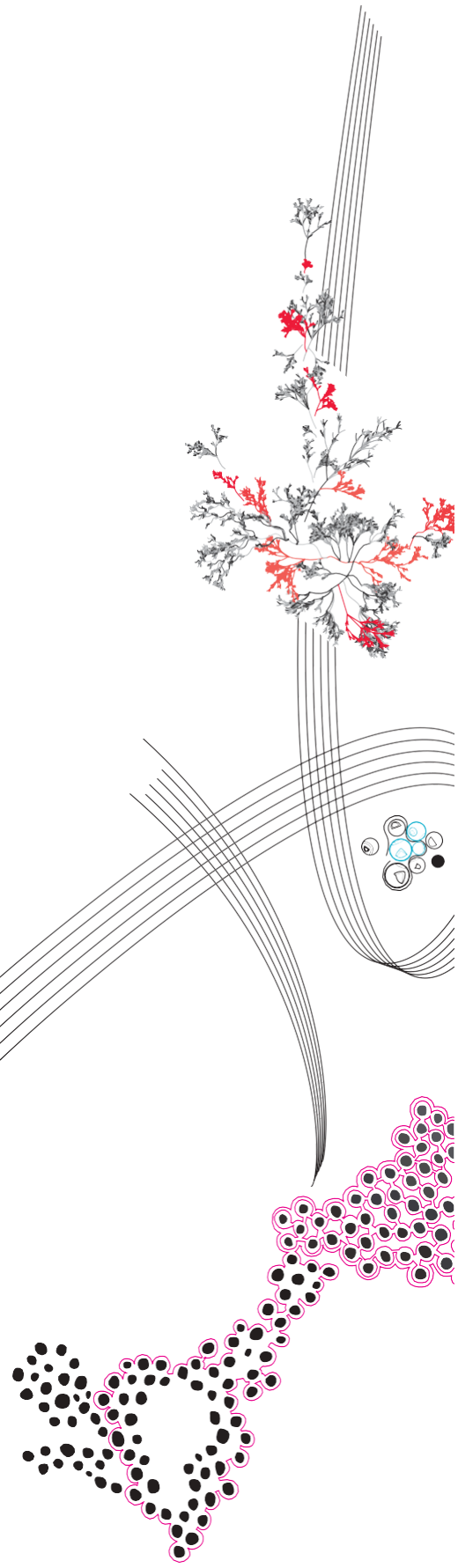

Creative Technology
Module 7 “Hands-on AI”
Project Report

GestureXR

Tran Quy An (s2752379)
Carlos Romero (s2325225)
Yuqing Zhao (s2649780)
Xinran Zhi (s2874695)

Supervisors: Angelika Mader, Luc Schoot Uiterkamp, Jannick Siderius

April 8, 2024

Department of Interaction Technology,
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

1 Introduction

The idea started with making a holographic tap track glove. The design is based on the most advanced Tap Strap 2 product for gesture keyboards (Tap strap 2 2024), the gesture is inspired by Apple Vision Pro Gestures, an advanced gesture recognition product that can recognize different gestures [2]. This interaction was inspired by the holographic table scene in the Iron Man movie[3] where the main character interacts with the holographic table to prototype his super armor by hand. Our goal is to replicate the most accurate interactions, just like in the movies. We designed a three-finger glove to interact with 3D objects in real-time through different IMUs (Inertial Measurement Units, a unit that measures body movement through acceleration and rotation) [1] attached to the fingertips. Users can grab, rotate, zoom in, zoom out, detach, and build objects with their hands or custom gestures to control virtual objects or extend use for any IoT devices. We built a prototype with 3 IMUs units attached to the finger of the user and a smaller version of the ESP32 S3 chip, which collected the data from the IMU and preprocessed it. By using a combination of a Long Short Term Memory deep learning model and a Convolutional Neural Network Model (CNN), we train with the collected data and test it using scikit-learn model evaluation score metric. After training, the model is imported back into the ESP32 S3 chip to act as an edge device, allowing it to recognize gestures and interact with 3D objects in real time. For the visualizing and representing our product, we plan to use Unity to create a 3D space with floating materials and objects represented as a holographic interactive model. The predicted data we got from the ESP32 S3 chip will be sent to Unity using Wi-Fi. The application for this project aims to create the first prototype for fast, visually real time prototyping for mechanical engineers or industrial designers who need to brainstorm, and real time prototyping within a team

2.1 Hardware

Design Requirements are as followed:

- The design should fit with different hand sizes. This is crucial for our data collection part since loosening feeling, could lead to unwanted noise and inaccuracy training process.
- The design should be as light and comfortable as possible for the user. It should be capable of making the user feel more free and more natural when posing different gestures. Moreover, it is aimed to create a more natural variance dataset.
- Placement of the sensor and microcontroller needs to follow the guidelines from the manufacturer to increase accuracy and precision, avoid hardware failure, and increase reliability on the recorded data.
- The design has to be capable of withstanding the gestures while in usage.

With the design requirement we had above, it is possible to come up with some hardware and sketches. A half glove to hold and secure the microcontroller in the back of the hand. In addition, with each finger, we designed a small ring to secure the IMUs to the second joint of the finger. The second joint of the finger is the most defined joint for the movement of the finger, so the rings will be located preferably at the joints or the tips of the fingers. Figure 2.1 depicts the sketch ideas of what our products will look like.

1. MicroController

There are plenty of small but powerful microcontrollers out there for wearable device and can run Machine Learning model inside it. From our research, we come down with 2 ideal candidate, the ESP32 S3 and the M4 chips. Each come with different strength and weakness when coming to our task.We deices to choose the ESP32 S3[4] over M4F[5] Nordic chips because of the wireless Wifi and Bluetooth connection and price. Even though the M4 is equipped with the dedicated TPU to boost the performance in decimal operation. the ESP32 S3 is performance better in calculating whole number. and also better power management compare the the M4 chips. The price is also promising. We will deal with the decimal point number problem in latter chapter where we are cleverly feature engering the data to decrease the computation time for the model. For the ESP32 S3 chip we use XiaoESP32S3 dev board.

Table comparison between two chips

	ESP32 S3 (Xiao ESP32 S3)	ARM M4F Cortex (Xiao nRF52840
Processing Power	dual-core design and clock speeds up to 240 MHz	Single-core Cortex-M4 running up to 64 MHz

Memory and Storage	512 KB SRAM 8MB PSRAM and 8MB Flash on board	256 KB RAM 1MB Flash on-chip and 2MB Flash on Board
Machine Learning Capability	dedicated hardware accelerators for neural network inference support better integer calculation, and floating point calculation is emulated through software.[1]	Have dedicated FPU to processing floating point units, which will accelerate much better in machine learning and neural network
Transmission	Have Wifi and Bluetooth	None

2. IMUs sensor

For the choice of IMUs, we select the BMI270 from Bosch Technology. Since these are the state-of-the-art IMUs used in industrial and wearable devices [6], it has high sensitivity, and low noise and are best for VR, and AR applications, which is suitable for our main project goal.

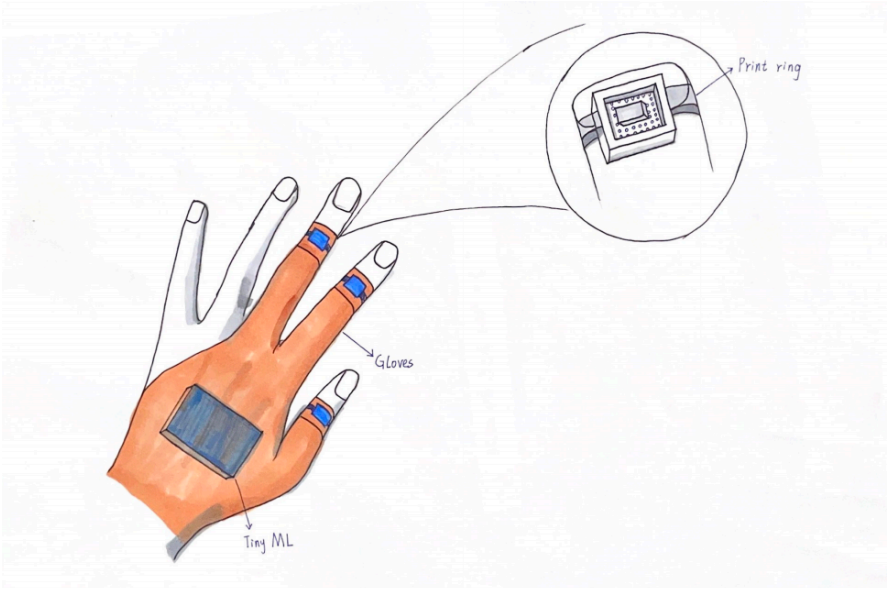

Figure 2.1: Low-fi prototype 3-finger half glove design drawing

3. Glove and ring material.

The material of the gloves is composed of sponge cloth, suede cloth, and elastic cord, which is wear-resistant and can be used with hands of different sizes. The ring holder is 3D printed, with a Self-adhesive Hook and Loop attached to both sides. This design is conducive to adjusting the size of the ring. The esp32 and the glove are also connected through a Self-adhesive Hook, which helps separate the overall electronic components from the glove for easy cleaning and re-soldering. Picture 2.2 shows the final look of the product

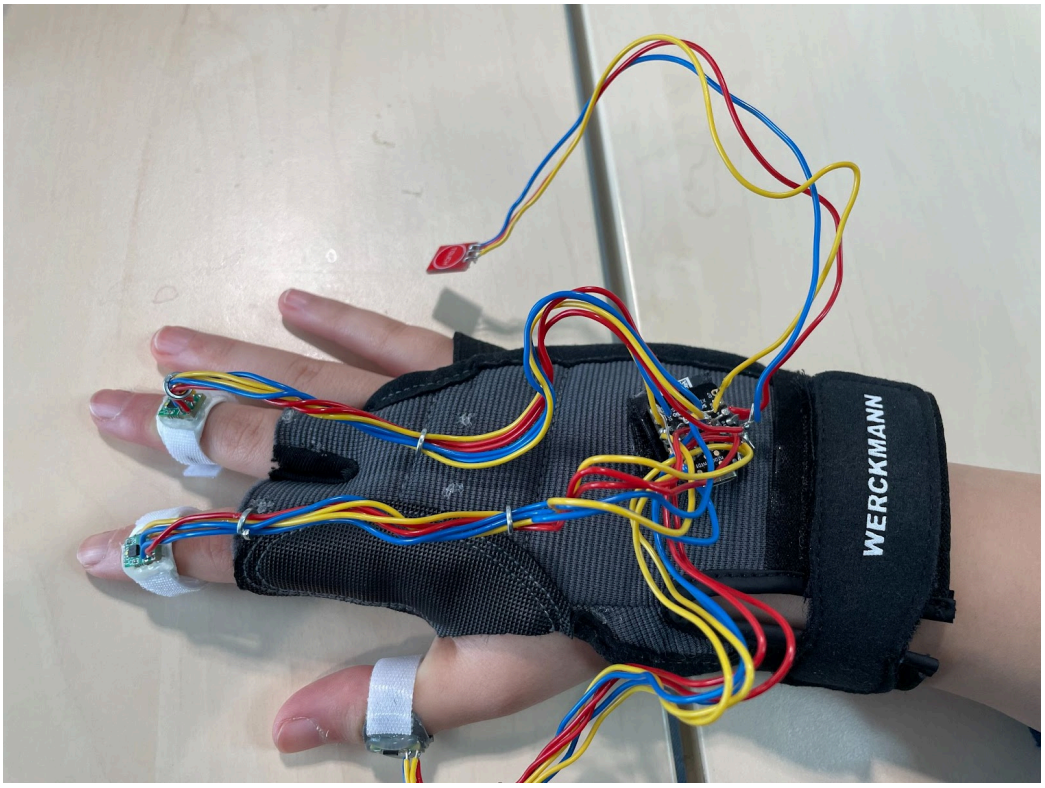

Figure 2.2: Hi-fi prototype 3-finger half glove

3.1 Process of data collection

The front ends of the thumb, index finger, and middle finger on the glove are connected to the IMU respectively. The sensor inputs data into Arduino through esp32. In addition, there is a capacitive sensor. The Arduino will only read the data when the capacitive sensor is triggered. The code in Arduino is used for reading the sensor directly. We put all the sensor data we collected from the IMUs into an array. Then read the value output from Arduino in Python, save it as a new JSON file, and add a label during the collection of data. Before starting data collection. Everyone who is not a creator must sign an informed consent form. A consent form is provided by UT [11].

For collecting data, participants need to wear special gloves and make several gestures, which are pinch (thumb and index finger), relax, rotate 90 degrees, repeat pinch, relax, rotate back, 3-finger pinch (thumb, index finger, and middle finger), rotate 90 degrees and repeat the 3-finger pinch, relax, rotate back, and make a fist. This is a gesture in 1 piece of data. A total of 10 pieces of data are collected for each person. In fact, there are only 5 gestures we need, they are pinch (thumb and index finger), relax, relax, 3-finger pinch (thumb, index finger, and middle finger), relax (3-finger), and fist. The repeated gestures after rotation are for better training and simulate gestures at different angles. Each person collects data repeatedly, not only to collect more data faster but also to reduce individual differences and errors caused by unskilled first-timers. When participants make gestures, two people are needed to coordinate and help. They need to guide the participants to make gestures, input labels before making gestures, and trigger the capacitive sensor in a timely manner when participants make gestures to artificially reduce noise.

3.2 Madgwick Orientation Filter vs Traditional Integral

At the beginning, we retrieved raw data from the IMUs and fed it into machine learning. However, the model could not converge that well so we decided to feed into the model the relative position of each finger. In order to do it, we came up with a way to integrate the value of the acceleration raw measurement to get the velocity and then integrate one more time to get the position we had for each finger. The acceleration value only works well for low frequency value so we also integrate the value from the gyroscope to auto correct the value of acceleration. Combine two values, we extract it as features to feed it into our model. However we found out that there are two problems with it.

- Noise
- Need frequent calibration.

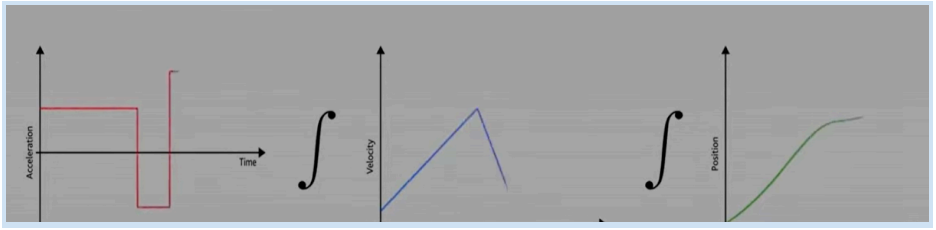

Figure 3.1 Double integrate to compute the position.

For the processing on the IMUs, we found an algorithm to transfer the raw data from the IMUs[13], including the acceleration meter and the gyroscope meter into the Quaternion of that single IMUs. In mathematics, the quaternion number system extends the complex numbers, it provides a definition of the quotient of two vectors in a three-dimensional space. The advantage of a Quaternion value is that each value ranges from 0-1 and represents the rotation relatively to the calibration value. This is very convenient for later in training the data since we do not have to perform Scaling or Normalization toward the data. Additionally, the relative rotation value is best for feature extraction from the data. Unlike the previous traditional integral we have discussed before, the rotation value is easily drifting away from the calibration value, even when we apply Karmal Filter, it is still affected by the noise and in the long term, the value from the IMUs is inaccurate.

$$
\begin{aligned}
f(\mathbf{q}, {}^{E}\mathbf{d}, {}^{S}\mathbf{s}) &= \mathbf{q}^{*}\,{}^{E}\mathbf{d}\,\mathbf{q} - {}^{S}\mathbf{s} \\
&= \begin{bmatrix} 2d_x(\frac{1}{2} - q_y^2 - q_z^2) + 2d_y(q_w q_z + q_x q_y) + 2d_z(q_x q_z - q_w q_y) - s_x \\ 2d_x(q_x q_y - q_w q_z) + 2d_y(\frac{1}{2} - q_x^2 - q_z^2) + 2d_z(q_w q_x + q_y q_z) - s_y \\ 2d_x(q_w q_y + q_x q_z) + 2d_y(q_y q_z - q_w q_x) + 2d_z(\frac{1}{2} - q_x^2 - q_y^2) - s_z \end{bmatrix}
\end{aligned}
$$

Figure 3.2 Madgwick Orientation Filter uses Gradient Descent Algorithm to compute the solution for the Quaternion.

3.3 Preprocessing of data

Before preprocessing the data, we have a look into what the data looks like, what is abnormal, so we could have a glimpse of what we should plan for preprocessing the data as well as feature engineering if necessary. We saw that there are several problems with the data.

- Some values of position, which should contain a 1D list of 12 values, are empty.
- Some values of position, also include string value read from Serial COM.
- The quaternion float value we collected was converted to string when transmitting data through Serial COM.
- Labels need to be converted into One Hot Encoded.

After knowing which problems we have to tackle, we then merge the 220 pieces of JSON data into one large data file. We did that by using the os module from Python to access the directory of all of the data samples we collected. Then we merge them into one large JSON file.

When we preprocess the data, we realize that it is too late for us to redo it again so we have to reverse it into a list containing an integer number. Data Encoding is performed after this. Since the labels in the data are in text form, they need to be converted into numbers through one hot encoding. The last and most important thing is to convert the data type. Since the code saves the collected data as string type when collecting data, but integers and floating point numbers are more suitable for model training, converting string to numeric type is the most important thing for preprocessing data.

3.4 Description of the data set

Originally, The final collected data structure was as follows. There were 5 columns in total, namely timestamp (hour: minute: second), acceleration, gyroscope, label, and separator. The data in the acceleration and gyroscope columns are all lists. The first three values in the acceleration list are the acceleration of the first finger, the values 4 to 6 belong to the acceleration of the second finger, and the values 7 to 9 belong to the acceleration of the third finger. The list of gyroscopes is similar to the list of acceleration. Each three numbers represents the x, y, and z values of the three fingers. The timestamp added before each receipt data is to capture the temporal order and correlation in the data to better handle continuous changes in gestures. Each piece of data has an average of about 3,000 rows, and there are 200 pieces of data in total, with a total of about 600,000 data points. This amount of data is relatively small for training an LSTM model. This may cause overfitting and poor generalization. This is also one of the limitations of this project. After collecting the data, we found that due to the contact of the capacitive sensor, the data of the last gesture was not recorded or was incompletely recorded. The situation of data you can see in Figure 3.3.2. Of course, some other gestures are not recorded. To deal with this situation, we recorded another 20 points of fully recorded data. However, incomplete data will have a certain impact on the accuracy of the final prediction. The link of data can be seen in the appendix (collected data - Google, z.d.).

Timestamp	Accel	Gyr	Label	Sep
14:36:00	['-952', ' -775', ' 1811', ' 413', ' -49', ' -136', ' -164', ' -2135', ' -833']	[' -5087', ' -779', ' -1432', ' 2115', ' 339', ' -173', ' 232', ' -78', ' -1345']	pinch2finger	FALSE
14:36:01	['-1064', ' -752', ' 1827', ' 443', ' -129', ' -71', ' -98', ' -1915', ' -1020']	[' -5088', ' -847', ' -1546', ' 2111', ' 305', ' -167', ' 260', ' -50', ' -1355']	pinch2finger	FALSE

table 3.3.1: Slice of data

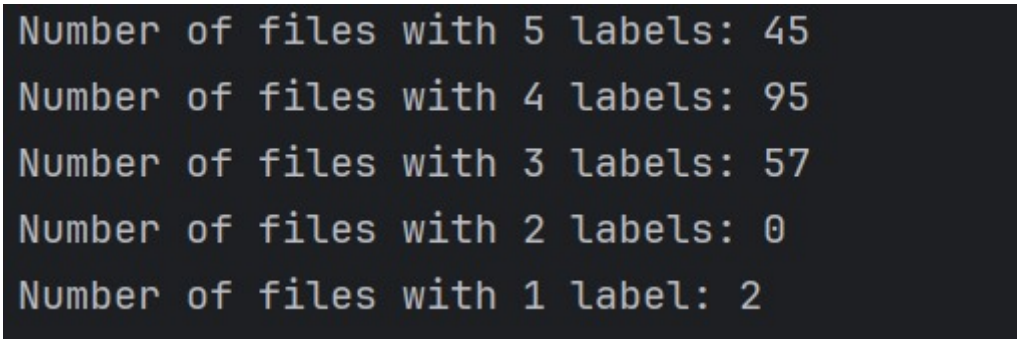

Figure 3.3.2: A graph that label being record

4.1 AI algorithms applied

originally, it was planned to use a combination of CNN (Convoluted Neural Network) and LSTM (long short-term memory) for this AI algorithm. The reason for choosing these models would be the excellent performance on long-term memory while being able to make motion estimation over a sequential series of data. This exactly fits the needs of our data training ([7], see figure 4.1) because LSTM is a special kind of RNN (a kind of neural network that deals with sequential data), it can memorize very long sequential data and forget about the data that we don't need, making the data framework more suitable for our needs [8]. Simultaneously, since the input to the model is continuous and represents the continuous movement or change of gestures, our data has a sequence structure, so it is very appropriate for this project to in addition use a CNN model to represent the data as a frame that demonstrates a change in position per each input value that is introduced into the model. In addition, the STML model has high model complexity, especially when dealing with long sequence data, so the STML model can improve accuracy, but it should also be noted that a problem of this model is being prone to overfitting and requiring a large amount of data.[9] It is important to notice that the reason this AI model was developed is because the way the IMU's record data. The data is sequentially saved and it can be treated like a frame from a video. The difference in between inputs will represent the desired gesture that is being recreated. Thus the data first goes to a CNN model, and later the output of this model is given as input to the LSTM model.

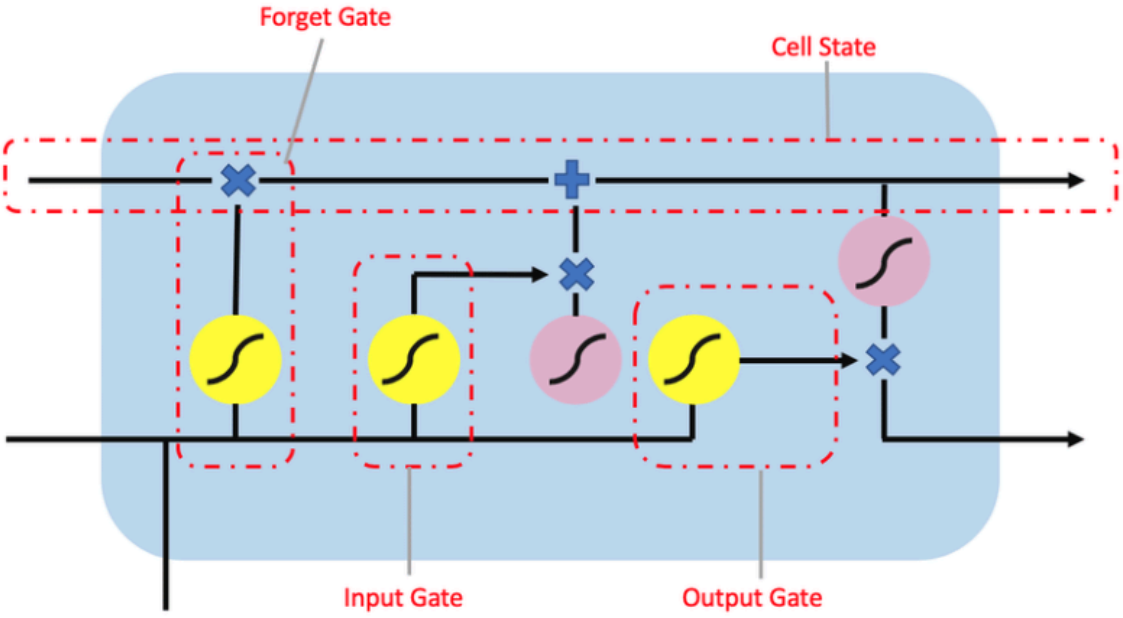

figure 4.1 Implementation of a LSTM unit [10]

A common LSTM unit comprises several components:
Cell: The core of an LSTM unit that remembers values over arbitrary time intervals.
Input Gate: Determines which new information to store in the current state.
Output Gate: Controls which information from the current state to output.
Forget Gate: Decides what information to discard from the previous state.

By selectively outputting relevant information from the current state, LSTMs maintain useful, long-term dependencies for making predictions across multiple time steps. In theory, classic RNNs can track arbitrary long-term dependencies in input sequences, but their computational limitations lead to vanishing or exploding gradients during training. LSTMs address this by providing a robust short-term memory that can contain extensive time intervals. This model's versatility aligns perfectly with your data training requirements.

The model is fed the values read from each IMU sensor while a specific gesture was done using one-hot encoding. This means the data is composed by the sequential read of acceleration measurements in the XYZ axis, the gyroscope measurements in the XYZ axis, and a one-hot encoded list that represents the gesture that is being recorded with a one and the other four gestures are represented by zeros. Consequently, for the LSTM, the model uses a categorical cross entropy loss method because there is more than one label being predicted. Moreover, a stochastic gradient descent (SGD) optimizer with a learning rate of 0.001 is used. This means the gradient is calculated using a randomly selected instance of the training data instead of all the training data.

However, in the end, we only used the CNN model for training and classification data. This CNN model was trained using the data that was generated using the Madgwick algorithm. This algorithm helped develop constant data that could be used for representation of the selected gestures. The data consisted of a list with the values from the Madwick algorithm. Inconsistencies with the LSTM model made it impossible to work while trying to develop a reliable source of predictions for the labels or gestures.

4.2 Quality of the AI model

While writing the model code, we created a method called "evaluate" to evaluate the

performance of the model on the test set "X_test" and output the relevant metrics: loss rate, accuracy, recall, and precision with some graphics to show the results of our tests. We will explain the functionality and implementation of the method step by step:

Firstly, the model evaluation is performed to calculate the loss value and accuracy on the label “y_test” corresponding to “X_test”, and “batch_size” formulates the batch size during the evaluation, and 'verbose' controls the level of detail of the output during the evaluation. After that, prediction and processing of predicted results. The results of the test set “X_test” are predicted using “predict” and the results are stored in “y_pred”. Then, the “np.argmax” function is utilized to find the index along the second dimension (i.e., the category dimension of the prediction) where the maximum value of the prediction for each sample is located to get the final category of the prediction. Finally, the prediction results are output, here output here outputs the overall prediction result “y_pred” and the category of the prediction for the first sample in the test set (X_test[0]).

In addition to this, we use two aids for visualizing the model training and evaluation process. “plot_history”, which is used to plot the accuracy curve during the model training process, including the trend of the accuracy of the training and validation sets over the epoch, and to set the labels, titles, and legends of the graphs, as well as to limit the y-axis range to [0, 1], which represents the percentage of the accuracy. This step allows you to visualize the progress of the model during training, determine whether the model is overfitting or underfitting, and whether the results on the validation set are improving.

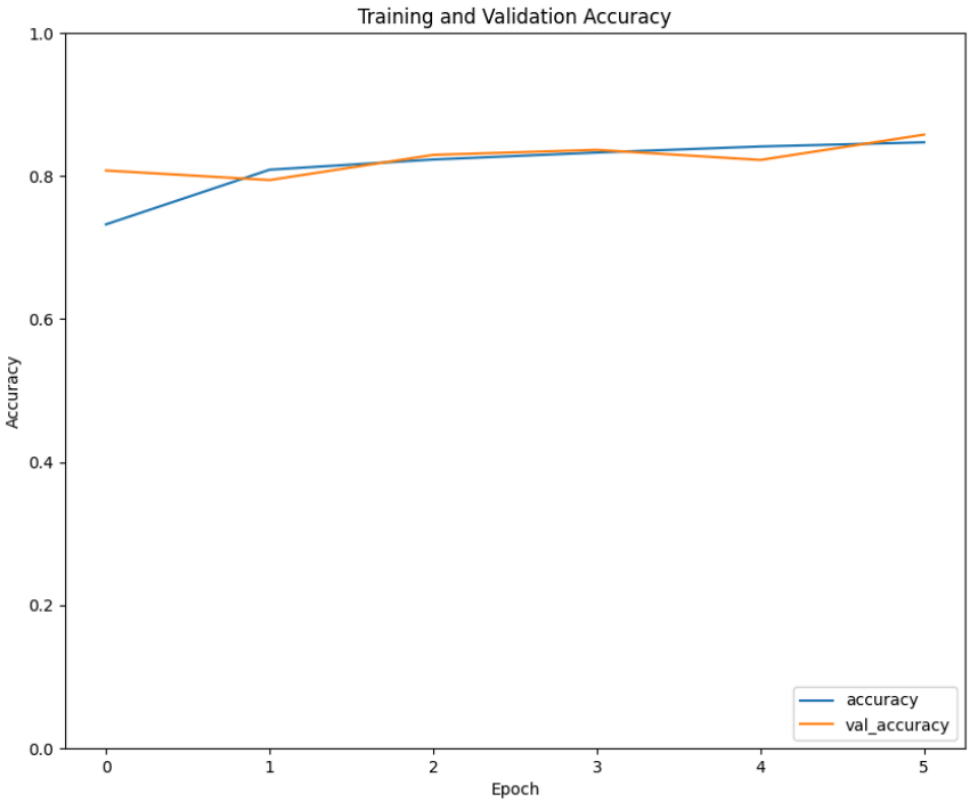

Figure 4.1 History training of accuracy and validation accuracy of CNN model

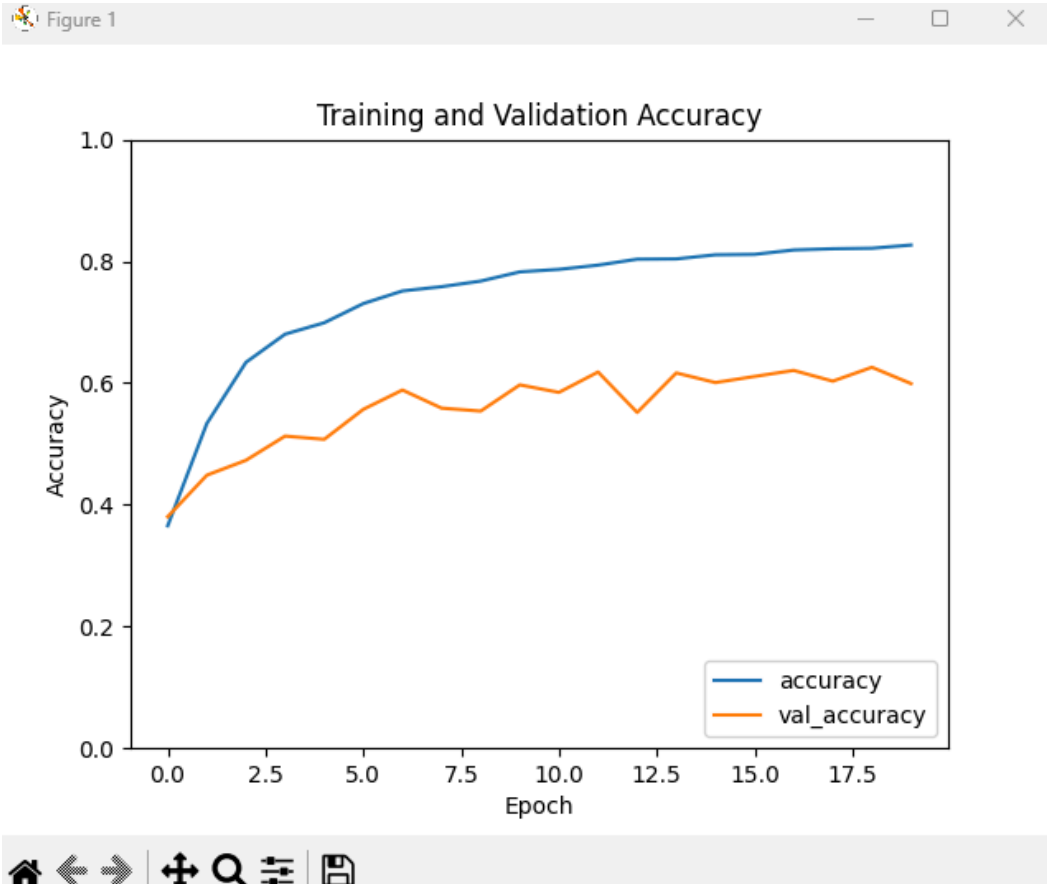

Figure 4.2 History training of accuracy and validation accuracy of LSTM model only

“plot_confusion_matrix”, this method is used to plot the confusion matrix, showing the correspondence between the model's predictions on the test set and the actual labels. This step provides a visual understanding of the model's prediction performance on non-features, determining whether the model has a specific category of prediction bias, as well as further analysis of the model's performance and optimization direction.

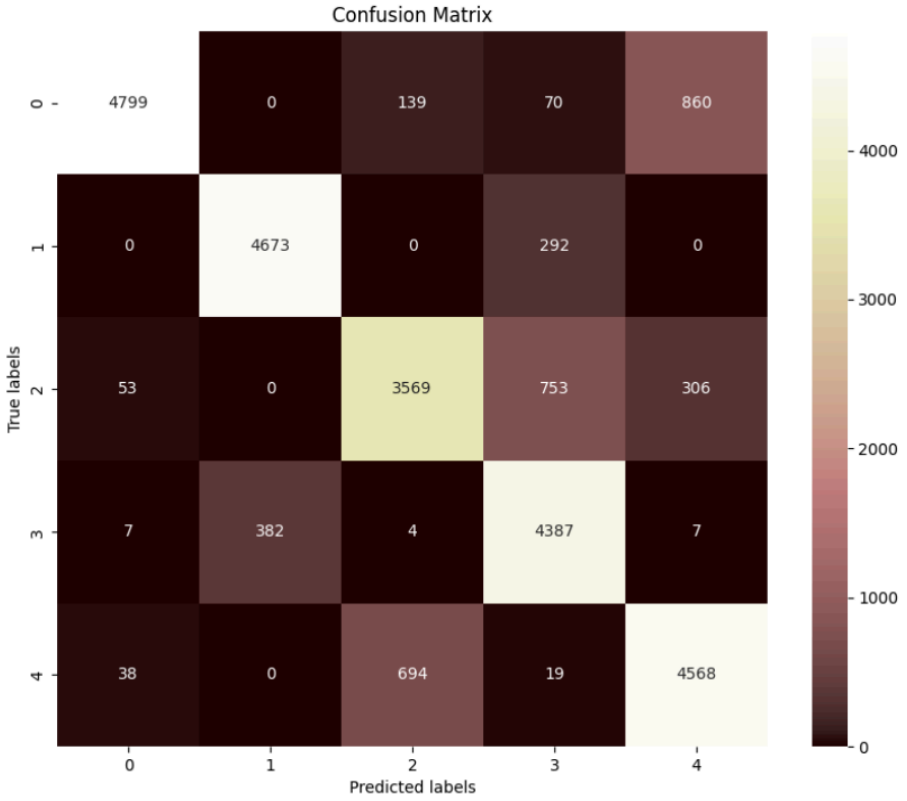

Figure 4.2 Confusion matrix of CNN model with label respectively as "fist, pinch2finger, release2finger, pinch3finger, and release3finger"

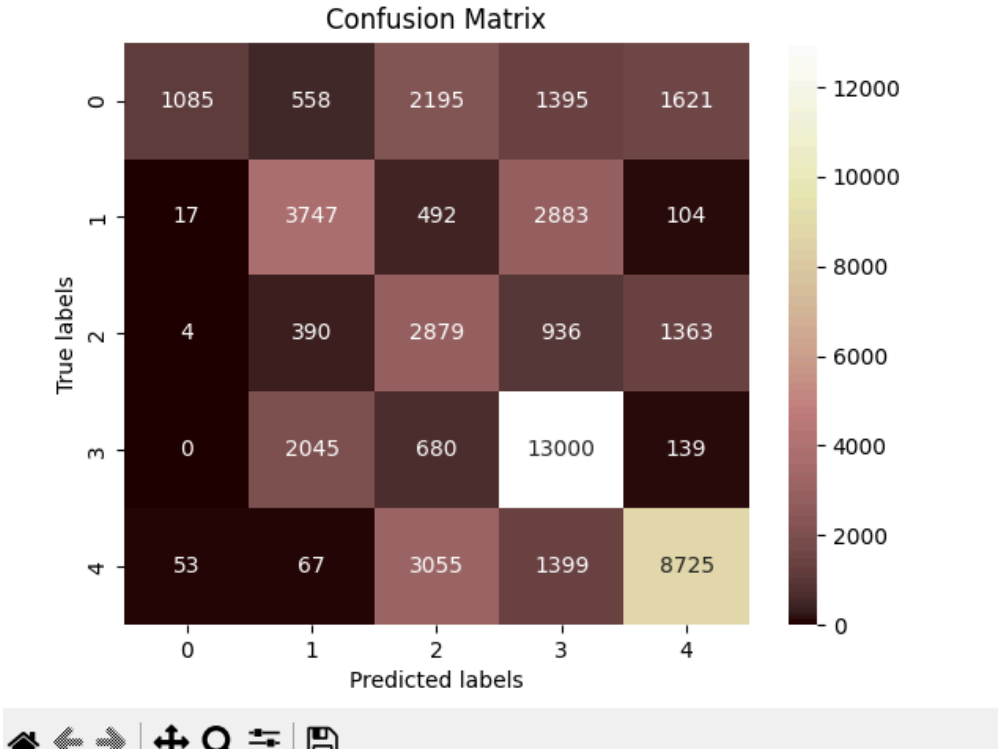

Figure 4.3 Confusion matrix of LSTM model with label respectively as "fist, pinch2finger, release2finger, pinch3finger, and release3finger"

Both a CNN model and LSTM model were tested. As it can be seen in the confusion matrices of both models, it is possible to simply notice that the LSTM model suffered a lot of wrong predictions, both false positives and negatives. The LSTM model tested with an accuracy of 75% compared to the CNN model's 85% accuracy. The only uncertainty that the CNN model had was that it was trained using data that was designed for the LSTM model. So, instead of showing the CNN model many frames that represented one single static gesture, we showed the model the movement of the fingers when making a specific gesture. This type of data worked but created some inconsistencies. For example, when the model tries to predict at the beginning of a gesture, it might confuse it with another gesture which ended in the same position as the new beginning gesture

5.1 Description of the Installation

As described in section 2. In hardware, our device has the appearance of a soft and easy to wear glove (figure 2.2), and on top of the glove is mounted a microcontroller (ESP32S3) containing our training model and various training data, which acts as the 'brain' of the whole operation. It directs all commands and fulfills our requirements. This microcontroller is connected to 3 IMUs (inertial measurement units), which are a crucial part of our setup. Its role is to analyze the changes in the accelerometer and gyroscope data built into the IMUs according to the movement of the user's hand, package the data to the microcontroller, predict it with the model in the microcontroller, and then analyze it to understand the meaning of the user's actions and release the corresponding information to match the visualization effect of our target. (pygame) to show a complete interactive device. In terms of Wearable, the user can observe that each IMU is supported by a relatively soft, deformable TPU finger ring to maximize the data detection function of the device.

5.2 Evaluation

Our criteria for evaluating the soundness of the device are divided into two parts, the first of which is its use in collecting data. Our goal in this section was to get a device that is easy to use, easy to put on, easy to take off, supports the hardware perfectly, and fits the user's hand size and finger size to the greatest extent possible. Therefore, our evaluation will be based on absolutely random people. This has already been realized in our data collection process. Evaluation steps: We randomly select a student (regardless of the student's gender, height, weight) for data collection, the first step is to let the student wear the gloves we made, adjust the width of the wrist through the Velcro of the gloves, and match the thickness of the student's fingers through the other softer Velcro of the finger area, and with the assistance of an assistant, the student will be able to use the gloves to adjust the width of the wrist. A total of 50 students participated in our data collection process, and each student found the most suitable size after certain adjustments to the glove, and felt comfortable wearing it throughout the entire operation.

Another direction of Evaluation is whether our trained model can accurately recognize the user's gestures and thus get satisfactory results. We were able to make the model consistently predict the gestures that were being imputed. With an accuracy value of 85%, it is safe to say that the model in combination with the hardware has proven to be successful

6 Conclusion

During the production process, we faced two major challenges. Our team is dedicated to handling complex welding tasks. Every connection on an integrated circuit board (PCB) requires precision. The delicate dance of soldering components (resistors, capacitors, microcontrollers) requires expertise and patience. Ensuring a reliable connection while avoiding short circuits or cold joints is a meticulous process. Welded joints are often damaged and disconnected due to gesture movements, requiring repeated welding. Also, we encountered some problems when trying to collect data. Accurate data helps in training and fine-tuning our system. However, collecting this data is very time-consuming. Volunteers wear the devices and instructors guide the data collection process. Balancing efficiency (collecting sufficient data) and quality (avoiding noise) is a major challenge.

Our LSTM model has not been successful so far. In fact, we try model building and training using small amounts of data before collecting large amounts of data. However, because the code for collecting data has changed when collecting large amounts of data before, the collected data contains strings in the list instead of direct ints. The data type did not match the data required by the model, and after a significant amount of time changing the data type still succeeded, so we decided the best approach was to re-gather the data. It is also a warning and learning for us. Before we start collecting large amounts of data, we must double-check that the correct type of data is being collected and that all data has been collected. Although the results are not ideal at present, we are full of enthusiasm in this process, from product design, data collection to model construction and training. Moreover the usage of raw data from the IMUs was not convenient but it was later improved thanks to the Madgwick algorithm. Consequently, since the previous LSTM model was not working as expected, the implementation of the new CNN model did greatly increase the reliability of the product. A satisfactory outcome has been reached together with the hardware and selected models that have been trained.

References

[1] ScienceDirect(2022). *Inertial measurement.* Bistatic Synthetic Aperture Radar. URL: https://www.sciencedirect.com/topics/engineering/inertial-measurement

[2] Juli Clover(2023, June, 8). *Apple vision pro gestures, These Gestures Are How You Control Apple Vision Pro.* MacRumors. URL: https://www.macrumors.com/2023/06/08/apple-vision-pro-gestures/

[3] Rafael A. F. Silva(2011). Iron Man. Youtube. URL: https://www.youtube.com/watch?v=D156TfHpE1Q&t=51s

[4] Espressif Systems. (2021, August). ESP32-S3 Datasheet (Rev. 1.0) [Brochure]. Retrieved from: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf

[5] Nordic Semiconductor. (2019, December). nRF52832 Product Specification (Rev. 1.4) [Brochure]. Retrieved from： https://infocenter.nordicsemi.com/pdf/nRF52832_PB_v1.4.pdf

[6] Geek & Sundry. (2015, September 15). Critical Role: Episode 26 - Consequences and Cows [Video]. YouTube. https://www.youtube.com/watch?v=Cr2grRn3ru4

[7] Ashish Thakur(2024, April, 18). *What is LSTM? Introduction to Long Short Term Memory.* intellipaat. URL: https://intellipaat.com/blog/what-is-lstm/

[8] ResearchGate. (2020. January). Deep Learning With Dynamically Weighted Loss Function for Sensor-based Prognostics and Health Management - Scientific Figure on ResearchGate. Retrieved from https://www.researchgate.net/figure/The-LSTM-unit-contain-a-forget-gate-output-gate-and-input-gate-The-yellow-circle_fig2_338717757

[9] (2020, January 7). *How to diagnose overfitting and underfitting of LSTM models.* MachineLearningMastery.com. URL: https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/

[10] Aishwary(2023, December, 4). *Introduction to Recurrent Neural Network.* geeksforgeeks. URL: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[11] *Hands-on AI Project. (n.d.). Module M7 of Creative Technology. University of Twente*

[12] *Tap strap 2* (2024) *Tap*. Available at: https://www.tapwithus.com/product/tap-strap-2/ (Accessed: 18 April 2024).

[13] xioTechnologies(2013). Open-Source-AHRS-With-x-IMU. URL: https://github.com/xioTechnologies/Open-Source-AHRS-With-x-IMU

[14] *GitHub official page of the project*: Phickies/GestureXR (github.com)

Appendix

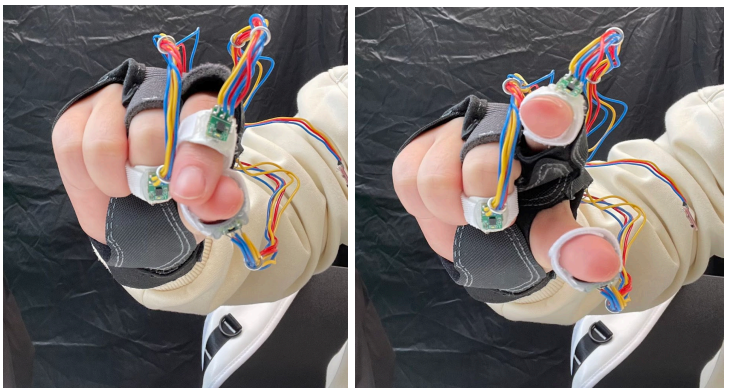

Figure 6.11: pinch (thumb and index finger)

Figure 6.12: Release 2 pinch

Figure 6.13: rotate 90 degrees, repeat pinch,

Figure 6.14: Release 2 pinch

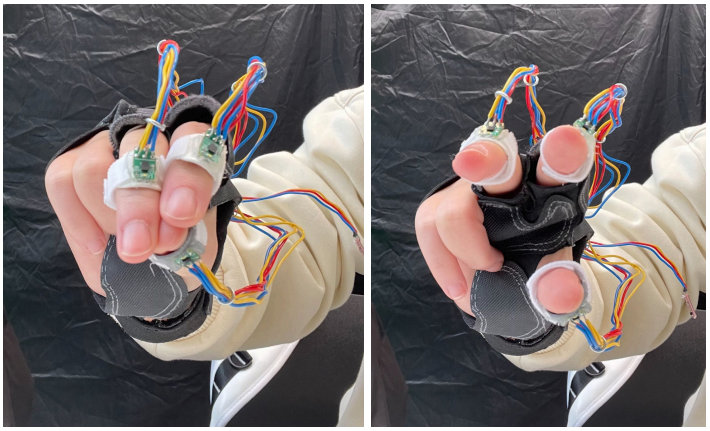

Figure 6.15: rotate back, 3-finger pinch (thumb, index finger, and middle finger)

Figure 6.16:Release 3 pinch

Figure 6.17: rotate 90 degrees and repeat the 3-finger pinch

Figure 6.18: Release 3 pinch

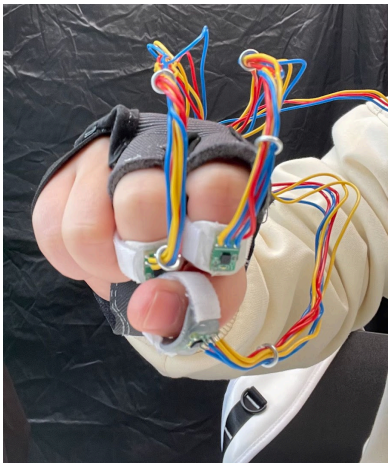

Figure 6.19: fist

Access the collected data:
https://drive.google.com/drive/folders/1qtBJ0BXPNz47YgEMSerCMwPltLadaVDN?usp=drive_link