Task 1: One-Layer Softmax Regression

In this task we implement a neural network consisting of one layer of 10 softmax neurons. The model is described by the softmax regression formula Y = softmax(X * W + b) and the loss function used is cross entropy (also known as negative log likelihood).

Task 2: Feedforward Neural Network with Back-Propagation

1. What is the maximum accuracy that you can get in each setting for running your model with 10,000 iterations?
    - With 4 sigmoids feeding into a softmax layer, the maximum accuracy achieved is 0.86720002. For the ReLU network, it is 0.8818.
2. Is there a big difference between the convergence rate of the sigmoid and the ReLU? If yes, what is the reason for the difference?
    - The ReLU appears to converge slightly faster than the sigmoid. The reason for this is the reduced likelihood of vanishing gradients due to the ReLU's constant gradient. The sigmoid function's gradient, by contrast, becomes smaller as X increases.
3. What is the reason that we use the softmax in our output layer?
    - The softmax function outputs probabilities in the range [0, 1] that sum up to 1, making it a very good candidate for classification when the output layer consists of more than one (binary) output.
4. By zooming into the second half of the epochs in accuracy and loss plot, do you see any strange behaviour? What is the reason and how you can overcome them? (e.g., look at fluctuations or sudden loss increase after a period of decreasing loss).
    - It looks like the model becomes overfited

Task 3: Regularization and Tuning

1. Explain during grading the motivation behind learning rate decay.
    - Learning rate decay can help speed up the training by shortening the step size as the learned weights approach a local optimum. With a fixed alpha and noisy data, it's possible to oscillate around an optimum or even overshoot and diverge. By slowing down the learning over time, the optimizer may oscillate in tighter and tighter areas around the optimum.
2. Explain during grading why dropout can be an effective regularization technique.
    - Regularization can be seen as a sort of ensemble learning, where a large neural network can be divided into various smaller ones. Thus, it allows us to train a set of weaker classifiers and use an average of their responses. This works well because it's possible that each "dropout network" learns a slightly different aspect about the input data, contributing new insight.
3. For each of the programming tasks plot accuracy and loss, and analyze whether your additions influence the accuracy/loss and if yes, in what way.

