UNIVERSITY OF CAMBRIDGE
Department of Engineering

# A Neural Network Approach For Supporting Gesture Keyboards in Augmented Reality

Author Name: Junxiao Shen

Supervisor: Professor Per Ola Kristensson

Date: 26th May 2020

I hereby declare that, except where specifically indicated, the work submitted herin is my own original work.

*Signed* Junxiao Shen *date* 26th May 2020

# *A Neural Network Approach For Supporting Gesture Keyboards in Augmented Reality*

***Junxiao Shen***[a,b,c]

[a] *Trinity College*
[b] *Department of Engineering, University of Cambridge, UK*
[c] *js2283@cam.ac.uk*

***Keyword:*** Human-Computer Interaction, Deep Learning, Machine Learning, Text Input, Augmented Reality, Gesture-based Keyboard

***Abstract:*** Augmented Reality (AR) can superimpose 3D virtual objects in a real environment to create a mixed reality experience that allows the user to simultaneously interact with digital and physical objects. The core value of AR is not a simple display of data, but the integration of immersive sensations and the natural interactions between the user and the virtual objects.
Text input is a key interaction method that allows users to enter text such as for sending messages or writing documents. Automatic Speech Recognition (ASR) is one of the methods. However, ASR is infeasible in noisy environments such as in factories or in crowded and public places. We propose the AR Gesture Keyboard (ARGK), which is an analogy to the word-gesture keyboard on smart phones [1].
The ARGK consists of a recogniser and a virtual keyboard. The recogniser maps the gesture traces to text, and we decided to use a machine learning model to build this recogniser. A simulated recogniser that simulates this machine learning-based recogniser was firstly designed to full-fill three goals. First, it can allow us to carry out user experiments to study user behaviour through the computation of micro metrics. Different metrics that measure the characteristics of the gesture data were proposed and analysed so that the user's behaviour can be explained in a more systematic way. The micro metrics describe the accuracy of the users while they gesture on the keyboard, how the users perceive where the keyboard lies and how consistent they are at gesturing the phrases on the plane relative to the virtual keyboard plane. Second, the text entry rate revealed from user experiments can show the potential of this new gesture input method to beat the state-of-the-art AR text entry techniques. Third, a large amount of data is required to build the machine learning-based recogniser. We can use the simulated recogniser to collect gesture trace data. Therefore, The outcomes of the simulated recogniser, not only the micro metrics but also the collected gesture trace data, of this project were anticipated to allow future work to generate artificial traces.
We carried out 20 user experiments to collect gesture data and performed a detailed analysis. The data collection experiment requires not only the HoloLens but also a tracking system that tracks the user's fingertip and the HoloLens so that the relative position and orientation between them can be computed. OptiTrack is used as the tracking

system since it offers a system with low latency and high accuracy. This simulated recogniser combined with the user experiments can not only enable us to study the user's natural behaviours but also give us insights on the characteristics of natural gesture traces so that we can implement these characteristics into the data synthesiser in the future. This is because a deep learning model may also require some handcrafted features implemented to improve the overall performance. Therefore, the project's goal is to capture and analyse the characteristics of the gesture trace, user behaviour and to assess the potential of the ARGK.

Since this project has completed the primary objectives, we began to investigate the generation of artificial traces. Therefore, the future work is to use deep learning to generate artificial traces and to map the natural and artificial traces to text. We proposed a neural network to generate artificial traces. The neural network consists of two LSTM layers and one attention mechanism. Different optimisers and training techniques were used to optimise the training process and performance. The performance of this synthesiser can be measured not only by classifiers (if the synthesiser has a representative output, then the classifier can not distinguish artificial and natural gesture traces) but also by the micro metrics proposed in the early stage.

In general, the whole project contains three main stages. First, develop the ARGK with a simulated recogniser; Second, run user experiments to collect gesture trace data and obtain the micro metrics quantitatively from the collected data; Third, develop a full ARGK with a machine learning-based recogniser by using a combination of collected data and synthetic data (which is synthesised from the collected data using another neural network). The fourth-year project contains the first and second stage. The second stage is critical for the third stage. The trace data is limited and the second stage can help us to generate synthetic data. Moreover, the second stage can quantitatively show the great potential of the ARGK system with the machine learning-based recogniser. Nevertheless, preliminary results from the simulated recogniser show the potential of the ARGK to support text entry in AR.

**Contents**


**1 Introduction** **5**

**2 Project Plan** **6**
2.1 ARGK with a Simulated Recogniser . . . 6
2.2 ARGK with a Machine Learning Based Recogniser . . . 7

**3 Background and Theory** **7**
3.1 Augmented Reality . . . 7
3.1.1 Frame of AR Systems . . . 7
3.1.2 Target Recognition, Tracking and Registration Technologies . . . 8
3.1.3 Displays . . . 8
3.1.4 Interactive Technology . . . 9
3.2 Neural Networks . . . 9
3.2.1 Shallow Feed-Forward Neural Network (SFFNN) . . . 10
3.2.2 Shallow Recurrent Neural Network . . . 11
3.2.3 Deep Feed-Forward Neural Network (DFFNN) . . . 12
3.2.4 Deep Recurrent Neural Network . . . 12

**4 Method** **13**
4.1 System Design . . . 13
4.1.1 Data Collection . . . 13
4.1.2 Virtual Keyboard Key Features . . . 14
4.1.3 Experiment Protocol . . . 15
4.1.4 Process the Raw Data . . . 15

**5 Results** **16**
5.1 Simulated Text Entry Rate . . . 16
5.2 Data Analysis Through Micro Metrics . . . 16
5.2.1 Gesture Accuracy . . . 17
5.2.2 Gesture Consistency . . . 17

**6 Data Synthesis** **20**
6.1 Quality Check . . . 20
6.2 Model Overview . . . 21
6.2.1 Long Short Term Memory (LSTM) . . . 22
6.2.2 One-Hot Encoding . . . 22
6.2.3 Attention Mechanism . . . 22
6.2.4 Mixture Density Network (MDN) . . . 24
6.3 Optimisation Methods . . . 24
6.3.1 Cost Function . . . 25
6.3.2 Back-Propagation . . . 25
6.3.3 Gradient Descent . . . 25
6.4 Training Techniques . . . 26
6.4.1 Dropout . . . 26
6.4.2 Batch Training . . . 26
6.4.3 Padding . . . 26
6.4.4 Gradient Clip . . . 26
6.4.5 Learning Rate Schedules . . . 27
6.5 Data Synthesis Results . . . 27

**7 Discussion** **27**
7.1 Limitations . . . . . 27
7.2 Future Work . . . . . 28

**8 Conclusions** **29**

**Appendix A Risk assessment** **29**

**Appendix B Covid-19 disruption** **30**

## 1. Introduction

AR is a technology that can superimpose images, text, video and audio components onto existing images or spaces. AR blends computer-generated perceptual information with the real-world environment through multiple sensory modalities such as auditory, visual and olfactory. AR technology combines target detection, target recognition, virtual modelling and other technologies. It is widely used in archaeology [2], architecture [3], education [4, 5], industrial manufacturing [6] and other fields.

So far, AR can be applied to many platforms, such as computers, tablets and smartphones. Human-machine interaction in the traditional sense usually needs the help of hardware equipment such as a mouse to enable humans to interact with machines. In AR, humans respond to the data information, such as images and audio, obtained from external sensors with gestures and speech to realise the manipulation of virtual objects. This AR interaction between humans and virtual objects brings users a mixed reality experience. Therefore, how to enhance the experience of interacting with virtual objects in the real world has become the focus of most researchers.

Text input is an essential way to interact with the AR device under some circumstances such as sending short messages in industrial applications. A keyboard-based text entry technique is necessary for AR Head-Mounted Displays (HMDs). This is because other input methods such as Automatic Speech Recognition (ASR) cannot be used in certain circumstances, for example, in industrial situations where the environment is noisy or in public situations where privacy is important. Currently, HoloLens 1, a commercially available AR HMD, has a text entry technique based on a gaze-then-gesture paradigm. Dudley et al. [7] found that this strategy yields a mean entry rate of approximately 15 words per minute (WPM). HoloLens 2 has a state-of-the-art text entry technique which is similar to the VISAR keyboard proposed by Dudley et al. [7] and user can directly 'touch' keys with two fingers. However, there are several challenges which are not solved by the above techniques:

1. **Delimitation Problem**: It is hard for the sensor of current AR HMD to find the parallel to 'touching' on a capacitive touchscreen.
2. **Correspondence Problem**: There are a misalignment and delay between the actual finger trace that a user sees and the measured virtual finger trace sensed by the sensor.
3. **Feedback Problem**: It is difficult to provide visual and haptic feedback in a mid-air text entry technique.
4. **Data Sparseness Problem**: As it is known, training a neural network requires a large amount of data. There is no gesture trace data available since there is no such gesture keyboard existing in AR. Therefore, we have to collect our own trace data and then generate synthetic trace data from the collected trace data.

Therefore, an AR Gesture Keyboard (ARGK) innovated from the word-gesture keyboard proposed by Kristensson et al. [8] is proposed to tackle the above-stated problems. Figure 2 illustrates how ARGK works: the user uses their fingertip to draw the trace from $h$ to $d$ to enter the phrase 'hello world'. The main focus of this project is the development of this ARGK with a simulated recogniser on Microsoft HoloLens 1. The aim of this project is to assess the potential of the gesture-based keyboard in AR and characterise the user behaviour that emerges from the gesture traces.

The AR Gesture Keyboard (ARGK) is a system which consists of a virtual keyboard and a recogniser which can map gesture traces (marked by the red curves in Figure 2) to text. Future work will be designing a deep learning model to perform the recognition from gesture trace to text. The way a keyboard translates gesture inputs into text is similar to how an ASR system translates voice inputs into text [9]. Therefore, the previous research in ASR can be leveraged here in this gesture input technique.

