Neural Network for Realtime Handwriting Recognition

Nomaira Javaed   
*Computer Science*Indianapolis, USA  
[njavaed@iu.edu](mailto:njavaed@iu.edu)   
  
Roman Jennings   
*Computer Science*Indianapolis, USA  
  
  
Mory Kaba   
*Computer Science*Indianapolis, USA  
[djkaba@iu.edu](mailto:djkaba@iu.edu)   
  
Emmanuel Kareem   
*Computer Science*Indianapolis, USA  
[ekareem@iu.edu](mailto:ekareem@iu.edu)

Noah Khamliche   
*Computer Science*

Indianapolis, USA  
[nkhamlic@iu.edu](mailto:nkhamlic@iu.edu)

Luke Kim  
*Computer Science*Indianapolis, USA  
[lukim@iu.edu](mailto:lukim@iu.edu)

Meenakshi Kommineni   
*Computer Science*

Indianapolis, USA  
[mkommin@iu.edu](mailto:mkommin@iu.edu)

Jianbin Lao  
*Computer Science*

Indianapolis, USA  
[jianlao@iu.edu](mailto:jianlao@iu.edu)

*Abstract*—The purpose is to train a neural network to recognize a person’s handwriting. The user enters each letter one by one in a UI window during the data collection period. The users input data will be passed to the Capsule Neural Network, which will compute the geometry of the image, and map it to the current user, creating a user specific model. These models will be saved and retrieved for each user. Once the user finishes the training period, the user will be recognized after submitting a writing sample in the application.

Keywords—convolutional neural networks, training, data, capsule neural networks, handwriting

# Introduction

This project is about training a neural network to recognize the handwriting of a person. The MNIST database is used for this purpose. We start with working with capsule neural networks and then moving on to convolutional neural networks. Capsule neural networks (NN) have some advantages over convolutional neural networks which are discussed in this paper.

## Why chose neural networks?

There are many forms of handwriting styles. With the addition of digital writing we need a system that can work with unexpected writing styles, characters and symbols. Before deep learning, we had Optical Character Recognition (OCR) which was based on hardcoded features. This method did not work for unexpected characters as its features were manually added. Neural network works with unexpected characters as well. This is why it is important to use neural networks for handwriting recognition.

The neural networks use an activation function called a sigmoid function. In simple terms this sigmoid function converts any given real number to a number between 0 and 1. It only works with normalized values. It helps in logical regression. The plot of a sigmoid function is shown in *Fig. 2*

# importance of handwriting recognition

Handwriting recognition is important in several domains. Two of the most important domains are:

## Security

Handwriting recognition helps keep our banks secure. It will immediately recognize if the person who is signing the check owns the account or not.

## Forensics

Handwriting recognition can help in solving criminal cases by matching the handwriting of a person to a database.

A picture containing drawing, necklace

Description automatically generated

Fig. 1. Layers of capsule neural network

*A close up of a piece of paper

Description automatically generated*

Fig. 2. Layers of capsule neural network

# capsule neural networks

In a typical neural network only the output of a single unit is squashed by a non-linearity. We have a set of output neurons and based on the output of each we apply non-linearity to each of them. In Capsule neural network (CapsNet), instead of applying non-linearity to each individual neuron we group these neurons into a capsule and apply non-linearity to the entire set of neurons (the vector of those neurons).

Fig. 1 shows how we can think of layers of capsules arranged in a program. Each layer in the capsule when they forward propagate data, it goes to the next most relevant capsule.

# Convolutional Neural Networks

In convolutional neural networks (CNN), we have a filter that we slide over the input image. Here we have a binary image of a character that is drawn in the UI window. This filter moves across the entire image and multiply its values with the original pixel values. The summation of this gives us one number. This summation uses weights and biases. Adjusting weights and biases helps prevent overfitting and underfitting of data.

# comparison of capsule with convolutional

According to the article by Rachel Wiles, there two main advantages of CapsNets over regular CNNs.

1. Reduces the effects of spatial invariance which is found using CNN methods.
2. Reduces the amount of data required for training.

Furthermore, capsule neural network replaces max pooling by agreement. It replaces the scalar output feature detectors of convolutional neural network with vector output.

# application

## First Approach

In our application we started with using the Capsule neural networks. We chose to use a Capsule Networks specifically because they are more powerful than conventional Convolution Neural Networks. One of the main capabilities of a capsule network is to detect order. Below is a simple example where CNN’s fail to recognize correctly but Capsule Networks recognize the image as correct.

*Fig. 3* image is of a face that is deconstructed. To a CNN, this image has the correct face shape, two eyes, a mouth, a nose. All of the things that a face has. Even the best of CNN facial recognition networks will fail some of the time on images such as this one. Capsule Networks mitigate this hiccup by keeping track of the individual position and order of these elements, rather than if they exist in the image or not. This will help with the variability in rotation and scale of writing. Using a traditional CNN, we would need to line up the images every time we needed to run an image through the network. Using Capsule Networks allows us to screen capture the GUI rather than fit a specific image every time.

