Handwritten text to Synthesized Speech

Attiano Purpura-Pontoniere *Student*, Brian Tehrani *Student*, Mike Briggs *Professor*

***Abstract*— The purpose of this project is converting handwritten text to synthesized speech. This project is motivated by the desire to have computers understand, process, and output handwritten text. Our approach is to apply image processing techniques and machine learning to accomplish that task. We are able to successfully output single handwritten words and digits as speech.**

***Index Terms*— Binarize, Gray-Scale, Image Analysis, Image Filtering, Image processing, Machine Learning, Neural Networks**

I. Introduction

THIS section covers the history, sources, and global constraints.

A. History

The earliest ideas related to handwritten character recognition can be traced to Fournier d'Albe's Optophone and Tauschek's Reading Machine in the late 1800s and early 1900s [1]. In the 1930-1950s the first handwritten character recognition tools were invented and used in industry by the Intelligent Machines Research Corporation, and they were able to interpret text and decode Morse code [2]. Then more recently, beginning in the early 2000s, in part due to Kurzweil Computer Products, OCR software is readily available from companies like Adobe Acrobat, WebOCR, and Google.

B. Global Constraints

The main external global constraints we faced in our project were that of time and computing power. We had ten weeks to complete our project, in addition to the fact that we only had CPU based computers without dedicated GPUS to apply to the training of our artificial neural network.

The main local constraints we faced were related to our choice of programming languages and machine learning packages. We had to use the image processing modules given to us in Python, such as OpenCV, and to use the interface given by Keras/Scikit-learn for machine learning. Our computer processing power and memory constraints were due to our choice of computer architecture.

II. Motivation

The motivation for this project is to expand the area of handwritten recognition coupled with a computer's ability to output that processed handwritten data as speech. We wanted to essentially teach a computer how to read handwritten notes. This could be applied in a series of ways: It could help blind and illiterate people read. It could be used as a way of creating a handwritten turning machine. It could be used to scan books and convert them to speech files to have audible books more readily available.

III. Approach

The overall group design, implementation, and results are outlined within this section.

A. Team Organization

The course of the project was divided into three parts: image processing, implementation of the neural network, and speech output. Since the implementation of our project involved the use of using two different data sets and means of image processing, the tasks amongst group members remained consistent between the two data sets. Separate research was conducted upon the respective topics. Brian Tehrani was in charge of the implementation of the image processing and speech output. Attiano Purpura-Pontoniere was in charge of the neural network implementation.

B. Plan

The first step in the project design was to crate a proposal outlining the general overall goal and assignment of individual group tasks. The outline follows a break down of the project on a tri-weekly basis with weekly individual journals to chart progression. General research was conducted on topics within character recognition and machine learning and within the first week the team collaborated with the Professor Briggs on the overall approach and change in future plans when the task becomes too difficult within the time frame of the project.

This research paper was submitted for review on March 23rd, 2019. This work was supported in part by Professor Dennis Briggs at University of California, Los Angeles, and teaching assistant Anastasios Papathanasopoulos.

Author Brian Tehrani is graduating from the department of Electrical and Computer Engineering from University of California, Los Angeles in 2019. Los Angeles, CA, 90024 (e-mail: btehrani@ucla.edu).

Author Brian Tehrani is graduating from the department of Electrical and Computer Engineering from University of California, Los Angeles in 2019. Los Angeles, CA, 90024 (e-mail: attiano@ucla.edu).

The tri-weekly goals are broken down as follows. Week 3 involved both team members to gain an understanding of the dataset that involved handwritten words, the used software and having the software read in the dataset. Week 7 involved the complete implementation of the neural network on the training set with an acceptable level of error. Week 10 involved the completed speech output for any given handwritten word test dataset.

Due to the difficulty of the overall task, the short time frame of the project, and coupled with having a beginner's knowledge of the field of machine learning the overall plan had to go through several weekly modifications. The dataset we had originally planned on using did not have an easily understood way of performing character separation. Additionally, the character separation techniques used involved a deep knowledge of individual word processing and filtering techniques that we did not feel comfortable implementing without gaining a thorough understanding on. After speaking to the Professor during week 3 about our concerns, we ended up finding two datasets that are great for beginner students who want to learn text processing using machine learning, the MNIST dataset and the Handwritten names dataset. By week 7 we have implemented the MNIST dataset with outputted speech, however with a low accuracy when processing handwritten digits. The use of a convolutional neural network was scrapped because the deep neural network holds inherent feature extraction and for the implementation of this project is unnecessary. By week 10 both datasets were implemented with an acceptable level of error and outputted handwritten digits and names to speech.

