CS 354 Lab 1 Report

Anurag Shah
shah447@purdue.edu

March 2021

1 Github Repository

https://github.com/Anurag-Shah/CS390NIPLab2

2 Resources Used

Lecture Slides
https://keras.io/api/datasets/
https://keras.io/api/layers/convolution_layers/convolution2d/
https://keras.io/api/layers/pooling_layers/max_pooling2d/
https://keras.io/api/layers/normalization_layers/batch_normalization/
https://keras.io/api/optimizers/
https://keras.io/api/optimizers/learning_rate_schedules/exponential_decay/
https://keras.io/api/optimizers/learning_rate_schedules/polynomial_decay/
https://keras.io/api/layers/regularizers/
https://keras.io/api/layers/preprocessing_layers/image_preprocessing/random_crop/

3 Parts completed

1. Neural Net Models
   (a) ANN, different from Lab 1
   (b) CNN
2. Accuracy for CNN:
   (a) MNIST digit accuracy of 99.22%
   (b) MNIST fashion accuracy of 92.99%
   (c) CIFAR 10 accuracy of 74.52%
   (d) CIFAR 100 Coarse accuracy of 54.06%
   (e) CIFAR 100 Fine accuracy of 41.22% (over 40% for extra credit)
3. Pipeline and Misc:
   (a) Code the pipeline to be able to use CIFAR-10 and CIFAR-100
   (b) Generated bar plots for accuracy over the dataset for the ANN and CNN
   (c) Added the option to preprocess the image with random crops

(d) Added the option to save and load network weights for both networks. I have included the weights the networks were trained with, both with and without random cropping, in the repository. There is a global variable SAVE_LOAD_WEIGHTS that is set to true, if you want to test the network without saved weights set it to false and the network will not load weights.
For some reason I was not able to figure out, saving the model leads to a slightly different accuracy than not having saving on. Its about the same and still meets the accuracy metrics, but the outputs provided below are from when saving is off on my local computer.

4 Questions

4.1 How is a CNN superior to an ANN for image processing

CNNs can use the 2 dimensional property of an image, rather than treating it as a 1d array. Learning filters to apply over various parts of the image is better at identifying patterns within the image than a single array, as it can identify patterns in multiple dimensions

4.2 Why do we sometimes use pooling in CNNs?

Pooling is a way to reduce the dimensionality, which has many advantages, like a faster computation time and less overfitting.

4.3 Why do you think the cifar datasets are harder than mnist?

There are a few reasons. First, cifar datasets have multiple channels for various colors. Second, cifar images are slightly larger than mnist images. Finally, for cifar 100, there are a lot more classes to learn as compared to mnist.

5 Increasing the accuracy of my CNN

I started out with a simple architecture to test if it works, just 2 convolution layers, a pooling layer, and a few dense layers, and it already performed really well from the start, nearly making all the accuracy requirements as it was. The big problem I noted was a ton of overfitting, which I tried to mitigate by adding dropout and batch normalization. I added a very slight decay to the learning rate, which increased the accuracy pretty significantly over a lot of epochs. I also tried adding l1 and l2 regularization to the layers, but that ended up increasing overfitting, no matter how large or small I made λ. In the end, I added a second dense layer, increased the dropout, and got to the final model.

6 Hyperparameters

Learning rate: 0.001, with an exponential decay of 0.98 every 9,000 steps
Dropout rate: 40%
Architecture:
3x3 convolution, 32 filters $\rightarrow$ 3x3 convolution, 64 filters $\rightarrow$ 3x3 Max Pooling $\rightarrow$ Dense layer, 512 Neurons $\rightarrow$ Dense layer, 256 Neurons $\rightarrow$ Output Layer
Minibatch size: 128
Epochs: 30
Split for validation set: 15%
Random crop size: 25x25 for MNIST, 29x29 for CIFAR (Disabled by default)

7 Outputs

Note: no confusion matrix for CIFAR 100 dataset, as it did not fit in the terminal window without truncation.

7.1 ANN Outputs (No random cropping)

7.1.1 MNIST Digits

```
Testing TF_NN.
Classifier algorithm: tf_net
Dataset: mnist_d
Classifier accuracy: 98.070000%

Confusion Matrix:
[[ 974    1    1    0    0    1    1    1    1    0]
 [   0 1128    2    0    0    1    2    0    2    0]
 [   3    2 1012    1    2    0    2    7    3    0]
 [   1    0    2  989    0    6    0    7    2    3]
 [   1    0    3    0  959    0    5    3    0   11]
 [   2    0    0    4    1  876    5    2    1    1]
 [   4    3    1    0    2    3  944    0    1    0]
 [   2    4   10    1    0    0    0 1010    0    1]
 [   7    0    1    8    3    6    3    4  937    5]
 [   2    3    0    3    9    5    0    9    0  978]]
F1 Score Matrix
[0.986 0.991 0.981 0.981 0.98  0.979 0.983 0.975 0.976 0.974]

Time Elapsed: 285.9 seconds
```