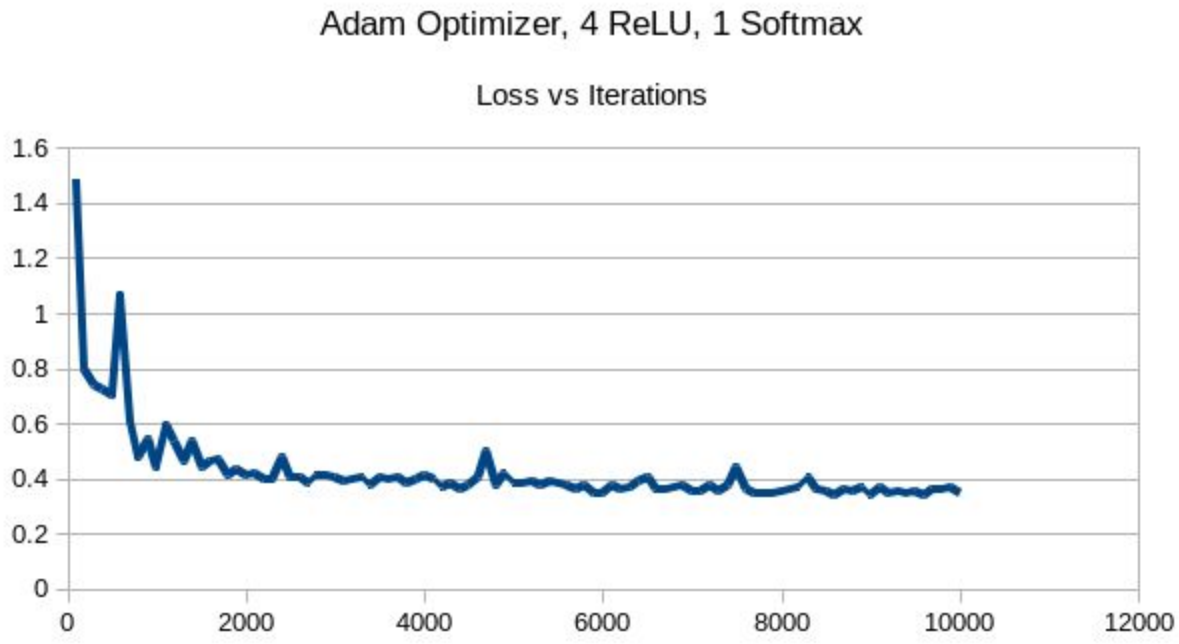
Task 4: Convolutional Neural Network

1. Define the output structure of the convolutional layers based on the given stride.
    -

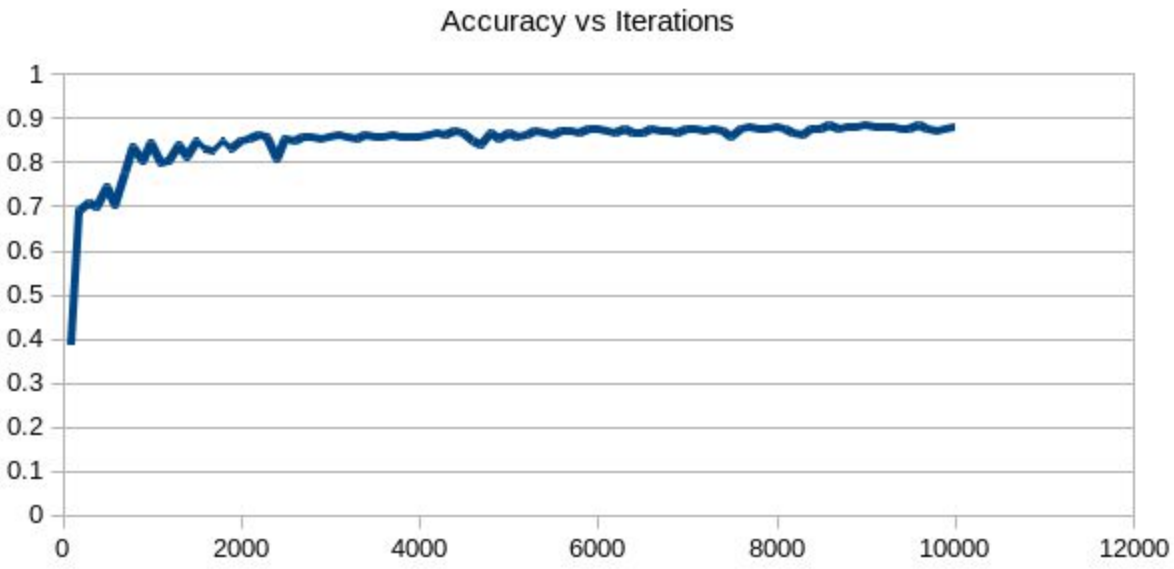
2. For each of the programming subtasks 2-4 point out the changes that happen to the accuracy and error and explain why your modifications caused those changes.