C. Standard

A beginner's guide to machine learning starts with the ability for the computer to recognize handwritten digits within the MNIST dataset. The general knowledge is found directly from the TensorFlow website tutorials [3]. To understand the tutorial and its implementation of the deep neural network used, and additional tutorial of Keras is needed [4]. Processing of the digit images followed a standard threshold technique to blacken and enhance pixels for the computer to recognize the test digit. The handwritten names data set follows template matching and character separation. Binarization after character separation follows a simple function call. The speech output is by a function call inherent to the software being used [5].

D. Theory

In order for the computer to intake the datasets, broad knowledge of image processing is required. The MNIST dataset had a simple threshold to distinguish between the darker and lighter pixels. Deep neural networks were performed with Keras and Tensorflow as outlined by Code Snippet 1. Much of the image processing performed with the handwriting dataset was done through the use of OpenCV and Pandas image processing and template matching functions. A clustering algorithm is used to separate the characters form the handwritten word. Bounding boxes were used to surround the characters in order to distinct characters from one another. Deep neural networks contain inherent feature extraction and therefore the use of a convolutional neural network would be redundant for our simple case of feature extraction. Speech output for both datasets were performed with an inherent windows speech client available for Windows 10. In order for the speech command to work on other operating systems other specific clients must be called for the specific operating system. For the handwritten names dataset the individual character images had to be converted to a string and concatenated to resemble the desired name for a proper pronunciation to be outputted to speech.

E. Software / Hardware

The hardware used for this project was a single desktop computer with an Intel i7 8700k CPU and 32 Gbytes of RAM controlling most of the processing. Speech output was noted though the computer speakers. Pictures of the test images were taken with an iPhone 5 and 6.

The computer software used to compile the project is a lightweight and user configurable compiler Visual Studio Code on a Windows 10 operating system. The programming language used through the entire project is Python. This is because the language has many readily usable machine learning libraries and tools that support machine learning functions. For the MNIST dataset, the machine learning functions used are Tensorflow with Keras implemented over it. For the handwriting dataset the machine learning libraries used were Sci-Kit Learn [6]. Both image processing and text recognition tools were taken from the OpenCV library [7].

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256,
activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(128,
activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(10,
activation=tf.nn.softmax))
```

Code Snippet 1: The above code is the deep neural network constructed with Keras to classify the 10 digits.

