Kade Boltjes, kboltjes@purdue.edu

Github Repo Link: https://github.com/Kboltjes/CS390Lab2

Resources Used

- https://keras.io/guides/serialization_and_saving/
- https://keras.io/api/layers/regularization_layers/dropout/
- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html

Parts Completed

I completed all the sections of the lab that were not extra credit. The only extra credit I did was saving my network weights to a file. You toggle that by setting the global variable USE_PRETRAINED_WEIGHTS to True. I also added a global variable that allows you to keep training your net using pretrained weights, so if you set TRAIN_USING_PRETRAINED_WEIGHTS to True as well, it will begin the training using the pretrained weights as a starting point.

Questions

How is a CNN superior to standard ANNs for image processing?

A CNN uses convolutions to look for patterns in the image. The convolutions will look for more and more complex patterns as the data goes through the network. That gives CNNs a major advantage over ANNs.

Why do we sometimes use pooling in CNNs?

It allows you to reduce the width and height of your image, so the CNN can learn more general overall patterns from the data. It also reduces the number of weights we need to train.

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

Firstly, a cifar image has almost 4 times the input data as an mnist image. That makes it so the network will have to be much larger. Also, the cifar-100 dataset had much more classes that it had to predict, so there is a higher chance of error there. Another thing that makes it difficult is the cifar dataset is split into classes like airplanes and horses, which have a lot of variation, but the mnist only had numbers, which don't have near as much variation.

How I Increased the Accuracy of my CNN

Whenever I hit a wall on accuracy, I would start off by running only 7 epochs. Then I would compare different drop rates to see which would give me the best accuracy. After that, I would try different numbers of nodes in my convolutional layers. If that didn't increase the accuracy, I would add more layers or change the activation functions.

Hyperparameters

CNN Hyperparameters					
Dataset	**mnist_d**	**mnist_f**	**cifar_10**	**cifar_100_c**	**cifar_100_f**
Epochs	10	15	10	10	10
Drop Rate	0.2	0.3	0.2	0.3	0.4
Batch Size	100	100	100	100	100
Initial Activation Function	Sigmoid	Sigmoid	Sigmoid	Sigmoid	Sigmoid
Interior Activation Functions	Relu	Relu	Relu	Relu	Relu
Output Activation Function	Softmax	Softmax	Softmax	Softmax	Softmax
Loss Function	Categorical Crossentropy	Categorical Crossentropy	Categorical Crossentropy	Categorical Crossentropy	Categorical Crossentropy
Optimizer	Adam	Adam	Adam	Adam	Adam
Layers					
Layer 1 / # Neurons	(3x3)Conv2D / 32	(3x3)Conv2D / 32	(3x3)Conv2D / 32	(3x3)Conv2D / 32	(3x3)Conv2D / 64
Layer 2 / # Neurons	(3x3)Conv2D / 64	(3x3)Conv2D / 64	(3x3)Conv2D / 64	(3x3)Conv2D / 64	(3x3)Conv2D / 128
Layer 3 / # Neurons	(2x2)MaxPool2D	(2x2)MaxPool2D	(2x2)MaxPool2D	(2x2)MaxPool2D	(2x2)MaxPool2D
…	Dropout	Dropout	Dropout	Dropout	Dropout
…	(3x3)Conv2D / 32	(3x3)Conv2D / 48	(3x3)Conv2D / 48	(3x3)Conv2D / 32	(3x3)Conv2D / 64
…	(3x3)Conv2D / 64	(3x3)Conv2D / 64	(3x3)Conv2D / 64	(3x3)Conv2D / 64	(3x3)Conv2D / 128
…	(2x2)MaxPool2D	Dropout	(2x2)MaxPool2D	(2x2)MaxPool2D	(2x2)MaxPool2D
…	Dropout	Dense / 128	Dropout	Dropout	Dropout
…	Dense / 128	Dense / 64	Dense / 128	Dense / 128	Dense / 512
…	Dense / 10	Dense / 10	Dense / 128	Dense / 64	Dense / 256
…			Dense / 10	Dense / 20	Dense / 128
…					Dense / 100

Relevant Outputs

NOTE: All the relevant outputs were achieved only by changing the global variables DATASET, USE_PRETRAINED_WEIGHTS, and TRAIN_USING_PRETRAINED_WEIGHTS.

Here is my MNIST digit accuracy of 99% or higher

```
Epoch 10/10
600/600 [==============================] - 4s 7ms/step - loss: 0.0160
Testing TF_CNN on mnist_d.
Classifier algorithm: tf_conv
Classifier accuracy: 99.360000%
```

Here is my MNIST fashion accuracy of 92% or higher

```
Epoch 15/15
600/600 [==============================] - 4s 7ms/step - loss: 0.1452
Testing TF_CNN on mnist_f.
Classifier algorithm: tf_conv
Classifier accuracy: 92.350000%
```

Here is my cifar 10 accuracy of 70% or higher

```
Epoch 10/10
500/500 [==============================] - 5s 10ms/step - loss: 0.4903
Testing TF_CNN on cifar_10.
Classifier algorithm: tf_conv
Classifier accuracy: 75.950000%
```

Here is my cifar 100 course accuracy of 50% or higher

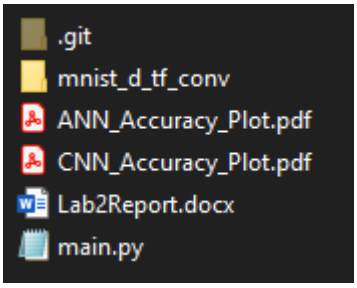
```
Epoch 10/10
500/500 [==============================] - 5s 10ms/step - loss: 1.4278
Testing TF_CNN on cifar_100_c.
Classifier algorithm: tf_conv
Classifier accuracy: 50.210000%
```

Here is my cifar 100 fine accuracy of 35% or higher

```
Epoch 10/10
500/500 [==============================] - 10s 19ms/step - loss: 2.0226
Testing TF_CNN on cifar_100_f.
Classifier algorithm: tf_conv
Classifier accuracy: 36.900000%
```

Here is an example of saving weights for MNIST digit

```
Epoch 10/10
600/600 [==============================] - 4s 7ms/step - loss: 0.0161
Saving Trained Model: mnist_d_tf_conv
Testing TF_CNN on mnist_d.
Classifier algorithm: tf_conv
Classifier accuracy: 99.120000%
```



Here is an example of the network loading in weights and not training

```
Dataset: mnist_d
Shape of xTrain dataset: (60000, 28, 28).
Shape of yTrain dataset: (60000,).
Shape of xTest dataset: (10000, 28, 28).
Shape of yTest dataset: (10000,).
New shape of xTrain dataset: (60000, 28, 28, 1).
New shape of yTrain dataset: (60000, 10).
New shape of xTest dataset: (10000, 28, 28, 1).
New shape of yTest dataset: (10000, 10).
Loading trained model: mnist_d_tf_conv
Saving Trained Model: mnist_d_tf_conv
Testing TF_CNN on mnist_d.
Classifier algorithm: tf_conv
Classifier accuracy: 99.120000%
```

Here is an example of the network loading in weights and doing additional training

```
Epoch 10/10
600/600 [==============================] - 4s 7ms/step - loss: 0.0067
Saving Trained Model: mnist_d_tf_conv
Testing TF_CNN on mnist_d.
Classifier algorithm: tf_conv
Classifier accuracy: 99.360000%
```

Here are my generated plots, which can be autogenerated by just setting GENERATE_BAR_GRAPHS to True.