Task 5: Hyperparameter Optimization

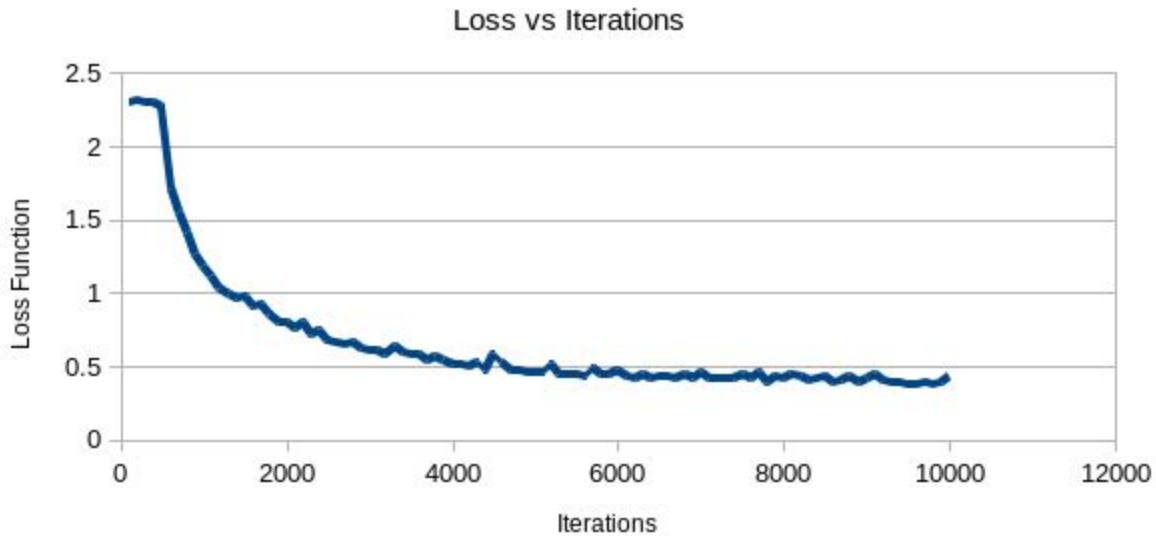
Graphs
Task 1
Task 2

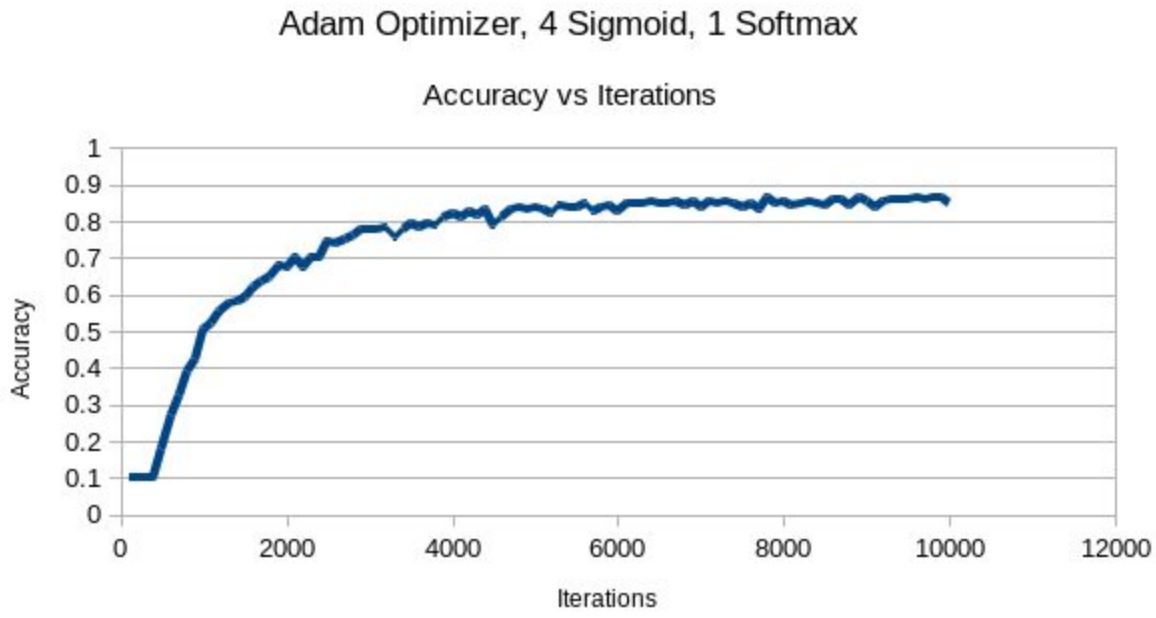


Adam Optimizer, 4 ReLU, 1 Softmax

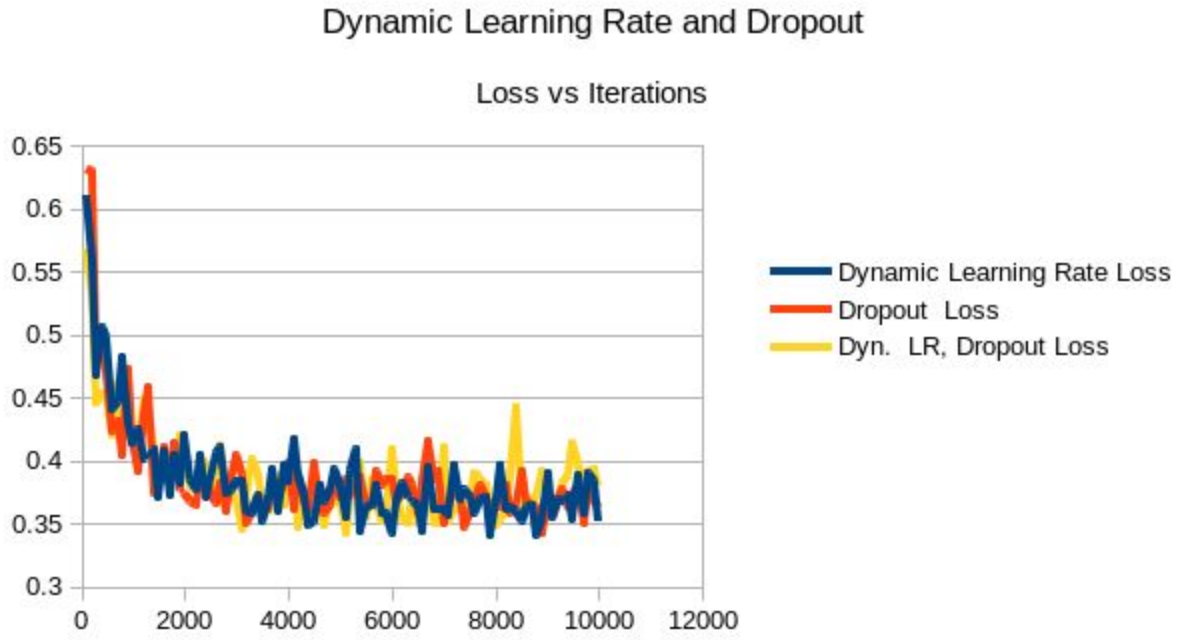


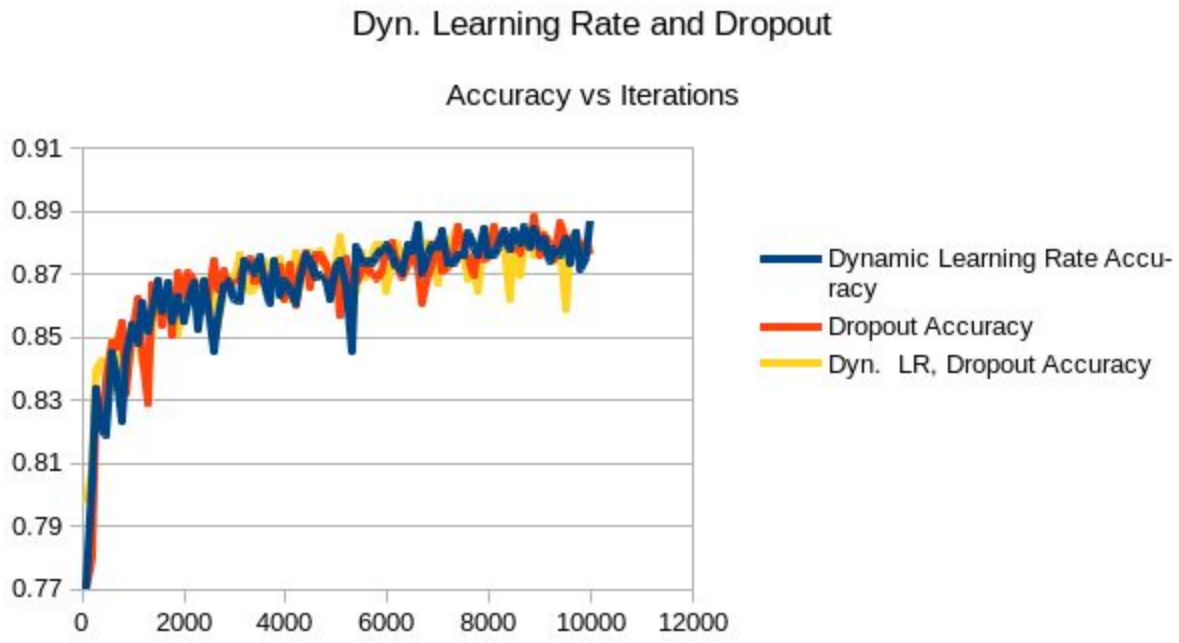
Adam Optimizer, 4 Sigmoid, 1 Softmax




Adam Optimizer, 4 Sigmoid, 1 Softmax
Accuracy vs Iterations
Accuracy
Iterations
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2000
4000
6000
8000
10000
12000

Task 3


Dynamic Learning Rate and Dropout
Loss vs Iterations
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0
2000
4000
6000
8000
10000
12000
Dynamic Learning Rate Loss
Dropout Loss
Dyn. LR, Dropout Loss


Dyn. Learning Rate and Dropout
Accuracy vs Iterations
0.91
0.89
0.87
0.85
0.83
0.81
0.79
0.77
0
2000
4000
6000
8000
10000
12000
Dynamic Learning Rate Accuracy
Dropout Accuracy
Dyn. LR, Dropout Accuracy

Task 4

Task 5