Topics in Artificial Intelligence – Homework 4 Results

Experiment 1: How does the choice of loss function effect the training convergence rate of your model? How does it affect the final accuracy?

Experiment 2: How does the choice of kernel type affect the training of your model? How does it affect the final accuracy?

Experiment 3: How does the choice of optimizer affect the training of your model? How does it affect the final accuracy?

Experiment 4: How does the choice of topology affect the training of your model? How does it affect the final accuracy?

Experiment 1a:

Hypothesis: Training should converge faster i.e. to reach 100 percent accuracy, when we apply no regularization as compared to L2 and L1 regularization.

Experiment Description:

Fixing the training data size to 5000, we measured the accuracy of the model for each of the loss functions of cross, cross-l1, cross-l2, indicating the loss function should be cross entropy, cross entropy with L1 regularization, cross entropy with L2 regularization.

Commands:

Training Data: 5000

python suits_generator.py data 5000

python cnn_net.py train 5square 2 cross adam model_loss data_train

python cnn_net.py train 5square 2 cross-l1 adam model_l1 data_train

python cnn_net.py train 5square 2 cross-l2 adam model_l2 data_train

Training Data	Loss function	Training Accuracy
5000	cross	Epoch 10/10 157/157 [==============================] - 1s 9ms/step - loss: 0.0011 - acc: 1.0000 Training data size: 5000
5000	cross-l1	Epoch 10/10 157/157 [==============================] - 2s 11ms/step - loss: 1.4454 - acc: 0.2562 Training data size: 5000 Time taken to train: 15.246774911880493
5000	cross-2	Epoch 10/10 157/157 [==============================] - 1s 8ms/step - loss: 0.0971 - acc: 0.9986 Training data size: 5000 Time taken to train: 13.916002988815308

Inference: When using Cross-L1 loss function, the model's training accuracy does not converge, and stops at 25%. When using cross-L2, the model learns training data much better with training accuracy of 99.8%. When using loss function with no regularization at all, the training accuracy reaches is 100%.

Experiment 1b:

Hypothesis: L1 loss function yields better results in CNN on the test data when compared with L2 and no regularizations.

Experiment Description:

Fixing the testing data size to 1000, we measured the accuracy of the model for each of the loss functions of cross, cross-l1, cross-l2, and validated the accuracy on the test data of each of the trained models.

Commands:

python cnn_net.py test model_loss data_test

Test Accuracy cross:

```
Testing data size: 1000
Time taken to test: 0.2507646083831787
Accuracy of model: 99.7
Confusion Matrix:
 tf.Tensor(
[[214   0   0   0]
 [  0 297   1   0]
 [  0   1 255   0]
 [  0   1   0 231]], shape=(4, 4), dtype=int32)
```

Commands:

python cnn_net.py test model_l1 data_test

Test Accuracy cross-l1:

```
Testing data size: 1000
Time taken to test: 0.25999975204467773
Accuracy of model: 23.200000000000003
Confusion Matrix:
 tf.Tensor(
[[  0   0   0 214]
 [  0   0   0 298]
 [  0   0   0 256]
 [  0   0   0 232]], shape=(4, 4), dtype=int32)
```

Commands:

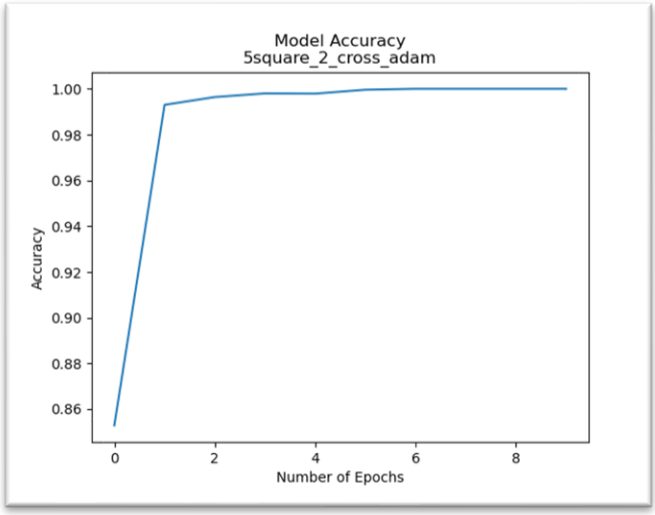
python cnn_net.py test model_l2 data_test

