ENPM673 - Perception for Autonomous Robots
Project 6: Convolutional Neural Network

Cheng Chen, Haixiang Fang, Timothy Werder

Due May 16, 2020

Introduction

Convolutional Neural Networks (CNN) are a type of feedforward neural networks that include convolution calculations. It is one of the representative algorithms of deep learning. After the 21st century, with the development of deep learning theory and the improvement of numerical computing equipment, convolutional neural networks have been rapidly developed, and are widely used in computer vision, natural language processing and other fields. In this project, CNN will be used to do images classifications between cats and dogs within a given dataset.

Output Video Link

`https://www.youtube.com/watch?v=WZvsU-t2cCo&feature=youtu.be`

Part 1: Importing and Tensoring the Data

The first part of the program is to import the images and convert them to tensors. The first aspect of this part was creating sub-directories with the labels of the data and the contents including all pictures under each label for the training data. We do this by searching the train sub-directory for the label and import pictures with those names. Finally we delete the sub-directory leaving only the the 'cat' and 'dog' sub-directories.
We then translate the data into tensor form, reshape the size of the images so they take less time to process, shuffle, and declare our batch size. Once this is complete the data was ready to be trained training.

Part 2: Construction of the CNN

Before training we needed to construct our forward operation of the CNN. For this we used a convolve, relu then pool, we then viewed, linearized, relu and linearized once again ending with two linear elements that could me min maxed into the labels. This section was largely based on a previous project that had been done using CNNs for the MNIST and CIFAR-10 data sets. We did not use dropout in our classifier as we were able to achieve good results without it. Additionally, we attempted to use a logarithmic softmax function, but the results were on par without its use but added additional time and was therefore cut.The main difference that allowed for us to achieve results with 95 average percent accuracy is because of the Relu function. The Relu (Rectified Linear unit) is a linearized activation function that allows for stochastic gradient descent while returning a a value that is greater than zero if the value is less than zero. This allows us to create a shallow variant of a deep neural network by bypassing heavy pooling straight to characterization requiring much less computation.

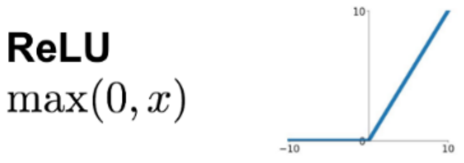

Figure 1: RELU Function and Mathematical Operation

Part 3: Validation and Training

Once the data was prepared the the CNN model was built we then trained it. Using 128 epoch and a batch size of 64 (see table 1 for full hyper parameters set). Our model took 9.5 hours to train with a picture size of 30x30.

Learning Rate: .1	Seed: 1	Layering
Time to Train: 9h, 38m	Epochs: 128	Conv, relu, pool, view (Flatten), linear, relu, linear
Batch size: 64	Optimizer: Adadelta	Loss Function: Cross Entropy Loss

Overall we were able to achieve an accuracy greater than 90 percent in validation with cross entropy for both validation and training both reaching zero.

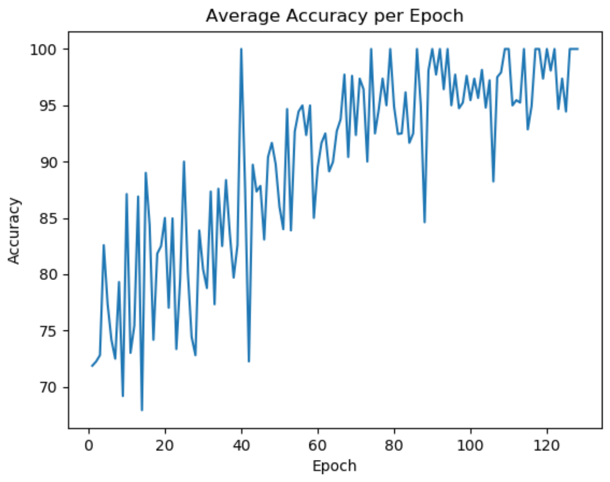

Figure 2: Average Accuracy between classes per Epoch.