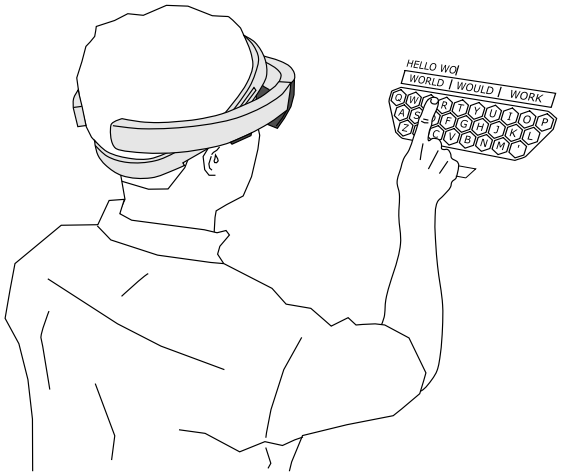


Figure 1: Illustration of a user typing on the gesture keyboard in AR, reproduced from Dudley et al. [7]

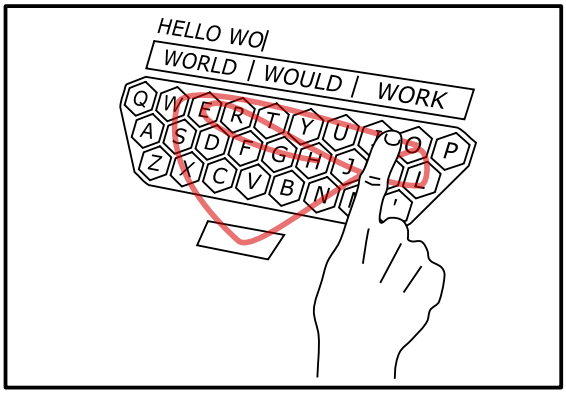


Figure 2: Illustration of a user's gesture trace

Therefore, the ARGK can tackle the above challenges as stated below.

1. **Delimitation & Correspondence Problem**: A machine learning-based recogniser is designed to be independent of the relative position between the gesture and the keyboard. This means that the recogniser translates the gesture trace to text only based on the 3D temporal and spatial shape of the gesture trace instead of its relative position to the keyboard. Therefore, even if the sensor is poor at tracking the relative position of the fingertip to the keyboard or it has a delay problem (large refresh rate), as long as there is a reasonably continuous gesture trace shape recorded by the sensor, the recogniser will be able to translate the gesture trace to text accurately. For a mid-air keyboard, it is not trivial to define what is equivalent to a physical touch. This means that it can be difficult to delimitate user action, and start and stop of user gestures.
2. **Feedback Problem**: The user receives feedback by simply looking at their actual fingertip. Furthermore, this gesture input method is intuitive and can provide a fluid text input experience.
3. **Entry Rate & Accuracy**: This ARGK can potentially boost the entry rate based on the simulated recogniser's tested performance. This machine learning-based ARGK can account for cognitive-motor errors manifesting as misspellings or similar traces such as 'Vampire' versus 'Value'.
4. **Data Acquisition**: We will build an ARGK with a simulated recogniser so that we can collect the trace data in the user experiments. The collected data will then be used to train the deep learning-based synthesiser to produce synthetic data. Therefore, this can successfully solve the data sparseness problem if the synthetic data can pass the quality check.

## 2. Project Plan

### 2.1. ARGK with a Simulated Recogniser

1. **Phase 1: System Design & Data Collection**: A large amount of data is required to train the deep neural network to achieve a good performance out of deep learning. However, it is not feasible to collect this amount of gesture trace data through user experiments. Therefore, synthetic data needs to be generated. The first phase is thus to build a simulated recogniser that can translate the gesture trace into text robustly and use this model to collect gesture

trace data from users. This simulated recogniser simulates a machine learning-based recogniser: the user needs to be given a phrase like 'hello world' to input on the ARGK first, then if this user draws from key *h* to *e* and so on until *d*, 'hello world' will then be shown as output. The difference between the simulated recogniser and the machine learning-based recogniser is that the former one requires the ARGK to know what phrases the user will input and the latter one does not. Therefore, this simulated recogniser is only built and used for collecting the data which will be used to generate synthetic data.

2. **Phase 2: Data Analysis**: The data we collect from the user experiments with the simulated recogniser will be analysed from different perspectives. Micro metrics such as Gesture Accuracy and Gesture Consistency are calculated and analysed to study the behaviour of the users. This is important not only because it can give the implication of how different users draw the trace and perceive the keyboard, but also gives us an understanding of the representative gesture trace data. This understanding can help us to verify the synthetic data generated in the future.

### 2.2. ARGK with a Machine Learning Based Recogniser

This is the future project that I will work on during the summer and PhD. However, I have already started the first phase of the future project since the proposed project has been completed.

1. **Phase 1: Data Synthesis**: The sparseness of the trace data is a critical problem since a large amount of data is required to train a machine learning-based recogniser in the ARGK. Combined with the data collected and the findings from the data analysis stage, we constructed a deep recurrent neural network and trained this neural network to generate synthetic data. This is a critical step because the volume of the currently collected data is not sufficient to train a deep neural network to recognise gesture traces to text based on other similar research and my own experience. Therefore, synthetic data needs to be generated using a deep learning model.
2. **Phase 2: Recogniser Design & Training**: Then the synthetic data combined with the collected real data will be used to train several specifically designed neural networks, and the performances of the trained models will be compared and analysed. The performance criteria include the accuracy, robustness and speed. The best one will then serve as the machine learning-based recogniser. This machine learning-based recogniser will thus replace the simulated recogniser in the ARGK. Therefore, a fully functional ARGK is built.
3. **Phase 3: Recogniser Assessment**: We will do another user experiment with this built ARGK and compare the results with state-of-the-art text entry techniques. Therefore, the built ARGK will be assessed to show the text entry speed it can achieve.

## 3. Background and Theory

### 3.1. Augmented Reality

AR is a collection of a variety of computer technologies and theories, including target tracking technology, display technology, and interactive technology. With the help of those technologies and theories, AR systems can integrate computer-generated virtual scene and the real scene, making the user perceive the virtual scene as part of the real world.

#### 3.1.1. Frame of AR Systems

A complete frame of a virtual reality system should include six main function modules, including scene acquisition, target recognition and tracking, target registration, virtual and real scene fusion, virtual and real interaction and image display modules. The target is the real object that virtual information is attached to.

1. **Scene Acquisition Module**: It stores and pre-processes the input image data collected from input devices.
2. **Target Recognition and Tracking Module**: It recognises and tracks the targets in the pre-processed images and obtains the relative position of the targets.
3. **Target Registration Module**: It can determine the location of the relative position between the virtual object and the real scene, so that it can register the virtual object onto the target location obtained from the target recognition and tracking module, thus to release the fusion of the virtual object and the real scene.
4. **Virtual and Real Interaction Module**: It obtains the user's operation instruction to achieve the control of the virtual object.
5. **Display Module**: Finally, the effect is presented to the user through the display module.

#### 3.1.2. Target Recognition, Tracking and Registration Technologies

Currently, the most commonly used tracking and registration methods in AR systems can be divided into three categories: (1) tracking and registration technology based on navigation and positioning sensors; (2) tracking and registration technology based on computer vision; (3) tracking and registration technology combining computer vision and tracking sensors.

Among the three categories, the tracking and registration technology based on navigation and positioning sensors is common in outdoor environments, mobile devices and wearable devices. This technology requires the use of different sensors to track the target. Computer vision-based tracking and registration technologies use image recognition algorithms to obtain the position of the target in the real scene in real-time, so as to realise the function of target tracking. The third category combines the advantages of both computer vision and the sensor technology so that it has the advantages of the outstanding performance in the outdoor environment and dynamic tracking function in the real scene. In terms of target recognition and tracking, there are many researchers continuously improving the efficiency of target recognition and tracking [10].

#### 3.1.3. Displays

AR has three main displays, including head-mounted displays (HMD), handheld displays and spatial displays. The different approaches to AR have their own advantages and disadvantages. We are using optical-see-through HMD in the project.

1. **HMD**: Head Mounted Display (HMD) is a display device worn on the head which places both images of the real and virtual environment over the user's view of the world. It has two distinct approaches: video-see-through and optical-see-through [11].

(a) **Video-see-through**: The user is required to wear two cameras on the head. The HMD requires the processing of both cameras so that it can provide both the augmented scene and the virtual objects. However, the resolution is unmatched. It has the advantage of better control over the visual result since the augmented view is already composed by the computer. Thus, control over the timing of the real scene can be achieved by synchronising the virtual image with the scene before displaying it [12].

(b) **Optical-see-through**: Optical-see-through employs specialised optics to allow views of the physical world to pass through the lens and graphically overlay information to be reflected in the user's eyes. There is a time lag introduced in the system by image processing. This results in a scene where virtual objects do not attach to the real objects to which they are supposed to be attached. Therefore, optical-see-through HMD is unstable, jittering, or swimming around [13, 14].

2. **Handheld**: Handheld displays, as implied by the name, can be held in the user's hands. It has microcomputers with a display, and it overlays graphics onto the real environment. This technique is a video-see-through technique. There are currently three distinct classes

of commercially available handheld displays for AR. They are smartphones and tablets. The advantage of the handheld displays is that they are widespread, so they have a large user base [15, 16].

3. **Spatial**: Spatial displays employ a larger device compared with the handheld displays. However, it requires neither wearable devices nor handheld devices. It can project graphical information directly onto physical objects through the use of tracking technology, optical elements, video-projectors and other hologram technologies. The spatial display allows users to collaborate with each other in a much more efficient way since there is no lag and difference between the scene perceived by the users. It also has three distinct ways that augment the environment, including video-see-through, optical-see-through and direct augmentation [17].

