ELEC4848 Senior Design Project 2022-2023

[Lo Ngai Chak] (3035685859)

[Deep-Learning-Based IMU writing glove]

Supervisor: Dr. Shiming Zhang

Second Examiner: Dr. G.W.Y. Fu

Abstract

Image-based approaches are used for most digital handwriting. The standard approach is optical character recognition in electronic devices such as smartphones and smart pads. In this paper, a deep-learning-based writing glove is designed and implemented to allow the user to perform text input through an IMU-based wearable device. The writing glove is designed to recognize 10 numbers with an inertial measurement unit (IMU) as the only type of sensor. Values drift in IMU is one of the main challenges to designing an IMU device. A customized filtering algorithm is designed and implemented to handle this challenge. Two types of deep-learning methods are trained and tested with the collected data of user writing. The methods include Artificial Neural Networks (ANN), and Convolution Neural Networks (CNN). The CNN outperformed ANN which achieves 92.2% testing accuracy. This device is designed as a text-inputting alternative that could be applied in scenarios that intend to provide high body mobility for user including virtual reality and augmented reality settings.

Keywords: handwritten character recognition, inertial measurement unit, deep learning

Acknowledgements

Firstly, I would like to show my gratitude to the Department of Electrical and Electronic Engineering, and Dr. Shiming Zhang for giving me a valuable chance to carry out the research, and for the support, they offer me for the project. I would also like to acknowledge the senior Ph.D. student, Xinyu Tian, who offer rich support on technical aspects and provide valuable suggestions during my research. Last, I would like to thank Miss Mable Choi for assisting me a lot regarding the technical reports.

1. 1. Introduction

1.1 Background

From the research carried out by B.Z Zhao, H.Z Lu, S.F Chen, J.L Liu, Convolution Neural Network (CNN) was experimented with to analyze against 8 sets of time series datasets, where a time series dataset is a sequence of measurements made over time. It was concluded that CNN can extract the internal structure of the time series data automatically and deep features for classification can be generated. [1]. Due to the time-series nature of the IMU data, it could be expected that CNN could give a better performance compared to traditional machine-learning approaches.

From the study of T.T. Alemayoh, M. Shintani, J.H. Lee and S. Okamoto. A force sensor and a 9-degree IMU was attached to a pen to a pen to classify 10 numerical numbers and 26 alphabetical characters using deep-learning approaches. CNN and ANN were implemented in that study, where CNN achieved 96.9% testing accuracy. ANN achieved 95.51% testing accuracy [2].

1.2 Objective

This project aims to design a wearable glove device with an inertia measure unit (IMU), which allow the user to draw numerical numbers [0-9] with the user's index finger. This study aims to implement Neural network models to classify the IMU data generated during the user's finger movements.

1.3 Project contribution

Compared to ordinary input approaches such as keyboards, mouses, and touchscreen, the biggest upside of the IMU keyboard glove is mobility. The mobility means that it will not restrict users' hand placement at all, the users will not be required to hold a controller with physical keys on it, which could potentially be utilized in a virtual reality environment, such as a virtual reality office. Consequently, it is hoped that this project could provide insight into the VR or AR companies that are developing new types of input approaches.

1.4 Outline of the report

The report is structured into four chapters.
Chapter 1 provides an introduction of the report.
Chapter 2 analyses and describe the methodology used in this project. The hardware components of the writing glove, software components including the filtering system of IMU data, data specification, data collection system, and proposed neural network models.
Chapter Five provides the conclusion of the report.

2. Methodology

In this section, the methodologies of the hardware system, software system, and deep-learning system are introduced. The hardware methodology includes ESP32, circuit design, and the assembly of the writing glove. Additionally, the software methodology includes data transmission systems, and IMU filtering systems. Finally, the deep-learning system includes data specification, data pre-processing, components of deep-learning models, and the proposed architecture of deep-learning models.

2.1 Glove components

In this section, the hardware of the IMU glove will be explained in detail. The glove consisted of an MPU6050, which is a 6-DOF IMU connected to the ESP32 microcontroller. The MPU6050 was stabled on the index finger location on a workout glove. The IMU data are delivered to a laptop by serial transmission to perform neural network computation. The two images below illustrate the glove connects to breadboard, the glove wear by a user.

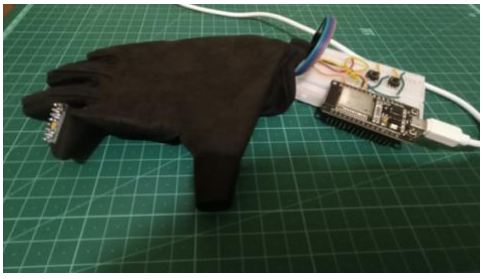

Fig 1: Writing glove connected to breadboard

Fig 2: Writing glove wear by a user

Fig 3 shows the circuit connection between ESP32 to two button and one IMU (MPU6050) on a breadboard.

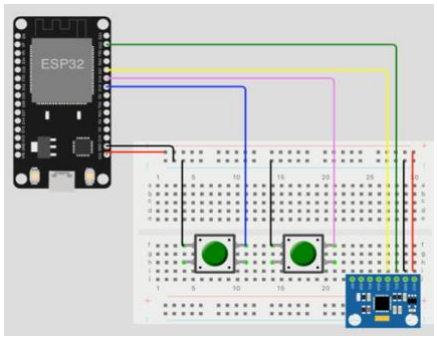

Fig 3 ESP32 Circuit diagram

In the design of the above circuit connection, the ESP32 is connected to two press buttons and one IMU (MPU6050). The left button is designed for resetting IMU data to zero manually when the user's hand is stationary, such design to make due to the problem of MPU6050 producing non-zero inertial measurement when the user's hand is stationary after a certain period of writing. The right button is designed to make an indication for the system that the user has started writing.

It should be noticed that the MPU6050 in the real setup is not located on the breadboard, it is located on the index finger position on the user's writing glove.

2.2 System overview

An overview of the system architecture is introduced in this section. Firstly, IMU data is transmitted from ESP32 to MacBook Computer through USB. The mechanism of transmission is USB to UART, which connects a microcontroller or other serial device to a computer via USB.

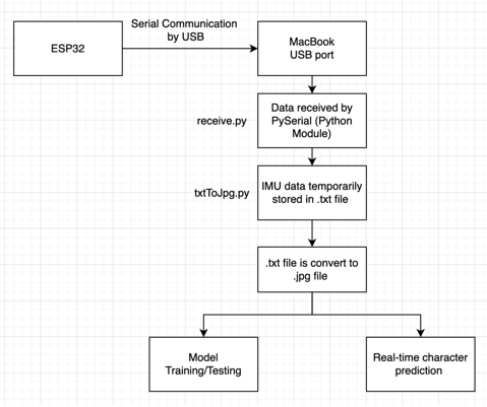

Fig 4 Data flow diagram

The data is received by a Python program (receive.py) which is built on top of pyserial, a Python library that support serial communication over a range of devices. "receive.py" is responsible to receive the IMU data and convert it to .txt files for temporary storage in the computer. Afterward, another Python program (txtToJpg.py) is responsible to convert the accumulated .txt files into .jpg image files for the convenience of model training or model testing process in the Pytorch framework.

A similar data flow is adopted for the real-time character prediction system, in which the user could receive the prediction result immediately after the writing motion is performed by the user.

2.3 Software System on ESP32

The ESP32 is responsible to receive the IMU sensor values and the inputs from the buttons and transmitting them to the computer. Additionally, IMU sensor values are processed by a filtering algorithm before they are passed to the computer.

2.3.1 Filtering System for IMU Data

Filtering is needed for IMU data because it is inherently noisy and prone to errors. As these measurements are taken, errors can accumulate, resulting in drift and a loss of accuracy over time. In this study, a customized IMU filtering system is designed and implemented.

The IMU data is observed when the IMU is stationary after certain hand movements. It is observed that the IMU data drifted to a certain bias value after certain movements are performed on the IMU. For example, after the user moved his hand and stopped the movement, the IMU values of one of the sensor degrees could be [4.08, 3.97, 4.03, 3.99, 4.02, 3.94, 4.00].

As the IMU data drifted to a certain range of bias values. The below algorithm is designed to measure the average of the drifted bias value of each degree of measurement. Firstly, this algorithm assumes that the user's hand is stationary, and the bias values are captured. For example, one of the degrees of the IMU could be [4.08, 3.97, 4.03, 3.99, 4.02, 3.94, 4.00, …]. Secondly, the algorithm calculates the average bias value based on the designed sample size. If the sample size is 6, the average value would be

$$\frac{4.08 + 3.97 + 4.03 + 3.99 + 4.02 + 3.94}{6} = 4.005$$

Thirdly, the upcoming measurement values are subtracted by the average bias value. If the upcoming measurement are [3.99, 4.02, 4.13, 3.92], the new upcoming values are

$$[\,(3.99 - 4.005)\,, (4.02 - 4.005), (4.13 - 4.005), (3.92 - 4.005)]$$

$$= [-0.015, 0.015, 0.125, -0.085]$$

The subtraction aims to reset the IMU reading back to around zero when the user's hand is stationary.

Fig 5 below illustrates the filter algorithm in a flow chart.

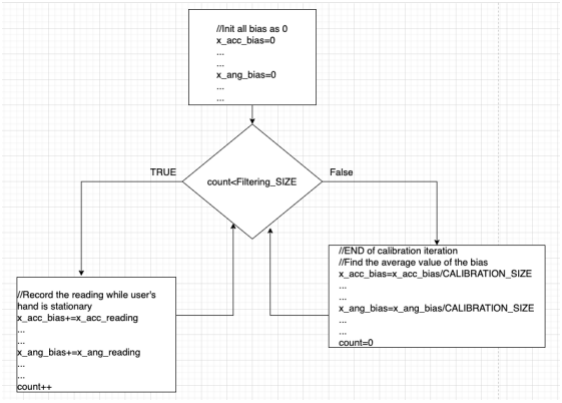

Fig 5 Filter system flow chart

At the beginning of the filtering process, the bias values of the IMU are set to zero for capturing the noise. Each iteration would capture the current bias value for all 6 degrees of the IMU measurement and add them to the corresponding bias term. When the count reached the filter sample size, denote as Filtering_SIZE above, the algorithm would calculate the average bias for all 6-degree measurements and update these values to the bias variable. In the serial transmission of IMU data, the values of (real

IMU measurement minus bias value) are transmitted.

2.4 Data Transmission System

A filter system is designed to measure the average bias value and adjust the IMU values back to zeros before the character writing, which is discussed in the previous section. The finite state diagram below illustrates the transition between the filtering process and the data transmission process under the control of a button.

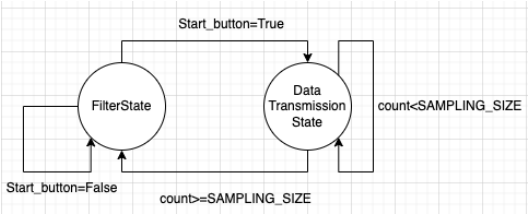

Fig 6: Data transmission finite state diagram

There are two states in the finite state diagram, which are the filtering state and the data transmission state. The filtering state is responsible to reset the IMU data to zero when the user is not performing the writing motion and assumes the user is stationary. And the data transmission state is the state where the user is writing a character.

In the filtering state, pressing the start button notifies the system that the user starts the writing process now, and the state is transited into a data transmission state. Otherwise, the state remains in the filter state if the start button is not pressed.

In the data transmission state, the state remained the same before accumulating the designated sample size of IMU. Upon completion of sampling, the state is transited back to the filtering state. It could be noticed that no filtering process occurred during the IMU data sampling in the data transmission state.

2.5 Data Specification

In this section, the data collection procedure will be introduced. All 6 degrees of IMU sensor values are captured, including 3 axes of linear acceleration and 3 axes of rotation rate. For each sample collection, the "start" button was connected to the ESP32 is pressed, and then 22 sets of IMU data were collected in approximately 0.8 seconds of the drawing period. Therefore, one sample had a sample shape of (H=22, W=6). It is also valid to treat one sample as an image with a height of 22 pixels and a width of 6 pixels respectively.

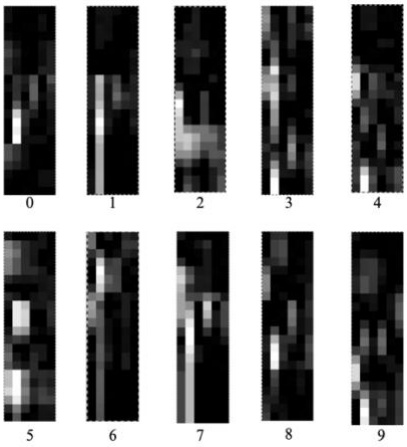

Fig 7 Sample input images for IMU data

The dataset could be classified into 2 categories that are training dataset and testing dataset. The training dataset is the dataset to be fed into the neural networks which were used for modifying the parameter of the network. For each numerical character, 300 training data were collected. In contrast, the testing dataset is the dataset for testing the

accuracy of the model, while this dataset is unseen during the training phase which would not be applied to update the model. For each numerical character, 20 testing data were collected.

2.6 Data Pre-processing

Before the training procedure, the dataset was shuffled in random order. And then every single sample was normalized to speed up the training process, followed by splitting into training and testing datasets. The same training dataset and testing dataset were applied in the two models which are ANN and CNN to produce a fair result.

2.7 Deep Learning

In this section, the ANN and CNN architectures that are implemented in this study are discussed

2.7.1 Artificial Neural Network (ANN)

ANN is regarded as the classical network in deep learning. The first layer of an ANN is called the input layer as the input values are passed to this layer at the beginning. The last layer is called the output layer which displays the results of the final computation. The layers in the middle are called the hidden layers which are regarded the layers to identify the complex pattern. Each neuron in the layer is connected to every other neuron, and the layer may use different activation functions.

Proposed design of ANN

In this study, an ANN is designed and implemented from scratch using the Pytorch framework in Python. The first layer is the input layer, which accepts a grey scale 2-D image with a shape of (H=22, W=6).

The input layer was reshaped into a 1-D array with a length of 132 as an ANN cannot take a 2-D array as its input. Next, the array was connected to 3 hidden layers with the number of nodes 560, 256, and 128 in sequence. The output layer had 10 nodes and was applied to a SoftMax function to compute the probability distribution of each numerical character [0-9]. The architecture of ANN was shown in Fig 8.

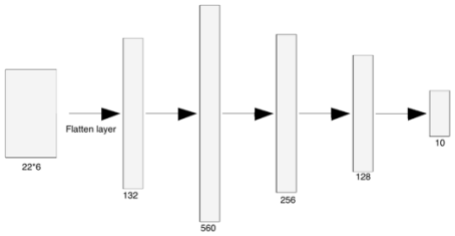

Fig 8: Proposed design for ANN architecture

2.7.2 Convolution Neural Network (CNN)

The second type of neural network is the CNN model. Although CNN is popular in image classification, it is suggested that CNN could also capture the features in time-series data, and thus classification could be accomplished. As the main processes of ANN are introduced in the previous section, it is noticed that CNN adopts a very similar computation process including forward propagation and backward propagation. Therefore, only the components that are different in CNN are introduced.

Proposed design of CNN architecture

In this study, CNN with two convolution layers was designed. The architecture of CNN is shown in Fig 9.

In the first convolution layer, the input data with shape (22,6) was processed with 128 convolution filters of size 3 x 3 and padding of 1 to extract the spatial features. This convolution layer was followed by a max-pooling layer with a size of 2 x 2 to down sampling the image. The second convolution layer was similar, which processed the output from the previous layer by 56 convolution filters of size 3 x 3, followed by a max pooling layer with size 2 x 1 and with stride (2,1). Next, the output from the second layer was reshaped into a 1D array, which had a size of 224.

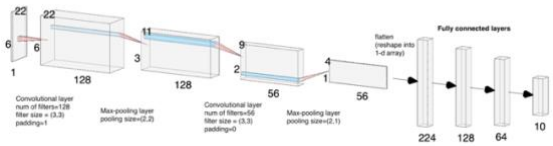

Fig 9 Proposed architecture for CNN

Lastly, the reshaped array was fed by two fully connected layers with nodes numbers of 128 and 64. The output layer had 10 nodes and was applied to a SoftMax function to compute the probability distribution of each numerical character [0-9].

2.8 Performance evaluation

After a neural network is trained, it is necessary to evaluate its performance on both the training dataset and the testing dataset. In this study, the only evaluation metric to be used is the accuracy metric which is defined as the sum of correct predictions divided by the total number of predictions.

2.9 Optimization in deep learning

Optimization techniques aim to improve the performance of the deep-learning model. In this study, two types of optimizations are introduced, including hyperparameter tuning and data augmentation.

2.9.1 Hyperparameters

Hyperparameters are the setting of the model architecture and the configurations of the models training.

For model architecture, they refer to activation functions, the number of layers in the network, and the number of nodes in each layer. For the training configurations, they refer to the number of iterations, learning rate, batch size, optimization algorithm, etc.

Hyperparameter tuning is the process aiming to search for the best set of hyperparameters that yields the best model performance. The process of hyperparameter tuning typically involves training multiple instances of a model with different values of the hyperparameters and measuring the performance of each model using a validation set. The hyperparameters that are experimented on in this study with their theoretical effect on the model performance are explained below.

Number of epochs

The term ‘epoch’ refers to one iteration through the entire training dataset during the model training. This results in the number of

epochs linearly correlated to the time of computation of the training. The model performance is affected by the number of epochs could affect the following aspects. If the number of epochs is too low, the model might not have enough time to adjust the weights to capture the underlying patterns of the training data, which results in underfitting.

Learning rate

The learning rate is a hyperparameter that determines the size of the step at which the model adjusts its parameter, that controls how fast the model learns from the data. The learning rate might affect the model performance in the following ways. When the learning rate is too high, the model may skip the optimal solution and never be able to minimize the loss function. This might result in never finding the optimal set of weights even if the number of epochs is high. On the other hand, if the learning rate is too low, the model may take a larger number of epochs to converge and might get stuck in suboptimal solutions (also known as a local minimum).

2.9.2 Data Augmentation

Data augmentation is a technique used in deep learning to increase the amount of training data by generating new data from existing data, which can reduce overfitting of a model. In this study, centre crop and random crop are investigated.

Centre Crop

Center Crop is a transformation that crops the center of the input image to a specific size. In this study, the center crop is chosen as it is assumed that the user might be performing the writing motion only in the middle of the sampling period.

The images are cropped from (22x6) images to (20x6) images. Only the heights of the images are reduced as it is necessary to keep all 6 degrees of IMU data. The heights are only reduced for 2 pixels as a short amount of writing time is already given to the user, cropping to a larger extent might lead to loss of important data.

Random Crop

The random crop is very similar to the center crop. The crop does not only take place at the very middle of the images but at a random position. In this study, the cropping output is the same as the chosen size as the center crop, which is (20x6) images.

The proposed architecture of ANN.

3. Result and Discussion

In this section, the experiment results for ANN and CNN regarding their performance on training and testing data are introduced. The detailed experiment procedures are explained, and the experiment results are briefly interpreted and discussed.

3.1 CNN experiment

The first type of model to be tested is CNN. For each type of model, the first hyperparameter to be tested is the learning rate. Learning rate(lr) is tested first as it is the most important hyperparameter that controls whether the model converges. Model convergence refers to a decreasing trend of loss value during training. In other words, the

model gives more accurate prediction at least on the training dataset. A grid search algorithm is applied to search for the learning rates that allow model convergence. The grid search is a classical hyperparameter tuning approach that tests the hyperparameters in a brute-force approach. In the grid search algorithm to be applied in this experiment, the search goes through a list of learning rates, which contains lr values 0.001, 0.0001, 0.00001, and 0.000001. As this is just a convergence test, a small value of epoch=10 is chosen which is smaller compared to a full training (e.g. epoch>100) to reduce the computation cost. The other hyperparameters are kept constant during this experiment.

Table 1: Training Configuration

Number of epochs	50
Batch Size	4
Optimizer	torch.optim.Adam
Learning rate	0.001,0.0001,0.00001,0.000001
Loss function	Categorical cross-entropy (Standard loss function for multi-class classification)

Table 2: Convergence result of CNN experiment 01

Learning rate	**Convergence**
0.001	**No**
0.0001	**Yes**
0.00001	**Uncertain**
0.000001	**No**

From the experiment result, a learning rate of 0.001 is too large to converge which might be caused by skipping the optimal solution. The learning rate of 0.0001 shows a clear upward trend for both training accuracy and testing accuracy. The learning rate of 1e-5 produces fluctuating accuracies, which should be able to converge for a larger number of epochs. Finally, the learning rate of 1e-06 does not produce a convergence result as the weight update rate might be too slow compared to the others. The convergence test, it shows that the learning rate between 0.0001 to 1e-5 is the range of convergence given the set of hyperparameters as above.

After the convergence test, full tests are performed in a range of learning rates. The model is trained with lr=1e-04, 1e-05 and 2e-05 until the model converges.

