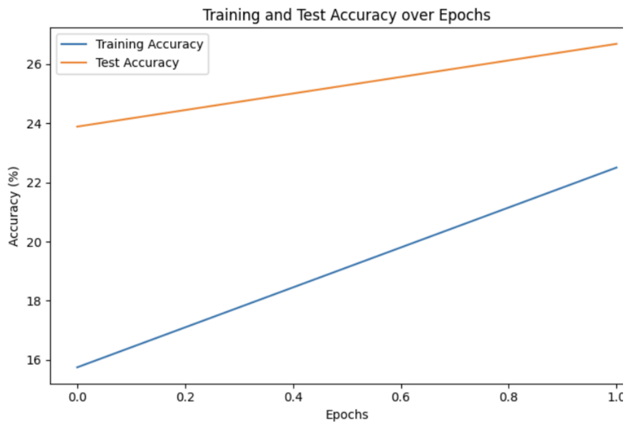
Adam (older config; beta1 = 0.9, beta2 = 0.999, epsilon = 1e-8)):



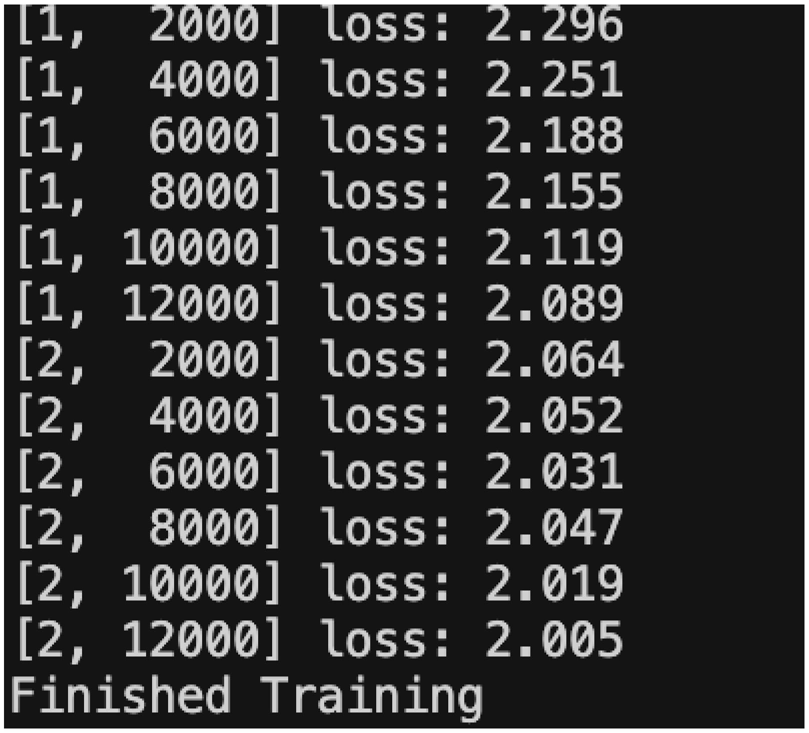
```
Starting training...
[1,  2000] loss: 2.305
[1,  4000] loss: 2.305
[1,  6000] loss: 2.305
[1,  8000] loss: 2.304
[1, 10000] loss: 2.305
[1, 12000] loss: 2.304
[2,  2000] loss: 2.307
[2,  4000] loss: 2.307
[2,  6000] loss: 2.304
[2,  8000] loss: 2.306
[2, 10000] loss: 2.305
[2, 12000] loss: 2.304
Finished Training
```

Dropout (on final config):



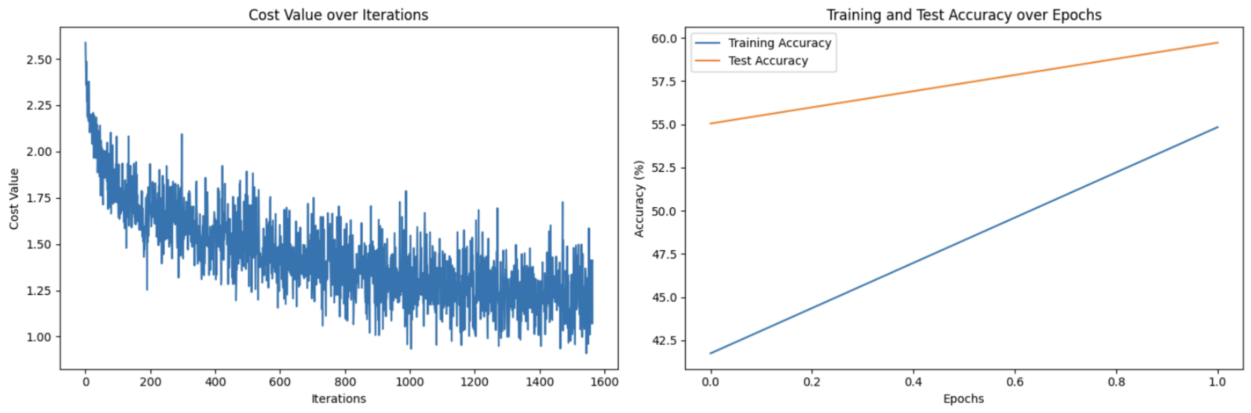


Dropout (on older config):