```
speak = wincl.Dispatch("SAPI.SpVoice")
for prediction in predictions:
    output = "The prediction for the number
input is " + to_str(np.argmax(prediction))
    print("The prediction for the number
input is: ", np.argmax(prediction))
    speak.Speak(output)
```

Code Snippet 2: The above code which would concatenate the characters from a name and output it to speech.

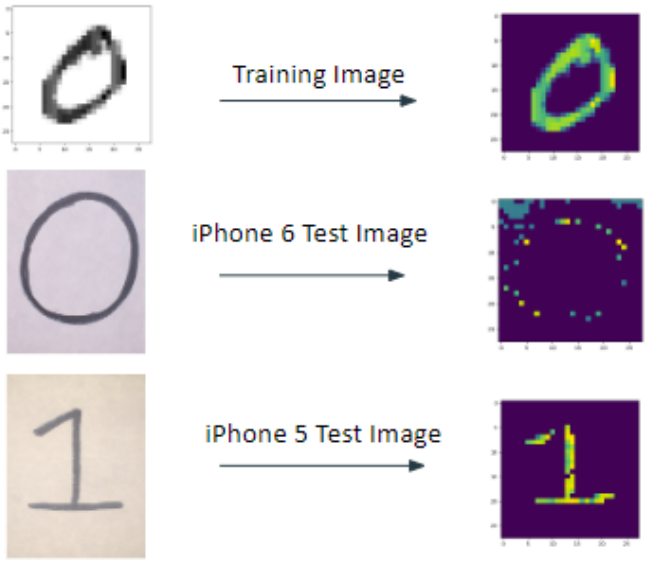

Figure 2: MNIST Processed Images and Training Set

F. *Procedure*

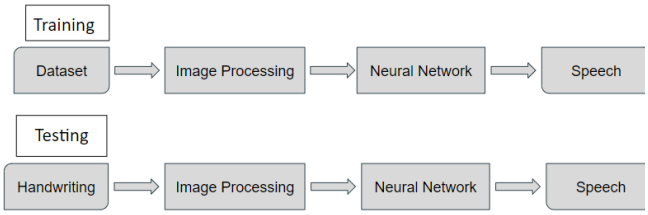

Figure 1: High-Level Approach of Project Implementation

The project implementation is shown as an overall flow diagram in Figure 1. Generally, put, a dataset is first read through our program where each image picture taken from the dataset is processed. The next step involves passing the processed images though the deep neural network for feature extraction and mapping. The datasets are trained several times to ensure highest accuracy and low tolerance for error. After the neural networks were fully trained a test image was taken with the iPhone 5 and passed though out system until the proper speech was uttered from the computer speakers.

Datasets were carefully chosen to fit the basics of neural network processing and character separation. The MNIST dataset is comprised of 60,000 handwritten digits as a training set with a test set of 10,000 [8]. The test set was hand drawn by one of us, and several images were processed to look like a heatmap image shown un the upper portion of Figure 2. The test images must match the MMIST training set image in order for the computer to be able to recognize the image and output the appropriate response. Each test image was process by first normalizing the RGB pixel values to a range from 0 to 254. A threshold was placed so that any pixels that were not apart of the text were turned 0 and any pixel above the threshold had their values doubled. The neural network is comprised of three layers ending with ten nodes which comprise the digits form 0 through 9. The code snippet shows how the training sets are output to speech.

The handwriting names dataset is comprised of over 125,000 names, where most of the names are French which adds the difficulty of recognition given the accent marks over certain characters [9]. The approach for the project follows in order: preprocessing of the dataset, preprocessing of the images, feature extraction, classifiers, interface, and validation. Much of the dataset preprocessing has been completed and formatted into a CSV file with URL links to the images of the handwritten words. The database itself was too large to fully download the entirety of the names as it would take several days to later train. A compromise was meet to process and train the dataset with an acceptable level of error by only utilizing 10,000 names in the dataset. Images of the words contain noise or unwanted text, such as NOM or PRENOM, because the original names were stored and printed through a machine. A template matching algorithm from the library OpenCV [10] is used to match the handwritten words with the machine text OM. What's left is the handwritten text which can then have their characters extracted. The characters are first binarized and then sent through a clustering algorithm and rescaled to a 28 x 28 image. Feature extraction is performed with a deep neural network due to its inherent feature extraction. The interface is the testing data, which is a test sample of our very own names to go though the process as outlined above. A validation report of the Scikit metrics are then shown to display the accuracy of the tolerance and fit of the training data as well as the testing data accuracy [11].

IV. Results

We had extremely good character level recognition results for both the MNIST and Handwritten Names dataset.

A. *Description of Results*