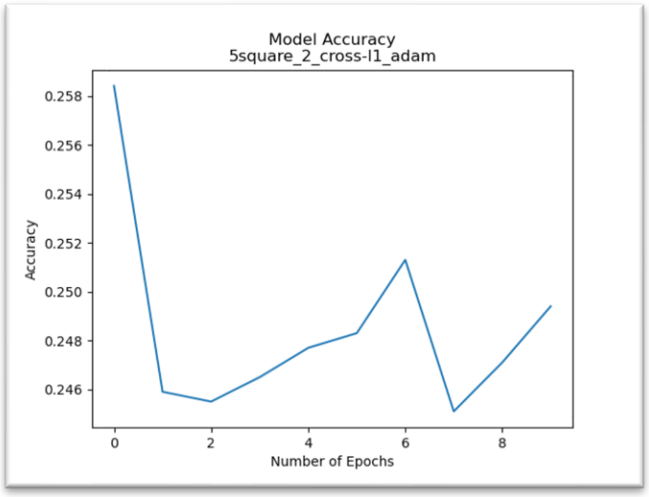
Test Accuracy:

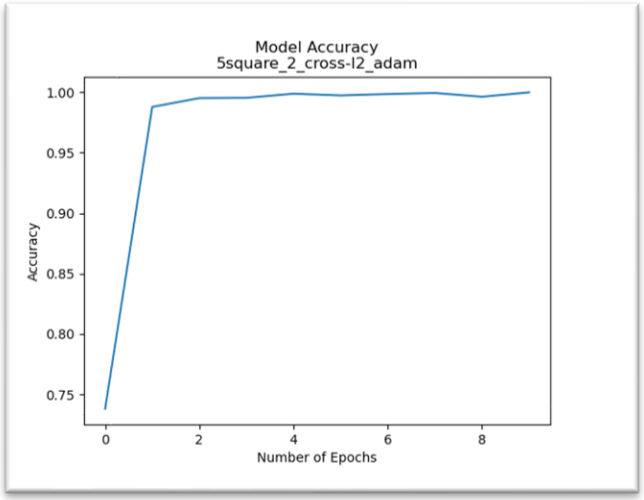
```
Testing data size: 1000
Time taken to test: 0.4280121326446533
Accuracy of model: 99.6
Confusion Matrix:
 tf.Tensor(
[[213   0   1   0]
 [  0 298   0   0]
 [  0   1 255   0]
 [  1   1   0 230]], shape=(4, 4), dtype=int32)
```

Inference: When using L1 loss function, the training accuracy is bad (~23%), since the model did not train properly on train data. The model using cross-L2 gives better test accuracy (~99.7%). The model with no regularization also produces similar results.

Effect of Loss function:







Experiment 2a:

Hypothesis: Kernel type 5square yields better train accuracy than 5diamond.

Experiment Description: We create 2 training models, one using 5x5 square kernel and one using 5x5 diamond kernel on train data of sample size 5000. We then compare their training data results to check the effect of kernel type on the LeNET.

Commands:

Training Data: 5000

python suits_generator.py data 5000

python cnn_net.py train 5diamond 2 cross-l2 adam model_5diamond data_train

python cnn_net.py train 5square 2 cross-l2 adam model_5square data_train

Training Data	kernel_ type	Training Accuracy
5000	**5diamond**	Epoch 10/10 157/157 [==============================] - 3s 16ms/step - loss: 0.0800 - acc: 0.9984 Training data size: 5000 Time taken to train: 24.710729360580444
5000	**5square**	Epoch 10/10 157/157 [==============================] - 2s 13ms/step - loss: 0.1046 - acc: 0.9980 Training data size: 5000 Time taken to train: 21.519373178482056

Inference: When using 5x5 square kernel, we observed better convergence on training data when compared to the diamond kernel. This could be because the shape diamond kernel may work well only for diamond class, but not for the rest of the classes in general.

Experiment 2b:

Hypothesis: Kernel type 5square yields better test accuracy than 5diamond

Experiment Description: We test the 2 models created using 5x5 square kernel and 5x5 diamond kernel on test data of sample size 1000. We then compare their test data results to check the effect of kernel type on the LeNET.