#### 3.1.4. Interactive Technology

Interactive technology in AR has changed the traditional way of human-computer interaction. Users can send instructions and obtain responses to computers without the help of external hardware devices. Through the interaction and control of virtual objects, people and computers can communicate with each other. The realisation of AR interactive technology is based on target recognition, tracking and registration. We are mainly studying the text input approach, which is as one of the interactive methods.

1. **Text Input in AR**: Google Glass is one of the early optical-see-through HMDs that was developed by Google X. There are a few pieces of literature that are on the text entry in AR and developed on Google Glass. Wang et al. [18] presented PalmType, which uses palms as interactive keyboards in AR. The PalmType displays a QWERTY keyboard onto a user's palm, and it enables the user to type without requiring users to pay any visual attention to their hands. However, this system lacks the auto-correction function, and the text entry speed is below 10 WPM under a variety of user experiments. There is a rich body of research that looks at mid-air text entry methods which can be implemented on AR devices. Dudley et al. [7] presented the Virtualised Input Surface for AR (VISAR) keyboard where users can select keys through single-hand typing. The maximum entry rates achieved in this text input system is 23.38 WPM, and the mean entry rate is 17.75 WPM. Sridhar et al. [19] proposed a multi-finger gesture set for text entry.
2. **Gesture-based Text Entry**: Gesture typing provides users with an efficient way to input text using continuous traces created by finger or stylus. Grossman et al. [20] proposed a gesture-based text entry on the side touchpad of smart eyewear. The gesture involves swiping to select key groups and then select the desired letter. This system achieved a WPM rate of 9.73 after the proposal of the SwipeZone, which replaces diagonal gestures with zone-specific swipes. Most of the gesture-based keyboard research is applied in mobile phones. There are mainly two ways of encoding characters with gestures. The first one is a continues method that maps a word or a character to a continuous stroke such as MDITIM [21], Graffiti [22], EdgeWrite [23] and the Word-Gesture Keyboard (WGK) [24]. WGK was published in the early 2000s and then became one of the most popular text entry methods in smart mobile phones such as the Google Keyboard, and it was recently applied in the latest iPad [25, 26]. The second one is a discrete approach that maps each character to a number of symbolic tokens [27, 28].

### 3.2. Neural Networks

Artificial Neural Networks (ANNs) are now prominent in machine learning and were inspired by neuroscience. In 1943, McCulloch et al. [29] proposed the very early mathematical model of a single neuron that is defined as $y = \phi(\sum_i x_i w_i) = \phi(w^T x)$ with boolean inputs and outputs. The $\phi$ is a Heavyside step function, and the weights $w$ were set by hand. The model thus can compute a weighted sum of several inputs and impose a binary threshold to implement a linear discriminant. This finding showed it was possible to infer a connection between the

biological hardware of a spiking neuron and the categorisation (a function of the brain) from a mathematical model. One of the simplest class of ANN is the multi-layer perceptron, or feed-forward neural network, originating from the work of Rosenblatt in the 1950s.

The perceptron is based on McCulloh-Pitts neuron, and it is the first model that can learn weight from examples of two categories [30]. In 1960, Adaptive Linear Neuron (ADALINE) was developed that can perform linear regression $y = \sum_i x_i w_i + b$ from examples with $b$ as the bias. In 1980, Fukushima et al. [31] proposed the neocognitron, which is a hierarchical multilayered neural network for processing images such as handwritten character recognition. This is inspired by the structure of the mammalian visual system. This model is the basis for the modern convolutional neural network.

In 1986, Rumelhart et al. [32] suggested the parallel distributed processing, which is the prevailing connectionist approach today, that stressed the parallel nature of neural processing and the distributed nature of neural representations. However, the ANNs had poorer performance than some less brain-inspired models that use handcrafted representations between the 1990s and 2000s. Therefore, ANN research was only among a small community of scientists who did not think that the failure of the ANN was due to a fundamental limitation during that time.

It was only until the 2010s, ANNs achieved human-like performances. Krizhesvsky et al. [33] won the ImageNet classification competition [34] by a large margin using a Convolutional Neural Network model. This optimised CNN can match the discrimination ability of human for categorising objects, and it also shares the common features and mechanisms with the corresponding networks in the brain [35]. Moreover, Recurrent Neural Networks (RNN) also achieved a successful result in the processing of sequence data such as ASR [36].

ANN consists of a number of connected computational neurons, just like the biological neurons in Figure 3 arranged in layers. In Figure 3, each dendrite contributes to whether a signal is produced in different proportions and neuron produces an exciting or inhibiting signal along its axons in a non-linear way. The first layer of an ANN is called an input layer where input data enters the network and the second layer or more followed by is called hidden layers which transform the input data as it flows through. The last layer is called an output layer which produces the ANN's predictions. The computational neuron in Figure 4 takes a bias $\mathbf{w}_0$ and a weight matrix $\mathbf{w} = (w_1, ...., w_n)$ as parameters $\theta = (w_0, ...., w_n)$ to model a decision $f(x) = h(\mathbf{w}^T\mathbf{x} + w_0)$ using a non-linear activation function $h(x)$. There are many different activation functions, including the Sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$, the Hyperbolic tangent function: $\tanh(x) = \frac{e^x - e^x}{e^x + e^{-x}}$, the Softmax function: $\frac{e^{a_i}}{\sum_{j=1}^{N} e^{a_j}}$ and ReLu:$[a]_+ = \max(0, a)$. One of the most common features shared between ANN and biological neural network is that they all have a unit which sums the inputs and passes them through a non-linear activation function.

Neural Networks has different forms which can be mainly categorised through two different perspectives which are the number of hidden layers (shallow or deep) and whether the network is feed-forward or recurrent. Shallow Neural Network is more interpretable, and Deep Neural Network can perform more complicated tasks. Moreover, while feed-forward neural networks are universal function approximators, recurrent neural networks are universal approximators of dynamical systems [37].

### 3.2.1. Shallow Feed-Forward Neural Network (SFFNN)

There is no definite boundary between shallow feed-forward and deep feed-forward. However, usually, if the neural network has more than one hidden layer, then this is a deep neural network. Figure 5 shows such a SFFNN. This neural network with a single hidden layer can approximate any continuous function f(x) on a compact subset of $R^n$ (i.e. can in principle learn anything) including rectified linear units that most neural networks are using today [38]. It was generalised by various people in the 1990s [39]. A single layer network is conveniently

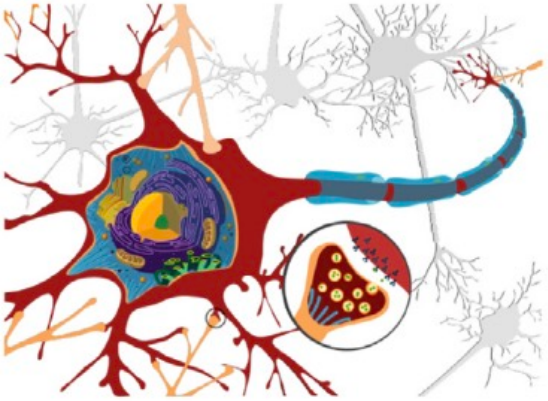

Figure 3: Biological neuron

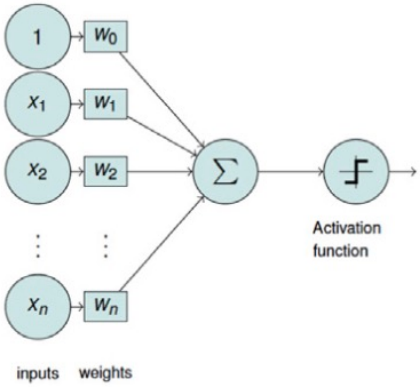


Figure 4: Computational neuron

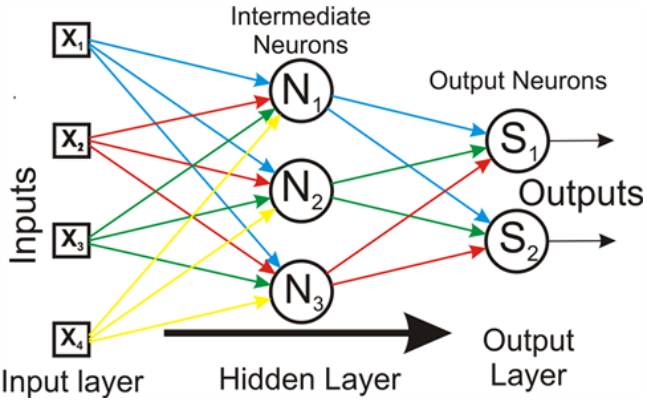


Figure 5: Shallow feed-forward neural network structure

summarised as a linear combination of N individual neurons. $\breve{f}(x) = \sum_{i=0}^{N-1} v_i h(w_i^T x + w_{0,i})$ using combination weights $v_i$. All trainable parameters of this network can be summarised as $\theta = (v_0, w_{0,0}, w_0, ...., v_N, w_{0,N}, w_N)^T$. The difference between the true function f(x) and its approximation $\breve{f}(x)$ is bounded by $\epsilon$ which decreases with increasing N for activation functions that satisfy the criteria such as monotonicity, boundedness, continuity. Therefore, if N is large enough, SFFNN can approximate any function.

