Source Code

Remote Repository: https://github.com/Mikestriken/Deep-Learning-Class
Name: Michael Marais
Student ID: 801177649
Homework Number: 2

Problem 1

Problem 1 (30 pts)

AlexNet is originally proposed for 227*227 image sizes. It may be too complex for the CIFAR-10 and CIFAR-100 datasets, in particular, due to the low resolution of the initial images; try simplifying the model to make the training faster while ensuring that the accuracy stays relatively high. Report the training loss, validation loss, and validation accuracy. Also, report the number of parameters in your modified version of AlexNet and compare it against the number of parameters in the original AlexNet architectures. Here is a good reference guide to AlexNet: https://www.kaggle.com/code/blurredmachine/alexnet-architecture-a-complete-guide

Explore the option of applying Dropout techniques for training your customized AlexNet. Compare the training and validation results against the baseline model without any dropout. Also, compare the results between CIFAR-10 and CIFAR-100.

Here are some information about CIFAR-100

https://paperswithcode.com/dataset/cifar-100

https://pytorch.org/vision/main/generated/torchvision.datasets.CIFAR100.html

Observations

Number of Parameters:
CIFAR-10: 36,187,914
CIFAR-100: 36,556,644
Original AlexNet: 62,300,000

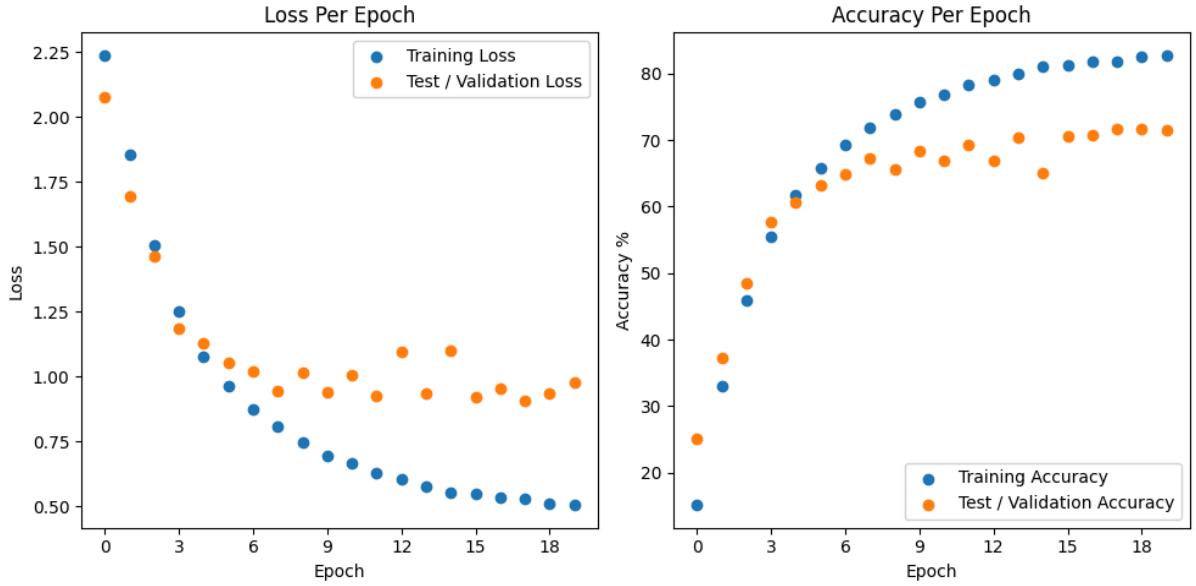
Baseline Vs. Dropout Enabled:
Using dropout in general reduced overfitting. It increased the training result loss and kept it more inline with the test / validation result loss. It also made the test / validation result loss more consistent (less variance).

When training on CIFAR-10 loss remained about the same with vs without dropout for the test / validation set. However, when training CIFAR-100, using dropout had a noticeable impact on reducing loss for the test / validation set.

CIFAR-10 Vs. CIFAR-100:
CIFAR-10 was a lot easier to train in terms of both time, accuracy and loss. CIFAR-100 reduced the performance benefits of using convolution layers to the point where the results looked similar to not using any convolution layers for CIFAR-10 in the previous homework.

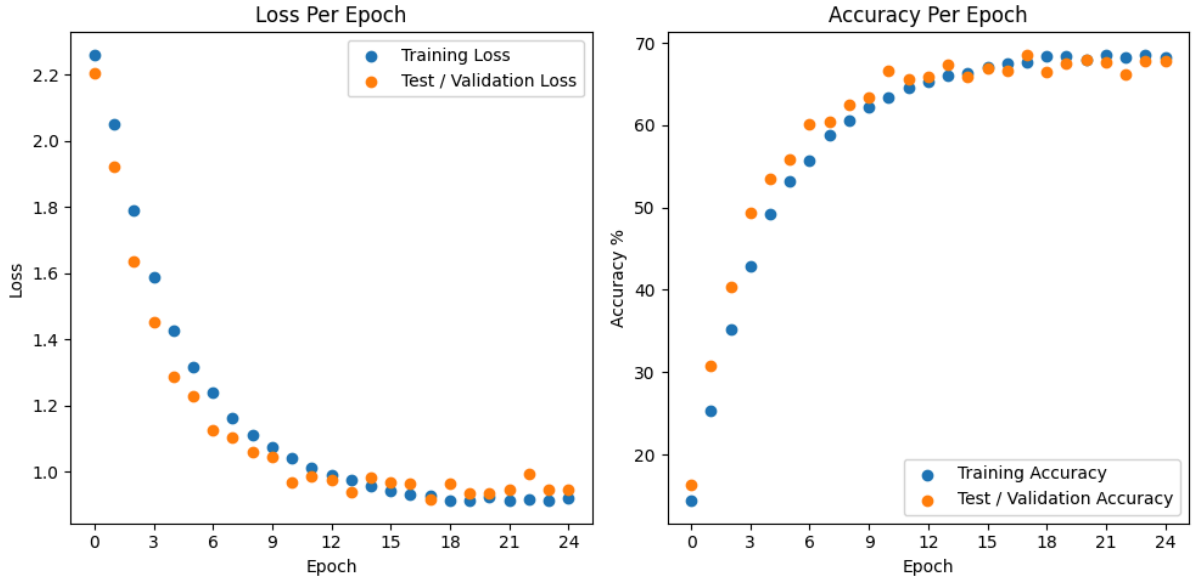
CIFAR-10

Baseline:



Last Best Epoch: 18, Train Time ≈ 18 min
Train Loss: 0.51034, Train Accuracy: 82.51%
Test Loss: 0.93697, Test Accuracy: 71.66%

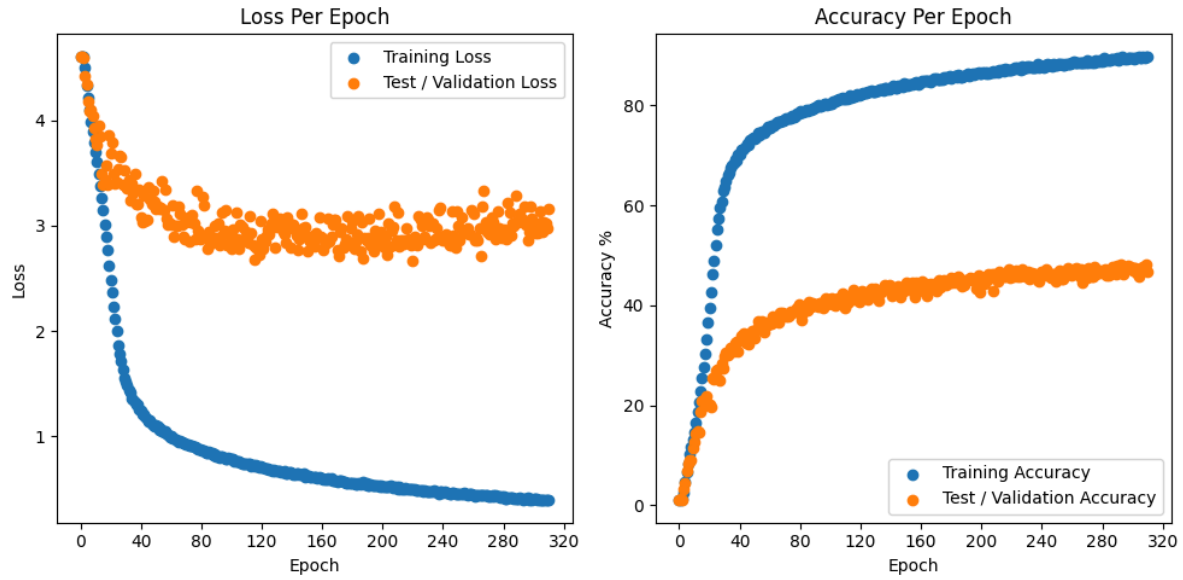
Dropout Enabled:



Last Best Epoch: 17, Train Time ≈ 15 min
Train Loss: 0.92624, Train Accuracy: 67.72%
Test Loss: 0.91852, Test Accuracy: 68.57%

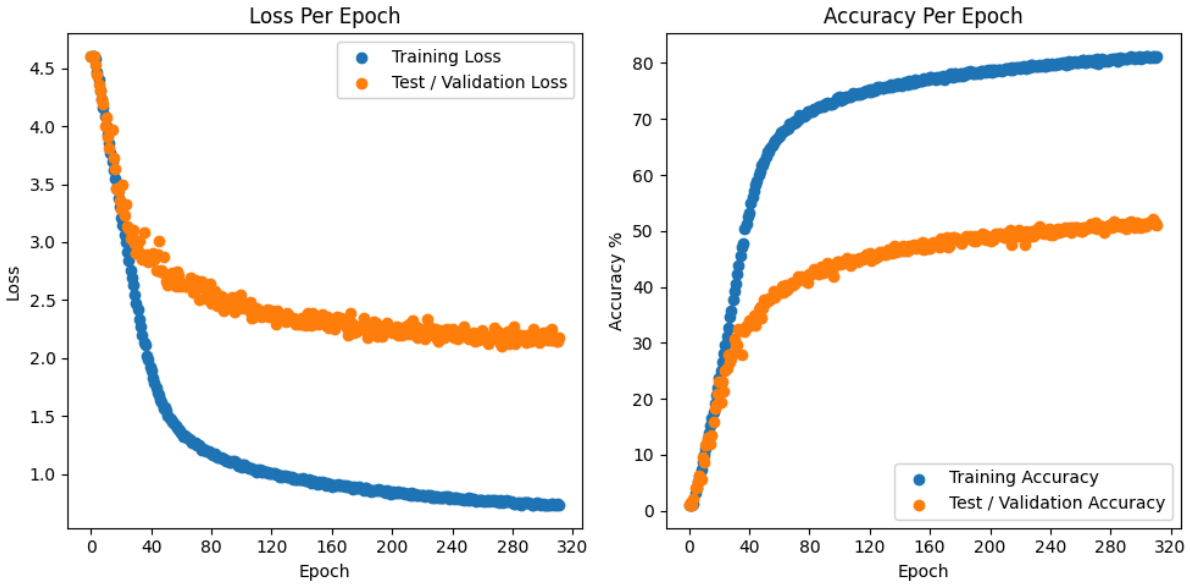
CIFAR-100

Baseline:



Last Best Epoch: 309, Train Time ≈ 3 hr, 52 min
Train Loss: 0.39752, Train Accuracy: 89.74%
Test Loss: 2.97470, Test Accuracy: 48.09%

Dropout Enabled:



Last Best Epoch: 308, Train Time ≈ 3 hr, 25 min
Train Loss: 0.73422, Train Accuracy: 81.26%
Test Loss: 2.17545, Test Accuracy: 52.09%

Problem 2

Problem 2 (30 pts)

Repeat the problem 1 this time for VGGNet. Identify the VGG configuration that matches the nearest number of parameters to the AlexNet Architecture that you did for problem 1 for CIFAR-10 and CIFAR-100 datasets. Compare your training and evaluation results against AlexNet in Problem 1.