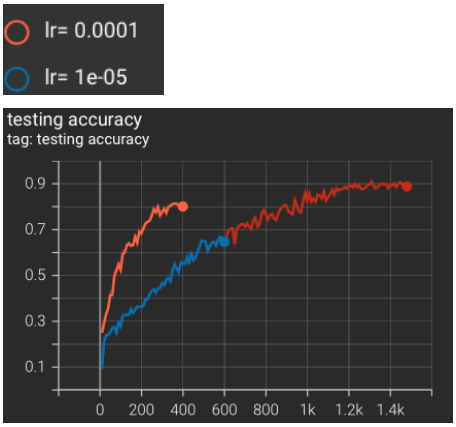

Fig 10 Testing accuracy for CNN experiment

The best training and testing accuracies after the additional training are shown below.

Training accuracy	Testing accuracy
96.6%	92.2%

From the experiment results above, it could be observed that both training and testing

accuracy are higher when the lr is lower despite longer training time.

Experiment with data augmentation

Experiments with data augmentation techniques are carried out. The best training and testing accuracies for the center crop experiment and random crop experiment are shown below.

Table 3: Accuracies for data augmentation

Data augmentation	Training accuracy	Testing accuracy
Centre Crop	90.1%	80.7%
Random Crop	87.5%	75.0%
Without augmentation	96.6%	92.2%

It is observed that the training and testing accuracies reduced after applying center crop and random crop. Two reasons are given to explain this phenomenon. Firstly, the crops eliminate some important parts at the beginning of the data or the end of the data. The eliminated parts could represent a stroke of a character, such that the model misclassifies the character. For example, a part of the horizontal stroke of character "7" could be shortened or eliminated and the model could misclassify it as a "1". Similarly, the last curved stroke of a "3" could be eliminated and the model could misclassify it as a "2".

The second reason is that data augmentation techniques are usually applied in models with high complexity. As our CNN architecture only consists of two convolutional layers and three fully connected layers, the complexity of the model might not be enough to handle the distortion of the data after data augmentation. This might result in a drop in training and testing accuracy.

3.2 Experiments with ANN

In the experiments with ANN, similar testing procedures as the previous one is carried out.

Table 4: Training configuration for ANN

Number of epochs	500
Batch Size	4
Optimizer	torch.optim.Adam
Learning rate	0.00005
Loss function	Categorical cross-entropy (Standard loss function for multi-class classification)

Experiment

The training and testing accuracies after the training are shown below.

Training accuracy	Testing accuracy
81.7%	76.0%

3.3 Comparison of ANN and CNN results

The CNN model shows better performance in both the training dataset and testing dataset compared to the ANN for around 15% higher accuracies as shown in Table 5. This is an expected result as it is discussed that CNN has a better ability to capture patterns of time-series data with higher noise-tolerance ability. [1]

Table 5: Accuracy comparison of ANN and CNN models

Model	Training accuracy	Testing accuracy
CNN	96.6%	92.2%
ANN	81.7%	76.0%

3.4 Limitations of study

In this section the limitation of the study in multiple aspects are discussed.

Limitation of dataset size

In this study, only 300 training data are collected for each number and 3000 training data are collected in total. The size of the dataset is small compared to many other deep learning research, which generally consist of at least thousands of training data for each class. The main drawback of training with a small dataset is the downgrading of deep learning model performance.

Limitation on data diversity

Secondly, all the data were collected from the same user which might reduce the diversity of the patterns in the data. The reduced diversity of data patterns might result in dropping in prediction accuracy when other users are attempting to write text with this device.

4. Conclusion

In this study, a deep-learning-based writing glove is developed in which the inertial sensor serves as the only type of sensor. IMU data of 10 numerical characters are collected with a customized filtering algorithm to handle the fluctuation of IMU data. Two types of deep-learning models are implemented and tested including the Artificial Neural Network (ANN) and Convolution Neural Network (CNN). It has experimented with that CNN has a strong performance to classify time series IMU data with a testing accuracy of 92.2%. While the dataset is relatively small in this study, this study has shown that the deep-learning approach for text input is reliable in the future given the larger scale of the dataset and more advanced deep-learning algorithms. Even though the performance is not reaching a practical performance, this study could be regarded as a basis of inertial-based character recognition by other researchers.

References

[1] B.Z Zhao, H.Z Lu, S.F Chen, J.L Liu, D.Y Wu, 2017. Convolution Neural Network for time series classification. National University of Defense Technology

[2] T. T. Alemayoh, M. Shintani, J. H. Lee, and S. Okamoto, “Deep-Learning-Based Character Recognition from Handwriting Motion Data Captured Using IMU and Force Sensors,” Sensors, vol. 22, no. 20, p. 7840, Oct. 2022, doi: 10.3390/s22207840.