### 3.2.2. Shallow Recurrent Neural Network

Recurrent Neural Network (RNN) is a network that contains one or more loops in its directed connection graph such as top-down feedback, and the internal state will evolve in discrete steps. The recurrent connectivity between neurons is that the neuron can receive input from any other neuron in the network and therefore, the activity of neurons in the network is affected not only by the current stimulus but also by the current state of the network shown by Figure 6. RNN is a discrete-time dynamical system that is defined by Equation 1 which has input $x_t$, output $y_t$ and hidden state $h_t$. $\sigma_h$ is a state transition function and $\sigma_y$ is an output function.

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{1}$$

$$y_t = \sigma_y(W_y h_t + b_y)$$

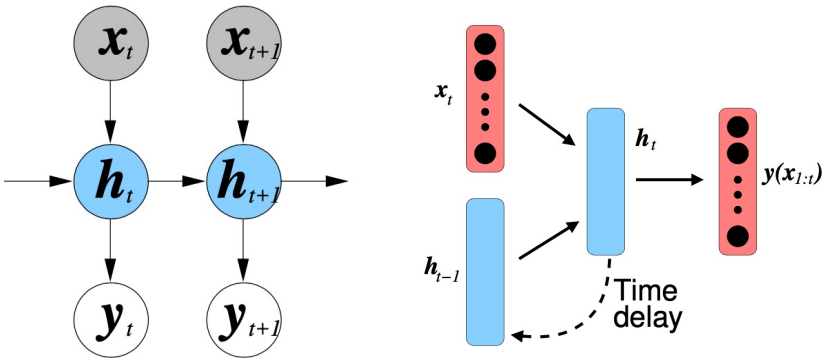


Figure 6: RNN structure

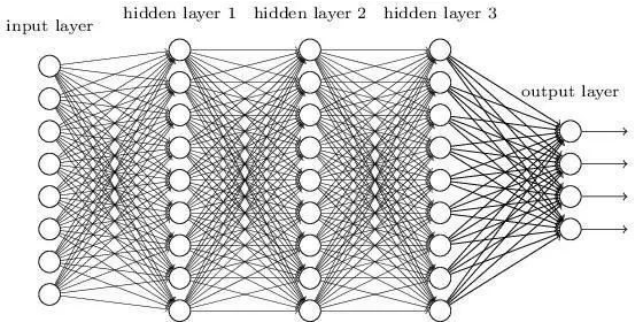


Figure 7: Deep feed forward neural network structure

### 3.2.3. Deep Feed-Forward Neural Network (DFFNN)

Figure 7 shows that DFFNN has a large number of hidden layers so that it enables the network to model highly non-linear and complex mapping. A DFFNN can be exponentially more efficient at representing some functions than a shallow one [40]. DFFNNs reach human-level performance in certain tasks, and early experiments indicate that they are capable of capturing characteristics of cortical function that cannot be captured with shallow models.

### 3.2.4. Deep Recurrent Neural Network

Depth in an RNN is not as clear as it is in feed-forward neural networks since RNN is already deep when unfolded in the time since it is a composition of multiple non-linear layers. An RNN is equivalent to a feed-forward network model with an infinite number of layers with each connected to the next by the same weights matrix. Therefore, a desirable function computed by a very large feed-forward network may be approximated by a small recurrent network when the number of units and connections are limited. There ware earlier version of deep RNN that was constructed by stacking multiple recurrent hidden states on top of each other which allows the hidden state at each level to operate at different timescale [41, 42]. This kind of deep RNN is called Stacked RNN. Pascanu et al. [43] extended explored different concepts of depth in RNNs and proposed two novel variants of deep RNNs (Deep Transition RNN and Deep Output RNN) and evaluated them empirically in objective tasks. The two novel deep RNNs outperformed the conventional RNN and the Stacked RNN on the task of language modelling. Therefore, RNNs with deep-hierarchical structure can outperform shallow RNNs, just like DFFNN can outperform SFFNN.

## 4. Method

We built the simulated recogniser and implemented it into the ARGK system. Therefore, we are expecting it can give the user the same experience as the experience from using the ARGK with a machine learning-based recogniser. This is important since we can collect natural trace data from user experiments. The collected data can then be used for user behavioural analysis through the computation of various micro metrics. Another use of the data is to generate synthetic data from a neural network constructed with recurrent layers. The reason to generate synthetic data is to solve the data sparseness problem since a large amount of data is required to train a neural network that can map the gesture to text.

### 4.1. System Design

Firstly, we would like the performance of the simulated recogniser to be as good as possible so that the data collected from this system can be representative of the actual gesture trace data that the user inputs. Thus high-quality synthetic data can be generated from a synthesiser in the future. Therefore, the ARGK with the simulated recogniser requires the OptiTrack tracking system. OptiTrack is a motion capture system that can accurately track the position and orientation of the markers that are placed above the targets such as the HoloLens and the user's fingertip with low latency. Therefore, OptiTrack is better than the HoloLens's inbuilt sensor. Figure 8 shows the set up of the OptiTrack system. The cameras fixed on the top of the stand in Figure 8 track the reflective balls attached to the user's fingertip and the HoloLens. Therefore I can use the position and orientation data obtained from the OptiTrack system to compute the relative position and orientation between the fingertip and the HoloLens.

There are two main stages in the system design:

1. **Transformation**: The challenge here is to transform the keyboard in the HoloLens coordinate system to the OptiTrack coordinate system so that the fingertip position tracked by the OptiTrack can be compared with the transformed keyboard position. There are two different coordinate systems in the transformation process: one is the OptiTrack coordinate, and the other one is the HoloLens coordinate. The reason that the transformation is challenging is that the convention of the two systems is different both in terms of the handedness and the order of Euler angles. This inconsistency in the conventions and the fact that the HoloLens coordinate is not fixed made the transformation challenging.
2. **Recognition**: After the transformation, a recognition algorithm is implemented to translate the gesture trace to text. The algorithm implements a relaxed region around the keys. As long as the fingertip position reaches the relaxed region of the desired key, the index of the phrase list will update. Finally, when all relaxed regions of the keys have been reached in order, the required phrase will be shown as an output above the virtual keyboard in AR. The tolerance distance determines how large the relaxed region is, so the more accurate the transformation algorithm is, the smaller the tolerance distance can be. Therefore, the accuracy of the transformation and the tolerance distance determines the entry rate: a more accurate transformation and a larger tolerance distance enable larger text entry speed. The vertical constraints on the keyboard are set to be 10 cm which means the user has to reach within 10 cm vertically to the keyboard plane to gesture effectively.

#### 4.1.1. Data Collection

The transformation and the recognition algorithms together formed the simulated recogniser, which was integrated into the ARGK. A user experiment was thus designed to use this ARGK to collect representative gesture data.

Three different sets of gesture data were of interest.

Figure 8: OptiTrack set up and data collection scene

1. **Word**: One word at a time such as 'good'. A word based gesture data set is desirable because we believe that the gesture to word mapping model is easier to train since the current state-of-the-art gesture keyboard - Google Keyboard uses a gesture to word mapping model. Therefore, the LSTM model used to map the 2D gesture data to the text proposed by Alsharif et al. [44] can be leveraged in the synthesising and training process.
2. **Phrases With Spaces**: One phrase at a time where the user must swipe to the space key for a space between the words, such as 'how_are_you'. Each underscore represents that the user needs to swipe to the space key like the trace in Figure 2. A gesture data set of phrase with space, we believe, may be of greater potential to have a better prediction ability and to be more robust than a single word decoder. Therefore, this gesture data set will have the most similarity with the speech signal, and thus much experience in ASR can be leveraged.
3. **Phrases Without Spaces**: One phrase at a time where the user does not need to swipe to space key for a space between the words, such as 'howareyou' so that it will be a continuous gesture input from $h$ key to $u$ key. This data set is desirable because we think that it is much more natural and more convenient to swipe through the keys without going to the space key every time there is a space between the words. Therefore, a recogniser based on this data set, if it is successfully built, will have a much higher entry rate than the first two, especially the first one based on inputting words. However, the difficulty in translating the continuous gesture trace to text is increased as well since this continuity introduces a lot of ambiguity when complicated and long phrases are entered. For example, the trace of a word may be very similar to a trace of a phrase. Then the recogniser might not be able to deal with this ambiguity.

#### 4.1.2. Virtual Keyboard Key Features

Figure 9 shows the layout of the virtual keyboard captured by the camera on the HoloLens. There is no backspace or deletion functionality in the virtual keyboard, so that error correction time does not contribute to the complete time of the gesturing. The 'Is she done yet' is a stimulus that tells the user what to gesture. If the user has drawn the gesture in the right way, another 'Is she done yet' will show up under the stimulus. Furthermore, the stimulus and the keyboard is on the same plane, and the user is required to draw as close as possible to that plane. The green button in Figure 9 is the *DONE* key. This key is the signal for starting and finishing the gesturing. Therefore, the system will begin to recognise the trace when the fingertip moves through the *DONE* key and finish the trace when the fingertip moves out of


Figure 9: First person view of the virtual keyboard in the HoloLens

the done key. This *DONE* key is placed 10 cm in front of the virtual keyboard and 10 cm below the centre of the virtual keyboard. 10 cm was chosen because we would like the *DONE* key to be close enough to the keyboard so that the user does not need to overly move their hand, and far enough so that the recogniser does not confuse the *DONE* key with other keys.

#### 4.1.3. Experiment Protocol

This experiment required the participants to gesture text in the virtual keyboard in AR by wearing the HoloLens under the OptiTrack system, which consists of multiple tracking cameras. In order to carry out this experiment, ethics approval was first obtained. As discussed before, three types of trace data were of interest. Here we selected the phrase with spaces mainly because this type of data would show signs of breaks and could be manipulated easier. The future project, which is the generation of synthetic data and the recognition from gesture traces to text, would be expected to be more feasible with this kind of trace data since experience from handwriting/speech generation and recognition can be leveraged. The speech data and other sequence data usually have specific signals showing the break between each word or characters.

