Hand-Written Digit Recognition

Mark Garcia

Alex Hwang

Hanson Nguyen

Andrew Phan

Anthony Reyes

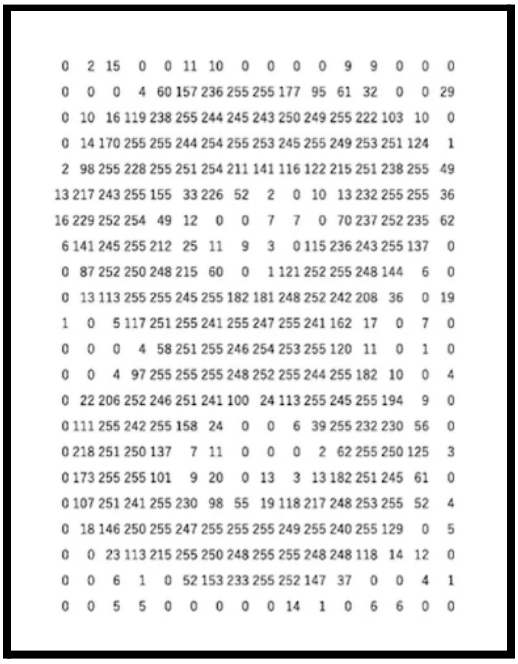
Linda Trinh

CECS 456 Machine Learning

May 3, 2022

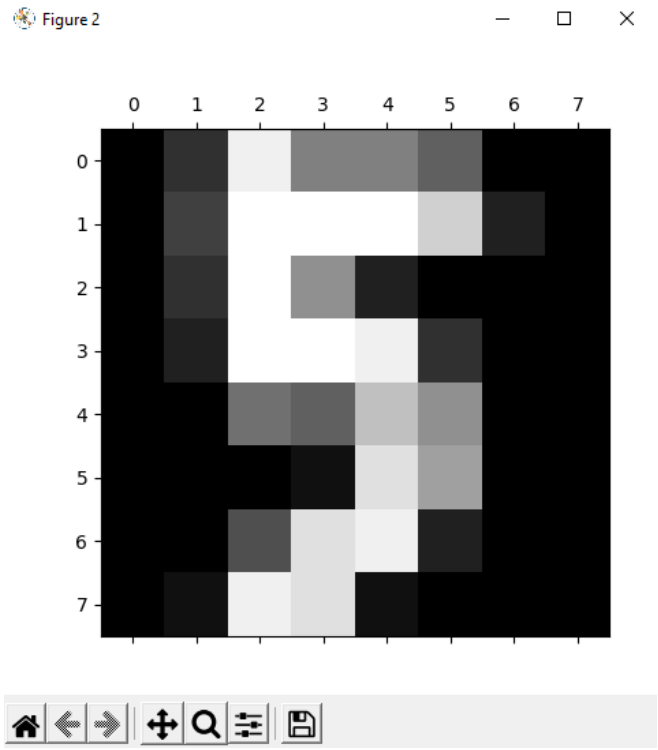
Image recognition and transfer into machine-readable data

A computer does not have eyes to perceive images, however, its ability to process data with an incredible speed in comparison to human is able to simulate such a process for machine learning. Image can be interpreted as a array of integers with values ranging from 0 to 255 for the brightness of the image in their respective color. The process of data interpretation start with taking in a dataset of image file, preferably png, and convert this image set to grayscale. The transfer of image into machine readable data is usually done with the tensorflow library which quickly convert the pixels of an image into the corresponding number values. The reason for using grayscale images is because they only have one integer value of color per pixel to determine the brightness of that pixel. Meaning that the data will have a faster calculation and interpretation time when working with 1 number instead of 3 per pixels. With the array of integers, the machine can now interpret combination of numbers as lines and strokes with through the training process. To the right is the converted example of a handwritten number 8 converted into a machine readable array of number. This machine data can now be pass for interpretation by the machine for training and image processing of handwritten digits.