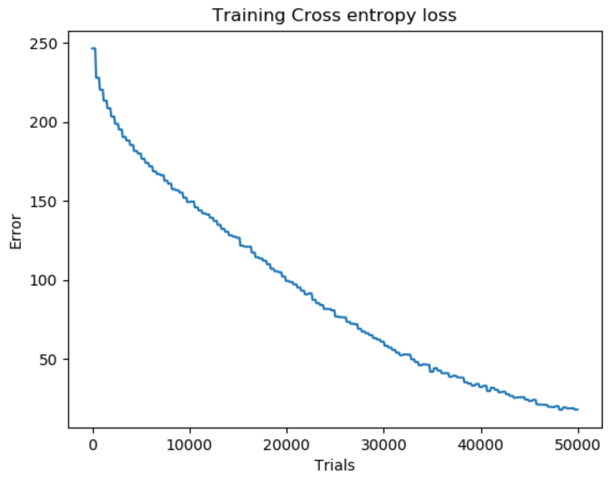

Figure 3: Training Cross Entropy per trial

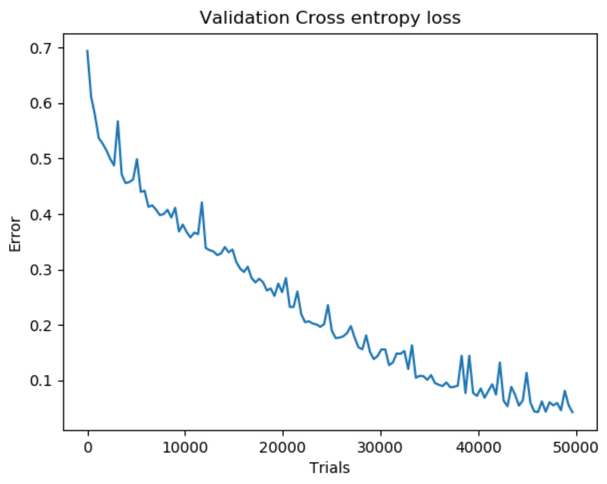

Figure 4: Validation cross entropy loss per trial

Part 4: Testing

For our testing we opted to test against all of the data in the test folder. We then selected a random set of 25 photos and displayed them.

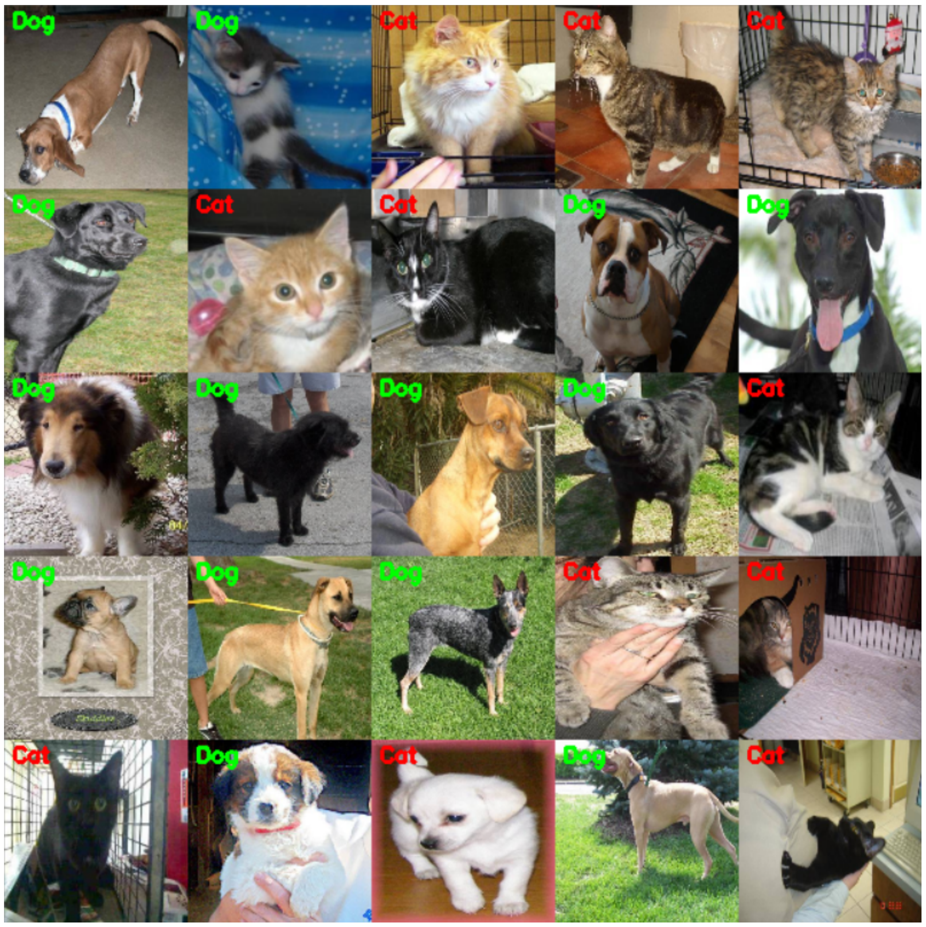

Figure 5: Test output with labels(1/2).

