Name(s): Brian (Boquan) Yin, Bowen Li
NetID(s): boquany2, bowenli6
Team name on Kaggle leaderboard: Toubat Brian

For each of the sections below, your reported test accuracy should approximately match the accuracy reported on Kaggle.

Briefly describe the hyperparameter tuning strategies you used in this assignment. Then record your optimal hyperparameters and test/val performance for the four different network types.

In this assignment, hyperparameter search was performed under 5 different dimensions: number of hidden layers (2, 3), optimization strategy (SGD, Adam), regularization constant (0, 0.01, 0.05, 0.1, 0.2), hidden layer size (20, 32, 48, 64, 100), as well as learning rate (1e-5, 1e-4, 1e-3, 1e-2, 1e-1). Over 500 different models were trained for more than 24 hours in order to try all combinations of hyperparameters.

```
reg_consts = [0.0, 0.01, 0.05, 0.1, 0.2]
hidden_sizes = [20, 32, 48, 64, 100]
learning_rates = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1]

accuracies = {}

# Train the model for each combination of hyperparameters
for num_layers in [2, 3]:
    for opt in ["SGD", "Adam"]:
        for reg_const in reg_consts:
            for hidden_size in hidden_sizes:
                for learning_rate in learning_rates:
                    file_name = f'checkpoints/{num_layers}_{opt}_{reg_const}_{hidden_size}_{learning_rate}.npy'
                    if os.path.exists(file_name):
                        input_size = 28 * 28
                        num_classes = 10
                        hidden_sizes = [hidden_size] * (num_layers - 1)

                        # Initialize a new neural network model
                        net = NeuralNetwork(input_size, hidden_sizes, num_classes, num_layers)
                        net.load_checkpoint(file_name)
                    else:
                        train_loss, train_acc, val_acc, net = train(num_layers, hidden_size, learning_rate, reg_const, opt)
                        net.save_checkpoint(file_name)

                    val_acc, test_acc = get_acc(net)
                    accuracies[(num_layers, opt, reg_const, hidden_size, learning_rate)] = (val_acc, test_acc)
                    print(f'num_layers = {num_layers}, opt = {opt}, reg_const = {reg_const}, hidden_size = {hidden_size}, '
                        f'learning_rate = {learning_rate} | val_acc = {val_acc}, test_acc = {test_acc}')

print("Training complete...")
```

Two-layer Network Trained with SGD

Best hyperparameters (if you changed any of the other default hyperparameters like initialization method, etc. please note that as well):

Batch size:	200
Learning rate:	0.01

Hidden layer size:	100
Regularization coefficient:	0.2

Record the results for your best hyperparameter setting below:

Validation accuracy:	0.8946
Test accuracy:	0.8915

Three-layer Network Trained with SGD

Best hyperparameters (if you changed any of the other default hyperparameters like initialization method, etc. please note that as well):

Batch size:	200
Learning rate:	0.001
Hidden layer size:	100
Regularization coefficient:	0.1

Record the results for your best hyperparameter setting below:

Validation accuracy:	0.8920
Test accuracy:	0.8904

Two-layer Network Trained with Adam

Best hyperparameters (if you changed any of the other default hyperparameters like initialization method, etc. please note that as well):

Batch size:	200
Learning rate:	0.001
Hidden layer size:	100
Regularization coefficient:	0.2
β_1	0.9
β_2	0.999

Record the results for your best hyperparameter setting below:

Validation accuracy:	0.8959
Test accuracy:	0.8924

Three-layer Network Trained with Adam

Best hyperparameters (if you changed any of the other default hyperparameters like initialization method, etc. please note that as well):

Batch size:	200
Learning rate:	0.001
Hidden layer size:	100
Regularization coefficient:	0.2
β_1	0.9
β_2	0.999

Record the results for your best hyperparameter setting below:

Validation accuracy:	0.8927
Test accuracy:	0.8918

Comparison of SGD and Adam

Attach two plots, one of the training loss for each epoch and one of the validation accuracies for each epoch. Both plots should have a line for SGD and Adam. Be sure to add a title, axis labels, and a legend.

Compare the performance of SGD and Adam on training times and convergence rates. Do you notice any difference? Note any other interesting behavior you observed as well.

The following two models were trained under the same hyperparameters:

- Learning rate: 0.0001.
- Regularization constant: 0.2.
- The number of layers: 3.
- Hidden layer size: 32.

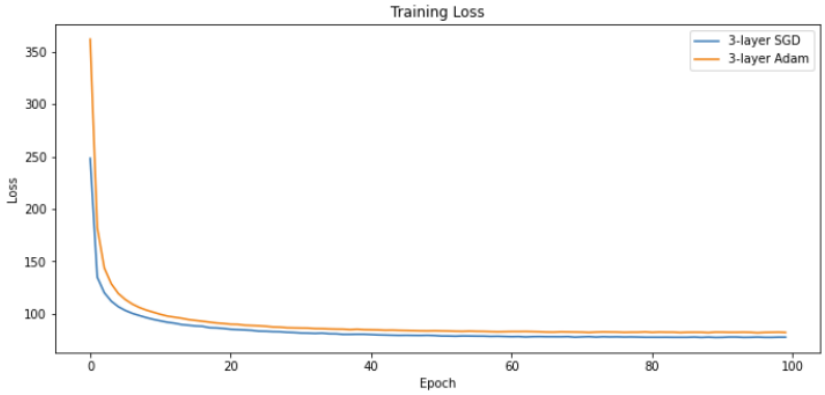

Figure 1. Training loss of SGD v.s. Adam

By looking at comparisons of training loss between SGD and Adam, we can see that their convergence rates are relatively the same, with SGD having slightly lower loss than Adam. For the initial epochs, Adam has a higher loss than SGD, which is probably caused by the random initialization of weight parameters. Adam also converges slightly slower than SGD, which can be seen at epochs 4 to 5.

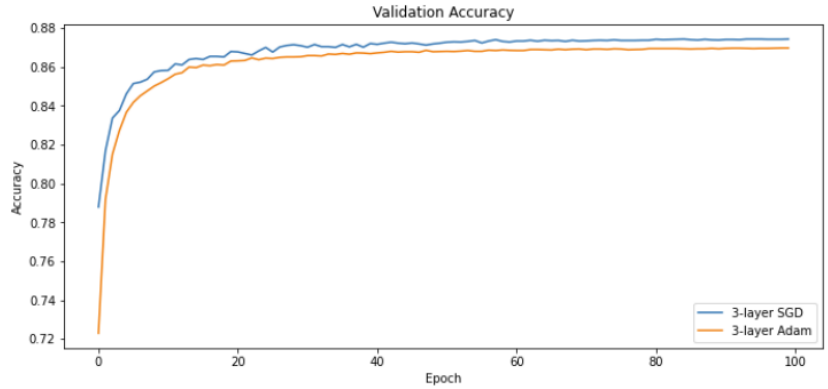

Figure 2. Validation accuracy of SGD v.s. Adam

For validation accuracy, SGD also performs slightly better than Adam. However, one interesting observation is that Adam has a more smooth accuracy curve than SGD, while SGD tends to have more noises. This is because Adam utilizes momentum techniques that keep track of the overall converging direction by combining history gradients, which would produce more stable gradient updates, whereas SGD updates parameters based on a mini-batch of data, so the direction of gradient would have more randomness than Adam optimizer.