AIR-DRAWN-HAND-WRITTEN DIGIT RECOGNITION USING A MULTI-LAYER PERCEPTRON PATTERN RECOGNITION SYSTEM

Lappeenranta-Lahti University of Technology LUT

BM40A0702 Pattern Recognition and Machine Learning

Project: Digits 3-D

2023

Durbar Hore Partha (001473021) [Data Analysis, Feature Preprocessing, Documentation]

Olivier Rutherford Dondjio Nguefack (001676938) [Model Building, Model Testing, Documentation]

Socrates Waka Onyando (001690286) [Feature Preprocessing, Model Building, Documentation]

LIST OF ABBREVIATIONS

BP	Back Propagation
FF	Feed Forward
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Unit

CONTENTS

1 INTRODUCTION

1.1 Project Background

Recognizing handwritten digits created in the air generally poses challenges due to variability in human writing patterns and styles. The complexity of digit shapes brings about more difficulties in sensor technology and its output. Precisely deciphering these representations, captured by sensors, holds promise for various applications, such as document processing [1] [2], handwriting analysis [3], and user authentication [4].

1.2 Project Context

This project aims to develop a pattern recognition system for recognizing hand-drawn digits. This project focuses on recognizing free-hand stroke drawn in the air using the index finger with 3-D location information of the fingertips which are captured by LeapMotion sensor. The location information of each finger stroke is collected from using the LeapMotion sensor which uses sensor technology for obtaining the inputs. The stroke data was then piled as both Matlab (.mat) and comma-separated values(csv) file format. Each file was named accordingly to the digit and a unique identification number: `stroke_DIGIT_NUM.mat |CSV` where `DIGIT` represents the digit label and `NUM` expresses a sequential identifier for the stroke.

1.3 Project Objectives

The overarching objective of this project is to design and implement a pattern recognition system as a robust digit recognition system. Specifically, the aim is to classify the hand-drawn digits $\{0, 1, ..., 9\}$ correctly based on the time series derived from the 3-D fingertip location data.

1.4 Structure of Report

In this report, we present the methodologies for feature extraction as well as an intuitive explanation of the classifier used in the pattern recognition system. Finally, the results of our pattern recognition system on a small test dataset are presented.

2 METHODOLOGY

The section is divided into three major parts: feature preprocessing, digit classification system and performance evaluation.

2.1 Feature Preprocessing

2.1.1 Trajectory Smoothing

Given the nature of the data collection process, some jaggedness in the trajectory of the fingertip location is inherent. This jaggedness can be caused by a number of factors, including: sensor noise, hand tremors and variations in drawing styles (some people draw their digits with smooth, fluid strokes, while others draw them with more jagged, shaky strokes). To deal with such issues, smoothing the trajectory of fingertip location is beneficial since it can make it easier for the classification algorithm to identify the true shape of the digit.

Equations 1, 2 and 3 were applied to the fingertip 3-D data to smooth the trajectory.

$$x_t = \frac{1}{n}\sum_{t=0}^{n} x_t \tag{1}$$

$$y_t = \frac{1}{n}\sum_{t=0}^{n} y_t \tag{2}$$

$$z_t = \frac{1}{n}\sum_{t=0}^{n} z_t \tag{3}$$

where t is the individual position in a trajectory and n is the number of points used for smoothing [2].

After experimentation, it was established that the optimal n would be 5 in the case of the present dataset (relative to the performance of the classifier). Moreover, using a too high n smooths the trajectory too much, which can distort the shape of the digit [2]. Using a lower value on the other hand does not normalize the trajectory adequately [2].

2.1.2 Root Point Translation

It is also inherent that data collected using the sensor will have different initialisation/starting points subject to the person writing the digits. To address this issue, root point translation is employed to generalize the starting point of each digit. This technique involves shifting all the starting points of the digits to a common origin. By removing the impact of random starting positions, the digit recognition system can focus on the intrinsic features of the digit itself.

Equations 4, 5 and 6 were used for this translation operation.

$$x_{t_{new}} = x_t - x_0 \tag{4}$$

$$y_{t_{new}} = y_t - y_0 \tag{5}$$

$$z_{t_{new}} = z_t - z_0 \tag{6}$$

where $[x_t, y_t, z_t]$ is the location co-ordinate at position t before translation, $[x_0, y_0, z_0]$ is the starting co-ordinate of a sequence and $[x_{t_{new}}, y_{t_{new}}, z_{t_{new}}]$ is the location co-ordinate at position t after translation [2].

Figures 1 and 2 show the impact of trajectory smoothing and root point translation on the 3D trajectory path. Specifically, Figure 1 shows a sample trajectory before applying the aforementioned preprocessing steps whilst 2 shows the same trajectory after preprocessing.

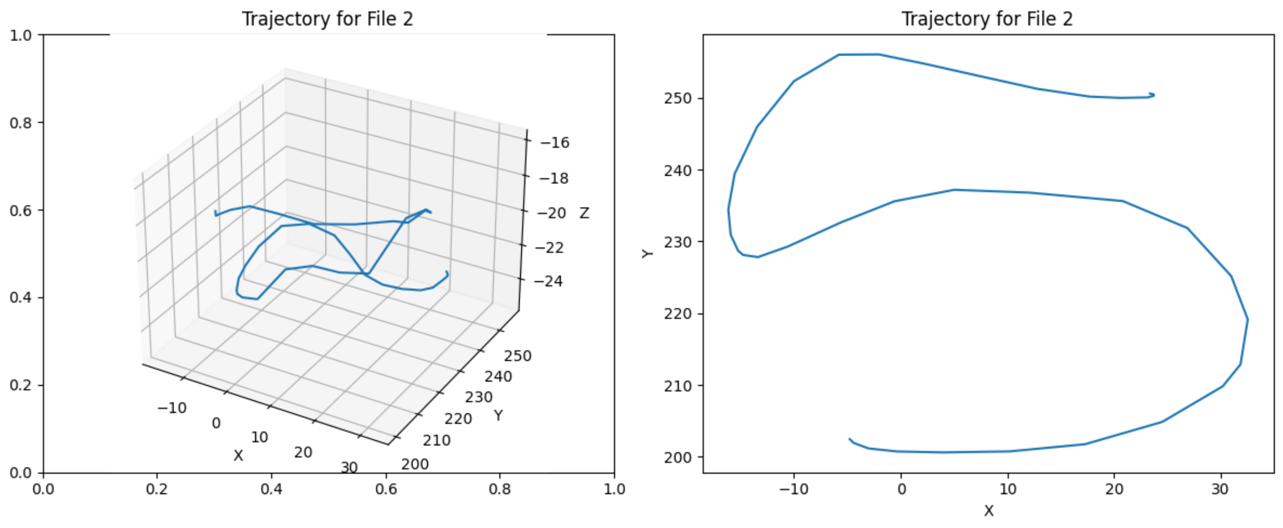


Figure 1. Before trajectory smoothing and root point translation

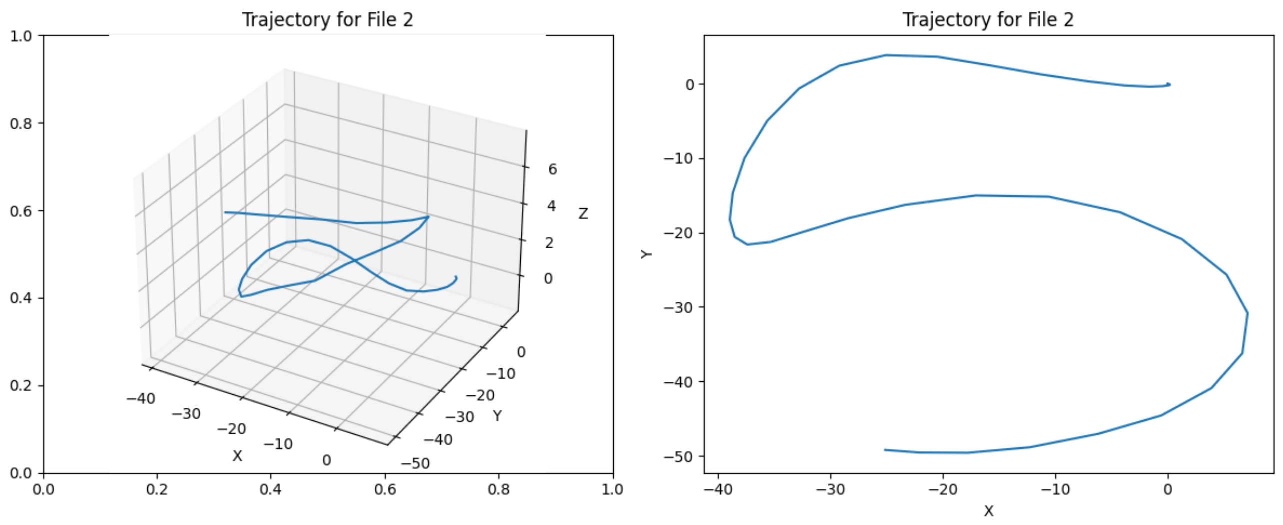


Figure 2. After trajectory smoothing and root point translation

2.1.3 Pre-sequence Zero Padding

Another inherent issue with the data is that of inconsistent/irregular sequence lengths. Time-series data frequently exhibits varying lengths [5]. This poses challenges when employing machine learning algorithms that demand uniform-length input sequences. A good example is the scenario where two users draw the same digit (e.g., 3) but with different lengths of fingertip location data.

It was therefore necessary to pad the sequences. For the current project, zero padding was utilised, in which case zeroes were added to the start of the sequence until the desired length was reached. The logic for utilising pre-sequence zero padding was because of the fact that after root point translation, the initial starting point for all sequences would be $[0, 0, 0]$ coupled with the fact that adding zeroes before the sequence would not distort the shape. Thus, pre-sequence zero padding enabled the preservation of the spatial relationships and temporal patterns inherent in the fingertip location data, whilst ensuring all the sequences had a consistent length.

2.1.4 Data Flattening

This step in the preprocessing pipeline was influenced by the chosen classifier architecture (Multi-Layer Perceptron (MLP)). The logic behind flattening is to transform the 3-D structure into a linear, one-dimensional representation that can be easily processed by the MLP's dense layers.

2.2 MLP

2.2.1 Algorithm

A MLP is a type of artificial neural network that consists of multiple layers of neurons, each connected to the next [6]. There are two major operations that take place in a MLP network: the Feed Forward (FF) phase and the Back Propagation (BP) phase.

- **FF Phase**

  For the FF operations, information is transmitted from the input layer through the hidden layers and out to the output layer. Therefore, each neuron receives information from all the neurons in the previous layer of neurons. Thereafter, the neuron applies an activation function to this input and transmits its output to all neurons in the next layer [6]. Some of the common activation functions include the sigmoid function, hyperbolic tangent function, Rectified Linear Unit (ReLU) function and Softmax function [6].

- **BP Phase**

  For the BP operations, the MLP network learns by adjusting its weights to minimize the difference between the actual and predicted values. This can be achieved through gradient descent method. Using this method, errors are computed for each output using a specified cost function. This error is then propagated backwards through the network, adjusting the weights along the way [6].

These two operations are carried out until an optimal set of weights is arrived at.

2.2.2 Project Model Architecture

The Multilayer Perceptron (MLP) architecture used in this project works to classify the 10 digits in the dataset. For the activation functions, we have used ReLU function for the hidden layer and Softmax for the output layer. We have built a function to intialise the weights and biases randomly. The weights are initialised using a small random value, and the biases are initialized at zero.

For the feed forward pass, the input data is changed using the initialised weights and biases of the hidden layer. The ReLU activation function is then applied element-wise to introduce non-linearity. The result of this operation is then passed through the weights and biases of the output layer. Afterwards, the Softmax activation function is applied. This guarantees that the output values represent probabilities corresponding to each class.

For the loss function, we have used categorical cross-entropy loss. The goal during training is to minimize this loss. The backward pass, or backpropagation, calculates the gradients of the loss with respect to the model parameters (weights and biases) and updates these parameters using gradient descent. This process iterates over the training data for a specified number of epochs, adjusting the model to improve its performance. The architecture is flexible, allowing adjustments to the number of neurons in the hidden layer, the learning rate, and the number of training epochs.

2.3 Performance Evaluation

2.3.1 Accuracy

Accuracy is a performance evaluation measure commonly used in classification problems to assess the overall correctness of a model's predictions. It is defined as the ratio of correctly predicted instances to the total number of instances in the dataset and is given by Equation 7.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{7}$$

In terms of interpretation, it is intuitive that the higher the accuracy the better the classifier.

2.3.2 Time

To evaluate the computational efficiency of the classifier, the computational time across several runs will be evaluated and assessed, with a less time being more favourable.

2.4 Software and Code

The project's analyses were conducted using Python within the Google Colab environment.

3 RESULTS

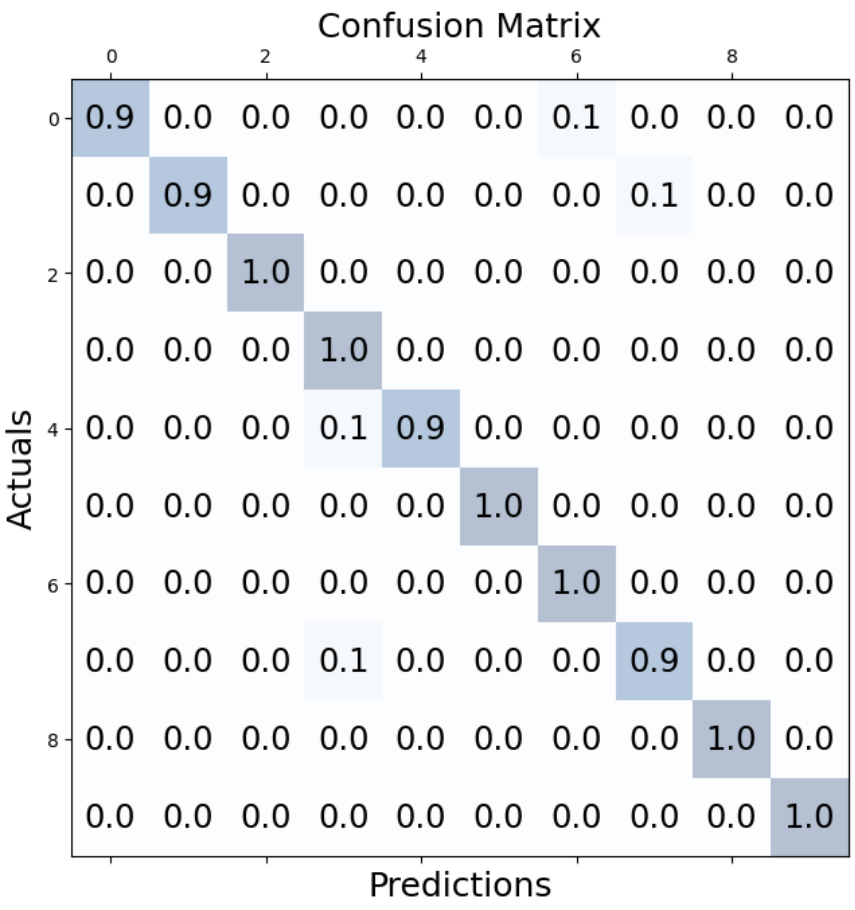
3.1 Hyperparameters

Whilst the tuning of the hyperparameters was not done within a formal experimental setup, manual adjustments to the parameters was done to evaluate the model's performance to concerned parameter's adjustments. Particularly, experimentation was done on the number of epochs, learning rate and random state initialisation of the weights of the network.

3.2 Model Performance

3.2.1 Accuracy

In order to validate the model, 10% of the data was used as the test set. Whilst using random samples as the test set across different runs, the obtained accuracy scores varied between 90% and 96%. Figure 3 illustrates the confusion matrix of one of the runs.



	0		2		4		6		8	
0	0.9	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0
	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.1	0.9	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.9	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Figure 3. Confusion Matrix

3.2.2 Time

The MLP classifier performed well in terms of computational efficiency with the network requiring less than a minute to train across various training runs.

REFERENCES

[1] Chengzhang Qu, Dengyi Zhang, and Jing Tian. Online kinect handwritten digit recognition based on dynamic time warping and support vector machine. *Journal Of Information &Computational Science*, 12(1):413–422, 2015.

[2] Md Shahinur Alam, Ki-Chul Kwon, Md Ashraful Alam, Mohammed Y Abbass, Shariar Md Imtiaz, and Nam Kim. Trajectory-based air-writing recognition using deep neural network and depth sensor. *Sensors*, 20(2):376, 2020.

[3] Andreas Klaß, Sven M Lorenz, Martin W Lauer-Schmaltz, David Rügamer, Bernd Bischl, Christopher Mutschler, and Felix Ott. Uncertainty-aware evaluation of time-series classification for online handwriting recognition with domain shift. *arXiv preprint arXiv:2206.08640*, 2022.

[4] Liyang Xie, Zhongcheng Wu, Xian Zhang, Yong Li, and Xinkuang Wang. Writer-independent online signature verification based on 2d representation of time series data using triplet supervised network. *Measurement*, 197:111312, 2022.

[5] Philip B Weerakody, Kok Wai Wong, Guanjin Wang, and Wendell Ela. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, 441:161–178, 2021.

[6] Rudolf Kruse, Christian Borgelt, Christian Braune, Sanaz Mostaghim, and Matthias Steinbrecher. *Multilayer Perceptrons*, pages 47–92. Springer London, London, 2016.