The trace data gestured from 20 participants was collected. Each participant was required to gesture 51 phrases. No participant would see the same phrase. This whole stimulus phrase data set consists of 1029 phrases, and these are 1817 unique words and 6530 words in general. 1020 out of 1029 phrases was used for the data collection. The average number of words in a phrase is 5.348. This phrase data set was collected from two existing phrase data sets, which are the Enron data set [45] and MacKenzie data set [46]. The criteria for selecting the phrases was to select phrases between 5 and 45 characters. The whole stimulus phrase data set contains 500 phrases (average number of words in a phrase 5.428) from MacKenzie data set and 529 phrases (average number of words in a phrase 5.268) from Enron data set.

Each experiment lasted approximately 40 to 50 minutes, among which 20 minutes was for training and the rest of the time was for actual data collection. Different users showed a variety of behaviour, including the perception of the depth of the virtual keyboard, and adaptation to the keyboard system such as the learning rate and the speed of gesturing.

Different micro metrics were used to analyse user behaviour and assess user performance, and this would help us to have a deeper understanding of the actual gesture traces. Therefore, we can use these metrics to develop a better synthesiser to generate artificial trace data. We can also have a better measure to evaluate the performance of the synthesiser.

#### 4.1.4. Process the Raw Data

The data is logged starting from the time that the stimulus begins to show up on the keyboard, and the user may need time from taking their hand on the desk to the ‘ready

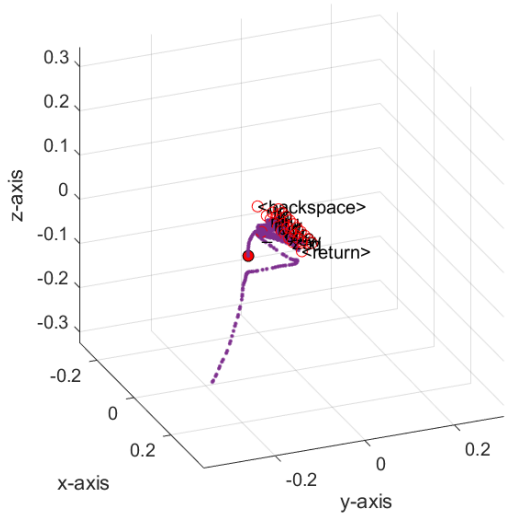


Figure 10: Unprocessed trace data, with tails and background noise

position' which is the *DONE* key position. Therefore, many different methods were used to cut the tails of the trace in Figure 10, which is referred to as 'dehooking'. We tried to determine the point where the tail stops at the keyboard plane by finding the first minimum in the fingertip velocity curve. However, this method turned out to be not reliable since users have a variety of gesture behaviours and the theory above does not apply to every user. We then determined the point by finding when the gesture tracepoints began to move on the same plane. Figure 11 shows the processed trace.

## 5. Results

### 5.1. Simulated Text Entry Rate

We would like to analyse the text entry rate because this is an important quantity in text entry research. The potential of the ARGK can be analysed through this simulated text entry rate. The simulated ARGK we have completed will be representative of the completed ARGK with a machine learning-based recognition model. Therefore, we believe that the simulated entry rate indicates the true entry rate that can be potentially achieved by the completed ARGK.

Words per minute (WPM) is the number of words entered divided by the time taken. The number of words is counted to be the effective word count which is a nominal word length of five 'keystrokes' divided by the total character length of the phrases (including spaces). The result showed that the maximum entry rate is 36.0 WPM, and the minimum entry rate is 12.2 WPM. The standard deviation is 6.2 WPM. Figure 12 shows that there is a general trend that participant typed faster over the experiment, which is expected due to the increased learning curve of the user.

Figure 13 shows that there are two subsets of participants who can gesture really fast and who has a slow gesture speed, respectively. However, most of the participants lie within the 16 to 24 WPM interquartile range. Therefore, it shows the potential of the ARGK to beat most of the state-of-the-art text entry methods in AR.

### 5.2. Data Analysis Through Micro Metrics

We use different micro metrics to measure the performance of the simulated recogniser and to analyse the behaviour of the users. Micro metrics are important since they are quantitative

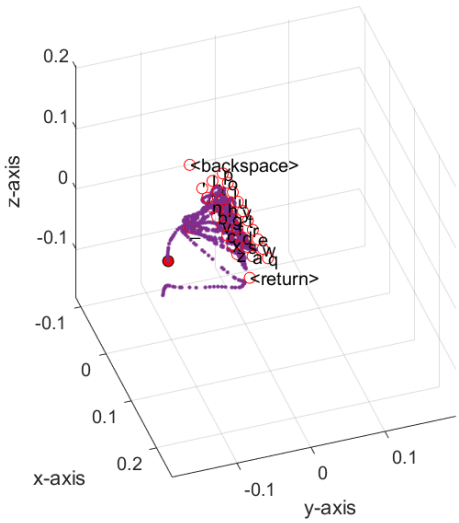


Figure 11: Processed trace data, without tails and background noise

and thus provide us with a rigorous user behavioural result. We can also use the micro metrics to analyse the synthetic data generated from the data synthesiser. The micro metrics results obtained from the currently collected data will be compared with the micro metrics obtained from the synthetic data. The comparison will be analysed, and if the difference of the micro metrics is small, this suggests that the synthetic data is representative of the collected natural data. Therefore, another importance of micro metrics is for the quality check of the synthetic data.

### 5.2.1. Gesture Accuracy

Gesture accuracy represents the accuracy of the user's typing behaviour. We would like our keyboard to have an accuracy small enough to make an accurate recognition and smart prediction, and large enough so that the system is robust.

The accuracy is measured by the smallest tolerance length that the trace can pass the recognition model. A tolerance length of 5 cm is set, and we believe if we make the tolerance length larger, the entry rates will be higher since the simulated recogniser will be more robust. However, a small tolerance length (relaxed region) was used in order to ensure that the gesture trace data collected is accurate and representative of typical human behaviour.

Figure 14 shows the trend of different participants for the number of failed phrases against the change of the tolerance length. The failed phrases refer to the corresponding traces that fail to pass the simulated recognition model. Most participants show a decreasing gradient in the middle of the curve, and this suggests that they have achieved good accuracy.

Figure 15 shows that most participants can achieve accuracy within 3 cm, and this means that if the tolerant length is set to be 3 cm, most of the data collected can still be valid. Whereas a small subset of the participants can not draw an accurate trace since they do need 5 cm to achieve a large number of successful phrases.

### 5.2.2. Gesture Consistency

Gesture Consistency is an important metric that analyses the user's perception of the virtual keyboard. Different user may have a different depth perception of the virtual keyboard shown in the HoloLens. Therefore, the depths of the keyboard perceived by different users are very different and the traces they gesture fit into planes with a variety of displacement and rotation angles from the virtual keyboard. Therefore, based on this stated assumption, we set a large

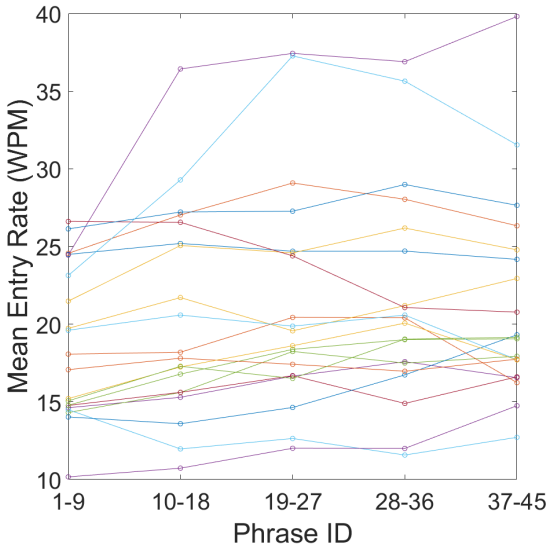


Figure 12: Text entry rate (WPM) over the experiment for different participants

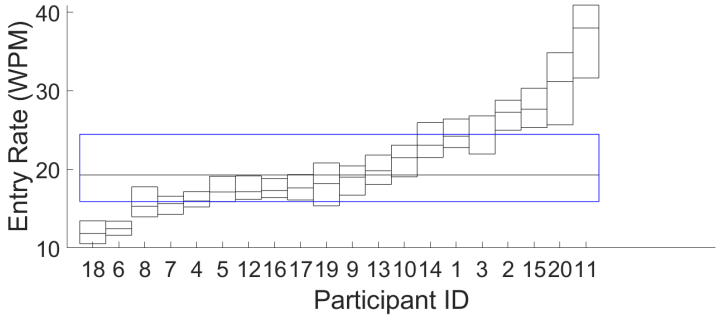


Figure 13: Median and interquartile range for the text entry rate for each participants and across all the participants

tolerance vertically through the keyboard; that is, the user is allowed to gesture above or below the virtual keyboard without affecting the recognition performance. This assumption was proved to be true after computing this depth consistency metric from the collected data. Different participants show a variety of sense of depth.

Figure 16 shows that participants tend to gesture away from the keyboard as the experiment goes on since they tend to relax over the experiment. Therefore, the depth difference is larger at the end of the experiment.

Figure 17 shows that most people will tend to gesture in front of the keyboard instead of right on it or behind it. However, some people still tend to gesture behind the keyboard but by a little distance. There are several outliers who do not have a strong depth consistency.

As discussed above, we assumed that the user may have a different perception of the keyboard position and orientation. Therefore, the keyboard plane perceived by the user may not be aligned with the actual virtual keyboard plane. This results in an angle and a displacement between the plane that the user draw gesture on and the actual keyboard plane. This micro metrics can tell us how good the user is at perceiving the relative position and orientation of the virtual keyboard.

Here we focus mostly on the rotation angle since the depth/displacement has been analysed. Therefore, to verify this assumption from the relative angle perspective, Figure 20 to 22 show

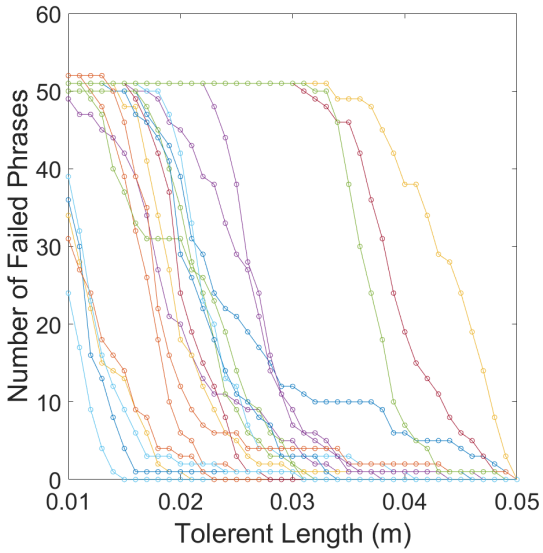


Figure 14: Number of failed phrases against the tolerant length

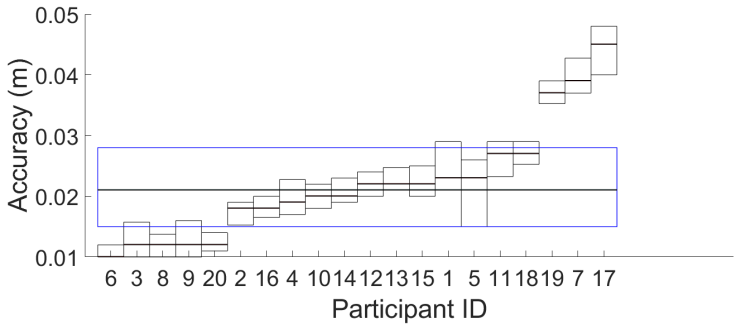


Figure 15: Median and interquartile range for the tolerence length for each participants and across all the participants

that the gesture data does fit into a plane that has a certain angle relative to the actual keyboard. Theta $\theta$ is the angle between the normal vector to the plane and the Z-axis in the Z-X plane, and Beta $\beta$ is the angle between the normal vector and the Z-axis in the Z-Y plane as shown in Figure 18 and Figure 19. In other words, Theta angle is the rotation angle between the plane that the gesture trace fit on and the keyboard plane with the Y-axis and the Beta angle is the rotation angle with the X-axis.

The values of the median and interquartile range for both the Theta $\theta$ angle and the Beta $\beta$ angle in Figure 20 and Figure 21 are expected. Beta angle is caused by the fact that the keyboard is not entirely vertical and the user will tend to gesture vertically. Theta angle is caused by the fact that the keyboard sometimes does not show exactly in front of the user but with some angle tending towards the right or left. RMSE is the root mean square error between the trace data points and the plane that they fit into, and it shows how closely they fit together.

Figure 22 shows that users are relatively good at locating a plane and are consistent at gesturing on that plane since the RMSE has a median around 5.7 mm.

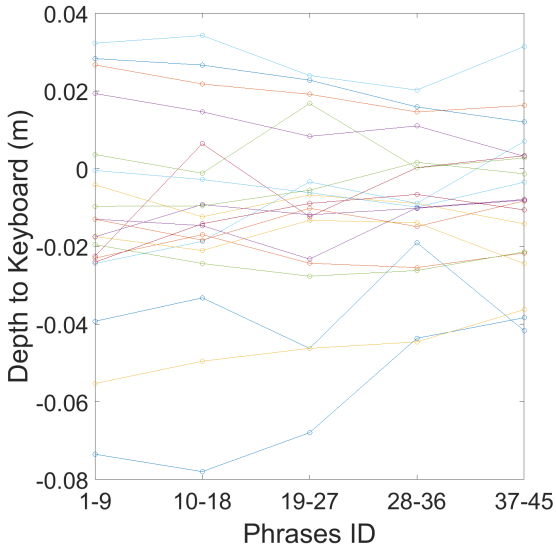


Figure 16: Depth to the keyboard over the experiment

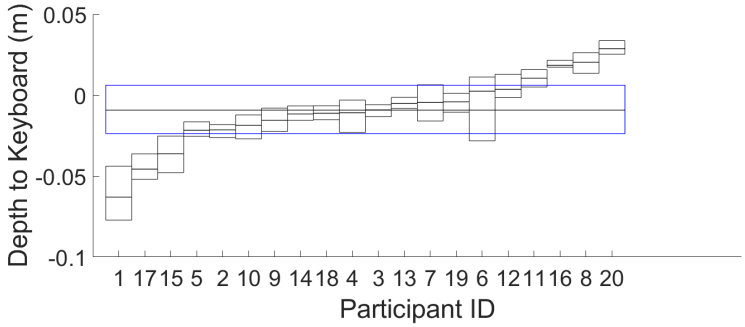


Figure 17: Median and interquartile range for the depth to the keyboard for each participants and across all the participants

## 6. Data Synthesis

We will generate synthetic data based on the data we obtained from the user experiments. There are many state-of-the-art generative models, one of the most popular ones is the Generative Adversarial Networks [47] which uses a generator to produce training samples from the vector of random noise and uses a discriminator to classify whether a training sample is real or produced by the generator. However, this model is mainly used in computer vision, but this can still be of use to our project. More relevant models in Text to Speech can be important as well, such as Recurrent Neural Network. WaveNet is one of the most well known RNNs [48]. However, we tend to use a simple model without a deep hierarchical structure since we don't have that much data, and overly complicated model such as WaveNet may lead to over-fitting.

### 6.1. Quality Check

A quality check needs to be conducted to see whether the synthetic data is representative of the real data. Different methods can be used such as calculating the KL divergence like TSNE (T-Distributed stochastic neighbour embedding) which learns a low-dimensional embedding of a feature vector by minimising KL between similarities in both spaces [49]. Classifier approach can also be used for quality control purposes. If it fails to distinguish the real data and synthetic data, then it is a good indicator. Lastly, the micro metrics obtained from the collected

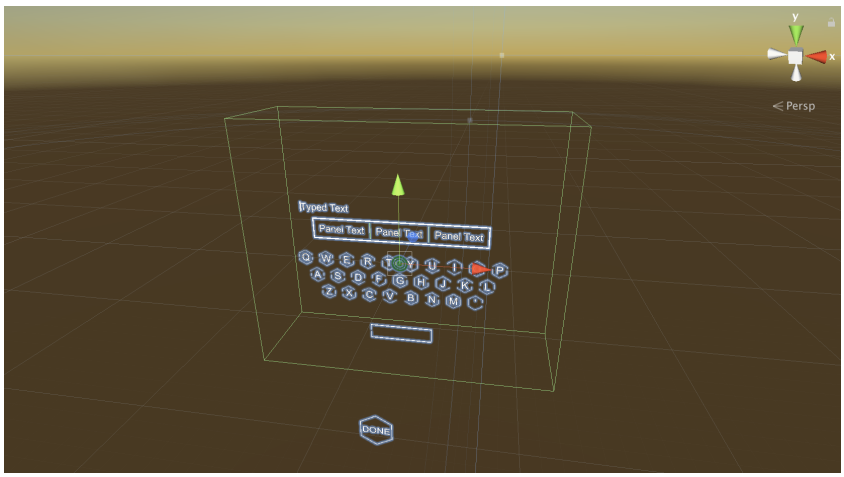


Figure 18: A view of the virtual keyboard to show the X-Y-Z coordinates and orientation

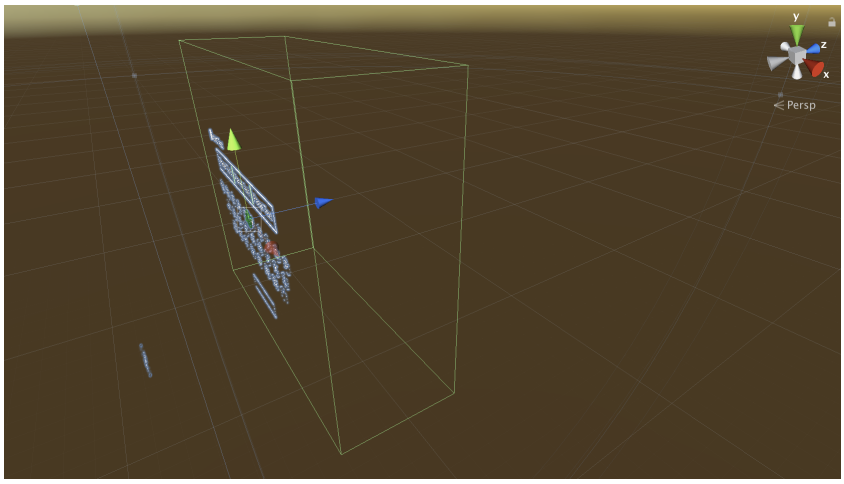


Figure 19: A view of the virtual keyboard to show the X-Y-Z coordinates and orientation

representative data can be used to compare with the micro metrics computed from the synthetic trace data to make sure the synthetic data shows a variety of user behaviours.

### 6.2. Model Overview

RNNs are often used to solve sequence problems. The input sequence is long so the neural network requires large memory. Therefore, I would attempt to use a neural network based on Long Short-Term Memory (LSTM) [50, 51]. The backbone of the model consists of three LSTM cells and one attention mechanism, as shown in Figure 23. The number of the layers depends on how deep I would like to make the network to be. Based on the current data volume, three layers is enough. A deeper network might generate better results, but that requires a large data set. A deeper neural network is harder to train as well. There is a custom attention mechanism which digests a one-hot encoding of the phrase we want the model to synthesis. This attention mechanism is used because the input and the target output have different lengths. The Mixture Density Network on top chooses appropriate Gaussian distributions from which to sample the next tracepoint, adding some natural randomness to the model. Details of the specific structure are discussed in the following subsections.

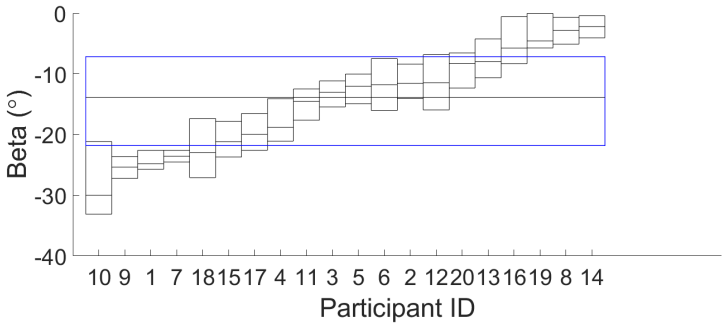


Figure 20: Median and interquartile range for the beta angle for each participants and across all the participants

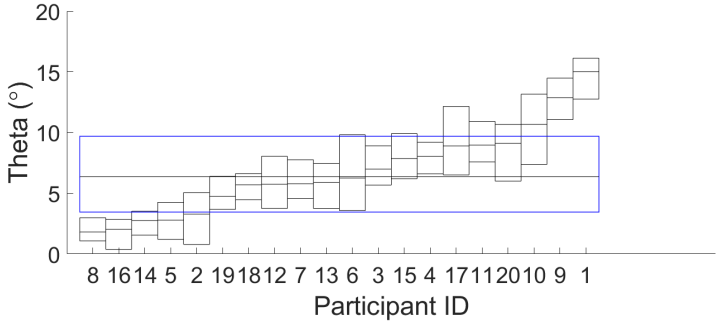


Figure 21: Median and interquartile range for the theta angle for each participants and across all the participants

### 6.2.1. Long Short Term Memory (LSTM)

LSTM is a special case of RNN. An LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is able to connect previous information to the present tasks. LSTM can handle long term dependencies much better than the standard version of RNN. Alsharif et al. [52] proposed a hybrid approach which is a combination of recurrent neural network (LSTM) and the conventional Finite State Transducer decoding to address some of the challenges in gesture typing. The challenges include elision, co-articulation and high variability.

### 6.2.2. One-Hot Encoding

One-hot encoding is a process by which categorical variables are converted into a form that could be provided to a machine learning algorithm to do a better job in prediction. Therefore, each of the phrases is converted to a vector using one-hot encoding. There are other coding methods that we can use, but considering the amount of the phrases that we were processing, one-hot encoding is more than enough. It is a popular encoding method in machine learning.

### 6.2.3. Attention Mechanism

An attention mechanism is an important tool in sequence to sequence (seq2seq) problems. The seq2seq problem relies on reading a complete sequence and compress all information into a fixed-length vector. Attention mechanism fixes this problem since it allows a machine translator to look over all the information the original sentence hold, then generate the proper character according to the current character it works on and the context. It can also allow the translator to focus on local or global features. I have used the attention mechanism for this reason. The input and output have very different lengths. The attention mechanism I have implemented

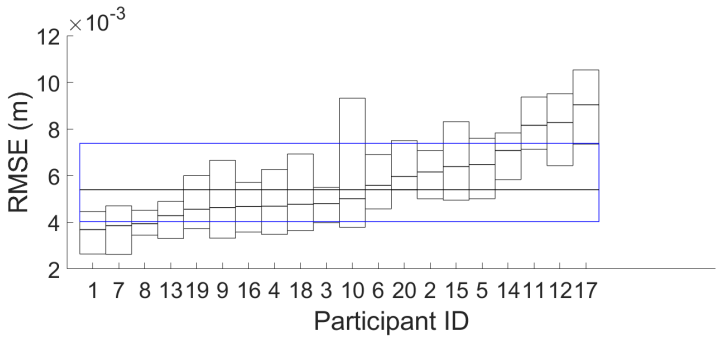


Figure 22: Median and interquartile range for the RMSE value for each participants and across all the participants

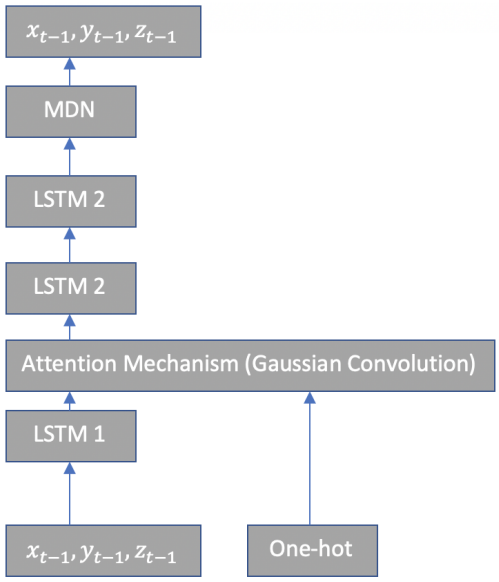


Figure 23: Model Overview, it shows the overall structure of the neural network model to generate synthetic trace data

performs a Gaussian convolution over a one-hot encoding of the input text using a mixture of Gaussians. Its final output is a soft window into the one-hot encoding of the character the model thinks it is drawing.

There are three parameters controlling the character window $w_i$ according to

$$(\hat{\alpha}_t, \hat{\beta}_t, \hat{\kappa}_t) = W_{h^1 p} h_t^1 + b_p$$

Each of these parameters are outputs from a dense layer on top of the first LSTM which we then transform according to:

$$\alpha_t = \exp(\hat{\alpha}_t)$$
$$\beta_t = \exp(\hat{\beta}_t)$$
$$\kappa_t = \kappa_{t-1} + \exp(\hat{\kappa}_t)$$

From these parameters we can construct the window as a convolution:

$$w_t = \sum_{u=1}^{U} \phi(t, u) c_u$$

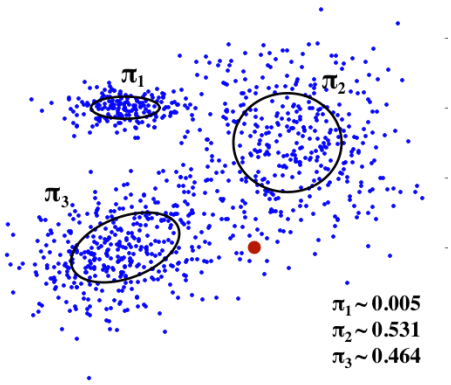


Figure 24: Mixture Density Network, the $\pi$ values in are likely values for the red dot. The probability that it came from the first distribution $\pi_1$ is really unlikely, but the probabilities are evenly balanced between the second two

where

$$\phi(t,u) = \sum_{k=1}^{K} \alpha_t^k \exp\left(-\beta_t^k(\kappa_t^k - u)^2\right)$$

An intuition is provided by Graves et al. [53] for the roles of $\alpha$, $\beta$, and $\kappa$ in his paper: "Intuitively, the $\kappa_t$ parameters control the location of the window, the $\beta_t$ parameters control the width of the window and the $\alpha_t$ parameters control the importance of the window within the mixture".

#### 6.2.4. Mixture Density Network (MDN)

The core idea is to have a network that predicts an entire distribution of the gesture trace-points. Here we are predicting a mixture of Gaussian distributions by estimating the means and covariances with the output from a dense neural network. In effect, the network will be able to estimate its own uncertainty. When the target is noisy, it will predict diffuse distributions, and where the target is really likely, it will predict a peaky distribution. A high-order (P order) Gaussian function is Equation 2. Here we use a 3D Gaussian distribution then P equals to 3.

$$f(x) = A \exp\left(-\left(\frac{(x - x_o)^2}{2\sigma_X^2}\right)^P\right) \tag{2}$$

If Figure 24 represents the target data space. The MDN will fit Gaussian distributions as shown. Since we are drawing from a mixture of Gaussians, we use the network to predict $\pi$, which defines how likely a given point was drawn from each Gaussian in the mixture.

Then we can transform the neural network outputs into parameters for the Gaussian Mixture Density network as shown below:

$$e_t = \frac{1}{1 + \exp(\hat{e}_t)} \qquad \pi_t^j = \frac{\exp(\hat{\pi}_t^j)}{\sum_{j'=1}^{M} \exp(\hat{\pi}_t^{j'})} \qquad \mu_t^j = \hat{\mu}_t^j \qquad \sigma_t^j = \exp(\hat{\sigma}_t^j) \qquad \rho_t^j = \tanh(\hat{\rho}_t^j)$$

### 6.3. Optimisation Methods

Optimisation is a very important concept in training the deep neural network. Training a neural network means optimising the cost function using gradient descent as input data flows through the network. The gradient of the cost function with respect to the weight is computed using the chain rule. One important contribution to deep learning is the separation of the optimisation problem into two pieces which is the back-propagation algorithm and the cost

function. Back-propagation algorithm is used to reduce the errors of the outputs by iteratively making changes to the weights such that Shallow Feed-forward Neural Networks (SFFNNs) can be automatically trained [32]. The cost function is the objective function that the model is trained to optimise. Often, there is no need to choose an optimisation algorithm, but it is necessary to set the right cost function in a deep learning problem.

