Final Project Report for CS 175, Spring 2018

Project Title: DIGIT RECOGNITION
Project Number: "I'm not sure what goes in here, since I work on this project solo, I didn't sign up for a group and therefore don't have a group number"

Student Name(s)
Tri Hoang, 15681623, trih1@uci.edu

General Instructions:

- Your report should be 5 to 7 pages long in PDF

- If you want to add more details (e.g., additional graphs, examples of your system's output, etc) beyond the 7-page limit, feel free to add an Appendix to your report for additional results

- **What the Team submits to Canvas (one student submits the items below on behalf of the team)**
    - Your report entitled **FinalReport.pdf**
    - A zip file called **project.zip** that contains the following in 1 directory called project/
        - A README file that contains a 1 line description of each file in project/
        - A Jupyter notebook (called **project.ipynb**) that can be run directly and that demonstrates your project. Your notebook can import a sample of the data that you used, import 1 or more models that you built, and generate examples of the types of predictions or simulations your model can make. The notebook should not take any longer than 1 minute to run in total (if you have models that require a lot of training time, train them offline and just upload the models and some sample data to illustrate them). Feel free to generate examples of your model(s) in action, e.g., for reviews you could generate examples of reviews where the models work well and reviews where the models work poorly.
        - Also save a .html version of your notebook called project.html, showing the outputs of all the cells in the notebook.
        - Upload any data files needed to run project.ipynb – keep your data sets to 5MB in total or or less.
        - Also include a subdirectory called src (within the zipped project/ directory) with all of the individual code (scripts, modules) for Python (or equivalent for other languages) that your team wrote or adapted– these don't need to be called by the project.ipynb notebook but need to be in the src/ directory

- Note that we don’t necessarily plan to run all your code, but may want to look and run parts of it.

- **Individual Contribution**

  Each team member needs to submit a ½ page of additional text as an individual, titled “**IndividualContribution_ID.pdf**” that provides an honest assessment of which parts of the project you contributed to and which parts were worked on jointly. This should be written individually – you may wish to discuss the plan of what you will write with your project partner, but the page you write should be generated separately by each individual.

1. Introduction and Problem Statement (1 or 2 paragraphs)

For this project, I will design an application that can detect a digit from a handwritten input. The input of this project will be a dataset of thousands of scanned handwritten digits. The goal is to identify thousands of images that are provided from the dataset and to have as least error as possible. For some scenarios, the handwritten digits will be nice. However, there will be cases when the digit isn’t easy to identify even for the human eyes.

NOTE: This is a Kaggle project. The project can be found at: https://www.kaggle.com/c/digit-recognizer

2. Related Work: (1 or 2 paragraphs)

While doing the research for the project, I found 2 related work that were considered useful for the software.

1) *A program that helps me read input from the dataset, extracts it and produces some useful functionalities. More details can be found at the* **Software** *section.*

Link to the resource: https://github.com/petewarden/tensorflow_makefile/blob/master/tensorflow/contrib/learn/python/learn/datasets/mnist.py

2) *A tutorial that walks through the dataset, Tensorflow and many other useful informations that I needed to get started. NOTE: I didn’t consider everything in the tutorial into my project, it’s only a reference to my research and although there are sniper of codes here and there in the tutorial, I definitely didn’t copy everything as I only look at it for reference to start the project and have an idea how to work on the algorithm.*

Link to the resource: https://www.tensorflow.org/versions/r1.0/get_started/mnist/beginners

3. Data Sets

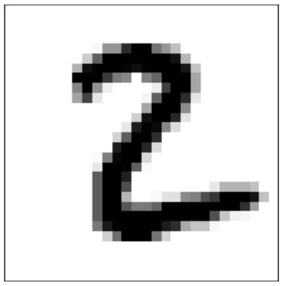
The dataset can be found from the MNIST (Modified National Institute of Standards and Technology) database: http://yann.lecun.com/exdb/mnist/index.html

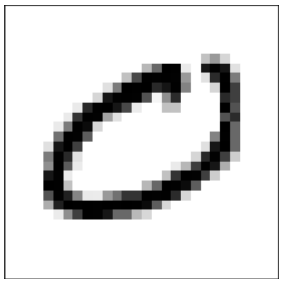
There are 2 datasets, one for training and one for testing. Here is a brief explanation:

"The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image."

"Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive."

Sample data:





4. Description of Technical Approach [at least 1 page]

- *The first thing I did to approach the project was doing some research of how to read and extract the datasets after downloading it. From there, I found my first resources that is described in the* **Related Work** *section. After I extract the datasets, I play around with all the functionalities and print out things to get me a better understanding of what I'm working with.*
- *Next, I declared some variables that are useful to include for my convolutional layers. While having 2 layers, the first one will have 16 filters and the second will have 36. For the fully connected layers, there will be 128 neurons that will be processed.*
- *MISCELLANEOUS: At this point, I should mentioned that I started out the project with the Pytorch framework in mind. However, after making progress into the project and with some research that I found, there are things that Tensorflow is better in term of executing and for this project, it was a better idea to use Tensorflow. It's still doable with Pytorch, but I wish I have saved some screenshot that I was originally worked with Pytorch.*
- *Using some functionalities that I write in the helper.py (as describe in the* **Software** *section below). I created 2 convolutional layers, with the first one being 16 filters and the second one with 36 filters.*
- *Next, I flatten the data that was output from the convolutional layers and feed that to the fully connected layer.*
- *Next, I feed what was output from the fully connected layer to the output layer. The input will be filter into one of the ten filters (for ten digits) in this layer.*

- *Next, to create a final prediction, I find the mean of the cross entropy (the loss) results. I also created two optimizer, Adam and RMS, to help with the optimizing process. After some testing, Adam seems like the better one so that's what I kept in my project.*
- *Next, using Tensorflow framework, I created a session that will process the data. I can run the session as many time as I want for tuning purposes or create a new one if I would like.*
- *Lastly, I run it with the test dataset and output the accuracy result. At the end of my project, I also plot 10 random examples to show that the output is correct.*

4. Software [at least ½ a page]

Software used:

- *Jupiter Notebook: This is an application that allows me to develop a .ipynb application for my project. Additionally, I used this application to write my Python codes as well.*
- *Tensorflow library: This is the main library/framework that I used in order to create my convolutional neural network layers.*

My codes:

- *helper.py: This file consists all the helper functions that I write to assist my main software*
    - *flatten: This function is used to adjust the dimension of the image in order to feed it to the fully connected layers*
    - *convolution_layer: This function will do all the processing job within the two convolutional layers in my software. It utilized the Tensorflow.nn library with many helpful functionalities such as .conv2d, .bias_add, .max_pool, .relu*
    - *fully_connected: This layer will take the output of the convolutional layer as an input, turn them into small neurons (the number of neurons depends on what state it's in) and ultimately proceed with the classification progress.*
    - *choose_optimizer: give an input of either "Adam" or "RMS", it will return the optimizer that you would like to use for the software.*
    - *get_accuracy: This function will return the final prediction of the image(s) and the accuracy of the test set.*
    - *plot: This function will help plot the image for sample outputs*
- *digitRecognizer.ipynb*
    - *This is the notebook that contains the main logic of my software. More details can be found at the* **Description of Technical Approach** *section above.*

Other resources:

- *get_data.py*
    - *With some research, I found a helper program that would help me read the datasets and extract it for my project. This program would read the datasets and additionally, return four useful*

functionalities that I used throughout the project: data.images (returns the hand written image that I need to work with), data.labels (returns the label or the correct result of the output), data.num_examples (returns the numbers of the dataset) and data.next_batch (return the next batch of the dataset)

- *Link to the resource:* https://github.com/petewarden/tensorflow_makefile/blob/master/tensorflow/contrib/learn/python/learn/datasets/mnist.py

5. Experiments and Evaluation [at least 1 page]

Dataset:

- *The dataset was found at the MNIST website. (more details can be found at the* **Data Sets** *section)*
- *To read and extract the data, I use an outside resource (more details can be found at the* **Related Work** *section)*

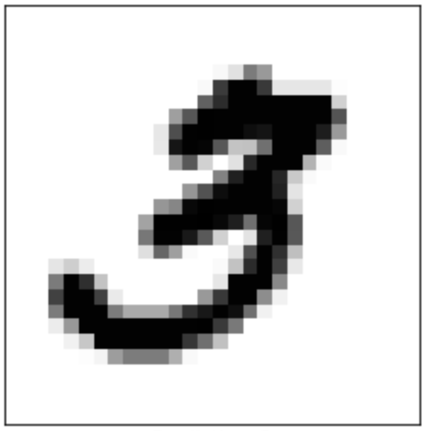
Convolutional Neural Network:

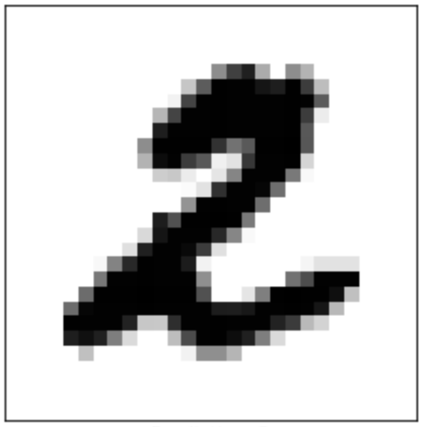
- *Convolutional Layer 1: This layer will break the image into 16 pieces. It will perform a scan by using multiple functionalities from the Tensorflow framework. Then it will feed the 16 small pieces into the second convolutional layer.*
- *Convolutional Layer 2: This layer perform the same task as the first layer, except it will break the image into even smaller 36 pieces to perform a better scanning process. When it's done, it will be flatten to get the right dimension for the fully connected layer*
- *Fully Connected Layer: This layer consists of 128 neuron. It will be feed into the last and final layer with only 10 neurons, with each neuron being the class of the image (aka the digit)*
- *Output Layer: There are total of 10 filters (each filter is basically one digit). It will take the output from the fully connected layer as an input and feed it into one of those 10 filters. The output will be a single digit.*

Finding Loss and Mean:

- *By using one of Tensorflow functionalities, I was able to calculate the loss function with cross entropy. Again, by also using Tensorflow, I was able to compute the mean of the losses. (Note that in order to calculate the mean, I had to use tensorflow.reduce_mean(). Originally, I used np.mean() and that didn't work very well).*

Result/Output: The result is being plotted with a sample input and the prediction that is made by the software. Some sample outputs are below:


Prediction: 3


Prediction: 2

6. Discussion and Conclusion [at least ½ a page]

To conclude the project, I learned a lot about how convolutional neural network works and its different layers it has to go through. I had a lot of confusion after working on homework 4 with Tensorflow and Pytorch, but to able to apply it to an actual project is pretty cool and I truly enjoy working on this project. At first, convolutional neural network sounds like a very complicated idea to implement and to be honest, I've spent a lot of my time in this project since I don't have a partner, but looking back at it, Tensorflow is so much useful and the biggest thing I get out of this project is the practice of using both Tensorflow and Pytorch framework that I could potentially use again in the near future project, or even at a professional career.