Here is a good reference guide to AlexNet: https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide

Explore the option of applying Dropout techniques for training your customized AlexNet. Compare the training and validation results against the baseline model without any dropout. Also, compare the results between CIFAR-10 and CIFAR-100.

Observations

Number of Parameters:
AlexNet CIFAR-10: 36,187,914
AlexNet CIFAR-100: 36,556,644
VGGNet CIFAR-10: 42,160,970
VGGNet CIFAR-100: 42,529,700

VGGNet Vs. AlexNet:
Unlike with AlexNet, I could not use the full VGGNet-16 architecture proposed in the reference reading. Firstly the number of parameters way over exceeded that of AlexNet to be proportional, secondly, the model became way too complex and during training for CIFAR 100 the training loss would decrease but the test loss would remain relatively static with a test accuracy of 1%.

I looked online for some inspiration VGGNet architectures that would work with CIFAR but they all used batchnorm which the reference reading doesn't show. Thus I assumed batch norm was not allowed for VGGNet as it wasn't allowed for AlexNet for this assignment.

Without batch norm layers I was forced to use a very small VGGNet 6 Implementation in order to get the model to work with both the CIFAR 100 and 10 datasets.

Getting VGGNet 6 to work with CIFAR 100 was the pain point of this problem. I know that the number of parameters for this VGGNet implementation exceeds my AlexNet implementation but any added complexity and the model doesn't learn in CIFAR 100, reduced complexity has the same outcome.

Overall with this implementation of VGGNet, it performed 10% worse in test accuracy both with and without dropout for CIFAR 10, and around 20% worse for CIFAR 100. The training time for CIFAR 100 was markedly improved, going from 3 hours 25 minutes to only 51 minutes for the CIFAR 100 dataset with dropout.

In my journey to try get decent scores with VGGNet on CIFAR 100 without batch norm I did find another poor soul on reddit that never found a good solution.

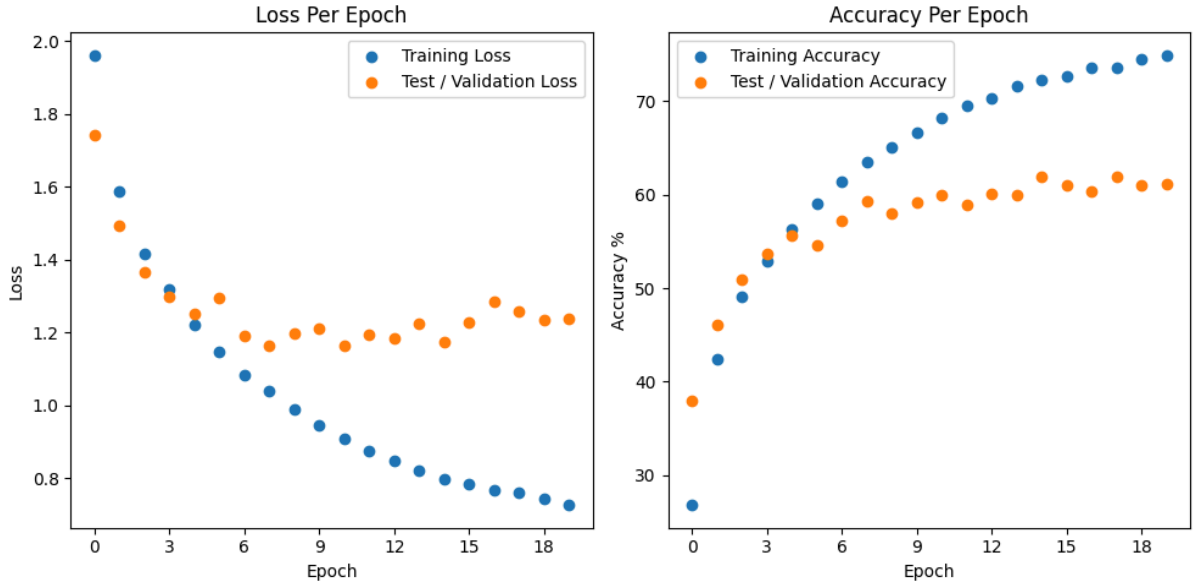
Baseline Vs. Dropout Enabled:
Using dropout in general reduced overfitting. It increased the training result loss and kept it more inline with the test / validation result loss.

When training on CIFAR-10 loss remained about the same with vs without dropout for the test / validation set. However, when training CIFAR-100, using dropout had a noticeable impact on increasing loss for the test / validation set.

CIFAR-10 Vs. CIFAR-100:
CIFAR-10 was a lot easier to train in terms of both time, accuracy and loss. CIFAR-100 was a terrible dataset for VGGNet 6 to train on without batch normalization.

