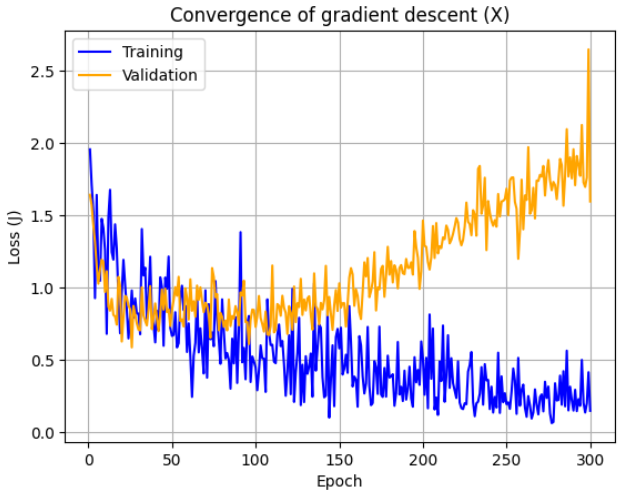
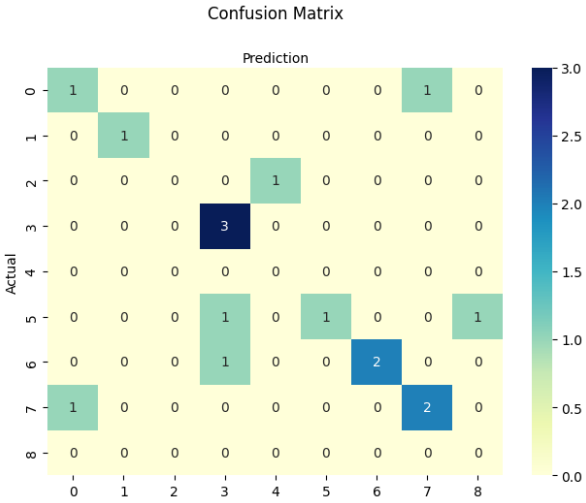
ECGR 5105 Homework 7

Owen Bailey-Waltz (801488178)
GitHub Link: https://github.com/obaileyw-uncc/ecgr5105/tree/main/hw07_cnns