## Graphical User Interface (GUI)

Our Implementation uses a UI System that allows a user to draw in the window. The user then types the letter + “Shift” for a capital letter, and simply the letter for the lowercase letter. This allows the user to swiftly enter the characters with one hand and draw with the other, using a mouse or digital pen. The user enters each letter one by one during the data collection period. The users input data is passed to the Capsule Neural Network or Convolutional Neural Network, which computes the geometry of the image, and map it to the current user, creating a user specific model. These models are saved and retrieved for each user. Once the user finishes the training period, the user is recognized after submitting a writing sample in the application.

*Fig. 4.* Shows the GUI of our application which is developed using python. In addition, we have used tkinter and numpy libraries.

##### page2image25352912Fig. 3. A face with its features scattered

##### A screenshot of a cell phone Description automatically generated

Fig. 4. The GUI of our application

# achievements

Unfortunately, the Capsule Network we originally planned to have implemented for the recognition network, proved to be too difficult for our group to implement. Capsule Network implementation is an extremely complicated form of neural networks and require much more experience writing and building networks than any member of our group has. However, we did successfully implement a network that recognizes handwritten numbers. We utilized the MNIST dataset to train our model. The model consists of a recurrent neural network made up of the following layers.

1. LSTM Layer (Long Short-Term Memory)
2. Dense Layer

The LSTM layer processes sequences of data and not individual data points. This allows the layer to save the general shape of a number, and not the exact shape. Thus, the network can recognize slight changes to the structure of the number. The dense layer condenses the output of the of the LSTM into 10 classes for each number we are looking for, 0-10. The resulting structure allowed us to achieve an accuracy of 98% on a portion of the dataset the network had never seen before.

# Implementation

To implement the objective of the project for the Neural Network for Real-Time Handwriting Recognition we set a goal to follow 5 steps to achieve a working Real-Time Handwriting Recognition Application as follow:

1. Step 1: Installation of the database of images from EMNIST which is a dataset of handwritten character digits derived from the NIST special database 19. It has about 131600 handwritten characters images with their labels. We will use these to train a network. Then given a new image, the network should be able to classify it as the right handwritten characters.
2. Step 2: We will set up a neural network with the required number of layers and nodes, we will also tell the network how it has to train itself
3. Step 3: We will feed the training data to the neural network
4. Step 4: We will check how the output is for one image.
5. Step 5: We will feed a test dataset with thousand images to the trained neural network and then see its accuracy.

## Capturing Training Data

Input: It is gray-value image of size 128x32. Often, the images from the dataset do not get this size, therefore it is resized without distortion. Then copy the image into the target image of size 128x32. Finally normalize the gray-values of the image. The integration of copying images to random positions is best suited compare to the alignment of the images to the left or the random resizing of it.

## Training the Neural Network

It requires the creation of database to store all the inputs captured from different users in order to make the program learn from it. Its operation requires the structure of the neural network (NN) which consist of convolutional neural network (CNN) layers, the recurrent neural network (RNN) layers and the connectionist temporal classification (CTC) layers.

CNN: Gets the input image fed into its layers. These layers are trained to extract relevant features from the image. Each layer is composed of three operations. First, convolution applies a filter kernel of 5x5 in the first two layers and 3x3 in the last three layers to the input. Afterward, the non-linear RELU function is applied. Finally, a pooling layer summarizes image regions and outputs a downsized version of the input.

RNN: the feature sequence contains 256 features per time-step, it spreads relevant information via the sequence. The Long Short-Term Memory (LSTM) uses the implementation of RNN, like it is capable of spreading information via longer distances and produces a better robust training-characters. The RNN output sequence is mapped to a matrix of size 32x80. The IAM handwriting dataset which collect several writers data and then classify writers according to their writing styles but for this program not the full IAM dataset is required but some authentic features subset which can be used to training such as the top persons who contributed the most towards the dataset.

CTC: while training the NN, the CTC is given the RNN output matrix and the ground truth text and it computes the loss value. While finishing, the CTC is only given the matrix and it decodes it into the final text. Both the ground truth text and the recognized text can be at most 32 characters long.

## Testing Handwriting Recognition

Our test is predicting the input image which could be the most rewarding of all if all processes before it was properly implemented. *Fig. 5.* Shows how a user writes something into the window developed using python, tkinter and numpy. The user can enter words or characters in this window using either keyboard or directly entering text using GUI’s keyboard.

# Summary

The Real-Time Handwritten Recognition Application using Neural Network was hard to implement to our specific model therefore, we could not produce a working application to recognize user handwriting. We did our best to train the classifiers but the amount of time we had and not having enough experience in this field, we were not able to come up with a fully functional application. We have a solid foundation on how to implement neural networks but the calculations for this type of work are too intense to work on.

A close up of a logo

Description automatically generated

Fig. 5. The GUI application working example

##### References

1. HandWriting Recognition - Offline Approach. (n.d.). Retrieved from https://cs.stanford.edu/people/adityaj/HandwritingRecognition.pdf
2. Wiles, R. (2019, June 27). Have we solved the problem of handwriting recognition? Retrieved from <https://towardsdatascience.com/https-medium-com-rachelwiles-have-we-solved-the> problem-of-handwriting-recognition-712e279f373b