Karan Bhargava

Final Project Report

Machine Learning

**Project Outline and Purpose:**

This project is meant to analyze the performance of a CNN trained on Google Streetview Digits and tested on a classical MNIST dataset to see how robust a CNN model trained on images with a high level of noise is able to distinguish numbers in a much “cleaner” MNIST dataset. This is meant to be a training exercise to see how I can work with training models to be versatile enough on data that is similar in concept, but different in presentation.

One of the challenges that the model will face when training on Streetview digits is that the model will train on “perfectly drawn” numbers: numbers that are “perfect” in shape but imperfect due to the noise present in each of the images. Whether it be inconsistent lighting, the presence of distracting digits, or poorly angled images, the numbers will vary in how they will be read. This may effect the portions of the CNN that specialize and filter the images to reduce “noise” since the MNIST dataset will not have the same type of noise that the streetview images will have.

Ideally, I would like to use the experience from this project to then start creating models that train and read human handwriting on letters and words and synthesizing it with pictures of “perfect letters” to make a model that can read words accurately. In the long run, the goal is to then apply this to education so that AI can “grade” homework, tests, etc.

**Work**:

The dataset was taken from [www.ufldl.stanford.edu/housenumbers/](http://www.ufldl.stanford.edu/housenumbers/) . I talked with Tovio to give me an idea on how to scope the project to make it do-able within the time-frame of when the project was assigned and when it was due. The basic model structure was based in the Practice 4 class notes that were found in blackboard. I tweaked the model to handle the Streetview images, though. I also took the data as a .mat file, saved each individual image to a folder, and then processed the images into the model since the .mat file had a bizarre 4d implementation on how the data was stored. I also changed the images to grayscale to match the types of images that the model will train on in MNIST. Model evaluation code was pulled directly from the blackboard code. I also increased the size of the MNIST images to match the size of the Streetview images (32x32). I changed the optimizer to ‘rmsprop’ based on recommendations from websites on optimal combinations of parameters for a CNN.

The only code I directly copied was a sorting algorithm that I cited in the program. It sorted ‘listdir’ and its confusing ordering of folder files.

*Data Collection:*The .mat file was not usable in its basic state. The only data I was able to pull from it that was useful was the labels for the images. Because of its bizarre 4d ordering, I had to manually pull each image and save it to a new folder. So I made a folder for test data and a folder for train data. I included the python file that pulls the matlab file data and outputs it to a folder titled ‘train’ and a folder titled ‘test’. If you run the python file named ‘PicChanger’ make sure to delete all the contents in the ‘train’ and ‘test’ folders so that you don’t have double the images in them. If you don’t want to run the ‘PicChanger’ folder then just use the StreetViewNN python file and it will read the images in the ‘train’ and ‘test’ folders I already made.

I also included code (but commented it out) that would save the pre-processed image to their own files so that you didn’t need to create a new set and so that the images didn’t need to be stored in the computer’s RAM.

*PreProcessing:* I manually wrote the pre-processing for the mean image on the train data and then implemented library normalize functions to the data to normalize the mean subtraction

I also implemented the Roberts edge detection to see if standardizing the images to be similar to MNIST might improve performance.

The model I used was primarily the CNN used for Practice 4. Since that model was based on the MNIST dataset and had great performance, and this model is meant to analyze a dataset presented in a similar style to MNIST, I reasoned that this would be a sufficient model for the problem I am trying to solve.

**Libraries:**

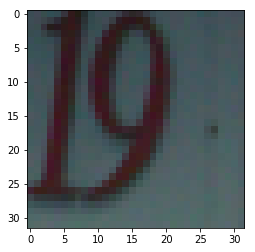
Keras, Sklearn, tensorflow, cv2, numpy, pandas, matplotlib, warnings, os, skimage, scipy, re (regular expression)

All of these are standard libraries for deep learning and image/computer vision data processing.

However, re was a direct code copy from stack exchange which I cited in my python file for sorting the ‘listdir’ function.

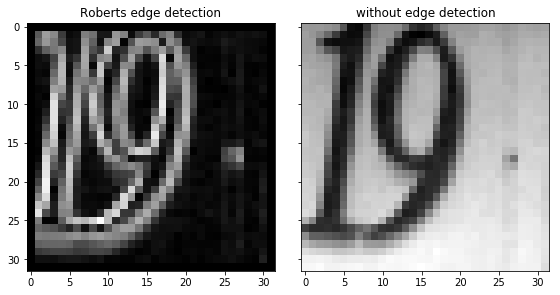
**Pre-Processing:**

**Original rgb2gray mean subtraction/normalized**



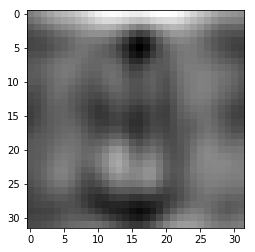
**Roberts Edge MNIST + Roberts Edge**

**Detection + mean subtraction**

****

**Mean of All Individual pixels**

**Across training data (used for mean subtraction)**

****

**Analysis:**

Despite optimistic views, the amount of improvement from the pre-processing on the model was not significant. I started with about a 85% accuracy on the training data and about 67% accuracy on the MNIST test data and after all the pre-processing I ended with 67% accuracy after implementing Roberts edge detection and mean subtraction. Mean subtraction actually dropped the performance of the MNIST data set by about 6% and then mean subtraction + Roberts edge detection brought it back up to 67%. While it is discouraging to see that some of the pre-processing made the model perform worse, I am happy with the effort and thinking I applied to attempt that preprocessing on the data.

The reason I believe that the model didn’t improve much is because the influence of the distracting digit caused the classifier to misinterpret the numbers. Whereas the mean subtraction attempted to remove some of the noise, Roberts edge detection removed the noise entirely but then caused the impact of the distracting digit to increase. Perhaps adding noise to the Roberts edge detection pre-processing might improve the performance going forward.

Alternatively, I might make the data mute the noise on the outer edges of the image to help it focus more on center of the image. Since the images that will classified tend to be centered in the picture, it might help to mute the borders of the image to get a better performance.

Interestingly enough, the performance of the model might have improved more if I had used more epochs in the training. If you look at the performance of the training, it seems that the improvement is more linear and had more room to improve if there were more epochs to train on.

**Results:**

Performance of the trained CNN on the streetview image test set

A screenshot of a social media post

Description automatically generated

Performance of the Base CNN on the Base MNIST Dataset

A screenshot of a social media post

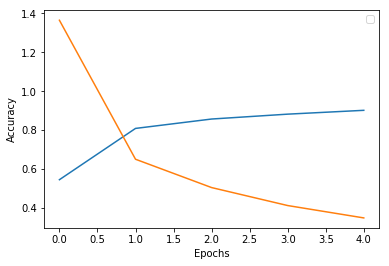
Description automatically generated

Performance of the CNN with mean subtraction and normalization on SV training dataset

A screenshot of a social media post

Description automatically generated

Training results with Mean subtraction and normalization (blue = acc, orange = training loss)



Performance of the CNN with mean subtraction and normalization on MNIST test dataset

A screenshot of a cell phone

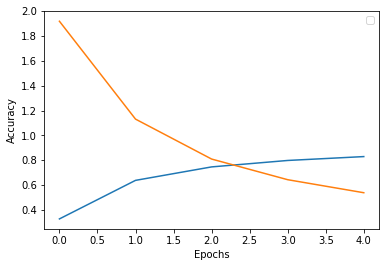
Description automatically generated

Performance with Roberts edge detection on training set

A screenshot of a social media post

Description automatically generated

Training loss on Roberts edge detection



Performance of the Test dataset with Roberts + mean subtraction on MNIST

A screenshot of a cell phone

Description automatically generated

**Potential Improvements:**

In the future, I would use a larger training set (my training set was only 10k images). I would also mix the MNIST dataset with the StreetView images so that I could actually have a robust model at classifying both human and artificial numbers. I would also consider applying transfer learning to the model to see how it effects the performance. Other things I would consider would be seeing if RGB changes the performance of the model at all (though I doubt it would).

**Why is this meaningful?**

The Pre-Processing work I did was meaningful in understanding the impacts of the noise in the dataset and how it effected the training of the CNN. It helped understand how edge detection can be both useful and problematic when the image that is being analyzed has too much noise for the model to clearly distinguish relevant from irrelevant data.

As stated earlier, I would ultimately like to work in applying Artificial Intelligence to education. I want to work on creating an AI based tutor. In order for this to become a reality, I want to begin by learning how models read human handwriting and human language. This would then lead to a development in learning NLP and written letters/alphabet. The project would also help me further understand how to analyze and preprocess data for computer vision. Other things that this project will help with is in understanding the limitations or advantages of processing data in grayscale as opposed to the classic RGB. Furthermore, this would also help understand more into CNN’s strengths and weaknesses when comparing it with images that are not in the same style of its training dataset.

This helps to create a versatile model that can read both artificial and hand-drawn numbers. By mixing MNIST and StreetView numbers, perhaps the model will be able to hit an accuracy of >95%.

I plan on tweaking the model further by adding layers, increasing epochs, doing more pre-processing, and increasing the size of the training set to see how it effects the model’s performance.

Sources:

**Roberts Edge Detection**

<https://scikit-image.org/docs/dev/auto_examples/edges/plot_edge_filter.html>

**Streetview Data set**

<http://ufldl.stanford.edu/housenumbers/>

**Choosing CNN final layer**

<https://www.dlology.com/blog/how-to-choose-last-layer-activation-and-loss-function/>

**Pre Processing Tutorial (I didn’t use the code, but it helped me learn pre processing techniques)**

<https://www.oreilly.com/library/view/programming-computer-vision/9781449341916/ch01.html>

**RE sorting code from stack exchange**

<https://stackoverflow.com/questions/4813061/non-alphanumeric-list-order-from-os-listdir>