Problem 1: CIFAR10 Image Classification

(a) Example neural network expanded to 10 features

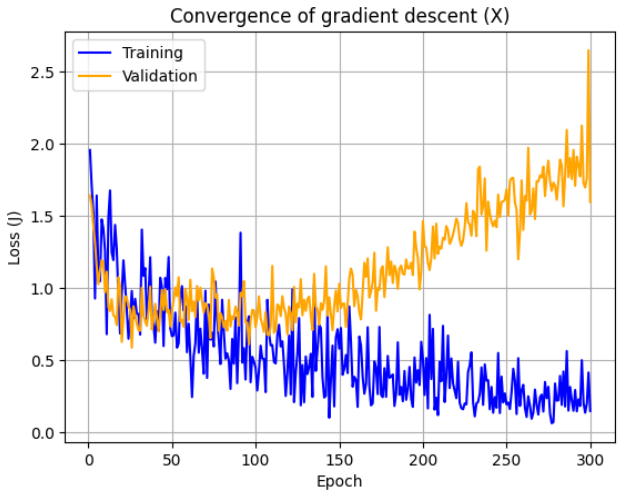


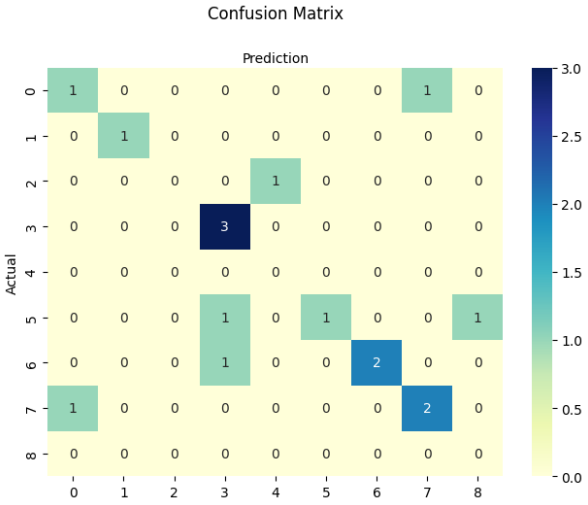


Accuracy 0.6136
Precision 0.5296
Recall 0.4630
F1 Score 0.4685

The shallow CNN classifier discussed in class extended to ten classes provides a much better fit than the fully connected neural network. The network still overfits after roughly fifty epochs, but the maximum accuracy is over 50% and occasionally nears 60% depending on the point in training. The minimum validation loss of 0.7332, indicating maximum generalization, occurs at epoch 49. Training time for the same network complexity decreased significantly due to the reduced number of input features – the training of this classifier took under an hour while the fully-connected neural network took longer.

(b) Deeper CNN classifier



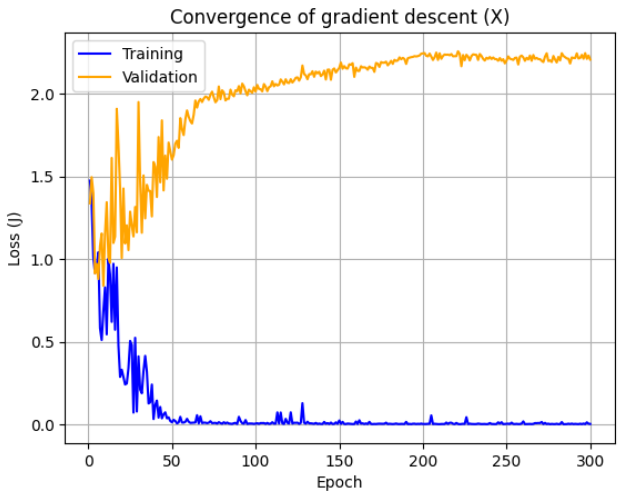


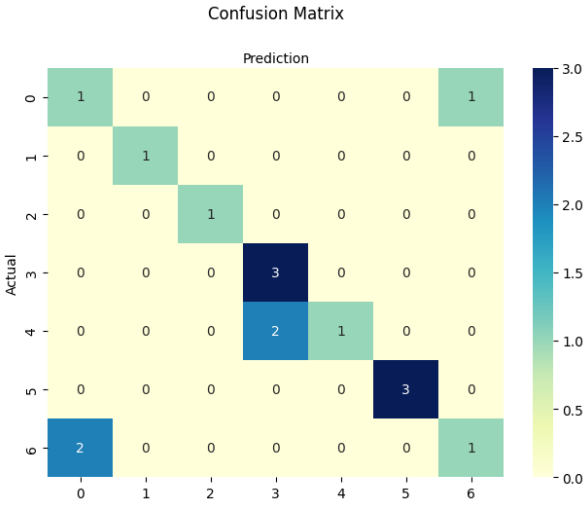
Accuracy 0.6893
Precision 0.6875
Recall 0.6250
F1 Score 0.6208

Adding an additional convolutional layer to the network reduces overfitting greatly. The network likely still requires a degree of precision that it has not yet attained to become more accurate as a classifier, but observing the training convergence reveals that, while the training process is much noisier and there are larger fluctuations in instantaneous loss than with the less complex network, the model does not overfit as much and validation loss remains constant around 1.0 rather than diverging as gradient descent fits closer with the training data. The ending F1 score after 300 epochs is 62.1%, which is approximately equal to the performance 124 epochs prior at Epoch 176, and the classifier's validation accuracy after training converges to values near 70%. All of these metrics show profound improvements compared to the shallower CNN classifier and the classifier which only uses a fully connected neural network. The additional layer of convolution further reduces input features, allowing another improvement to training time.

Problem 2: ResNet-10

(a) Basic ResNet-10



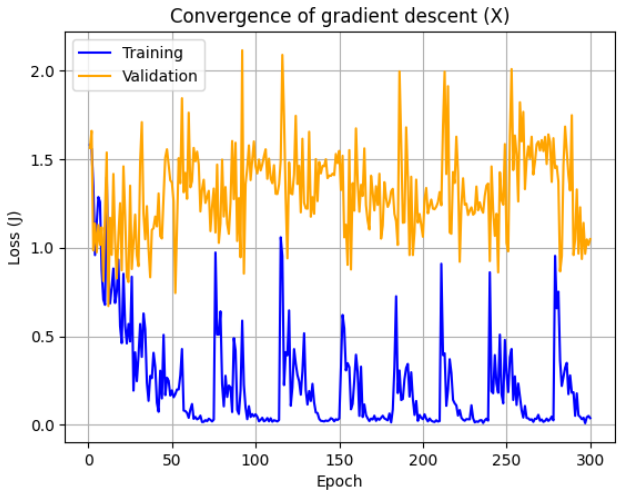


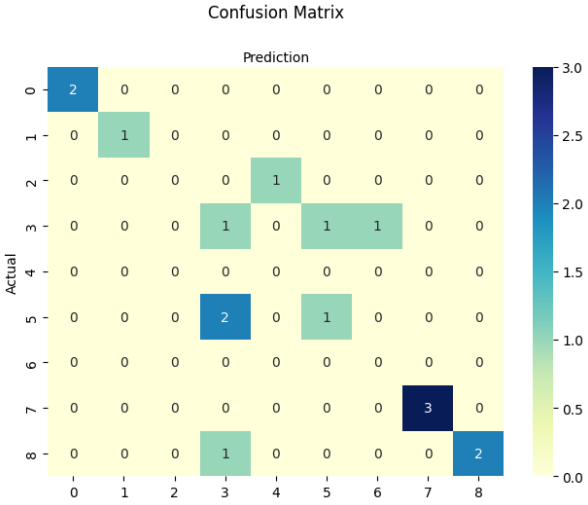
Accuracy 0.6553
Precision 0.7762
Recall 0.7381
F1 Score 0.7214

While the first attempt at a ResNet-based architecture does produce a large level of overfitting that is visible during training in the form of diverging losses, the fidelity of the classifier is nevertheless improved dramatically – with a much larger F1 score, precision and sensitivity than the previous example, ResNet's introduction of skip connections have made the model quicker-fitting than its counterparts. The level of the noise in the training from stochastic gradient descent also disappears as training progresses, hinting the model may be better able to learn as time goes on provided it does not overfit.

(b) ResNet-10 variations

Parameter penalties (weight decay, $\lambda = 0.001$)

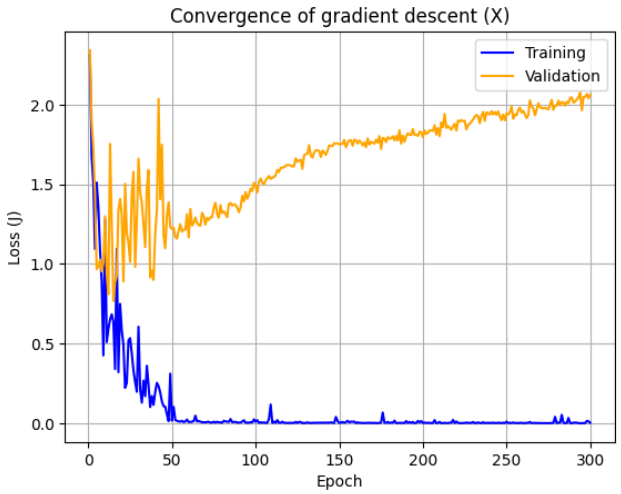


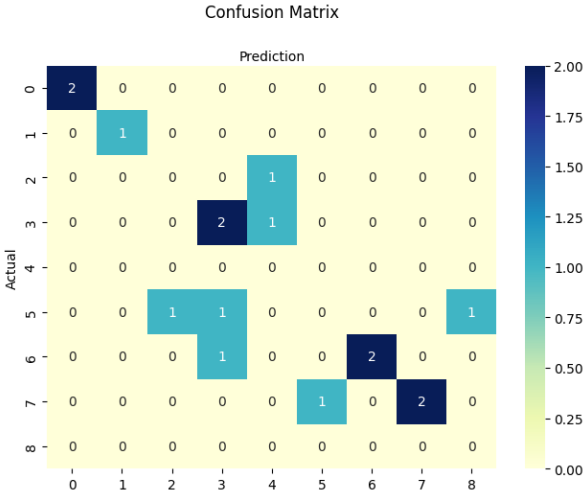


Accuracy 0.6835
Precision 0.5278
Recall 0.4815
F1 Score 0.4984

Adding parameter penalties produces a network which, while achieving an apparently better generalization initially, quickly begins to overfit and oscillate. The final result after 300 epochs of training does not perform well compared to the basic ResNet-10, which hints that, while this level of parameter penalty adequately addresses overfeeding, the level may be too high to get a high-quality fit.

Dropout

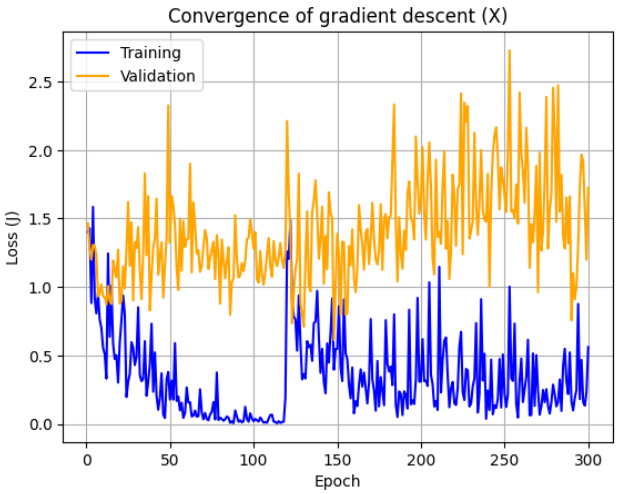


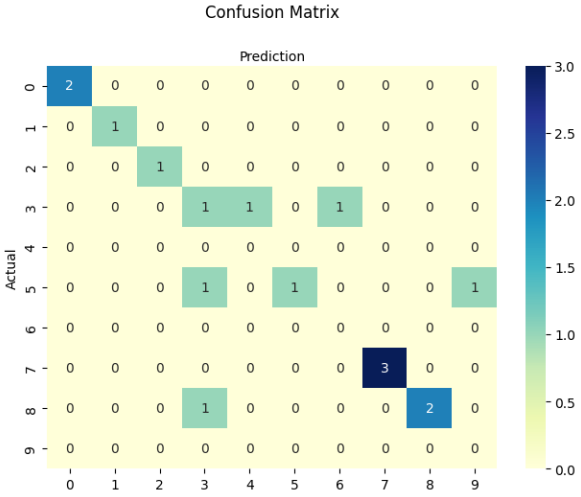


Accuracy 0.6714
Precision 0.5
Recall 0.4444
F1 Score 0.4634

While it does dramatically decrease the noise in training, introducing a dropout layer with p=0.3 to this network before the cascade of ResBlocks further degrades generalization performance for this example. This likely is a result of extreme overfitting, since, similar to the vanilla ResNet-10 example, the training loss in this example converged to a value of 0.0031 after 300 epochs of training and did not oscillate. Meanwhile, the accuracy reflects an improvement on the vanilla example even as generalization is impaired, indicating how dropout makes the model more certain in what it does know while compromising the impact of new data as training epochs increase.

Batch normalization





Accuracy 0.6594
Precision 0.6333
Recall 0.5333
F1 Score 0.5633

Introduction of batch normalization to the ResBlocks of the ResNet section creates an interesting behavior – this example converges to a very low level of training loss before it "bounces" off the bottom and becomes noisy again. This variation has trouble generalizing in the current configuration and also does not provide a good fit, however this technique appears to show promise if the overfitting issue that occurs early on in training can be addressed.