Image Classification with Convolutional Neural Networks

Bilal Ahmad

November 14, 2019

1 Problem

1.1 Applying CNN to a Dataset

For this exercise we are using convolutional neural networks (CNN) to perform classification on images. The CNN works by breaking an input image into a grid. A window is scanned across the image and hidden layers identify a characteristic of each window. CNNs have been shown to perform better than other algorithms on image processing tasks such as correctly identifying handwritten digits in the MNIST dataset.

1.2 Dataset

For this exercise, my dataset consists of images of dogs and cats. My goal is to train a CNN to be able to make a correct classification on an input image of a dog or cat. The dataset comes from kaggle; it was used for an online competition in 2016 [1]. The data consists of 25,000 images used for training and another 25,000 images used for testing. In addition, my younger sister works in Seattle doing marketing work for Microsoft. On the side, she likes to also pet sit. I included images of the dogs she has pet sit into the test dataset because I was curious how the algorithms would work on those dogs.

1.3 Implementing The CNN

For the CNN implementation, I followed a tutorial on pythonprogramming.net that primarily relied on tensorflow libraries [2]. The CNN has two five convolution layers and a fully connected layer to go with the input and output layers. I am using max pooling in the convolutional layers as well as a rectified linear activation function. In the fully connected layer, a softmax activation is being used. The code preprocesses the data by assigning each image a correct label, then the network is trained over 10 epochs, and finally the network attempts to make correct classifications on the test data. The full implementation of the code can be seen in the Appendix. The code primarily follows the tutorial but I wrote code to evaluate the network over all the images in the test directory rather than just a subset of the training data.

2 Results

I trained the data over 10 epochs. The accuracy of the network over the course of the 10 training epochs can be seen in Table 1 and Figure 1. I chose 10 epochs after some trial where I did not see improvement in the accuracy rate with the addition of more epochs. When running the network with the test data, I got a correct classification rate of .7812. The discrepancy between the success rate on test data and training data suggest that the network may be overfitted. I also used a recurrent neural network (RNN) from a previous exercise to classify the data and received a success rate of ..8081. I individually tested the dog pictures from my sister and they were all correctly classified as dogs except for one chihuahua that was classified as a cat. A sample of the test image classifications can be seen in Figure 2.

Epoch	1	2	3	4	5	6	7	8	9	10
Accuracy	.7171	.7419	.7689	.7854	.7961	.7887	.8356	.8438	.8449	.8623

Table 1: Training Data Accuracy Per Epoch

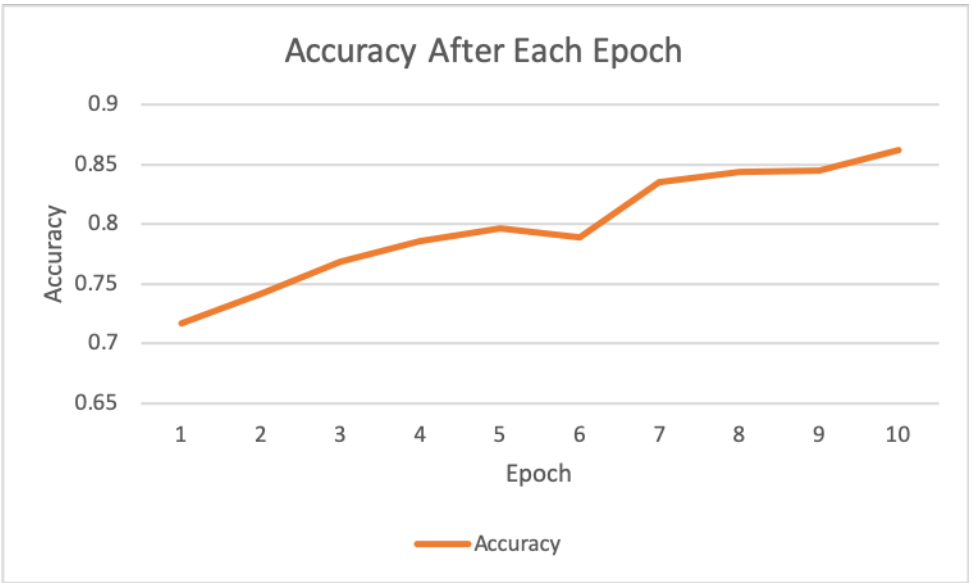

Figure 1: Accuracy After Each Epoch

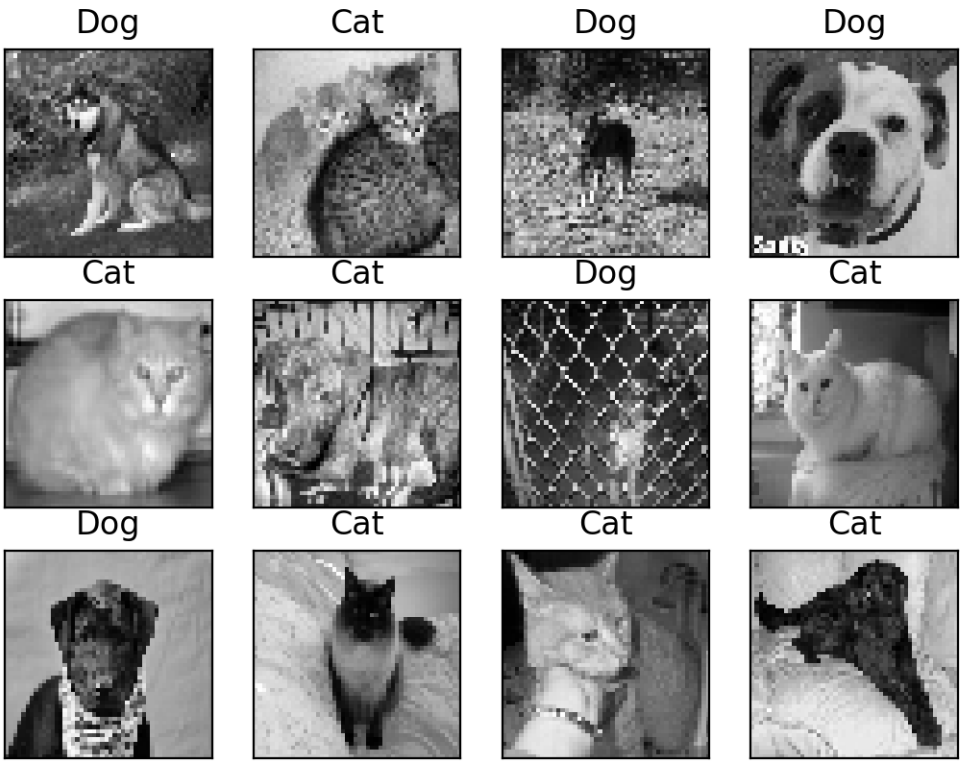

Figure 2: Sample of Test Image Classifications

3 Conclusion

The CNN performed well on the image processing with five convolutional layers. It should be noted however that during some experimentation with the network, I found that a network with only two convolutional layers performed quite poorly as did a network with 20 layers. It seems there is a point where networks can be too big or too small. In addition, because of the discrepancy between the test data accuracy and the training data accuracy, I think implementing some type of dropout would be beneficial to avoid over correction.

For further experimentation I would try to find the point at which adding more convolution layers is detrimental and further experiment with the window size used when performing convolutions. I think I would also try using a dataset with more than two

classes. I suspect that the RNN wouldn't perform as well compared to the CNN on image processing for more complex images. The data was grey scaled during the preprocessing step so I may also see how colored images perform on similar networks. Perhaps also I would filter out chihuahuas from the data set (Figure 3).

Figure 3: My Sister's Chihuahua Cat

4 References

[1] Dogs vs. Cats Redux: Kernels Edition www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data

[2] Deep Learning with TensorFlow pythonprogramming.net/tensorflow-neural-network-session-machine-learning-tutorial/