7.1.2 MNIST Fashion

```
Testing TF_NN.
Classifier algorithm: tf_net
Dataset: mnist_f
Classifier accuracy: 89.110000%

Confusion Matrix:
[[861   1  21  26   3   1  80   0   7   0]
 [  1 973   2  18   4   0   2   0   0   0]
 [ 14   0 849  13  76   0  47   0   1   0]
 [ 18  10  10 899  32   1  27   0   3   0]
 [  0   1 116  26 805   0  52   0   0   0]
 [  0   0   0   1   0 972   0  14   1  12]
 [143   2 102  24  57   0 664   0   8   0]
 [  0   0   0   0   0  23   0 958   0  19]
 [  2   0   5   7   4   1   0   4 977   0]
 [  0   0   0   0   0   7   1  39   0 953]]
F1 Score Matrix
[0.845 0.979 0.807 0.893 0.813 0.97  0.709 0.951 0.978 0.961]

Time Elapsed: 246.4 seconds
```

7.1.3 CIFAR 10

```
Testing TF_NN.
Classifier algorithm: tf_net
Dataset: cifar_10
Classifier accuracy: 38.300000%

Confusion Matrix:
[[435  52  80  30  36  12  36 100 183  36]
 [ 33 499  13  50  19  25  56  30 112 163]
 [109  34 170  33 284  77 162  93  29   9]
 [ 28  48 101 135 110 251 183  83  23  38]
 [ 55  22 104  32 411  60 191  87  24  14]
 [ 21  20  76 106 148 345 172  68  33  11]
 [  6  24  66  54 202  83 506  37   7  15]
 [ 37  35  72  55 196  55  76 418  20  36]
 [143 112  23  33  13  45  14  32 512  73]
 [ 52 197   3  54  13  13  51  76 142 399]]
F1 Score Matrix
[0.453 0.488 0.199 0.171 0.338 0.351 0.414 0.413 0.491 0.445]

Time Elapsed: 226.1 seconds
```

7.1.4 CIFAR 100 Coarse

```
Testing TF_NN.
Classifier algorithm: tf_net
Dataset: cifar_100_c
Classifier accuracy: 22.930000%


Time Elapsed: 194.2 seconds
```

7.1.5 CIFAR 100 Fine

```
Testing TF_NN.
Classifier algorithm: tf_net
Dataset: cifar_100_f
Classifier accuracy: 11.990000%


Time Elapsed: 217.9 seconds
```

7.2 CNN Outputs (No random cropping)

7.2.1 MNIST Digits

```
Testing TF_CNN.
Classifier algorithm: tf_conv
Dataset: mnist_d
Classifier accuracy: 99.220000%

Confusion Matrix:
[[ 976    0    0    0    0    0    2    0    1    1]
 [   0 1131    3    0    0    0    0    1    0    0]
 [   2    0 1025    0    1    0    0    4    0    0]
 [   0    0    0 1006    0    3    0    0    1    0]
 [   0    0    0    0  973    0    2    0    0    7]
 [   1    0    1    4    0  881    2    0    3    0]
 [   3    3    0    0    0    2  949    0    1    0]
 [   0    1    7    0    0    0    0 1019    1    0]
 [   1    0    2    0    0    0    0    1  968    2]
 [   0    0    0    0    8    2    0    3    2  994]]
F1 Score Matrix
[0.994 0.996 0.99  0.996 0.991 0.99  0.992 0.991 0.992 0.988]

Time Elapsed: 87.8 seconds
```

7.2.2 MNIST Fashion

```
Testing TF_CNN.
Classifier algorithm: tf_conv
Dataset: mnist_f
Classifier accuracy: 92.990000%

Confusion Matrix:
[[904   0  14   7   2   0  70   0   3   0]
 [  1 981   1  12   2   0   1   0   2   0]
 [ 19   0 916   5  30   0  30   0   0   0]
 [ 14   2   6 941  19   0  18   0   0   0]
 [  1   0  58  17 873   0  51   0   0   0]
 [  0   0   0   0   0 974   0  20   0   6]
 [100   0  57  22  40   0 775   0   6   0]
 [  0   0   0   0   0   2   0 981   0  17]
 [  7   0   0   5   1   1   1   4 980   1]
 [  0   0   0   0   0   3   0  22   1 974]]
F1 Score Matrix
[0.884 0.989 0.893 0.937 0.888 0.984 0.797 0.968 0.984 0.975]

Time Elapsed: 220.2 seconds
```

7.2.3 CIFAR 10

```
Testing TF_CNN.
Classifier algorithm: tf_conv
Dataset: cifar_10
Classifier accuracy: 74.520000%

Confusion Matrix:
[[812  13  33  20  15   7   8   9  60  23]
 [ 17 833   2   8   4   7  10   1  32  86]
 [ 75   4 589  71  70  97  54  23  15   2]
 [ 25   6  37 579  51 215  49  17   8  13]
 [ 20   3  56  81 672  47  40  75   6   0]
 [  7   2  25 183  32 708  12  25   4   2]
 [  6   2  44  65  22  45 808   5   2   1]
 [ 15   2  22  34  43  92   9 771   4   8]
 [ 55  18  10  16   5  10   6   1 861  18]
 [ 25  61   8  23   4  11   4   7  38 819]]
F1 Score Matrix
[0.789 0.857 0.645 0.557 0.701 0.632 0.808 0.797 0.848 0.831]

Time Elapsed: 943.0 seconds
```

7.2.4 CIFAR 100 Coarse

```
Testing TF_CNN.
Classifier algorithm: tf_conv
Dataset: cifar_100_c
Classifier accuracy: 54.060000%


Time Elapsed: 468.7 seconds
```

7.2.5 CIFAR 100 Fine

```
Testing TF_CNN.
Classifier algorithm: tf_conv
Dataset: cifar_100_f
Classifier accuracy: 41.220000%


Time Elapsed: 362.7 seconds
```