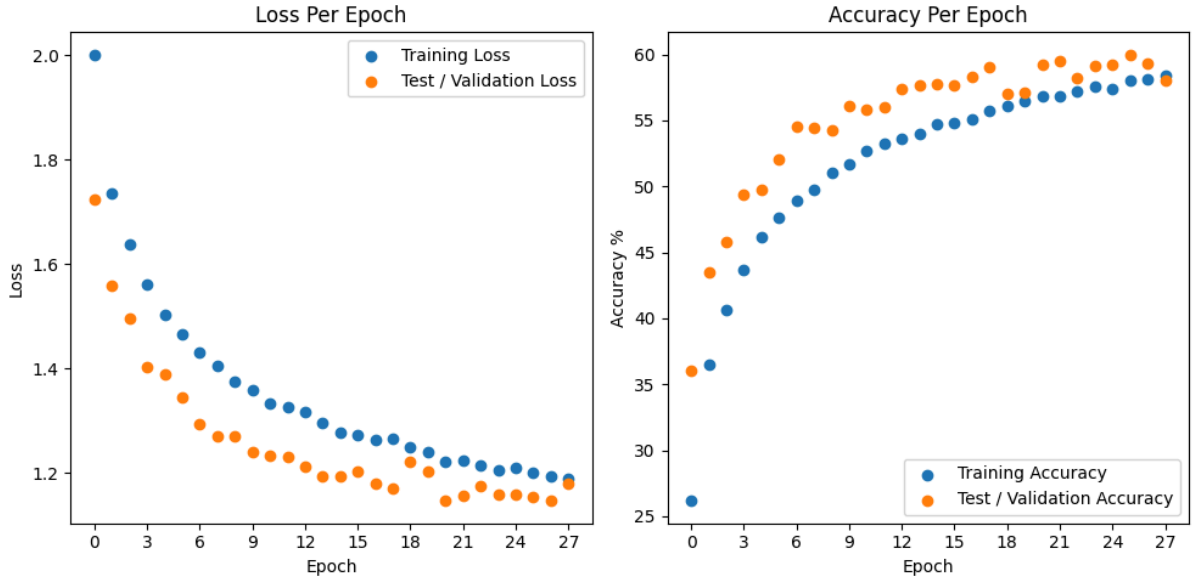
CIFAR-10

Baseline:



Last Best Epoch: 14, Train Time ≈ 0 hr, 11 min
Train Loss: 0.72636, Train Accuracy: 74.95%
Test Loss: 1.23712, Test Accuracy: 61.11%

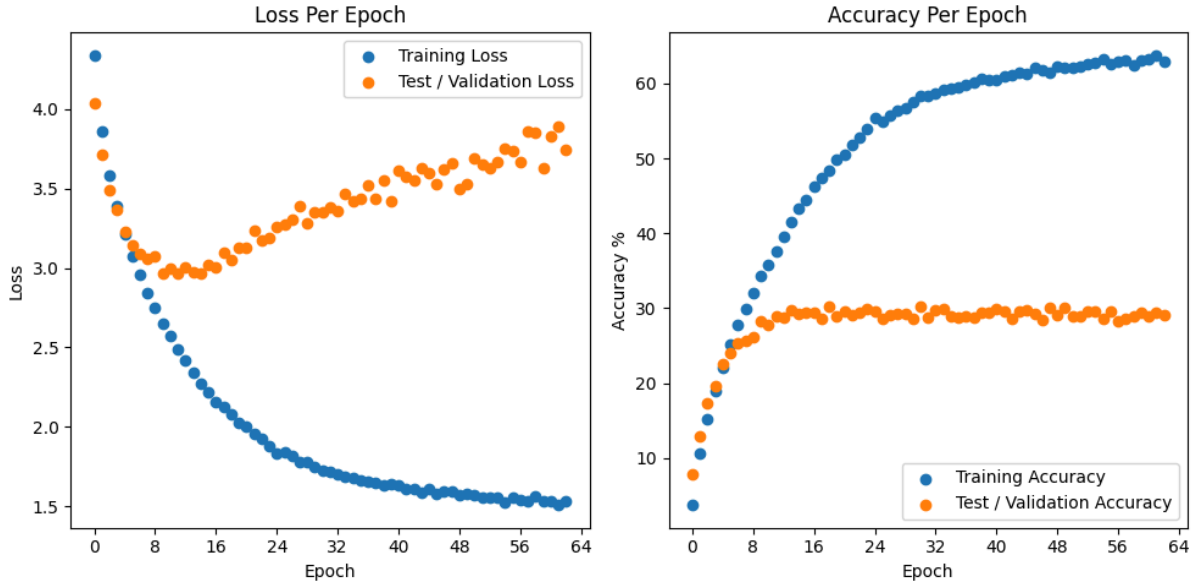
Dropout Enabled:



Last Best Epoch: 25, Train Time ≈ 0 hr, 19 min
Train Loss: 1.20018, Train Accuracy: 58.01%
Test Loss: 1.15498, Test Accuracy: 59.99%

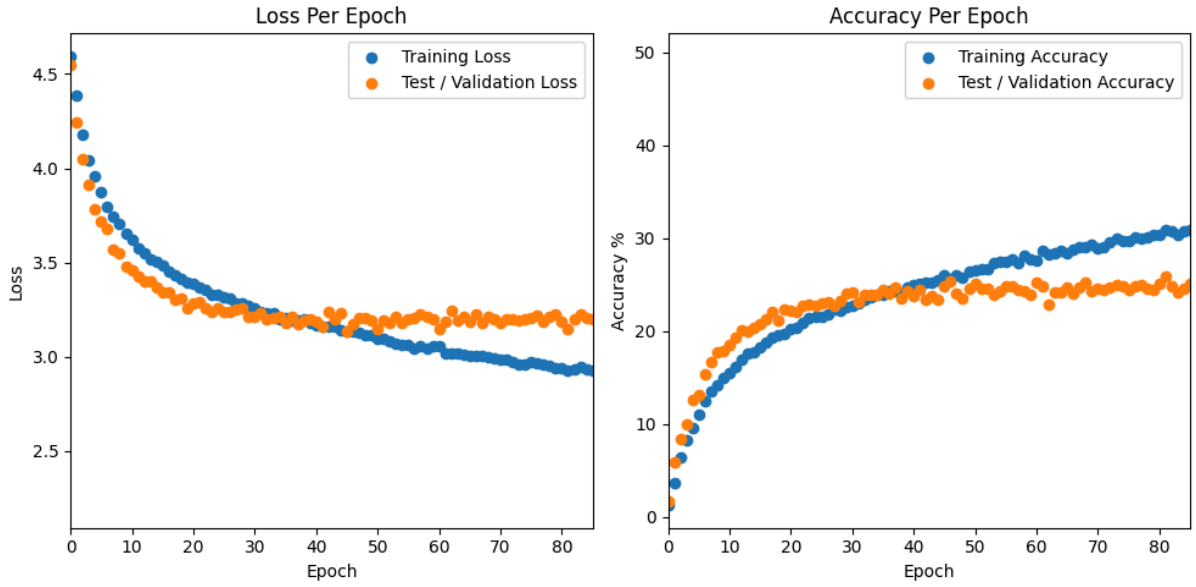
CIFAR-100

Baseline:



Last Best Epoch: 30, Train Time ≈ 0 hr, 20 min
Train Loss: 1.72385, Train Accuracy: 58.26%
Test Loss: 3.35307, Test Accuracy: 30.18%

Dropout Enabled:



Last Best Epoch: 81, Train Time ≈ 0 hr, 51 min
Train Loss: 2.92955, Train Accuracy: 30.83%
Test Loss: 3.14498, Test Accuracy: 25.92%

Problem 3

Problem 3 (40 pts)

The baseline model we did in lectures is called ResNet-11. Build a new version of ResNet (ResNet-18). Train it on CIFAR-10 and CIFAR-100 datasets. Plot the training loss, validation loss, and validation accuracy. Compare the classification accuracy, and model size across the two versions of ResNet (11, 18). How does the complexity grow as you increase the network depth?

You can find some references for ResNet 18 here:

https://www.kaggle.com/code/ivankunyankin/resnet18-from-scratch-using-pytorch

Explore the dropout option for the two networks and report your training results and validation accuracy. Also, compare the results between CIFAR-10 and CIFAR-100.

Observations

Number of Parameters:
CIFAR-10 + ResNet11: 30,343,594
CIFAR-10 + ResNet18: 165,877,202
ResNet14: 47,081,612
CIFAR-100 + ResNet11: 30,436,204
CIFAR-100 + ResNet18: 165,969,812

The network complexity grows approximately exponentially with added depth. This is weird because theoretically it should be linear as increasing the number of convolutional blocks is an additive process not multiplicative.

Doing some testing I noticed that the residual blocks cost a ton more parameters than the convolutional blocks, which is weird because the residual blocks are just made of 2 convolutional blocks (there is no 1x1 convolution as that was removed for the sake of improving model performance).

Residual blocks add approximately 76,096,672 additional parameters. A single convolution block adds approximately 19,024,168 additional parameters.

In conclusion, network complexity will grow linearly with added depth but it is depending on how the network architecture and how many residual blocks there are.

Baseline Vs. Dropout Enabled:
Using dropout in general reduced overfitting. It increased the training result loss and kept it more inline with the test / validation result loss. It also made the test / validation result loss more consistent (less variance).

It also however added a training time penalty for the CIFAR 100 dataset. Training time doubled for CIFAR 100, but CIFAR 10 showed no increase in training time to get the same results. Also note that by training time I mean the number of Epochs to train as well as time.

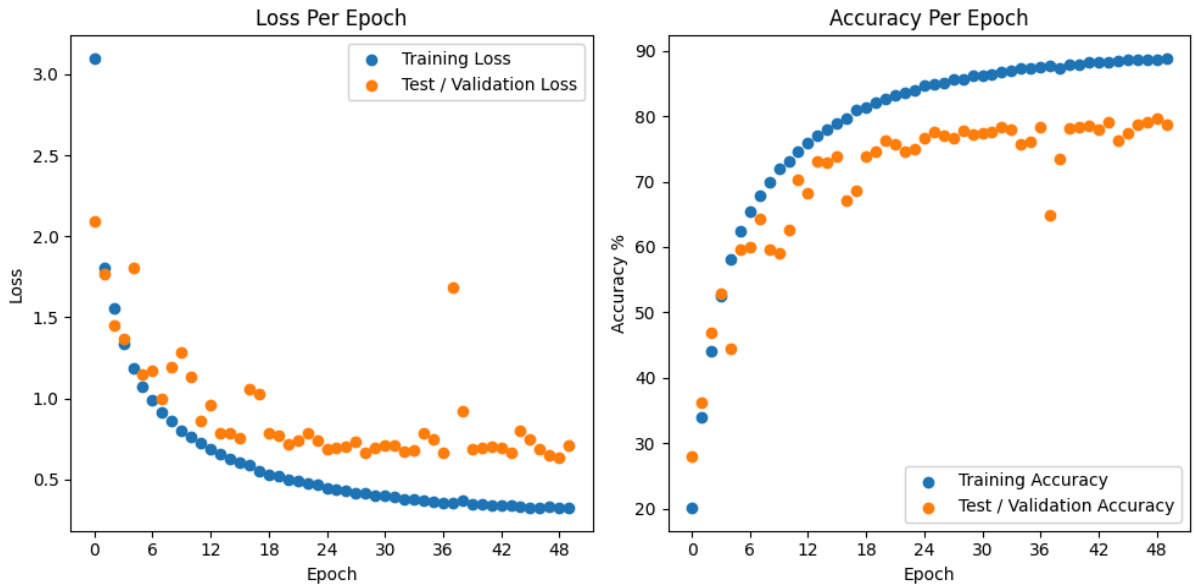
In general, loss remained about the same with vs without dropout for the test / validation set.

CIFAR-10 Vs. CIFAR-100:
CIFAR-10 was a lot easier to train in terms of both time, accuracy and loss. CIFAR-100 reduced the performance benefits of using convolution layers to the point where the results looked similar to not using any convolution layers for CIFAR-10 in the previous homework.

ResNet11 Vs. ResNet18:
Using the more complex ResNet18 did nothing but approximately double the time it took to train the exact same results. Overfitting did not improve. I think the reason is that the CIFAR dataset is too low resolution and both ResNet11 and ResNet18 are way too complex for the number of input features. If ResNet11 was already too complex, ResNet18 just made it worse meaning I had to train about double the number of parameters for a problem that didn't require that many parameters.

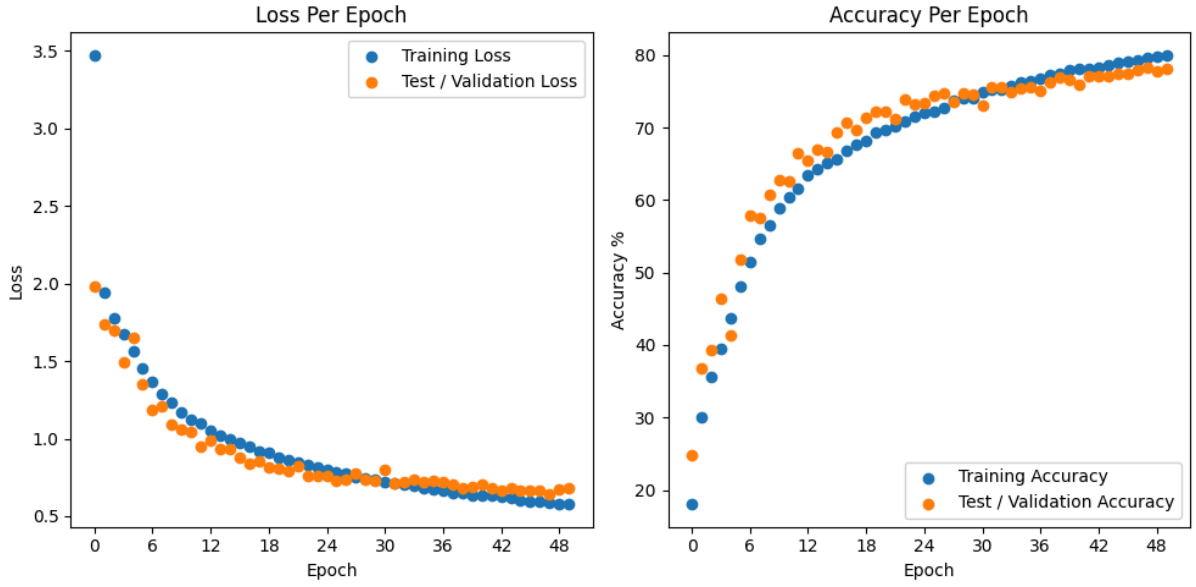
CIFAR-10 + ResNet11

Baseline:



Last Best Epoch: 48, Train Time ≈ 0 hr, 36 min
Train Loss: 0.32812, Train Accuracy: 88.68%
Test Loss: 0.63180, Test Accuracy: 79.60%

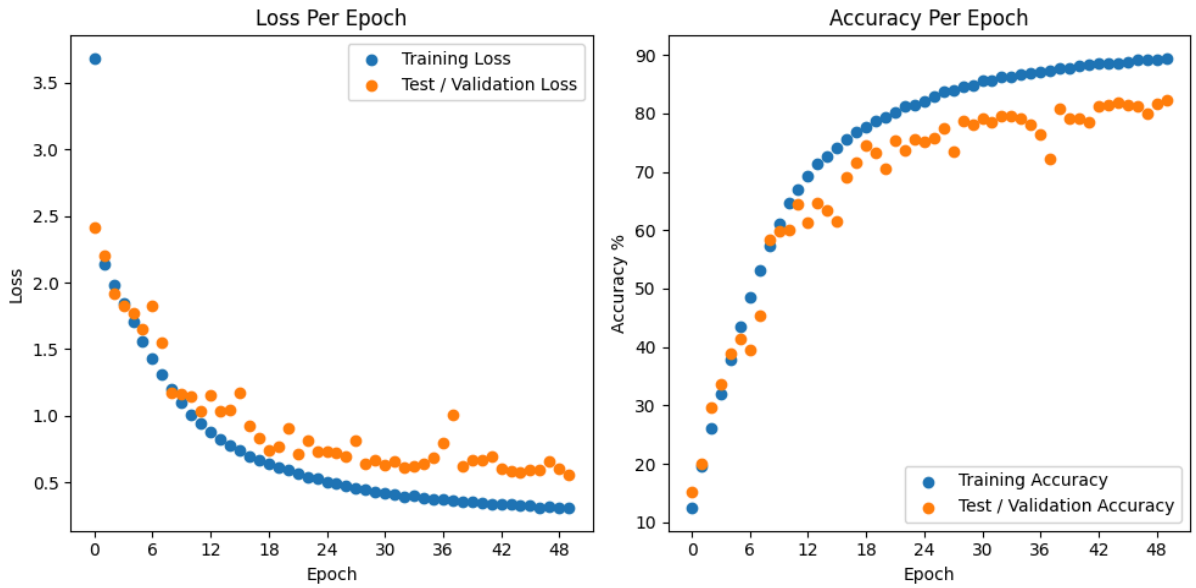
Dropout Enabled:



Last Best Epoch: 47, Train Time ≈ 0 hr, 35 min
Train Loss: 0.58785, Train Accuracy: 79.68%
Test Loss: 0.64036, Test Accuracy: 78.32%

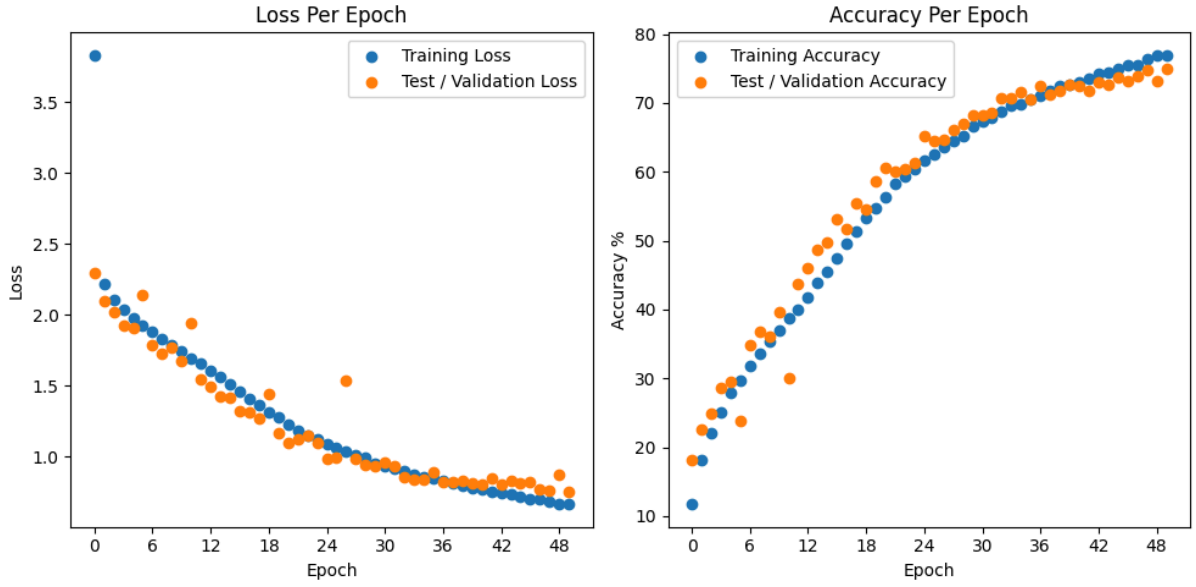
CIFAR-10 + ResNet18

Baseline:



Last Best Epoch: 49, Train Time ≈ 1 hr, 12 min
Train Loss: 0.30520, Train Accuracy: 89.44%
Test Loss: 0.55929, Test Accuracy: 82.18%

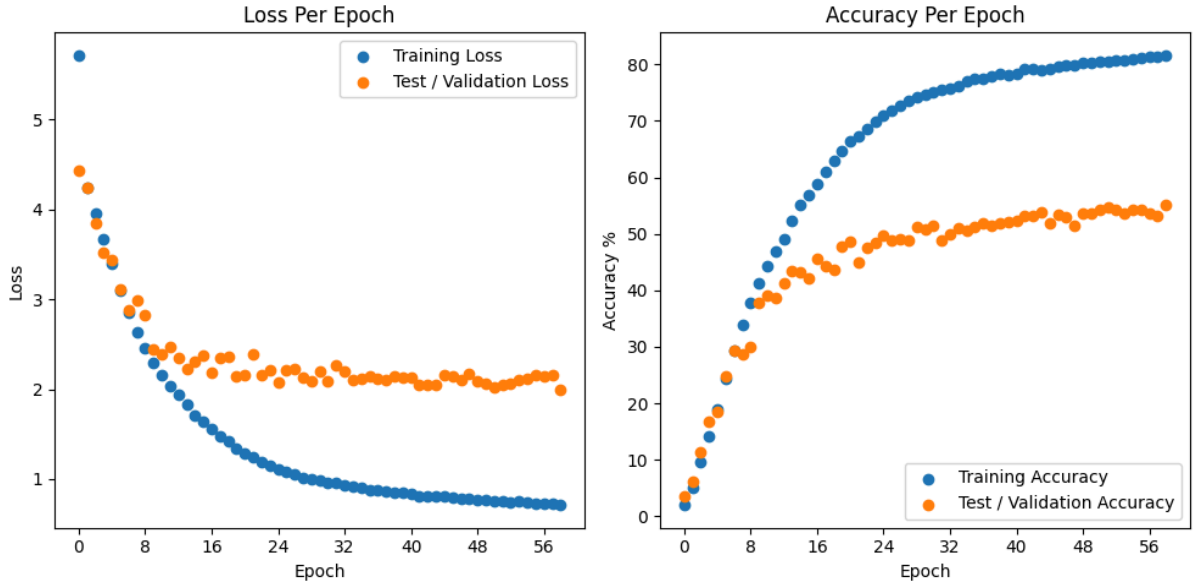
Dropout Enabled:



Last Best Epoch: 49, Train Time ≈ 1 hr, 12 min
Train Loss: 0.66167, Train Accuracy: 76.95%
Test Loss: 0.75462, Test Accuracy: 74.95%

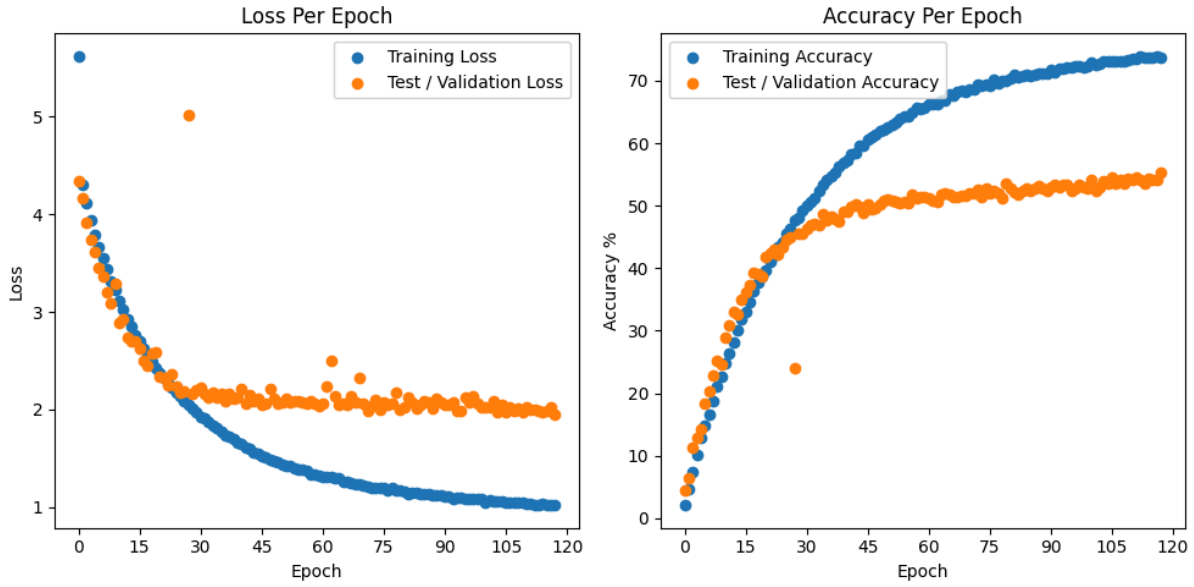
CIFAR-100 + ResNet11

Baseline:



Last Best Epoch: 58, Train Time ≈ 0 hr, 41 min
Train Loss: 0.70775, Train Accuracy: 81.64%
Test Loss: 1.99784, Test Accuracy: 55.18%

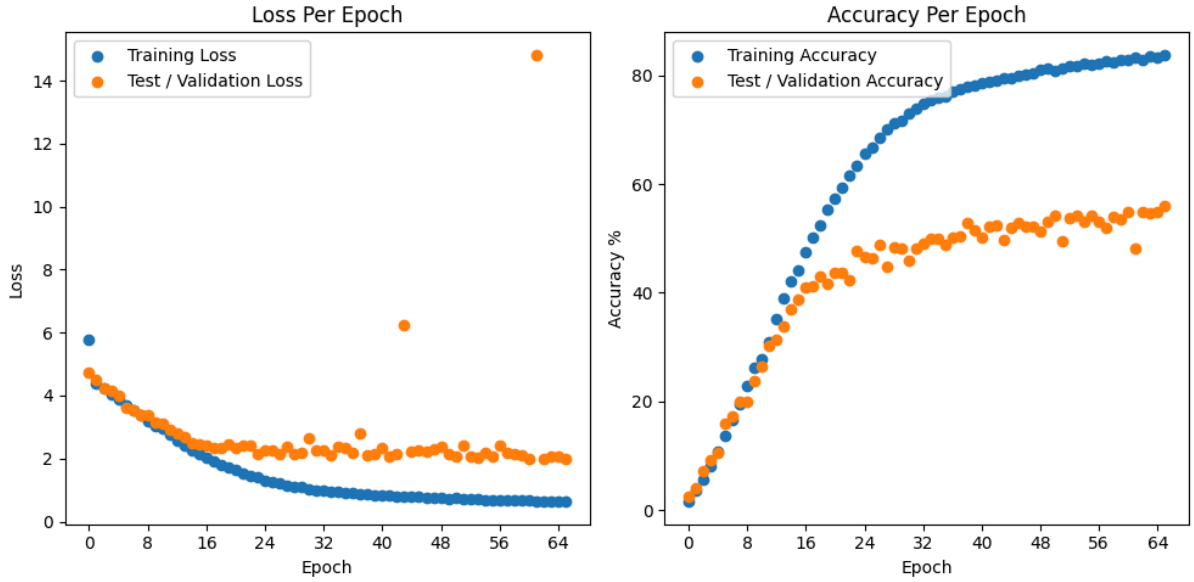
Dropout Enabled:



Last Best Epoch: 117, Train Time ≈ 1 hr, 34 min
Train Loss: 1.02294, Train Accuracy: 73.72%
Test Loss: 1.95096, Test Accuracy: 55.26%

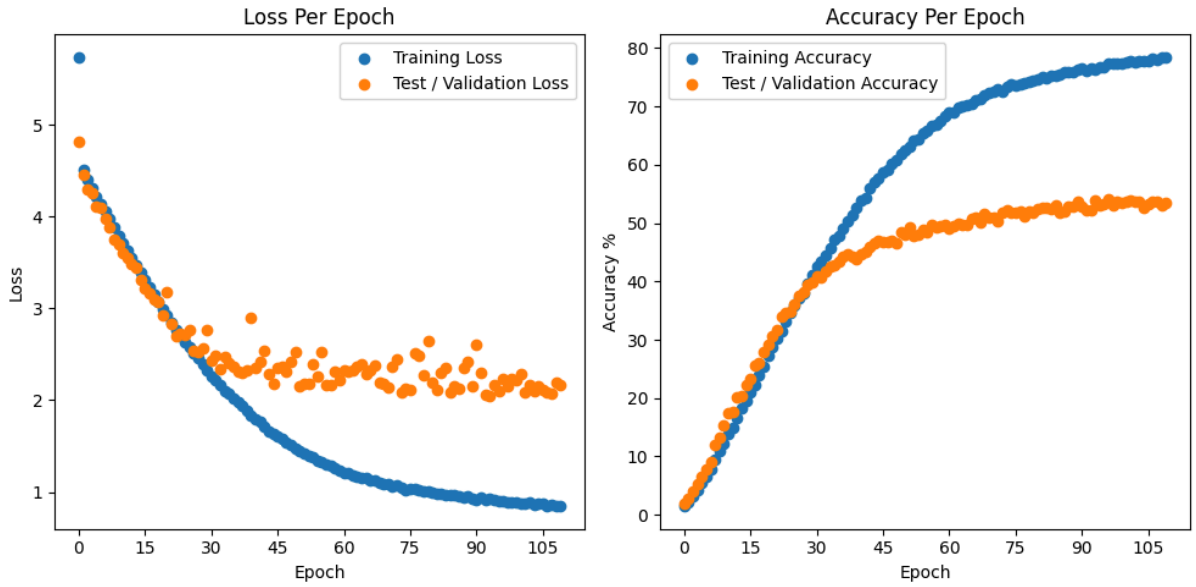
CIFAR-100 + ResNet18

Baseline:



Last Best Epoch: 65, Train Time ≈ 1 hr, 40 min
Train Loss: 0.62479, Train Accuracy: 83.80%
Test Loss: 1.97357, Test Accuracy: 55.92%

Dropout Enabled:



Last Best Epoch: 95, Train Time ≈ 2 hr, 20 min
Train Loss: 0.89489, Train Accuracy: 77.24%
Test Loss: 2.22619, Test Accuracy: 54.10%