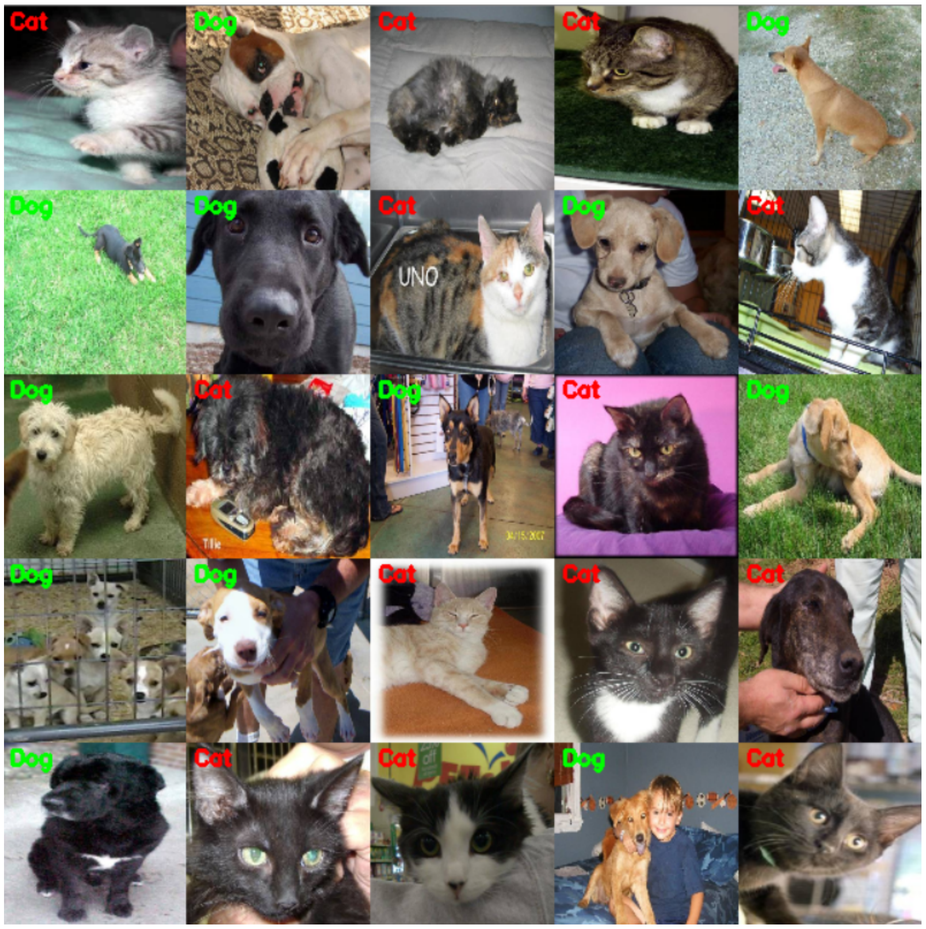

Figure 6: Test output with labels ((2/2).

Overall, running the testing data multiple time to see what results we would get showed that w were able to achieve very good results.

Part 5: Restoring and Saving

Once we had completed both testing and training we allow the user to save off the model for further analysis. Finally, if the user is done working on this program and wishes to restore the folders to the original form with all pictures being moved to the original sub-directory and deleting the class labeled sub-directories. This was mainly implemented for the Teaching Assistants so there wouldn't be issues with them running other codes.

Part 6: Conclusion and Lessons Learned

There were a few lessons learned with this project. First is that there are multiple ways to go about designing CNN's. Since CNNs are classified as Deep Neural Networks their models have an infinite variety to how they can be built initially we went with trying to design s VCC-16 but ultimately ended up deploying a shallow DNN using Relu. Second, the amount of time it takes to train can be astronomical without good parameter tuning. This would be particularly difficult given the image sizes as larger images, while providing more information that can be classified, will take significantly longer to complete training. This is particularly a problem for IDEs (like Pycharm our native developer), that don't support GPU acceleration. Third, file manipulation mechanisms are the easiest and most cost effective way to develop custom data into a data that can be used in Pytorch. We mainly attribute this to the difficulty it would be to develop an algorithm that would formulate the data in a way that Pytorch would be able to use. Instead, by simply formatting and reorganizing the data in the folders we were able to use inbuilt Pytorch functions to tesnorize the data.