Commands:

python cnn_net.py test model_5diamond data_test

Test Accuracy 5diamond:

```
Testing data size: 1000
Time taken to test: 0.2810842990875244
Accuracy of model: 99.6
Confusion Matrix:
 tf.Tensor(
[[214   0   0   0]
 [  0 297   1   0]
 [  0   0 256   0]
 [  3   0   0 229]], shape=(4, 4), dtype=int32)
```

Commands:

python cnn_net.py test model_5sqaure data_test

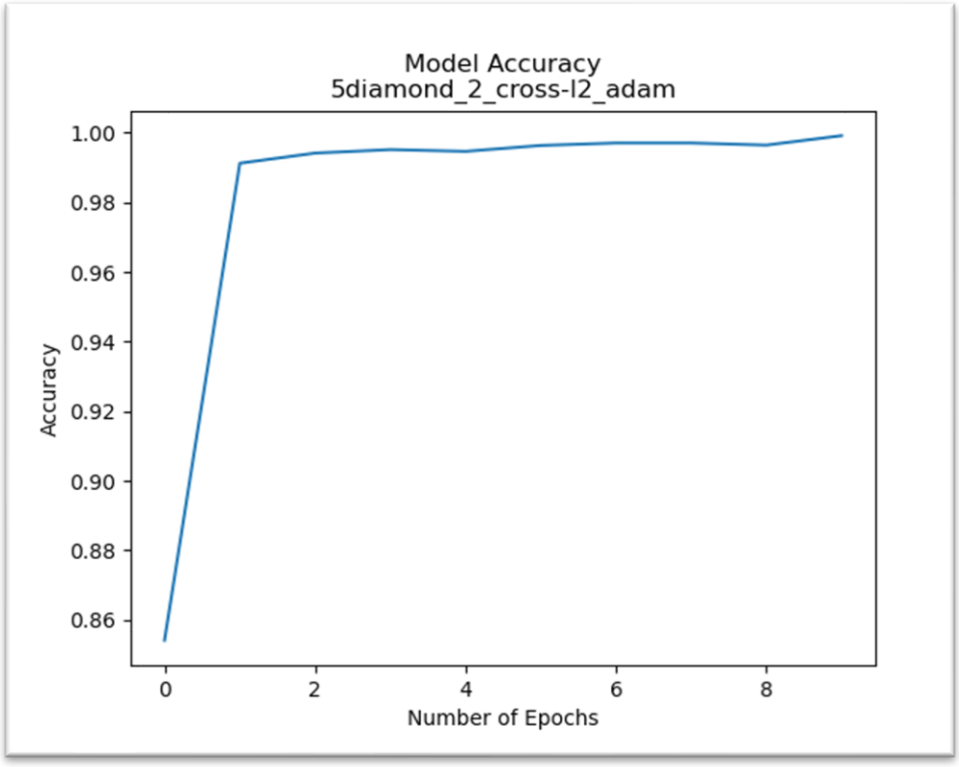
Test Accuracy 5square:

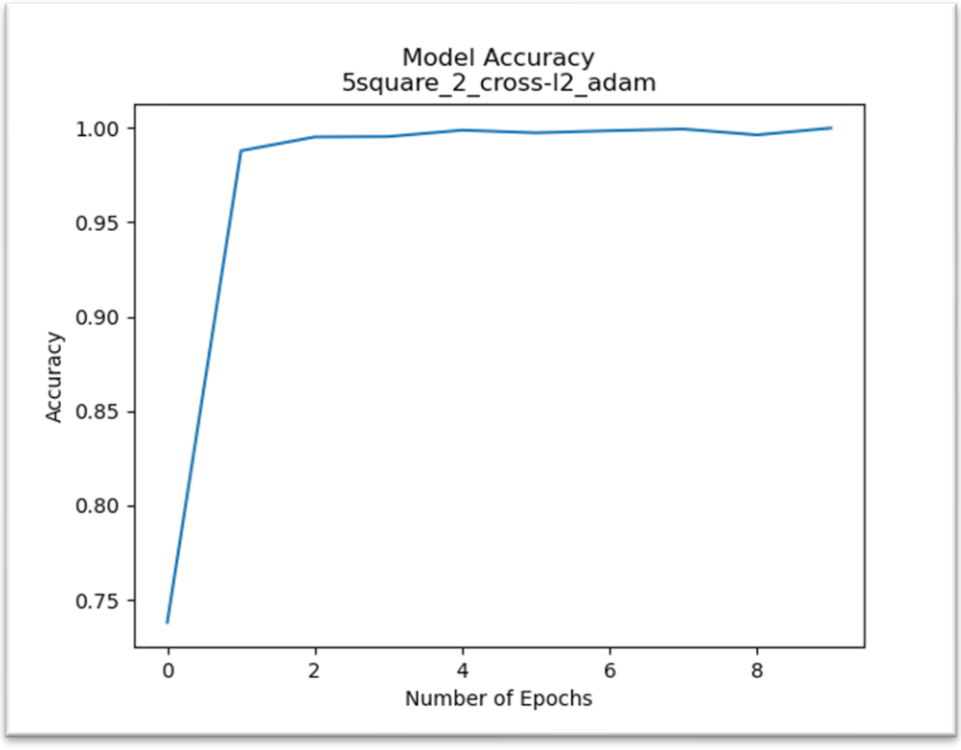
```
Testing data size: 1000
Time taken to test: 0.24073171615600586
Accuracy of model: 99.6
Confusion Matrix:
 tf.Tensor(
[[214   0   0   0]
 [  0 297   1   0]
 [  0   0 256   0]
 [  3   0   0 229]], shape=(4, 4), dtype=int32)
```

Inference:

When using 5x5 square kernel, we observed better accuracies on test data when compared to the diamond kernel. This could be because the shape diamond kernel may have trained well only on diamond class, but not for the rest of the classes in general.

Effect of Kernel Type:





Experiment 3a:

Hypothesis: Using adam optimizer yields better training convergence than using sgd optimizer

Experiment Description: We create 2 training models, one using adam optimizer and one using sgd optimizer on train data of sample size 5000. We then compare their training data results to check the effect of optimizer on the LeNET.

Commands:

Training Data: 5000

python suits_generator.py data 5000

python cnn_net.py train 5square 2 cross-l2 sgd model_sgd data_train

python cnn_net.py train 5square 2 cross-l2 adam model_adam data_train

Training Data	optimizer	Training Accuracy
5000	sgd	Epoch 10/10 157/157 [==============================] - 3s 16ms/step - loss: 0.5396 - acc: 0.9422 Training data size: 5000 Time taken to train: 24.250744581222534
5000	adam	Epoch 10/10 157/157 [==============================] - 3s 22ms/step - loss: 0.1019 - acc: 0.9988 Training data size: 5000 Time taken to train: 28.253603219985962

Inference: When using adam optimizer, we observed better convergence on training data when compared to the sgd optimizer.

Experiment 3b:

Hypothesis: Kernel type adam optimizer yields better test accuracy than sgd optimizer.

Experiment Description: We test the 2 models trained using adam optimizer and sgd optimizer on test data of sample size 1000. We then compare their test data results to check the effect of optimizers on the LeNET.

Commands:

python cnn_net.py test model_sgd data_test

Test Accuracy sgd:

```
Testing data size: 1000
Time taken to test: 0.2461872100830078
Accuracy of model: 96.1
Confusion Matrix:
 tf.Tensor(
[[200   2   2  10]
 [  0 282  14   2]
 [  0   7 249   0]
 [  1   1   0 230]], shape=(4, 4), dtype=int32)
```

Commands:

python cnn_net.py test model_adam data_test

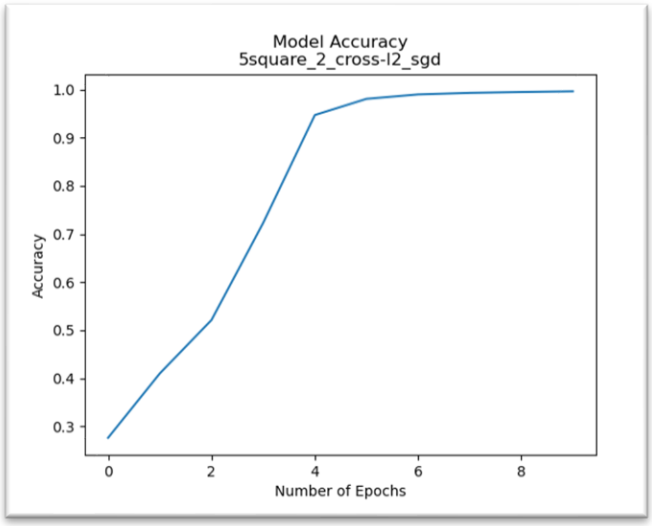
Test Accuracy adam:

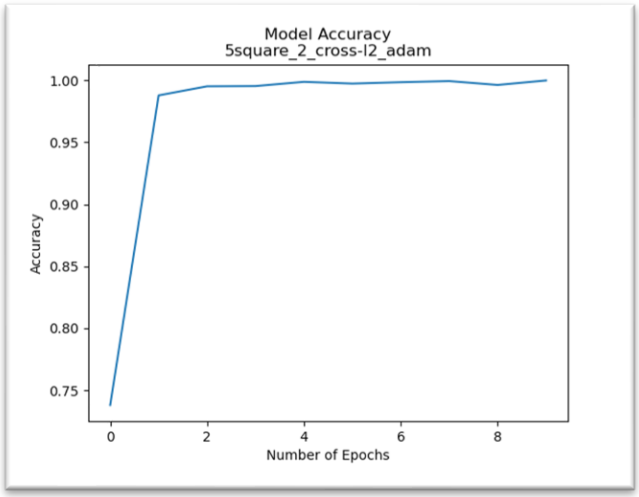
```
Testing data size: 1000
Time taken to test: 0.3072052001953125
Accuracy of model: 99.7
Confusion Matrix:
 tf.Tensor(
[[214   0   0   0]
 [  0 298   0   0]
 [  0   1 255   0]
 [  2   0   0 230]], shape=(4, 4), dtype=int32)
```

Inference:

When using adam optimizer, we observed better accuracies on test data when compared to the sgd optimizer.

Effect of Optimizer:





Experiment 4a:

Hypothesis: Topology 1 yields better train accuracy than topology 2.

Experiment Description: We create 2 training models, one using Topology 1 and one using Topology 2 on train data of sample size 5000. We then compare their training data results to check the effect of skip connections on the LeNET training.

Commands:

Training Data: 5000

python suits_generator.py data 5000

python cnn_net.py train 5square 1 cross-l2 adam model_top1 data_train

python cnn_net.py train 5square 1 cross-l2 adam model_top2 data_train

Training Data	Topology	Training Accuracy
5000	**1**	Epoch 10/10 157/157 [==============================] - 2s 15ms/step - loss: 0.1177 - acc: 0.9962 Training data size: 5000 Time taken to train: 24.399667024612427
5000	**2**	Epoch 10/10 157/157 [==============================] - 2s 14ms/step - loss: 0.1343 - acc: 0.9950 Training data size: 5000 Time taken to train: 24.041831731796265

Inference: When using Topology 1 and 2, we observed similar convergence on training data.

Experiment 4b:

Hypothesis: Topology 2 with skip connections yields better test accuracy than Topology 1

Experiment Description: We test the 2 models created using Topology 1 and 2 on test data of sample size 1000. We then compare their test data results to check the effect of skip connections on the LeNET final accuracy.

Commands:

python cnn_net.py test model_top1 data_test

Test Accuracy Topology 1:

```
Testing data size: 1000
Time taken to test: 0.24326229095458984
Accuracy of model: 98.4
Confusion Matrix:
 tf.Tensor(
[[214   0   0   0]
 [  1 288   9   0]
 [  0   1 255   0]
 [  5   0   0 227]], shape=(4, 4), dtype=int32)
```

Commands:

python cnn_net.py test model_top2 data_test

Test Accuracy Topology 2:

```
Testing data size: 1000
Time taken to test: 0.23997950553894043
Accuracy of model: 99.6
Confusion Matrix:
 tf.Tensor(
[[214   0   0   0]
 [  0 296   0   2]
 [  0   1 255   0]
 [  1   0   0 231]], shape=(4, 4), dtype=int32)
```

Inference:

When using Topology 2, we observed better accuracies on test data when compared to the one using Topology 1. This could be because of the slightly better training of the neural net with skip connections because of the availability of alternative paths which may be beneficial for the model convergence.

Effect of Topology: