# Low-Power Gesture Recognition Using Convolutional Neural Networks and Ambient Lighting

**Arne de Beer**[1]

**Supervisor(s): Qing Wang**[1]**, Ran Zhu**[1]**, Mingkun Yang**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

## Abstract

This paper presents a study focused on developing an efficient signal processing pipeline and identifying suitable machine learning models for real-time gesture recognition using a testbed consisting of an Arduino Nano 33 BLE and three OPT101 photodiodes. Our research aims to address the challenges of limited computational power whilst maintaining a high inference accuracy.

Experiments were conducted to optimise the signal processing and explore various machine learning model architectures, specifically revolving around convolutional neural networks. The data used for these experiments was gathered by creating a dataset of gestures from left- and right-handed participants. We took ethical considerations regarding participant recruitment and data security into account and we made sure to balance the dataset with both left- and right-handed participants as much as possible.

We obtained accurate gesture recognition results, surpassing the goal of a 75% success rate. Our machine learning models, trained on pre-processed 2D data, achieved near real-time inference times while running on the resource-constrained Arduino Nano 33 BLE.

The findings of this study contribute to the field of gesture recognition by providing insights into efficient signal processing techniques and identifying suitable machine learning models for resource-constrained devices. The developed system can be applied in various applications, ranging from games to healthcare. Furthermore, a dataset is contributed which can be used for further research.

## 1 Introduction

Recognising gestures has been of interest for over thirty years [1] [2]. The main limiting factors of gesture recognition systems were related to computing power and machine learning (ML) knowledge, specifically deep learning. With today's advances, however, we can detect and recognise hand gestures more efficiently, faster and in more compact form factors [3].

Meanwhile, humans have been putting artificial lighting infrastructures in as many places as we can, and these days artificial lighting can be found in almost all buildings. The abundance of this technology calls for other ways of using it and only recently have we started to transform these infrastructures to support other services besides lighting. For example, high-speed internet, or accurate indoor localisation, which is difficult to achieve with today's global navigation satellite systems like GPS, may finally be realised by this transformation of our lighting infrastructure [4].

Gesture recognition can be a powerful way to include human-computer interaction in a system. Including such a system allows users to control their computers, smartphones or other devices by performing specific gestures [5]. It can also be used for gaming, sign language recognition and even in healthcare for patients with physical disabilities [2] [6] [7].

Unlike other projects which made use of cameras or hardware that needs to be worn by the user [5] [1], this project aims at using the existing lighting infrastructure to implement a low-power system to recognise several gestures using the reflection of ambient light and simple photodiodes such as the OPT101 photodiodes [8]. This works because by performing specific gestures, the movement of our hands and fingers causes the surrounding ambient light to be reflected or blocked and then changes the amount of light which gets captured by the photodiodes. Using a machine learning model running on a microcontroller, the patterns of the changing captured light can be recognised and decoded into the performed gesture.

This study uses Tiny Machine Learning (TinyML) to run ML models on smaller, less powerful devices, such as microcontrollers [9]. We specifically make use of the TensorFlow Lite (TF Lite) library to build the models used in this study.[1]

In 2022, a similar project for the CSE3000 course at the Technical University of Delft (TUD) was started from the ground up. Since there were no real expectations and preparations a large amount of time was spent researching ways to approach the problem and designing hardware [10] [11] [12] [13] [14]. This year, however, this study was able to build off of what they created by reusing the Printed Circuit Board (PCB) they designed and learn from their mistakes by thoroughly going through their papers. Whilst the focus of last year was mostly on the optimal placement of photodiodes, this year, the focus was on experimenting with various ways to increase the overall performance of the gesture recognition system in terms of inference accuracy, inference latency and model space and also improve the overall pipeline.

The main question answered in this research is: *"Given a testbed with one Arduino Nano 33 BLE and three OPT101 photodiodes, how to detect and process the signals efficiently, which model could be used for gesture recognition based on 2D pre-processed data and how to compress the used deep learning model to make it run in real-time and be small enough in size to be able to be deployed on the microcontroller?"*

This main question is divided into three subquestions:

- How to efficiently read the signals from three OPT101 photodiodes and process them into 2D data?
- What kind of machine learning model is best to use for gesture recognition, such as swipe left or right, based on 2D pre-processed data, when the inference time must be near real-time and computational processing power is limited, whilst trying to keep the success rate above 75%?
- How to compress a machine learning model to make it real-time on an Arduino Nano 33 BLE? Real-time is considered to be below 200 milliseconds.

In this paper, various machine learning model architectures focused on convolutional neural networks (CNNs) are evalu-

[0]Parts of this paper were corrected by AI tools, such as ChatGPT.

[1]https://www.tensorflow.org/lite

ated and compared and the effects of various pre-processing and data augmentation techniques are analysed. Furthermore, a dataset with over 2300 performed gestures to train the models is presented. Finally, a microcontroller program as a proof of concept is contributed.

To achieve these goals, this paper is organised as follows. The next section will give a brief background on some of the lesser-known terms mentioned in this paper. After this, section 3 outlines the general methodology and steps taken during this research. Section 4 presents the contributions of our work, where the key findings and insights that can be valuable to the field of gesture recognition are discussed. Next, section 5 delves into the experimental setup and results, where the hardware and software setup is described in detail and the results of the analyses are presented. After this, in the responsible research section, the ethical considerations involved in this study are addressed, including participant recruitment and data security and anonymity.
In the discussion section, the strengths and limitations of our approach are examined, the findings with related studies are compared, and the implications of this research are highlighted.
Finally, the paper is concluded with a summary of the main findings and an outlining of possible future research in the field of gesture recognition.

## 2 Background

This paper assumes basic knowledge of machine learning, deep learning, and more specifically, convolutional neural networks. For a good explanation of how convolutional neural networks work please see [15]. Besides common machine learning terminology, the paper also focuses on 2D collected data, which may not be a familiar term. The data collected from the three photodiodes, which can be seen as a multivariate time series, can be interpreted as an image with the resolution being the number of samples by the number of photodiodes and the brightness of each pixel representing the value at that particular part of the sample. An example is illustrated in figure 1.

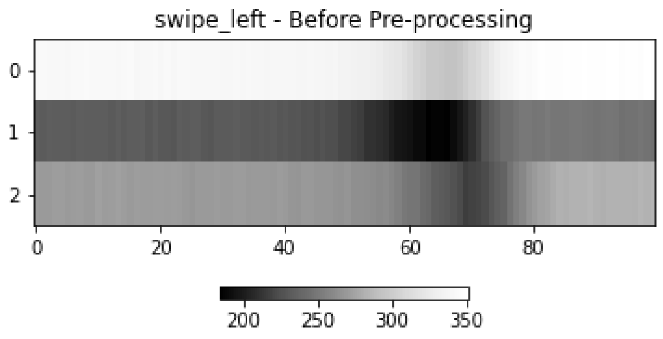


Figure 1: An example of a multivariate time series formatted as an image with the 3 photodiodes as its rows and the 100 samples as columns. A higher value, represented as a brighter pixel, means more light reached the photodiode. Before pre-processing, these values are the raw values coming from the sensors.

In this study, quantization was used to compress machine learning models. Quantization is a compression procedure where all weights and activation functions are converted from 32-bit floats to 8-bit integers. Although this sacrifices a bit on the inference accuracy, it greatly decreases inference time and model size, making it suitable for microcontrollers.

## 3 Methodology

This section goes over the methodology used for this research. Specifically, the steps followed to achieve the research objectives.

- Data collection: Participants were recruited to perform a range of hand gestures, such as swiping left or right. The OPT101 photodiodes on the PCB captured the changes in light intensity, generating raw data for further analysis. Participants were asked to perform the gestures with their dominant hand and were asked to perform them with their other hand if they were still willing. This way we were able to still get left-handed data, without solely relying on finding left-handed participants. More specific details on data collection can be found in subsection 5.3.
- Data pre-processing and augmentation: Data pre-processing and augmentation techniques play a crucial role in improving the performance of any ML model. Exploring various techniques and combinations, and comparing the performance of models allows one to find the most suitable combination of them and to get the most out of the collected data. Pre-processing steps can for example remove noise, rescale and normalise the signals. This makes it easier for the model to extract usable features and increase its accuracy. Data augmentation is used to modify existing data and generate new data from it by randomly adjusting the contrast and performing small translations. This allows machine learning models to improve performance even further [16].
- Constructing ML models: Several machine learning model architectures focused on CNNs were adapted to recognise multivariate time-series-based data. The models employed a deep learning architecture, consisting of several convolutional layers. This step is closely related to the next step, as an iterative approach was used where architectures underwent small adjustments depending on the results in the next step. Besides, new architectures were constructed using the knowledge gained from previously analysed architectures and papers such as [15].
- Training and evaluating: The constructed ML models were trained, evaluated and tuned. In CNNs, multiple hyper-parameters can be tuned in order to achieve the best inference accuracy. These hyper-parameters include the loss function, learning rate, number of epochs and batch size. Finally, accuracy, inference time and total model size were compared and analysed. Inference accuracy was measured by performing stratified K-fold cross-validation and data was split on a per-hand basis. Inference time was measured by getting the difference

between the current time before and after running inference on the microcontroller. The total model size was inspected both before and after quantizing. To have a fair comparison between the non-quantized and quantized models both are saved as TFLite files and the file sizes are then compared.

By following these steps we aimed to develop an accurate and reliable gesture recognition system using OPT101 photodiodes and the Arduino Nano 33 BLE microcontroller. Subsequent chapters will present the results and analysis based on this methodology.

## 4 Contributions

This study makes several contributions to the field of gesture recognition. The key contributions of our research are as follows:

1. **Dataset**
   An anonymised dataset with over 2300 performed gestures is contributed.
2. **Pre-processing and data augmentation techniques**
   This research proposes and explores various combinations of pre-processing techniques, like rescaling, normalisation and a low pass filter, to enhance the quality of the collected data and increase the ML model's accuracy. Additionally, the research explores some data augmentation techniques, which should improve the robustness and generalisation capabilities of the machine learning models.
3. **Analysis of various CNN ML models**
   In this paper an evaluation and comparison of different CNN ML models for gesture recognition is made. Through extensive experimentation the inference accuracy, inference time, and model size of each architecture is assessed, taking into account the effects of the pre-processing pipeline and data augmentation.
4. **Microcontroller program**
   A program that can be deployed to a microcontroller is contributed, specifically tailored for the Arduino Nano 33 BLE. The program incorporates a gesture start detector, the pre-processing pipeline, and the ability to incorporate and invoke inference on a compressed ML model. This program enables real-time gesture recognition directly on the microcontroller.

## 5 System Design

This section provides a comprehensive overview of the hardware setup and software tools that were used and the machine learning model architectures constructed during this study. The primary goal of this section is to present the experimental procedures and outcomes, showing the effectiveness and performance of various pre-processing pipelines and ML model architectures.

### 5.1 Hardware Setup

The microcontroller on which the final model runs is the Arduino Nano 33 BLE. It uses three OPT101 photodiodes placed on the custom-made PCB designed last year as its input. The PCB includes several digital switches that are able to change the sensitivity of the photodiodes using different combinations of resistors. The mentioned PCB can be found in figure 2.

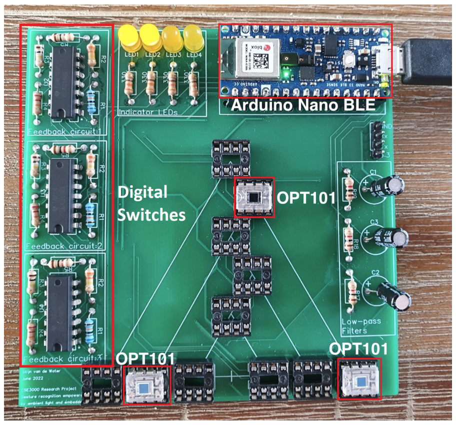


Figure 2: The PCB that was designed last year. It consists of a microcontroller on the top right, three OPT101 photodiodes in the middle, and some digital switches for adjusting the sensitivity of the photodiodes on the left side.

### 5.2 Software Setup

The microcontroller's code is compiled using the PlatformIO[2] framework and uses the TF Lite[3] library to load the model and run inference on collected data.

All ML models were created in Python using the TensorFlow library and then converted to a quantized version. How quantization works is explained in section 2. To deploy the quantized model on the microcontroller, the quantized model is exported to a C array using the Linux ”xxd” command.[4]

### 5.3 Data Collection

The dataset that was built for this study was created by using a bare-bone Arduino program which communicates with a simple interface over a serial connection. On command, the Arduino reads the output of the three photodiodes at a frequency of 100 Hz and sends the data over the serial connection, where the data is labelled and stored. 100 Hz was chosen as it was found to give a good balance between precision and redundancy [11]. After finding that all gestures can be properly and naturally performed within 1 second the time frame for data collection was set to 1 second. The recording time of 1 second and the sample rate of 100 Hz gives us 100 samples per recorded gesture. The participants were shown an example of how the gestures should be performed and were told to perform them as naturally as possible. Participants were initially asked to perform the gestures with their dominant hand. After collecting the data with one hand, they were asked whether they were also willing to perform all gestures

[2]https://platformio.org/
[3]https://www.tensorflow.org/lite
[4]https://linux.die.net/man/1/xxd

with their other hand. To perform the gestures, each participant was requested to repeat each gesture 5 times. In the end, the dataset consists of over 2300 performed gestures by 17 left and 26 right hands. The dataset consists of both male and female participants of varying ages, this metadata is not analysed or stored anywhere due to restrictions set by the ethics committee. For any further ethics-related concerns regarding participant recruitment and data collection, see section 7.

### 5.4 Model Construction and Configuration

For this study, multiple ML model architectures were constructed. The final model from last year [13] was used as a starting point and small modifications were made to explore and find out what changes increased the inference accuracy, whilst maintaining a reasonable model size. These small changes included trying out different amounts of filters per convolutional layer, trying out different kernel sizes, and adding or removing some convolutional or other types of layers. After this, architectures were constructed from scratch with inspiration from well-known image classification models such as LeNet [17] and AlexNet [18].

### 5.5 Model Analysing

All results were obtained through stratified K-fold cross-validation. Unless stated otherwise, the data was split on a per-hand basis, ensuring that gestures performed with each hand were treated separately during training and evaluation. This means that if a candidate performed all gestures with both hands, the gestures from one hand were separated from the other hand. This approach accounts for variations in how gestures are performed between hands, whilst making sure no data leakage occurs.

Throughout the experiments, a batch size of 256 and a fixed number of 768 epochs were utilised as the default settings. These settings were determined as the best-performing configuration, as discussed in detail later in this section. Any variations from these default settings are explicitly mentioned when discussing specific experiments.

### 5.6 Pre-processing and Data Augmentation

To improve the inference accuracy, a pre-processing pipeline was constructed to improve the quality of the data. Several techniques were explored and their effects on the model performance are discussed in the results subsection. To visualise the effects of the pre-processing techniques, appendix A shows the same sample with different combinations of pre-processing steps combined. The pre-processing techniques were applied individually to each photodiode signal. The first technique involved rescaling the data range from approximately $[200, 800]$ to $[0, 1]$ by dividing each value by the maximum value in the sample. This rescaling step ensured that all signals were within a consistent range for more effective model training.

Another technique employed was data normalisation, where the mean of the data was subtracted and then divided by the standard deviation. This is beneficial because it adds further interpretability by bringing the values to a standard distribution. Finally, to mitigate noise in the signals, a low-pass filter was utilised. Specifically, the Butterworth filter was implemented to attenuate high-frequency components and preserve the essential signal information. This noise reduction step aimed to improve the overall signal quality and allow the model to focus on the important patterns of the data instead of unwanted noise.

There are two versions of the pre-processing pipeline. First, during the model training phase, the data underwent the aforementioned transformations using a Python pipeline. This ensured that all data points experienced the same pre-processing steps. After deploying the machine learning model to the microcontroller, a separate pipeline was employed to process the incoming sensor data in near real-time, maintaining consistency with the training pipeline.

In addition to the pre-processing techniques, data augmentation methods were also explored to even further enhance the performance of models [16]. Specifically tailored for the 2D data, two data augmentation techniques were employed. Firstly, the random contrast techniques introduced variability by adjusting the magnitude of the signal's dips and peaks by a factor between [0.05, 1.95]. By randomly scaling the amplitude of the gesture, the models were exposed to a wider range of heights at which the gestures were performed. Secondly, the random translation technique introduced positional shifts within the gesture. By randomly shifting parts of the signal by a factor between [-0.3, 0.3] vertically and [-0.25, 0.25] horizontally, the model was exposed to slight temporal displacements, mimicking natural variations in gesture execution. The technique mainly aimed to improve the model's robustness on gestures that are performed slightly faster or slower.

While time warping is another commonly used data augmentation technique for time series data, it was not used in this study. Future investigations could explore this further.

By incorporating these data augmentation techniques, the study aimed to expand the diversity of the training data and improve the model's ability to handle variations and generalise better to unseen data. The effects of these data augmentation techniques on the performance of the models are discussed in the subsequent results subsection.

### 5.7 Microcontroller Program

As a proof of concept, a microcontroller program was developed to be deployed together with the quantized machine learning model on the Arduino Nano 33 BLE. This microcontroller utilises various buffers to detect when to start gathering data and adjust activation thresholds on the fly. After all data for a gesture is collected, the data from each photodiode individually goes through the aforementioned pre-processing pipeline. After this pipeline is finished the data is passed to the model and an inference is made. Finally, when the model inference is finished, the gesture with the highest confidence is selected. This program was used to collect the inference and processing times for the various models discussed in the results subsection.

## 6 Performance Evaluation

This section presents the key results of this study, which includes a range of experiments and analyses related to ges-

| | Non-Quantized | | | Quantized | |
|---|---|---|---|---|---|
| **Architecture** | **5-Fold Acc. and Loss** | **10-Fold Acc. and Loss** | **Size (bytes)** | **Size (bytes)** | **Inference Time (ms)** |
| BeerNet | $74.5\% \pm 4.1\%$, $2.20 \pm 1.04$ | $75.6\% \pm 7.7\%$, $1.99 \pm 0.79$ | 962,720 | 251,760 | - |
| BeerNet Lite | $76.2\% \pm 3.9\%$, $1.19 \pm 0.42$ | $76.8\% \pm 5.8\%$, $1.31 \pm 0.59$ | 494,752 | 131,744 | $38.3 \pm 0.04$ |
| Last year's model* | $72.9\% \pm 4.1\%$, $0.94 \pm 0.28$ | $72.8\% \pm 6.4\%$, $0.90 \pm 0.29$ | 19,472 | 10,232 | $21.2 \pm 0.02$ |

Table 1: This table compares the various architectures by average accuracies and losses over 5 and 10 folds on a non-quantized version and also reports the model's size. For the quantized versions, it reports the model size and the time it takes to run an inference. The comparison was made with the same pre-processing, data augmentation and hyperparameters. The inference time is recorded on the Arduino Nano 33 BLE, except BeerNet, which did not fit on the microcontroller. *The final model from last year was called 'NLCPP' [13].

ture recognition. The subsections that follow compare different model architectures, explore the effects of varying input shapes, investigate the effects of pre-processing and data augmentation techniques, examine the influence of different hyperparameter values, evaluate gesture confusion, assess the performance impact of handedness, and finally, analyse the effects of different data splitting approaches. Through these investigations, the study aims to provide a comprehensive understanding of the factors that contribute to gesture recognition performance in resource-constrained environments.

## 6.1 Comparing Different Architectures

In table 1 the different architectures that were experimented with are compared by looking at their respective 5-fold and 10-fold accuracy, quantized model size and inference time. As one can see the architecture "BeerNet Lite" performs best in terms of both 5-fold and 10-fold accuracy. The model beats the non-lite version by around 1%, showing that using more filters per layer is not always beneficial. This architecture was constructed during this study and has a standard structure commonly seen with convolutional neural networks. The structure can be seen in figure 5.

It is interesting to note that the model architecture from last year reaches comparable inference accuracies when using the best-performing input shape and pre-processing and data augmentation steps that were experimented with during this study. The model only reached around 65% with the unmodified input shape $(100, 3, 1)$, which was used last year. So, without changing the architecture and only by improving the pre-processing pipeline, adding data augmentation and changing the input shape, the inference accuracy improved greatly. With the miniature size of this architecture, this architecture may actually be preferred over the architectures with higher inference accuracies for some applications.

Even though the quantized model size of BeerNet Lite is over ten times as big as the model from last year, the inference time is only about 1.8 times as long. With both inference times still well below 50 milliseconds, the difference is almost indistinguishable to humans [19].

## 6.2 Comparing Different Input Shapes

Going further, only the results from the BeerNet Lite architecture are shown for further analysis. In order to find the input shape giving the best performance, many different input shapes were explored as one can see in table 2. The right-most number indicates the number of channels used as input to the model, and the other two numbers indicate the width and height of the input. The most obvious way of inputting the data is in the shape $(100, 3, 1)$. This is simply done by adding a third dimension to the original data, indicating the usage of a single channel. However, as experimentation showed, this is not the most optimal input shape. Instead, using $(20, 5, 3)$ gave the best results. This is most likely due to it having a good balance between the number of consecutive samples next to each other and the relation with samples later in the gesture that got put 'below' each other. The $(10, 10, 3)$ input shape most likely broke the signal down too much, ending up with a decreased performance.

| **Input Shape** | **5-Fold Accuracy** | **10-Fold Accuracy** |
|---|---|---|
| (100, 3, 1) | $71.6\% \pm 3.5\%$ | $72.9\% \pm 5.5\%$ |
| (25, 4, 3) | $74.6\% \pm 4.6\%$ | $75.8\% \pm 6.2\%$ |
| (20, 5, 3) | $76.2\% \pm 3.9\%$ | $76.8\% \pm 5.8\%$ |
| (10, 10, 3) | $72.0\% \pm 3.5\%$ | $71.1\% \pm 8.0\%$ |

Table 2: The best-performing architecture, BeerNet Lite, is used to inspect the effects of different input shapes. Other architectures had similar responses to the different input shapes.

## 6.3 The Effects of Pre-processing and Data Augmentation

In table 3 various combinations of pre-processing operations are compared to using no pre-processing at all. The results demonstrate the critical role of pre-processing in improving the model's performance. When no pre-processing is applied, the inference accuracy drops to around 10%, making it no better than random guessing. This poor performance is most likely due to the varying signal ranges observed in different lighting conditions. Without pre-processing, the model struggles to learn effectively from the data. That is also why only using the low-pass Butterworth filter is not making any improvements at all. In contrast, rescaling and normalisation separately both dramatically increase the inference accuracies to around 74% and 73% respectively. When combined, they mostly increase the 10-fold accuracy up to 74.5%. With rescaling and normalisation in place the low-pass Butterworth filter now actually does help and improves the accuracy for both 5-fold and 10-fold by around 2%. It is worth mentioning that all pre-processing step combinations are in the magnitude of microseconds, with all three operations together taking slightly over half a millisecond. These processing times are not the lower bound and further optimisation could still be done if desired.

In order to increase the inference accuracy even further, several data augmentation techniques were tried out. In table 4, the effects of the two augmentation techniques are shown separately and combined. It is interesting to see how only us-

| Pre-proc. steps | 5-Fold Acc. | 10-Fold Acc. | Processing Time |
|---|---|---|---|
| No pre-processing | $10.9\% \pm 0.7\%$ | $10.6\% \pm 1.2\%$ | $0\mu s$ |
| Rescaling | $74.4\% \pm 4.6\%$ | $73.5\% \pm 7.2\%$ | $209\mu s \pm 12\mu s$ |
| Normalisation | $73.1\% \pm 3.4\%$ | $72.3\% \pm 5.6\%$ | $277\mu s \pm 16\mu s$ |
| Butterworth filter | $10.8\% \pm 0.7\%$ | $10.7\% \pm 1.2\%$ | $179\mu s \pm 14\mu s$ |
| Rescaling & norm. | $74.3\% \pm 2.4\%$ | $74.5\% \pm 7.3\%$ | $436\mu s \pm 11\mu s$ |
| All 3 combined | $76.2\% \pm 3.9\%$ | $76.8\% \pm 5.8\%$ | $554\mu s \pm 12\mu s$ |

Table 3: The best-performing architecture is used to inspect the effects of different combinations of pre-processing steps. The processing time is recorded on the Arduino Nano 33 BLE.

| Data augmentation steps | 5-Fold Accuracy | 10-Fold Accuracy |
|---|---|---|
| None | $72.0\% \pm 4.1\%$ | $74.5\% \pm 7.1\%$ |
| Contrast | $73.0\% \pm 3.7\%$ | $74.2\% \pm 6.8\%$ |
| Translation | $75.1\% \pm 5.5\%$ | $75.4\% \pm 5.9\%$ |
| Contrast, Translation | $76.2\% \pm 3.9\%$ | $76.8\% \pm 5.8\%$ |

Table 4: The best-performing architecture, BeerNet Lite, is used to inspect the effects of different data augmentation steps.

ing contrast augmentation decreases the 10-fold accuracy by a small fraction, although this is almost negligible. The translation augmentation adds a bit to both the 5-fold and 10-fold accuracies, but especially the two augmentation techniques together add a noteworthy amount to both accuracies. With these two data augmentation techniques applied, the variation of the inference accuracy between folds seems to decrease as well, making the model have a more stable performance.

### 6.4 Experimenting with Different Hyperparameter Values

Finally, various values for the hyperparameters, batch size and the number of epochs, were explored and evaluated. The results are shown in table 5. The BeerNet Lite model was trained multiple times, whilst keeping the pre-processing and data augmentation techniques consistent. Among the different batch sizes tested (64, 128, 256, and 512), the batch size of 256 showed the highest inference accuracy of 76.8% in the 10-fold cross-validation run. Additionally, across all tested batch sizes, the runs with 768 epochs consistently demonstrated lower overall loss scores compared to those with 1024 epochs. These findings suggest that a batch size of 256 with 768 epochs yields the best performance in terms of accuracy and loss, and has a decent training time as well.

### 6.5 Evaluating Confusion Between Gestures

Figure 3 presents the confusion matrix obtained from evaluating the best-performing architecture and settings. The results demonstrate high accuracy for the four swipe gestures and the double tap gesture, with inference accuracy levels hovering around 90%. However, the rotational gestures, clockwise and counter-clockwise, show a moderate level of confusion, achieving accuracies of approximately 81% and 74%, respectively. Although the tap gesture performs reasonably well, it shows a considerable amount of confusion with the two zoom gestures, which only achieve accuracies of 50% and 40%. Moreover, the confusion matrix demonstrates significant confusion between these two zoom gestures, misclassifying between them approximately 25% of the time. These findings

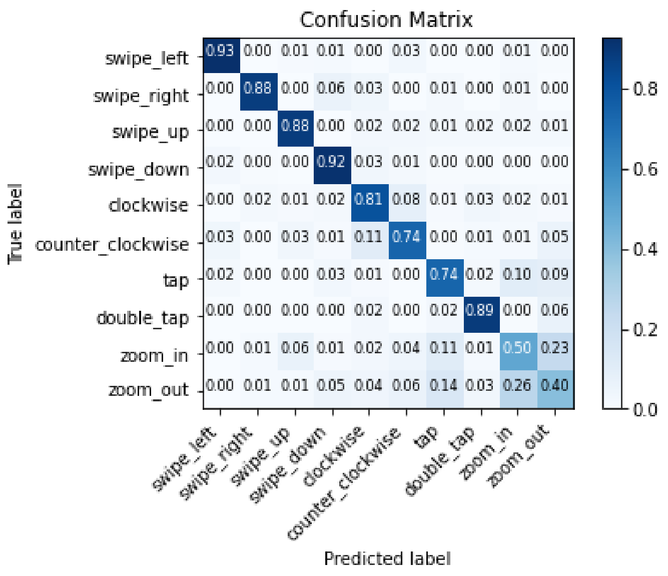


Figure 3: This confusion matrix shows the fraction of how many times the model predicted a sample to have label Y, whilst it is actually labelled X. In this confusion matrix, one can see that the model is pretty confused between the two zoom gestures.

highlight the strengths of the model in accurately recognising swipe gestures and the double tap gesture. However, further improvement is needed to enhance the distinction between the zoom in, zoom out and tap gestures.

### 6.6 Performance Impact of Handedness

Exploring the impact of training the model on only left- or right-handed data offers valuable insights into gesture performance. Table 6 presents the results of training the model exclusively on one-handed data. Training solely on left-handed data leads to an overall performance decrease of approximately 8%, likely due to the reduced dataset size and possibly due to less consistent gesture performance because of right-handed participants performing the gestures with their left hand. Additionally, the loss increases significantly, indicating challenges in properly capturing the patterns of left-handed gestures. Furthermore, the confusion between the zoom gestures intensifies even further, with the model misclassifying the zoom out gestures as zoom in gestures more frequently, as shown in appendix C. Surprisingly, training the model exclusively on right-handed data yielded similar performance to using the entire dataset, although slightly lower. This outcome suggests that certain gestures exhibit less variation between hands, and the slight decrease in performance can again be attributed to the reduced sample size when training exclusively on one hand. Moreover, the decreased variation between folds when only using right-handed data can be attributed to the more consistent manner in which gestures are performed, as the model does not need to account for variations arising from using left-handed data as well. These findings highlight the importance of accounting for handedness in gesture recognition systems.

| Hyperparameters | | 5-Fold | | | 10-Fold | | |
|---|---|---|---|---|---|---|---|
| Batch size | Epochs | Accuracy | Loss | Train Time | Accuracy | Loss | Train Time |
| 64 | 768 | $75.6\% \pm 2.4\%$ | $1.62 \pm 0.41$ | 22min | $75.6\% \pm 7.0\%$ | $1.79 \pm 0.87$ | 49min |
| 128 | 768 | $74.9\% \pm 4.0\%$ | $1.74 \pm 0.72$ | 10min | $75.8\% \pm 7.4\%$ | $1.57 \pm 0.75$ | 22min |
| 256 | 768 | $76.2\% \pm 3.9\%$ | $1.19 \pm 0.42$ | 6min | $76.8\% \pm 5.8\%$ | $1.31 \pm 0.59$ | 13min |
| 512 | 768 | $75.3\% \pm 5.7\%$ | $1.07 \pm 0.47$ | 4min | $75.2\% \pm 7.7\%$ | $1.13 \pm 0.55$ | 10min |
| 64 | 1024 | $73.6\% \pm 3.8\%$ | $2.14 \pm 0.73$ | 23min | $76.0\% \pm 7.2\%$ | $1.98 \pm 1.00$ | 58min |
| 128 | 1024 | $76.0\% \pm 3.4\%$ | $2.01 \pm 0.72$ | 13min | $75.1\% \pm 7.2\%$ | $1.84 \pm 0.85$ | 32min |
| 256 | 1024 | $75.6\% \pm 3.6\%$ | $1.57 \pm 0.51$ | 8min | $75.6\% \pm 7.3\%$ | $1.59 \pm 0.78$ | 18min |
| 512 | 1024 | $75.7\% \pm 3.5\%$ | $1.36 \pm 0.54$ | 5min | $76.2\% \pm 5.9\%$ | $1.36 \pm 0.54$ | 11min |

Table 5: The best-performing architecture, BeerNet Lite, is used to inspect the effects of different values for the batch size and the number of epochs.

| | 5-Fold | | 10-Fold | |
|---|---|---|---|---|
| Hands used | Accuracy | Loss | Accuracy | Loss |
| Only left | $68.5\% \pm 10.2\%$ | $2.18 \pm 1.63$ | $67.6\% \pm 17.2\%$ | $2.08 \pm 1.53$ |
| Only right | $73.6\% \pm 2.9\%$ | $1.44 \pm 0.18$ | $76.6\% \pm 4.3\%$ | $1.46 \pm 0.58$ |
| Both | $76.2\% \pm 3.9\%$ | $1.19 \pm 0.42$ | $76.8\% \pm 5.8\%$ | $1.31 \pm 0.59$ |

| | 5-Fold | | 10-Fold | |
|---|---|---|---|---|
| Data split | Accuracy | Loss | Accuracy | Loss |
| Per candidate | $74.8\% \pm 5.2\%$ | $1.43 \pm 0.42$ | $75.7\% \pm 6.9\%$ | $1.44 \pm 0.61$ |
| Per hand | $76.2\% \pm 3.9\%$ | $1.19 \pm 0.42$ | $76.8\% \pm 5.8\%$ | $1.31 \pm 0.59$ |

Table 6: The best-performing architecture, BeerNet Lite, is used to inspect the effects of using only the left- or right-handed parts of the data and splitting it in different ways.

### 6.7 Performance Impact of Data Splitting Approach

Additionally shown in table 6, comparing the data splitting approach between per-hand and per-candidate reveals marginal performance differences. Training on a per-candidate basis, where each participant's data is treated as a separate entity, shows slightly lower performance across all metrics. Furthermore, it is important to note that this approach may introduce a small bias towards right-handed participants, as not all participants performed gestures with both hands.

## 7 Responsible Research

This section discusses the responsible research practices employed during this study. We recognise the importance of ethical considerations in any research involving human participants. Therefore, we have taken several steps to ensure that this research is done responsibly and transparently. Specifically, The section goes over obtaining informed consent, handling participant privacy, handling data ethically and transparently, informing participants about potential risks and ensuring the reproducibility of the study.

### 7.1 Ethical Aspects

**Participant Recruitment and Selection**

Participants were recruited based on personal acquaintances, with no preference for age, gender, or previous experience with technology. To ensure participation was fully voluntary and informed, we obtained written consent from all participants prior to their participation in the study. The consent form, shown in appendix D, explained the purpose of the study, how the data would be collected, and the potential risks and benefits of participation. It also informed participants of their right to withdraw from the study at any time. All participants were also given the opportunity to ask questions before signing the consent form. To ensure a minimal amount of bias or discrimination in the selection process, potential sources of bias were carefully considered.

**Bias and Fairness**

We recognise that if no proper measures are taken, there may be a potential bias towards right-handed participants in our study. This is due to the unequal distribution of left- and right-handed individuals in the population, which may also be reflected in our participant distribution. The model may be trained on data from more right-handed individuals, making it harder to recognise some gestures performed with the left hand. Due to the time constraints, we made use of stratified K-fold cross-validation techniques to make sure the inference accuracies retrieved are representative of the population and to mitigate this bias towards one group as much as possible. Considering other potential sources of bias, even if they seem unlikely, is also important. Therefore, we also watched carefully for any biases related to age, gender, or previous experience with technology. By doing so, we aim to ensure that our study is fair and representative of the wider population.

**Privacy and Data Security**

Due to the low resolution of the photodiodes used in this study, minimal identifiable data was recorded. However, we took appropriate measures to handle all recorded data ethically and securely. Specifically, we did not store any personal information such as names or contact information together with the data. During the study, only our peer group had access to all the data and consent forms.

### 7.2 Reproducibility

To ensure the reproducibility of the study, the used tool for receiving and storing the data, and the minimal Arduino project which sends the data are made publicly available together with all data collected during the study. This enables other researchers to verify the data collection practices and the dataset itself and replicate the study. The repository is available on GitHub.[5]

[5]https://github.com/arnedebeer/CSE3000-DataCollection

### 7.3 Use of AI Tools

With the rise of AI tools that have become abundant in the last year, a lot of new ethical considerations come into play that were never needed before. It is crucial that Large Language Models (LLMs) are not used as a replacement for critical thinking, especially in studies and papers. This subsection shows the prompts that were used.

**ChatGPT**

ChatGPT was used a few times throughout the writing of this paper.[6] Specifically for improving word usage and spelling.

1. ”What is the right way of writing ”cross validation”? Cross validation or ”cross-validation”?”.
2. ”Is ”skewed fraction” a good way to phrase the difference in distribution? Or would you suggest using another wording? Maybe something with ”demographic”?”.
3. ”Is this sentence grammatically correct? ”With both inference times still well below 50 milliseconds, the difference is almost indistinguishable for humans.””.

**GitHub Copilot**

GitHub Copilot was used to greatly speed up the code-writing part of this project.[7] There are no specific prompts used since the Copilot gives suggestions whenever possible. All suggestions were thoroughly checked before applying.

## 8 Discussion

This study focused on developing efficient signal processing techniques, exploring suitable machine learning models and investigating compression techniques for real-time performance. Through thorough experimentation and analysis, we have uncovered valuable insights and made advancements in the field of gesture recognition. In this chapter, we critically examine our findings, discuss their implications within the context of previous research, address potential limitations, and provide explanations for unexpected results.

The study was able to build upon the foundations established by the previous project conducted last year. We were able to use their PCB design, which saved us a significant amount of time and has contributed to the overall success of our gesture recognition system. Furthermore, the previous project also explored the application of CNNs and Recurrent Neural Networks (RNNs) for gesture recognition. Their experimentation and analysis provided us with a good starting point on which we were able to refine the machine learning models employed in our research.

During our discussions with the authors of the previous project, we uncovered an important aspect related to the reported accuracies of their gesture recognition model. It was revealed that their method of splitting the dataset into train and test partitions introduced a potential data leakage issue, which must be considered when interpreting their reported results. Instead of splitting the dataset per candidate, the dataset was shuffled and randomly split. This approach resulted in a form of data leakage, where the model was exposed to information from the test set during the training process. Because of this, their reported accuracies may have been inflated and were not really representative of the model's performance on never-before-seen participants.

To ensure the integrity of any reported accuracies in this study, the splitting strategy was carefully designed to make sure the dataset was split on a per-hand basis. Doing so will provide a more robust and unbiased evaluation of our gesture recognition model's performance. One should still note that, due to the limited time span of this project, the collected data for the results is based on a single 5- and 10-fold run. Because of this, and the randomness involved with machine learning, the results may not have fully converged yet.

Finally, the reader should also note that the conclusions of this study are based on the results shown in section 5 and are by no means fixed, as various applications may require different characteristics. The reader is therefore encouraged to draw their own conclusions from the results.

## 9 Conclusions

The purpose of this study was to address the challenges of developing efficient signal processing techniques, identifying suitable machine learning models for gesture recognition with real-time constraints and limited computational power, and exploring model compression techniques for deployment on an Arduino Nano 33 BLE with three OPT101 photodiodes. Through extensive experimentation and analysis, this research provides insights into the effectiveness of various approaches and proposes practical solutions for gesture recognition in resource-constrained environments.

This research successfully demonstrated the effectiveness of the employed signal processing techniques on the microcontroller. The proof of concept program uses adaptive activation thresholds, enabling the system to operate in various and constantly changing lighting conditions. The constructed pre-processing pipeline, along with data augmentation, substantially improved the inference accuracy of all explored model architectures. Furthermore, the significance of considering handedness in gesture recognition models was highlighted and different data-splitting strategies were explored, with per-hand splitting showing marginal performance improvements.

As discussed in section 5 various model architectures were experimented with and compared to each other. As different applications may need to prioritise differently, there is no best model. However, if we were to focus on selecting a model that is deployable to an Arduino Nano 33 BLE specifically, and select on a high inference accuracy as well, the simple model shown in figure 5 came out on top.

The final results showcased a well-performing architecture with an inference accuracy of $76.8\% \pm 5.8\%$. Various hyperparameters were experimented with and the final model was trained with 768 epochs, a batch size of 256 and a reshaped input shape from (100, 3) to (20, 5, 3), separating the different photodiode signals into their own respective channel.

Although no data was collected in direct sunlight, the study demonstrated the applicability of the developed system in

[6]https://openai.com/chatgpt

[7]https://github.com/features/copilot

different indoor environments and the findings of this research have practical implications in real-world scenarios, such as deployment in patient rooms in hospitals or residential elevators. The proposed solutions, including efficient signal processing techniques, well-performing machine learning models given real-time constraints and limited computational power, and model compression, can be employed in resource-constrained environments to enable real-time gesture recognition.

In summary, this study contributes to the field of gesture recognition by providing insights into efficient signal processing techniques, well-performing machine learning models, and model compression for resource-constrained environments. This research furthers the development of practical and reliable gesture recognition systems in various real-world applications.

## 10 Future Work

Based on the analysis of the confusion matrix shown in figure 3, it is evident that the zoom gestures (zoom in and zoom out) have the lowest performance compared to other gestures. To enhance the overall performance of any model, it is recommended to focus on improving the recognition of these two gestures. This could be achieved by collecting more data, specifically for zoom gestures, or by redefining the gestures themselves, considering alternate ways of performing them or replacing them with two completely different gestures.

The dataset created and used in this study covers varying conditions in terms of the number of light sources, light intensity, and placement. In future research, it would be interesting to see the effects of specialised datasets tailored to specific application scenarios or environments on model performance. By focusing on a specific use case, one can assess how models perform under targeted conditions, which may provide insights into their suitability for real-world applications.

On a similar note, adding more data to this dataset will always be beneficial. With even more data, a model will be able to capture a larger variety of how gestures are performed and work better in more different environments. Specifically, collecting more data from a diverse set of participants in terms of age and technological knowledge, and different environments will improve the dataset's quality greatly.

Further refinement and experimentation with model architectures and their parameters should be explored. This could involve investigating even more different network architectures and optimising hyperparameters. Doing so may lead to even higher inference accuracies and lower inference times, and improved model size requirements. A promising-looking paper by Chien-Liang Liu et al. [20] on creating a CNN specifically for multivariate time series may be worth looking into.

By researching these aspects in future work, the performance, robustness and applicability of the gesture recognition system using three OPT101 photodiodes may be further enhanced.

## References

[1] D.L. Quam. Gesture recognition with a dataglove. In *IEEE Conference on Aerospace and Electronics*, pages 755–760 vol.2, 1990.

[2] R. Beale and A.D.N. Edwards. Gestures and neural networks in human-computer interaction. In *IEE Colloquium on Neural Nets in Human-Computer Interaction*, pages 5/1–5/4, 1990.

[3] Taegeun Yoo, Van Loi Le, Ju Eon Kim, Ngoc Le Ba, Kwang-Hyun Baek, and Tony. T. Kim. A 137-$\mu$w area-efficient real-time gesture recognition system for smart wearable devices. In *2018 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 277–280, 2018.

[4] Qing Wang and Marco Zuniga. Passive visible light networks: Taxonomy and opportunities. In *Proceedings of the Workshop on Light Up the IoT*, LIOT '20, pages 42–47, New York, NY, USA, 2020. Association for Computing Machinery.

[5] Kirti Aggarwal and Anuja Arora. An approach to control the pc with hand gesture recognition using computer vision technique. In *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 760–764, 2022.

[6] Yande Li, Taiqian Wang, Aamir khan, Lian Li, Caihong Li, Yi Yang, and Li Liu. Hand gesture recognition and real-time game control based on a wearable band with 6-axis sensors. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2018.

[7] Kalpana Lamb and Swati Madhe. Hand gesture recognition based bed position control for disabled patients. In *2016 Conference on Advances in Signal Processing (CASP)*, pages 170–174, 2016.

[8] Texas Instruments. *OPT101 Monolithic Photodiode and Single-Supply Transimpedance Amplifier*, 1 1994. Revised June 2015.

[9] Pete Warden and Daniel Situnayake. Tinyml: machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers, 2019.

[10] Femi Akadiri. Constructing a dataset for gesture recognition using ambient light, Jul 2022.

[11] Dimitar Barantiev. Designing a software receiver for gesture recognition with ambient light, Jul 2022.

[12] Matthew Lipski. Hand gesture recognition on arduino using recurrent neural networks and ambient light, Jul 2022.

[13] William Narchi. Recognising gestures using ambient light and convolutional neural networks: Adapting convolutional neural networks for gesture recognition on resource-constrained microcontrollers, Jul 2022.

[14] Stijn van de Water. Designing an adaptable and low-cost system for gesture recognition using visible light, Jul 2022.

[15] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network.

In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[16] Siham Tabik, Daniel Peralta, Andrés Herrera-Poyatos, and Francisco Herrera Triguero. A snapshot of image pre-processing for convolutional neural networks: case study of mnist, 1 2017.

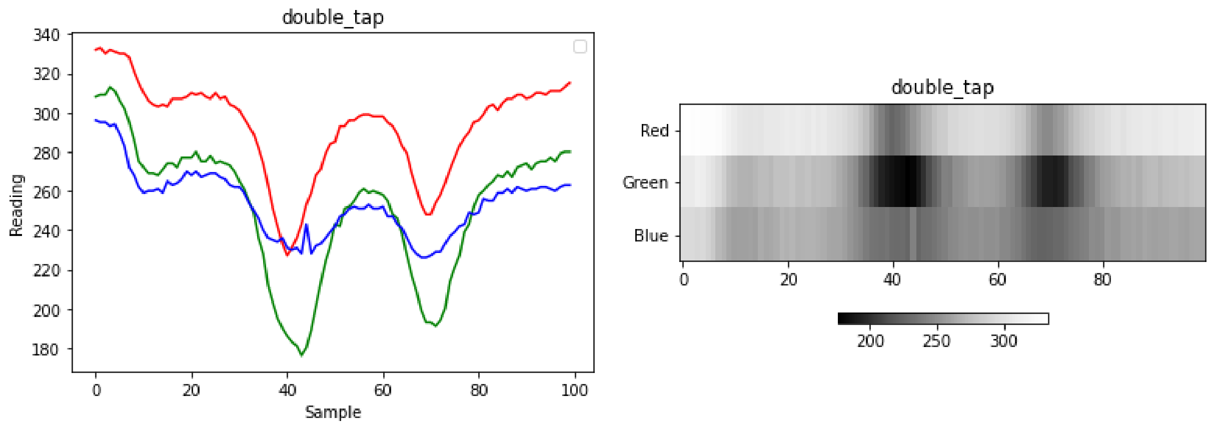
[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

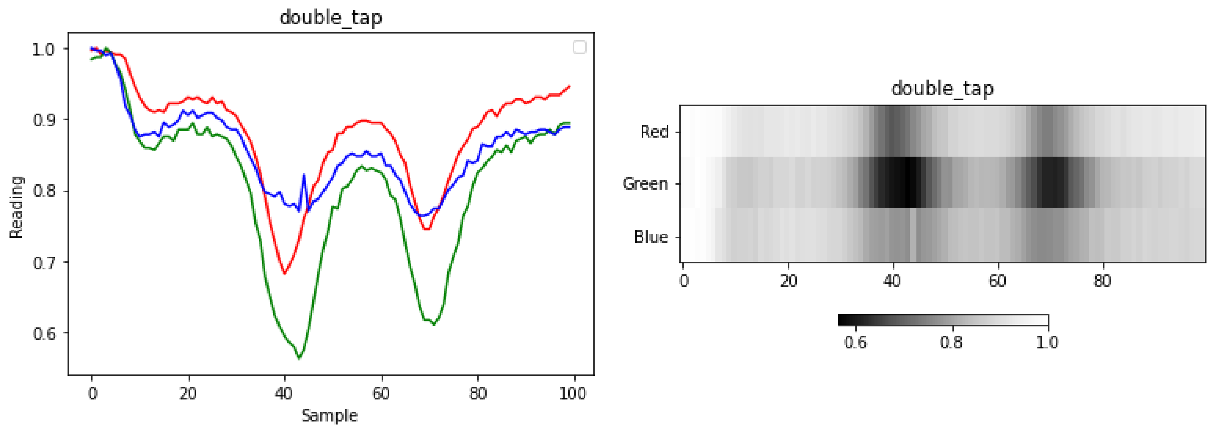
[19] Charles Arthur Nagler and William Merle Nagler. Reaction time measurements. *Forensic Science*, 2:261–274, 1973.

[20] Chien-Liang Liu, Wen-Hoar Hsaio, and Yao-Chung Tu. Time series classification with multivariate convolutional neural network. *IEEE Transactions on Industrial Electronics*, 66(6):4788–4797, 2019.

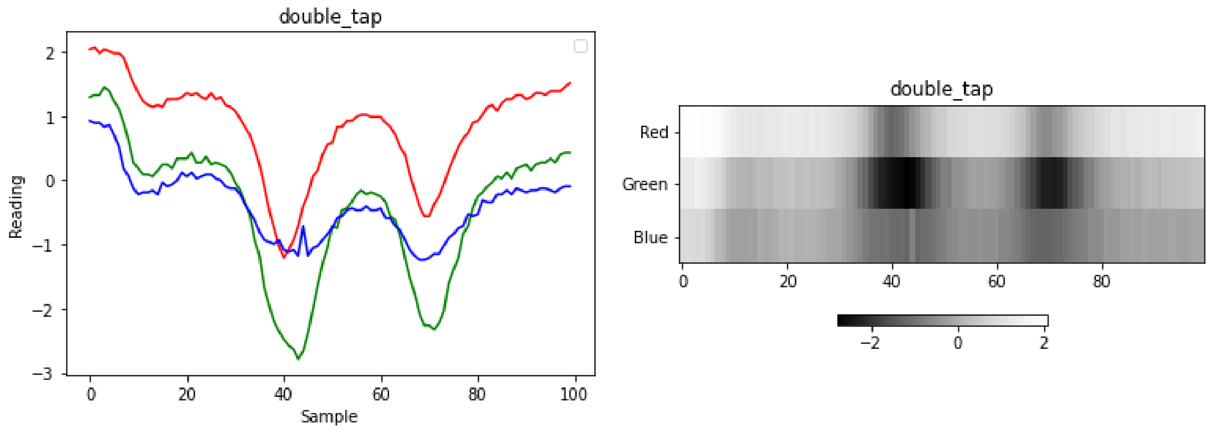
## A Pre-processing Techniques



(a) Graph and image representations of a performed double tap gesture with no pre-processing applied.

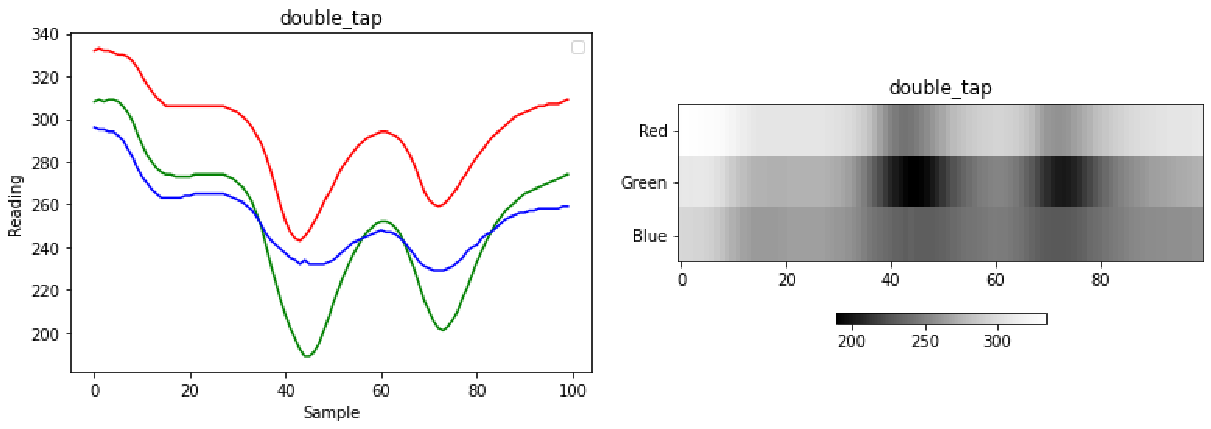


(b) Graph and image representations of a performed double tap gesture with only rescaling applied.

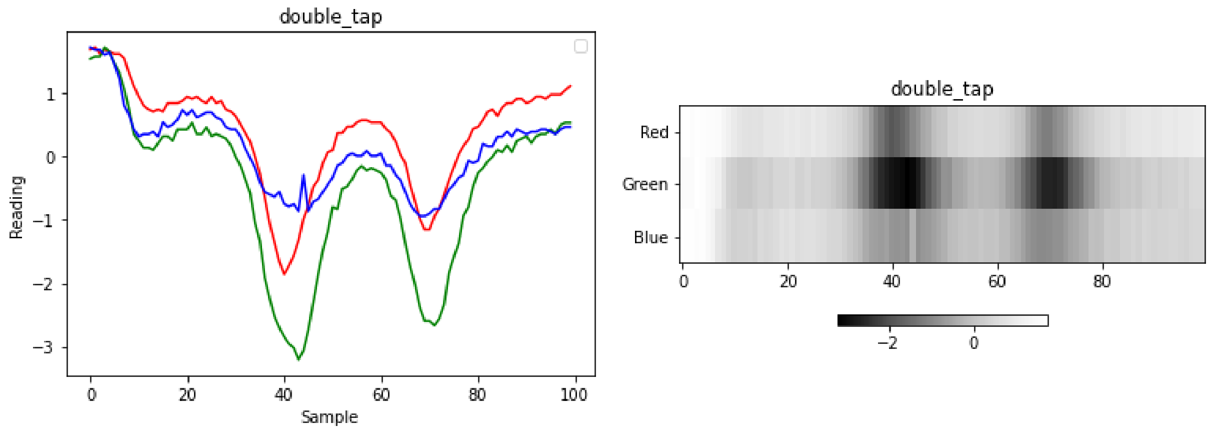


(c) Graph and image representations of a performed double tap gesture with only normalisation applied.

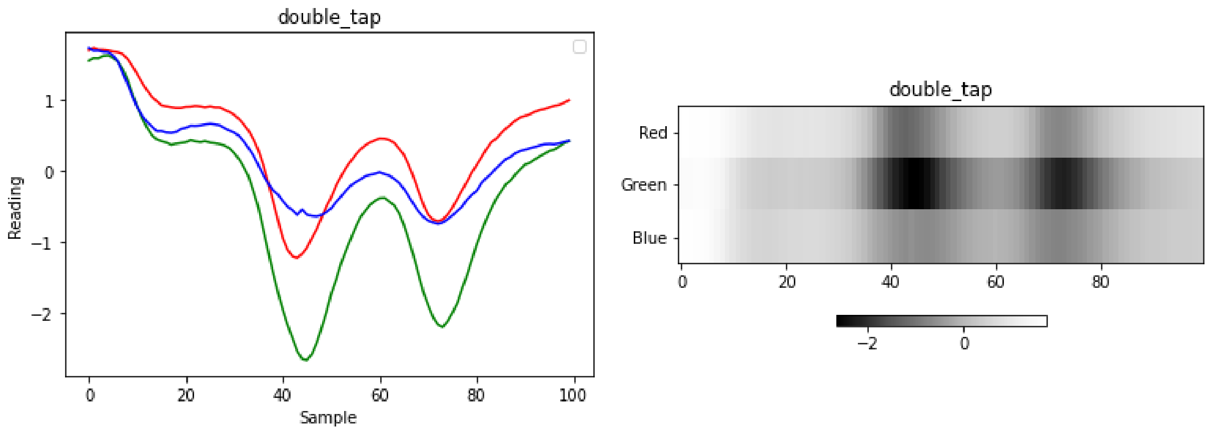
Figure 4: Graph and image representations of a performed double tap gesture with the combinations of the pre-processing steps that are discussed in the results subsection.



(d) Graph and image representations of a performed double tap gesture with only a low-pass Butterworth filter applied.



(e) Graph and image representations of a performed double tap gesture with rescaling and normalisation applied.



(f) Graph and image representations of a performed double tap gesture with all three pre-processing steps applied.

Figure 4: Graph and image representations of a performed double tap gesture with the combinations of the pre-processing steps that are discussed in the results subsection. (cont.)

## B Model Architectures

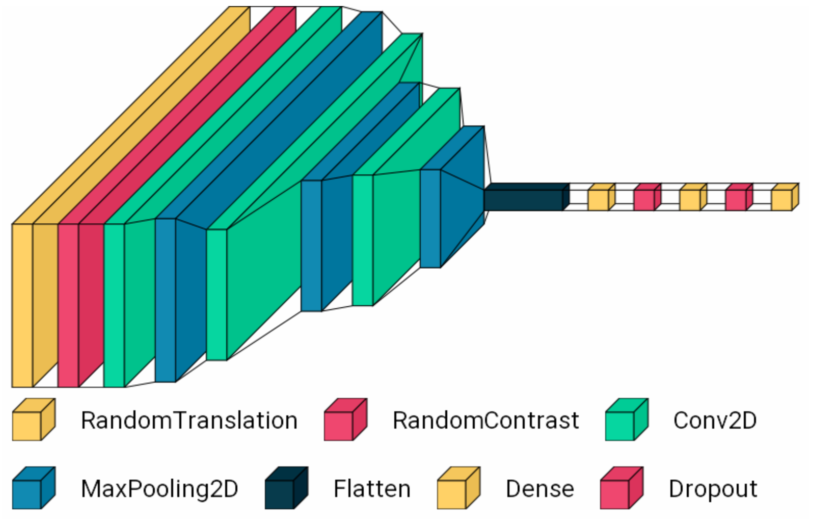
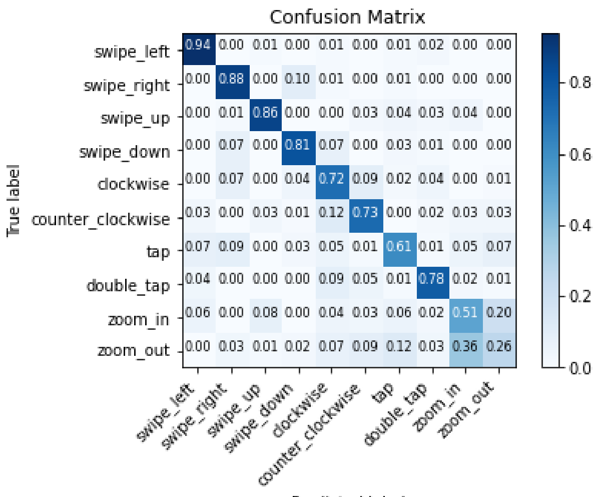
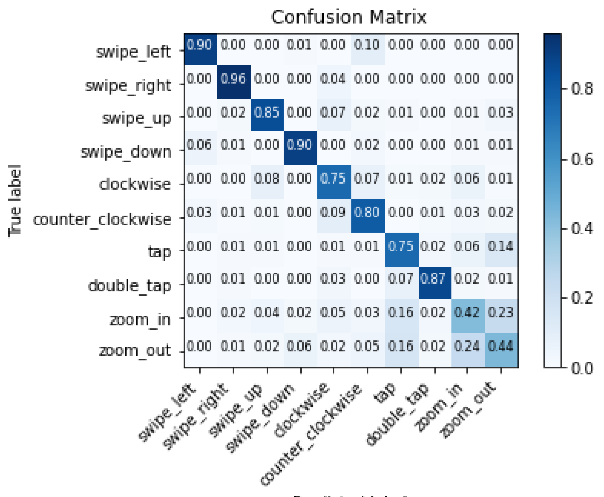


Figure 5: Model architecture BeerNet Lite. The first two layers are only used during training and take care of the data augmentation.

## C Confusion Matrices



(a) Left-handed data only.



(b) Right-handed data only.

Figure 6: The confusion matrices of architecture BeerNet Lite, trained on only the left- or right-handed parts of the dataset.

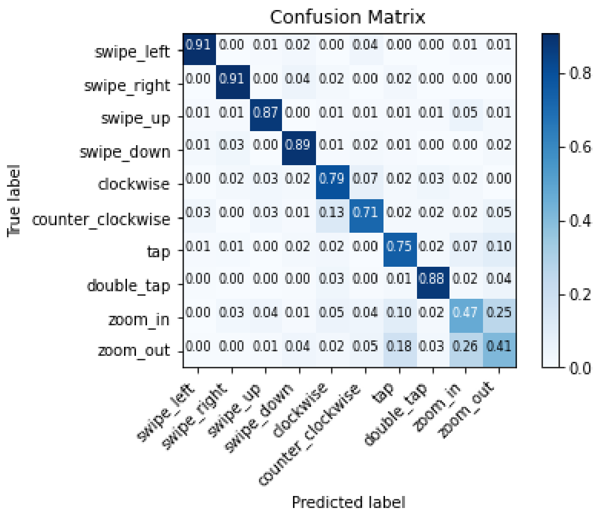
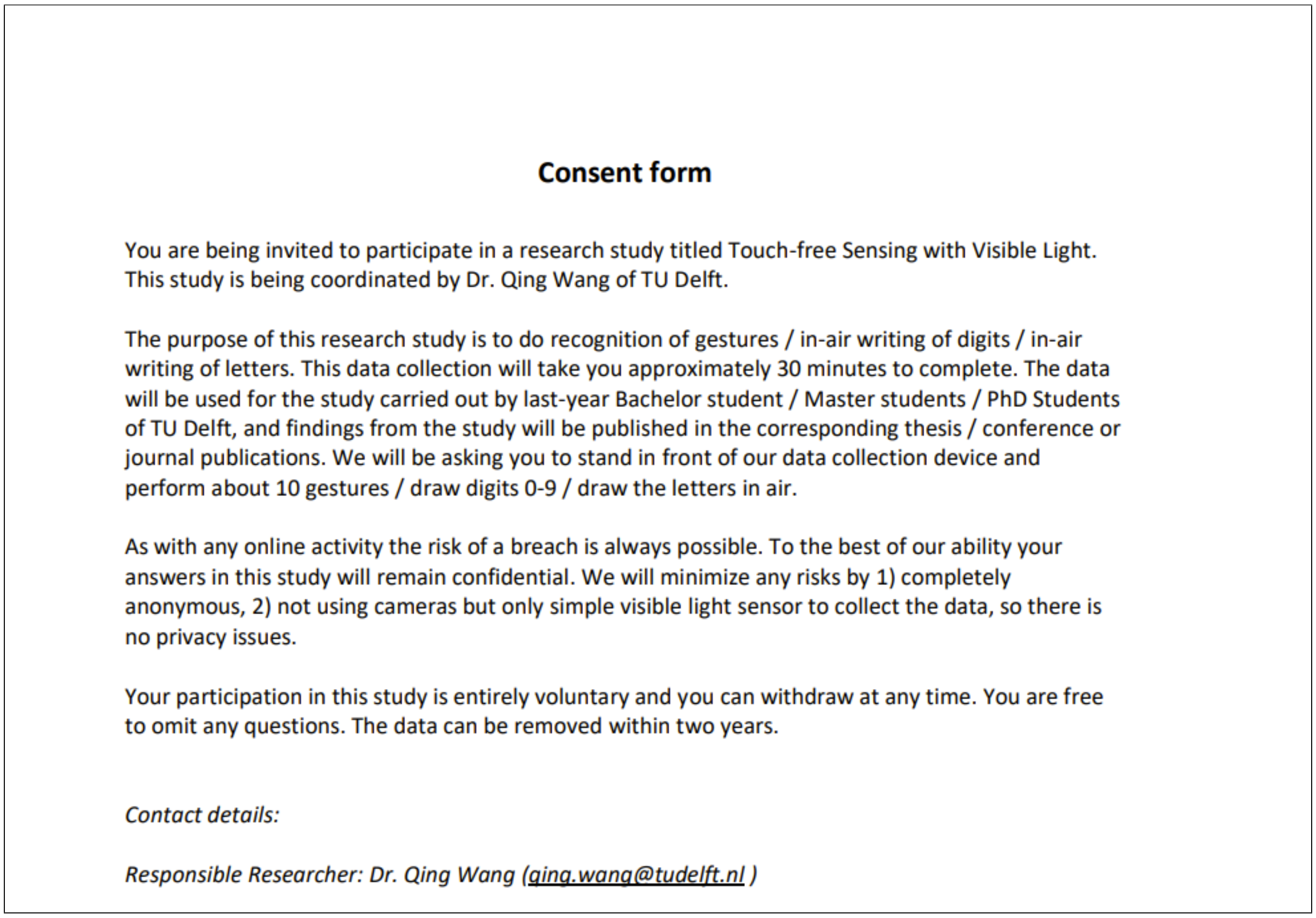


Figure 7: The confusion matrix of architecture BeerNet Lite, trained on data by splitting data per candidate instead of per hand.

## D Consent Form



**Consent form**

You are being invited to participate in a research study titled Touch-free Sensing with Visible Light. This study is being coordinated by Dr. Qing Wang of TU Delft.

The purpose of this research study is to do recognition of gestures / in-air writing of digits / in-air writing of letters. This data collection will take you approximately 30 minutes to complete. The data will be used for the study carried out by last-year Bachelor student / Master students / PhD Students of TU Delft, and findings from the study will be published in the corresponding thesis / conference or journal publications. We will be asking you to stand in front of our data collection device and perform about 10 gestures / draw digits 0-9 / draw the letters in air.

As with any online activity the risk of a breach is always possible. To the best of our ability your answers in this study will remain confidential. We will minimize any risks by 1) completely anonymous, 2) not using cameras but only simple visible light sensor to collect the data, so there is no privacy issues.

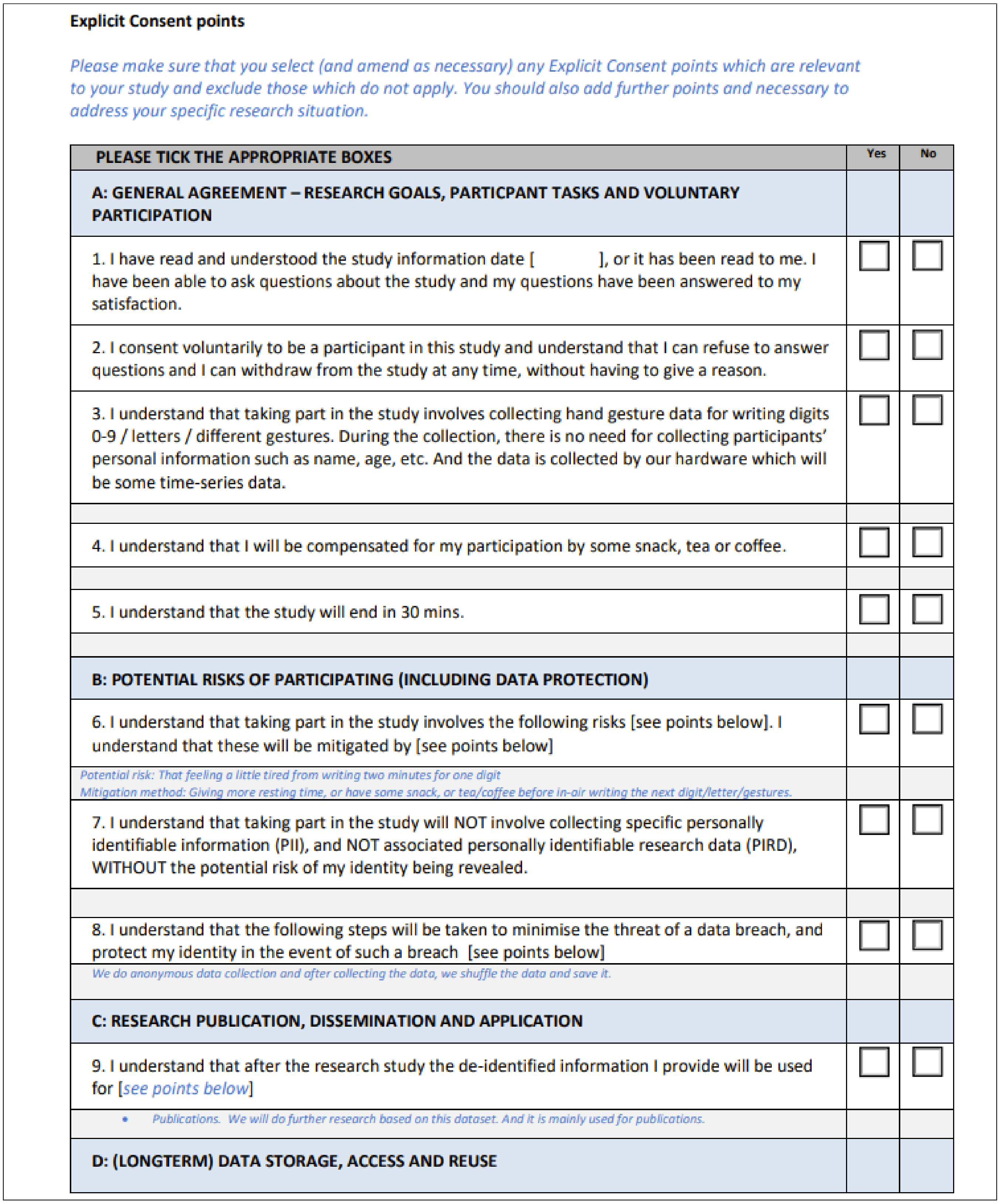
Your participation in this study is entirely voluntary and you can withdraw at any time. You are free to omit any questions. The data can be removed within two years.

*Contact details:*

*Responsible Researcher: Dr. Qing Wang (qing.wang@tudelft.nl )*

(a) The first page of the consent form includes an introduction of the study and roughly explains what the participant is asked to do for this study.

Figure 8: The consent form participants needed to fill in before data was collected.



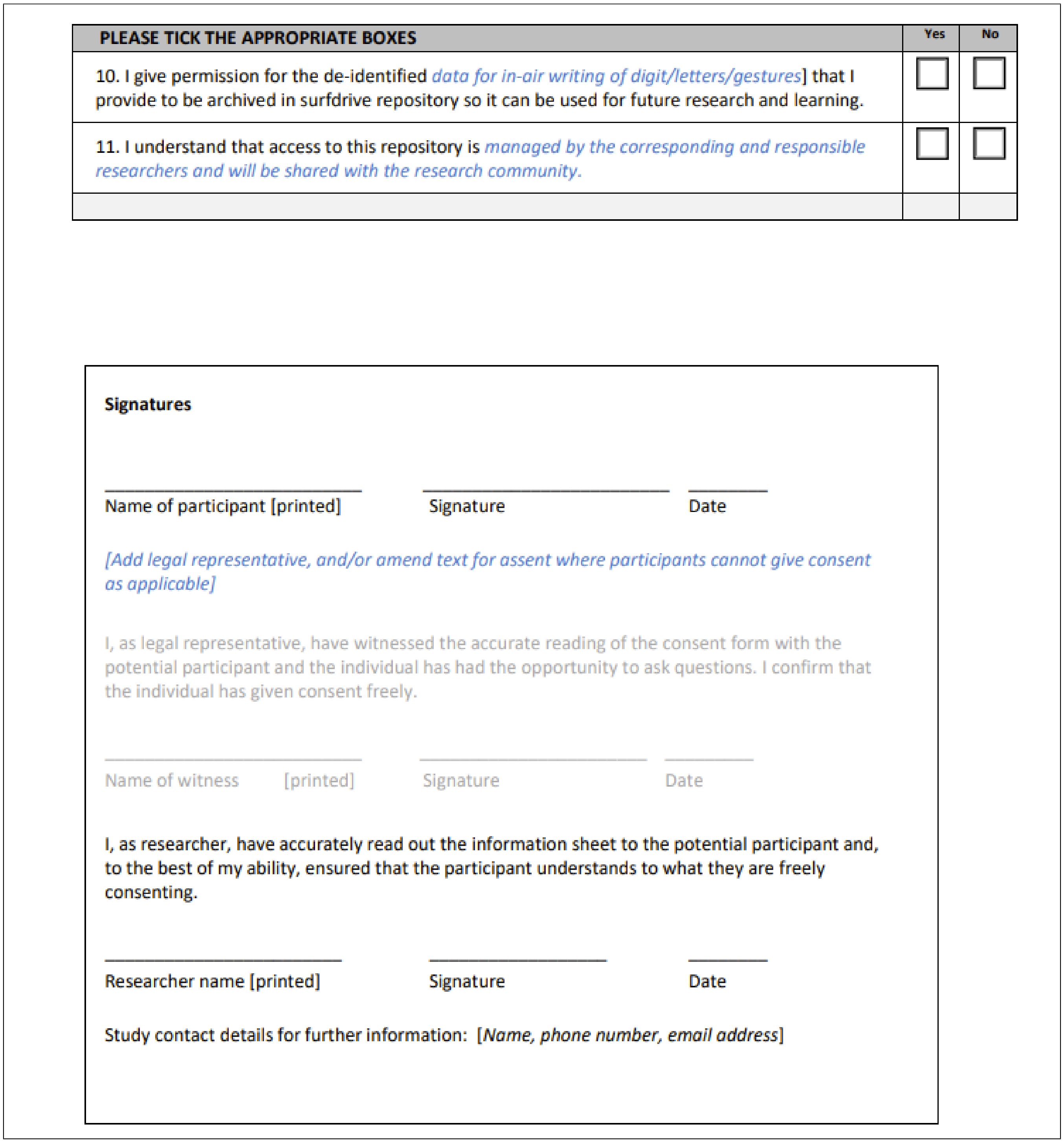
**Explicit Consent points**

*Please make sure that you select (and amend as necessary) any Explicit Consent points which are relevant to your study and exclude those which do not apply. You should also add further points and necessary to address your specific research situation.*

| PLEASE TICK THE APPROPRIATE BOXES | Yes | No |
|---|---|---|
| **A: GENERAL AGREEMENT – RESEARCH GOALS, PARTICPANT TASKS AND VOLUNTARY PARTICIPATION** | | |
| 1. I have read and understood the study information date [ ], or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction. | ☐ | ☐ |
| 2. I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason. | ☐ | ☐ |
| 3. I understand that taking part in the study involves collecting hand gesture data for writing digits 0-9 / letters / different gestures. During the collection, there is no need for collecting participants' personal information such as name, age, etc. And the data is collected by our hardware which will be some time-series data. | ☐ | ☐ |
| | | |
| 4. I understand that I will be compensated for my participation by some snack, tea or coffee. | ☐ | ☐ |
| | | |
| 5. I understand that the study will end in 30 mins. | ☐ | ☐ |
| | | |
| **B: POTENTIAL RISKS OF PARTICIPATING (INCLUDING DATA PROTECTION)** | | |
| 6. I understand that taking part in the study involves the following risks [see points below]. I understand that these will be mitigated by [see points below] | ☐ | ☐ |
| *Potential risk: That feeling a little tired from writing two minutes for one digit*<br>*Mitigation method: Giving more resting time, or have some snack, or tea/coffee before in-air writing the next digit/letter/gestures.* | | |
| 7. I understand that taking part in the study will NOT involve collecting specific personally identifiable information (PII), and NOT associated personally identifiable research data (PIRD), WITHOUT the potential risk of my identity being revealed. | ☐ | ☐ |
| | | |
| 8. I understand that the following steps will be taken to minimise the threat of a data breach, and protect my identity in the event of such a breach [see points below] | ☐ | ☐ |
| *We do anonymous data collection and after collecting the data, we shuffle the data and save it.* | | |
| **C: RESEARCH PUBLICATION, DISSEMINATION AND APPLICATION** | | |
| 9. I understand that after the research study the de-identified information I provide will be used for [*see points below*] | ☐ | ☐ |
| • *Publications. We will do further research based on this dataset. And it is mainly used for publications.* | | |
| **D: (LONGTERM) DATA STORAGE, ACCESS AND REUSE** | | |

(b) The second page of the consent form requests the participant to explicitly check all the consent points in order to take part in the study.

Figure 8: The consent form participants needed to fill in before data was collected. (cont.)



| PLEASE TICK THE APPROPRIATE BOXES | Yes | No |
|---|---|---|
| 10. I give permission for the de-identified *data for in-air writing of digit/letters/gestures*] that I provide to be archived in surfdrive repository so it can be used for future research and learning. | ☐ | ☐ |
| 11. I understand that access to this repository is *managed by the corresponding and responsible researchers and will be shared with the research community.* | ☐ | ☐ |
| | | |

**Signatures**

____________________________ ____________________________ _________

Name of participant [printed] Signature Date

*[Add legal representative, and/or amend text for assent where participants cannot give consent as applicable]*

I, as legal representative, have witnessed the accurate reading of the consent form with the potential participant and the individual has had the opportunity to ask questions. I confirm that the individual has given consent freely.

____________________________ __________________________ __________

Name of witness [printed] Signature Date

I, as researcher, have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

__________________________ ____________________ _________

Researcher name [printed] Signature Date

Study contact details for further information: [*Name, phone number, email address*]

(c) The third and final page of the consent form asks the participant to sign the consent form and also includes the signature of the researcher.

Figure 8: The consent form participants needed to fill in before data was collected. (cont.)