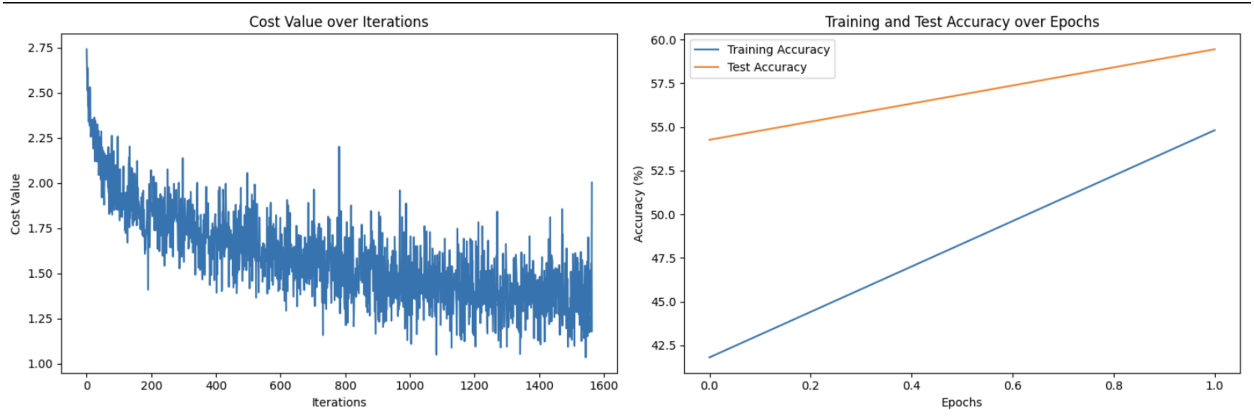


```
[1,  2000] loss: 2.296
[1,  4000] loss: 2.251
[1,  6000] loss: 2.188
[1,  8000] loss: 2.155
[1, 10000] loss: 2.119
[1, 12000] loss: 2.089
[2,  2000] loss: 2.064
[2,  4000] loss: 2.052
[2,  6000] loss: 2.031
[2,  8000] loss: 2.047
[2, 10000] loss: 2.019
[2, 12000] loss: 2.005
Finished Training
```

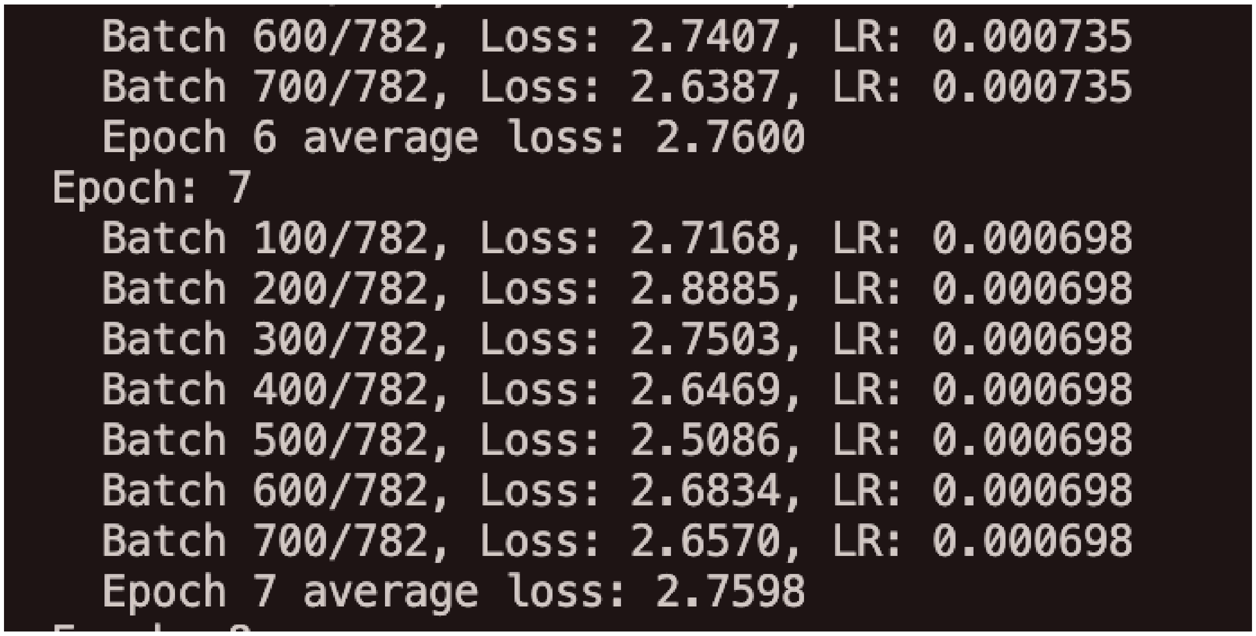
L2 regularization (final config, lamba: 0.001):



L2 regularization (final config, lamba: 0.01):

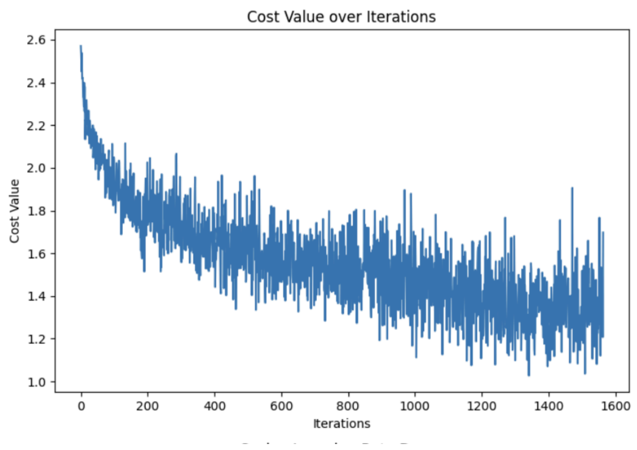


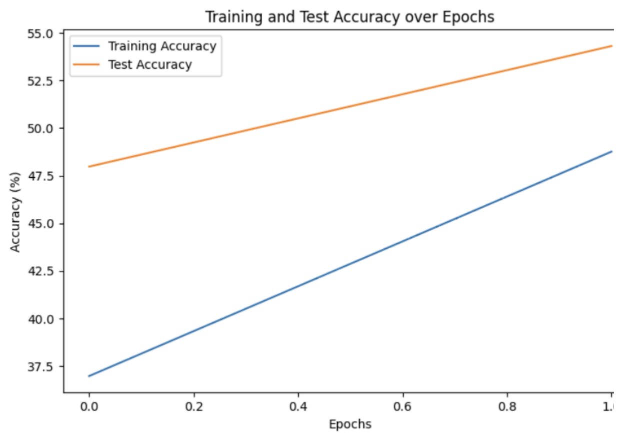
L2 regularization (older config, lamba: 0.01):



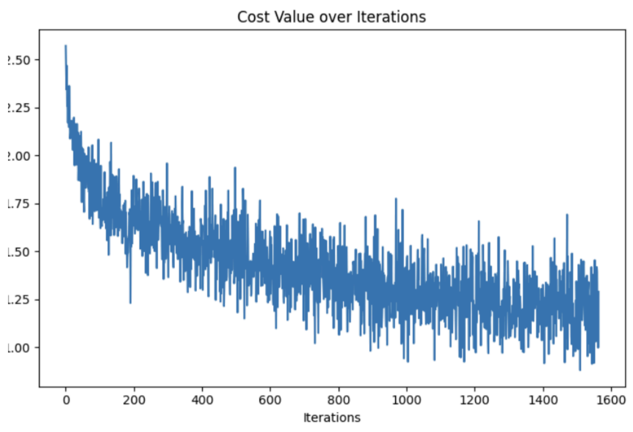
```
    Batch 600/782, Loss: 2.7407, LR: 0.000735
    Batch 700/782, Loss: 2.6387, LR: 0.000735
    Epoch 6 average loss: 2.7600
  Epoch: 7
    Batch 100/782, Loss: 2.7168, LR: 0.000698
    Batch 200/782, Loss: 2.8885, LR: 0.000698
    Batch 300/782, Loss: 2.7503, LR: 0.000698
    Batch 400/782, Loss: 2.6469, LR: 0.000698
    Batch 500/782, Loss: 2.5086, LR: 0.000698
    Batch 600/782, Loss: 2.6834, LR: 0.000698
    Batch 700/782, Loss: 2.6570, LR: 0.000698
    Epoch 7 average loss: 2.7598
```

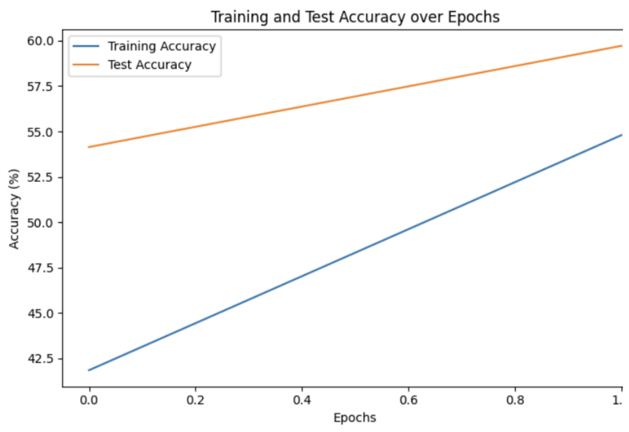
Cosign (final config):





Gradient descent with momentum (final config; momentum = 0.9):





4. Individual contributions

I am the sole group member, so I made all of the contributions.