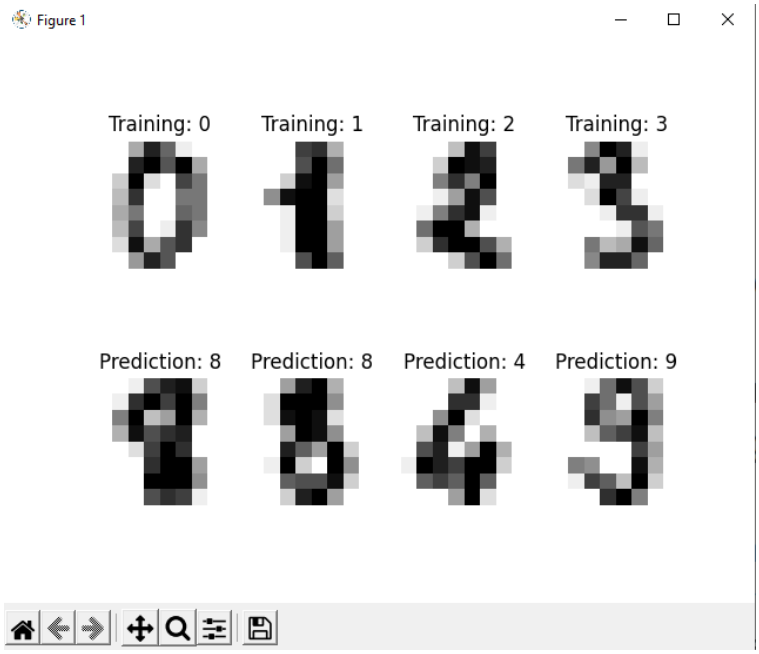


Splitting raw data input into recognizable patterns for training

A program can be trained to recognize patterns in numbers or data through classifiers. In this instance, our program splits all of the data into test or training data and then processes it to be able to recognize those numbers when they come up again. In our program, we used a Support Vector Machine classifier, or SVM for short because it had the highest accuracy report. We have a bunch of numbers in our training data as well as what those values actually are so the program has a reference guide. It then formats the numbers in an array-like graph like in this first figure.

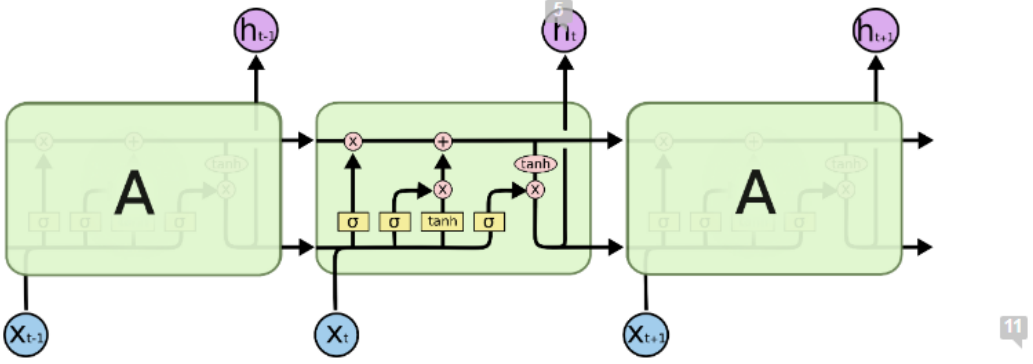


The program reads this pixelated graph and registers it as a 5. The next time a similar pattern or pixelated graph comes up it will refer to previous instances and try to come to a conclusion on what it is. This is done many times with many different numbers until the program is finished training and ready to test. Then, the test data will be used to test the program, which will use the previous training as a reference guide of sorts. The accuracy of our algorithm went up to 0.9688542825361512 with our current training and test data.



Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are a subcategory of recurrent neural networks (RNN). Sequential data is passed into layers of cells of a neural network. Input data is passed into one cell and that cell produces an output for the next layer of the neural network but also provides data for the next cell in the same layer. LSTM layers are commonly composed of cells that contain an input gate, output gate and forget gate.



The repeating module in an LSTM contains four interacting layers.

Middle cell contains a visual representation of an LSTM cell.

The first step in each LSTM cell is to decide what information the cell wants to keep from the previous node and what it wants to throw away from the previous cell state. The information that is no longer needed is stored in the forget-gate to be thrown out at a later step. Next step is to decide what new information should be added to the current cell state. This happens in two substeps; take input from the first sigmoid layer in the figure above called the input-gate, then decide what information from the input should be kept. The next step forgets the data from step 1 and combines the new information from the previous step to the updated current cell state. Finally the cell decides what it is going to output. The data passes through the final sigmoid layer that filters the current cell state and provides output for the next cell and layer of the neural network.

In our project, we constructed a RNN using LSTM layers to have better accuracy when predicting handwritten digits. Written in python, we trained our neural network model using the MNIST dataset provided by the tensorflow library that contains roughly 60,00 handwritten digits. The information of each of these digits are store in 28x28 2-dimensional arrays that contain numbers between 0 and 255. We then normalized the data so the numbers are between 0 and 1 so our model will be able to learn quicker. The model contains 2 LSTM layers and 2 normal dense layers. Each of these layers are followed by a dropout layer to help prevent overfitting in our model. We then train it with the data set mentioned above. To test our model, we wrote a number into a 28x28 pixel paint canvas. To interpret our drawing we used the cv2 library that reads in the image as an array similar to before. We also have to normalize this data to be between 0 and 1, so we divide

```
Image 0 is similar to: 0
Image 1 is similar to: 2
Image 2 is similar to: 2
Image 3 is similar to: 2
Image 4 is similar to: 4
Image 5 is similar to: 5
Image 6 is similar to: 6
Image 7 is similar to: 7
Image 8 is similar to: 0
Image 9 is similar to: 7
```

everything in the array by 255.0. We then predict what the image could be using our model giving us the output to the right. Our model had given us an accuracy of roughly 98%. Some of the output is incorrect, but this is because of our styles of handwriting.

Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed or undirected graph and this allows it to exhibit temporal dynamic behavior. It is important because in some scenarios reading sequential data instead of individual data is more important. For example, in an English sentence if you were looking at words individually it would be difficult to get an idea of what is going on and looking at the whole entire sentence would be better. RNNs have many different use cases such as language translation, natural language processing, speech recognition, and image captioning.

In our code, we make use of tensorflow and the MNIST database which consists of images of handwritten digits and their values. We split up our data into training sets and testing set. After that, we create a sequential model consisting of one flatten layer, two relu dense layers, and one dense softmax layer. We then compile our model and make it go through the training sets for 10 epochs. Finally, we are able to test our machine learning model's accuracy and loss. Our accuracy was 96.69% and our loss was 0.1267.

Convolutional Neural Network

A convolutional neural network (CNN) is an area of deep learning that specializes in image recognition and classification. It can perform image analysis as it is able to detect patterns

through its multi-layer neural networks: convolutional, pooling, fully connected, etc. Within those layers, it has filters, which perform pattern recognition. In our model, we used a 3 x 3 filter block and within that block, it specifies a pattern to look for. The convolutional layers receive input and then transform the input using the filter into a new interpretation, and this output would be the input of the next layer. With every convolutional layer, it performs feature extraction, only allowing positive values to bypass the next layer so the next layer is reduced in what is known as max-pooling. Before it can move to the fully-connected layers to be classified, it must be flattened from 2D to 1D. Once the model is created, we compile it and then fit the model to train it.