The results for the MNIST data set were stellar. We had 98% accuracy on the test set after training the artificial neural network for three epochs. In addition we were able to successfully convert images of digits taken with the Iphone 5 with high contrast lighting to the same form as that of the images from the MNIST dataset. We were not able to evaluate the percentage accuracy of images taken with an Iphone in the same way that we were able to evaluate the percentage accuracy on the test set of the MNIST dataset because we did not have enough sample images, we tried approximately 10 images of 5 different numbers. The MNIST dataset test set was approximately ten thousand images. Our algorithm to convert the Iphone images produced images that when passed into the trained neural network had varying results, but the varying results made sense, sometimes 7's were confused with 2's for example. See Figure 1. We were able to then output the predicted digit in speech.

Figure 1: MNIST ANN Iphone image results

B. *Discussion of results*

The results for the Handwritten Names dataset were also quite good, though not as great as that of the MNIST dataset. We had roughly 93% accuracy on the character level recognition and about 63% on entire word recognition. See Figure 2. We were not able to successfully convert images taken with the Iphone to that of the same form of the Handwritten Names dataset as the character segmentation in the Iphone images was considerably more difficult than those from the dataset. The size of the images in the Handwritten Names dataset was approximately 24x320 whereas the images taken from the iphone were closer to 1500x3000. This made it difficult for the Iphone images to maintain the same aspect ratio as the handwritten names dataset while shrinking it down to the same size to input to the neural network. In addition the Iphone image contained less contrast and had a slower gradient from black text to white background making it hard to segment based on intensity. We were however able to output the predicted text as speech. There were errors in full name predictions that made sense, such as an I being confused for a T, or an E being confused with an F. See Figure 3.

MLP only classification:

test_without

	precision	recall	f1-score	support
-	0.86	0.25	0.39	24
A	0.93	0.95	0.94	1686
B	0.69	0.82	0.75	98
C	0.92	0.90	0.91	305
D	0.88	0.78	0.83	183
E	0.96	0.91	0.93	1471
F	0.59	0.82	0.68	61
G	0.88	0.84	0.86	110
H	0.81	0.83	0.82	327
I	0.89	0.93	0.91	979
J	0.81	0.83	0.82	109
K	0.92	0.73	0.81	48
L	0.96	0.95	0.96	1000
M	0.91	0.85	0.88	571
N	0.91	0.95	0.93	1060
O	0.93	0.90	0.92	598
P	0.64	0.89	0.75	93
Q	0.80	0.29	0.42	14
R	0.94	0.89	0.92	537
S	0.90	0.95	0.92	402
T	0.90	0.96	0.93	518
U	0.88	0.90	0.89	374
V	0.86	0.75	0.80	105
W	0.76	0.74	0.75	38
X	0.92	0.91	0.92	102
Y	0.88	0.91	0.89	163
Z	0.93	0.70	0.80	37
micro avg	0.91	0.91	0.91	11013
macro avg	0.86	0.82	0.83	11013
weighted avg	0.91	0.91	0.91	11013

Full name test results:

Classifier	Correct percentage	Correlation ratio
MLP only	0.6384451544195953	0.9141435512066829

Figure 2: Handwritten Names ANN (MLP) character and whole word recognition results

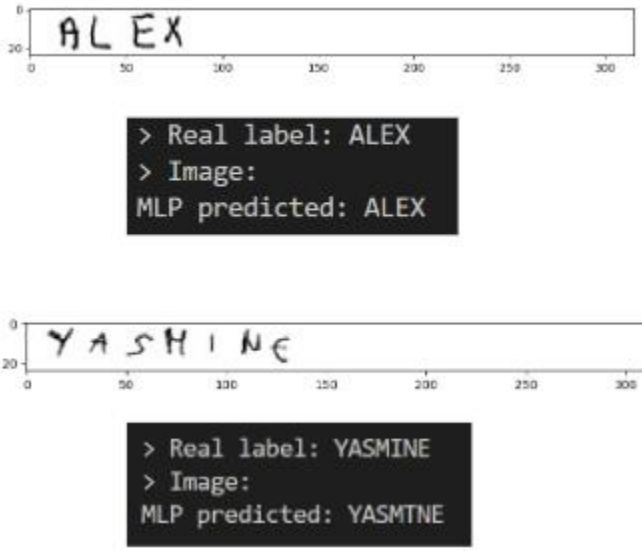

Figure 3: Handwritten Names ANN (MLP) example whole word recognition results

REFERENCES

[1] Schantz, H. F., "*The history of OCR: optical character recognition,"* in *Recognition Technologies Users Association*. 1st ed. Denver, CO, USA: 1982

[2] "The First OCR System: 'GISMO'1951," *The First OCR System: "GISMO" : HistoryofInformation.com*, 1951. [Online]. Available: http://www.historyofinformation.com/detail.php?entryid=885. [Accessed: 20-Mar-2019].

[3] "Getting Started with TensorFlow," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/tutorials. [Accessed: 22-Mar-2019].

[4] "Keras," *TensorFlow*."[Online].Available: https://www.tensorflow.org/guide/keras. [Accessed: 22-Mar-2019].

[5] "Text to speech," *Python*, 08-Mar-2019. [Online]. Available: https://pythonprogramminglanguage.com/text-to-speech/. [Accessed: 22-Mar-2019].

[6] "Sci-Kit Learn," *scikit*. [Online]. Available: https://scikit-learn.org/stable/. [Accessed: 22-Mar-2019].

[7] "OpenCV library," *OpenCV library*. [Online]. Available: https://opencv.org/. [Accessed: 22-Mar-2019].

[8] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST Database," *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 22-Mar-2019].

[9] "Data For Everyone," *Figure Eight*. [Online]. Available: https://www.figure-eight.com/data-for-everyone/. [Accessed: 22-Mar-2019].

[10] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[11] I. Garcia and E. Yerga, “Name Identification using ANNs,” *GitHub*.