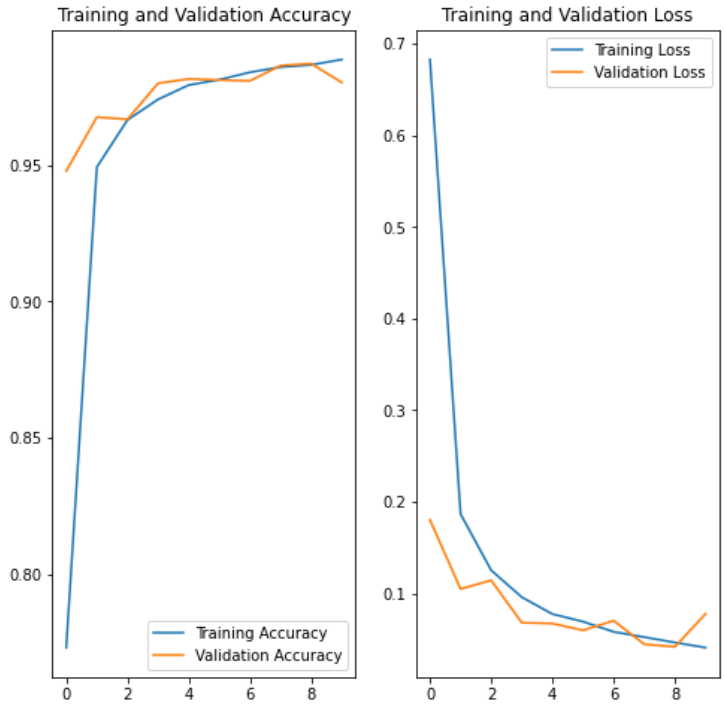
In our code, like RNN, we made use of TensorFlow and the MNIST database. We also split up the training and test datasets. We prepared the data as it needed processing on the images such as reshaping and normalizing the pixel values. Afterwards, we were able to build our CNN model. In our model, we had our model go through 3 of each being the convolution layers, max-pooling layers, and fully-connected layers before being compiled, trained and evaluated. To which our results were 98.30% accurate and a loss of 0.0678.

Comparing Results

When comparing the results of the three Neural Networks, LSTM is the most accurate for our dataset. Here we have the Training and Validation Accuracy and Loss graphs for each of the models. Both RNN and CNN have Overfitting Learning Curves, which means that they learned the training dataset too well. This is an issue because the models will not be able to learn new data as well as the training data. LSTM has a Good Fit Learning Curve. This means the training and validation loss values are stable and closer in value. The most accurate model is the LSTM

with an accuracy of 98.88% and a loss of .0411. Next we have CNN with an accuracy of 98.22% and a loss of .0641. And finally we have RNN with an accuracy of 96.69% and a loss of .1267.

LSTM



RNN


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Training Loss
Validation Loss

CNN


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Training Loss
Validation Loss

References:

1. “All Symbols in Tensorflow 2 :   Tensorflow Core v2.8.0.” TensorFlow, https://www.tensorflow.org/api_docs/python/tf/all_symbols.
2. Diaz, Emmanuel. “Using Convolutional Neural Networks to Read Handwritten Numbers.” Medium, Data Science Student Society @ UC San Diego, 28 May 2018, https://medium.com/ds3ucsd/using-convolutional-neural-networks-to-read-handwritten-numbers-c8c9a692003f.
3. “Handwritten Digit Recognition Deep Learning Project.” Analytics Vidhya, 26 Nov. 2021, https://www.analyticsvidhya.com/blog/2021/11/newbies-deep-learning-project-to-recognize-handwritten-digit/.
4. Michel Kana, Ph.D. “Recurrent Neural Networks (Rnns) for Dummies.” Medium, Towards Data Science, 21 Feb. 2020, https://towardsdatascience.com/recurrent-neural-networks-explained-ffb9f94c5e09.
5. “Understanding LSTM Networks.” Understanding LSTM Networks -- Colah's Blog, http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
6. Brownlee, Jason. “How to Use Learning Curves to Diagnose Machine Learning Model Performance.” *Machine Learning Mastery*, 6 Aug. 2019, https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.