### 6.3.1. Cost Function

The cost function is a function that measures the quality of our parameter set $\theta$. There are many different cost functions to choose from. Least squares error and cross-entropy are two of the most common training criteria. We define the cost function here in terms of the MDN parameters and the target data.

$$\mathcal{L}(x) = \sum_{t=1}^{T} -\log\left(\sum_{j} \pi_t^j \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j)\right) - \begin{cases} \log e_t & (x_{t+1})_3 = 1 \\ \log(1-e_t) & \text{otherwise} \end{cases}$$

### 6.3.2. Back-Propagation

Back-propagation, as mentioned earlier, is a successful learning algorithm for deep learning. A neural network will need to have a large number of units and weights such that a real-world task can be performed. This will make the global optimisation problem highly dimensional and intractable. Fortunately, the error between the desired outputs and the predictions is a locally smooth function of the weights, and the weight space contains many equivalent solutions. Therefore, a gradient-descent optimisation method is used to iteratively reduce the error by making small adjustments to the weights. The gradient is the derivative of the error with respect to the weight, and one way to compute the gradient is to propagate them backwards through the network. Thus the method is called back-propagation [54, 32]. Back-propagation through time is the application of the back-propagation training algorithm to recurrent neural network applied to sequence data like a time series.

### 6.3.3. Gradient Descent

Gradient descent is the most important technique and the foundation of how we train and optimise intelligent systems. Optimisation algorithms help us to minimise or maximise an objective function or sometimes referred to as error function which is a mathematical function dependent on the model's internal learnable parameters which are used in computing the target values from the set of predictors used in the model. It is one of the most popular optimisation algorithms used in optimising a neural network. We train a neural network via back-propagation in which we first propagate forward calculating the dot product of inputs signals and their corresponding weights and then apply an activation function to those sum of products. There are different variants of gradient descent, including the stochastic gradient descent and mini-batch gradient descent.

1. **SGD**: Gradient descent is a First Order Optimisation Method. It only takes the first-order derivative of the loss function into account and not the higher ones.
2. **Momentum**: This momentum is the same as the momentum in classical physics. Therefore, it leads to faster and stable convergence and also reduces oscillations. The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. This means it does parameter updates only for relevant examples.
3. **RMSProp**: RMSprop or Root Mean Square Propagation tries to dampen the oscillations, but in a different way than momentum. RMSprop also takes away the need to adjust the learning rate and does it automatically. Moreover, RMSProp chooses a different learning rate for each parameter.

4. **Adam**: Adam stands for adaptive moment estimation. Adaptive moment estimation is another method that computes adaptive learning rates for each parameter. Adam combines the heuristics of both Momentum and RMSProp.

### 6.4. Training Techniques

A larger number of hidden layers complicate the training process of the network parameters. Therefore, there is a number of training techniques to train deep neural networks. During training, updates in weights of a previous layer change the distribution of input values for a current layer. This slows down learning. The normalisation of outputs of a previous layer could help to alleviate this. Dropout is another technique that randomly sets some nodes to zero with probability for each forward pass. This is to prevent the network from over-fitting and from adapting to co-occurring features. It also introduces redundancy in representations such as multiple neurons performing similar tasks and increases robustness to missing feature responses or errors. For a large dataset, if the gradient is computed over all samples, the computational process can be very slow.

#### 6.4.1. Dropout

Large neural nets trained on relatively small datasets can overfit the training data. This has the effect of the model learning the statistical noise in the training data, which results in poor performance when the model is evaluated on new data, e.g. a test dataset. Generalisation error increases due to over-fitting. One approach to reduce over-fitting is to fit all possible different neural networks on the same dataset and to average the predictions from each model. However, this is not feasible in practice. Dropout is another method to solve over-fitting. It is a regularisation method that approximates training a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or 'dropped out'. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs.

#### 6.4.2. Batch Training

Training a neural network is the process of finding a set of weights and bias values so that computed outputs closely match the known outputs for a collection of training data items. Once a set of good weights and bias values have been found, the resulting neural network model can make predictions on new data with unknown output values. There are two general approaches for neural network training, usually called 'batch' and 'online'. The approaches are similar but can produce very different results. We used batch training in our project.

#### 6.4.3. Padding

The input vectors for the traces have different lengths. Padding is to add extra length to make the input vector in the same batch with the same length.

#### 6.4.4. Gradient Clip

Training a neural network can become unstable given the choice of the error function, learning rate, or even the scale of the target variable. Large updates to weights during training can cause a numerical overflow or underflow often referred to as 'exploding gradients'. The problem of exploding gradients is more common with recurrent neural networks, such as LSTMs in our project given the accumulation of gradients unrolled over hundreds of input time steps. A common and relatively easy solution to the exploding gradients problem is to change the derivative of the error before propagating it backward through the network and using it to update the weights. Two approaches include rescaling the gradients given a chosen vector norm and clipping gradient values that exceed a preferred range. Together, these methods are referred to as 'gradient clipping'. Therefore, it is essential to use gradient clip in our network.

### 6.4.5. Learning Rate Schedules

Choosing a proper learning rate can be difficult. With a high learning rate, the deep learning model would possess high kinetic energy. As a result, its parameter vector bounces around chaotically. Thus, it's unable to settle down into deeper and narrower parts of the loss function (local minima). If the learning rate, on the other hand, was very low, the system then would have low kinetic energy. Thus, it would settle down into shallow and narrower parts of the loss function (false minima). Therefore, in training deep neural networks, it is helpful to reduce the learning rate as the number of training epochs increases.

## 6.5. Data Synthesis Results

Artificial gesture traces are generated from the trained model. However, the generated trace is not representative of the true trace data. Figure 25 shows the time series plot of the soft window's position with different horizontal axis.

The soft attention window is the time series of one-hot encodings produced by the attention mechanism. The vertical axis is time in descending order. The horizontal axis in the left plot is the sequence of ASCII characters. The horizontal axis in the right plot is what the model sees when it looks through the soft window. The first one-hot vector is _ and the next one-hot vector is $a$ and so on as shown on the upper horizontal axis in Figure 27.

It can be clearly seen that the time series of the attention window has a large range and different characters seem to have similar time series but with a fixed displacement between each character. The right plot in Figure 25 shows a large soft attention window in _ and _ is in the phrase 'this should be okay'. Figure 26 and 27 shows the same plot for the collected real gesture trace for the phrase 'You Know Nothing Jon Snow'. Each character shows a characteristic time series in Figure 26 and 27. Therefore, the synthetic gesture trace is not representative of the real gesture trace.

However, this result was expected since training a deep neural network requires tuning the parameters and updating the training techniques for different iterations. It also required over 3 days to complete a full training process. Therefore, the iteration can be slow, especially under the current situation where I do not have convenient access to my desktop where I run the training on.

There are several things that I will do to improve the performance of this neural network. First, a network with a deeper structure can be used. I can put one or two extra LSTM layer in the current network. This is because a deeper network may have a better performance in finding representative distribution. However, a complicated network can potentially lead to an over-fitting problem. But again, this can be solved by dropout as discussed before. Second, I will tune the parameters of the network such as the batch size, initial learning rate. This is a necessary step in deep learning, and it is common that the researcher needs to tune parameters many times for the result to converge. Third, LSTM has different forms. I will look into the characteristics of the different LSTM cells and compare the performances of them.

# 7. Discussion

## 7.1. Limitations

This ARGK has its limitations. First, the ARGK requires sufficient space in front of the user. Users in a crowded place where they can not reach out their hand may not be able to use this system due to physical constraints.

Second, similar to the word-gesture keyboard in mobile phones, this ARGK is unable to output or predict unseen words or phrases such as a password.

Finally, we have used OptiTrack to track the fingertip of the user since the hand tracking module in the HoloLens 1 has a large latency and offset. However, the optical-see-through

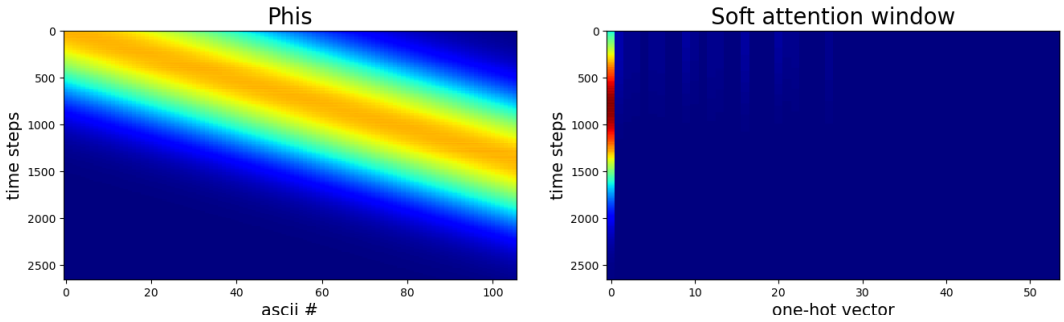


Figure 25: Plots for the phrase 'this should be okay'. Plots on the right and left are the time series plots of the soft window's position from the generated trace correspond to the phrase. However, they have different horizontal axes. The horizontal axis in the right plot is what the model sees when it looks through the soft window, and the horizontal axis in the left plot is the sequence of ASCII characters that the model is drawing. The colour represents a heat map that shows the magnitude of the time series.

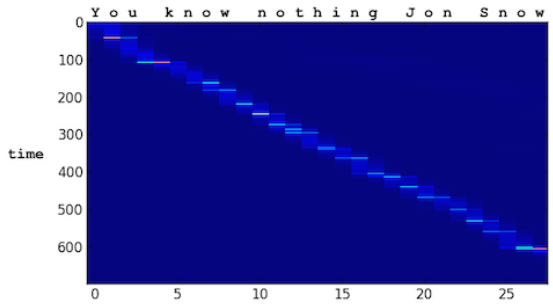


Figure 26: Phis plot for the trace with 'You Know Nothing Jon Snow'. The vertical axis is time in descending order. The horizontal axis is the sequence of ASCII characters. Each character has its own time series with different range and magnitude.

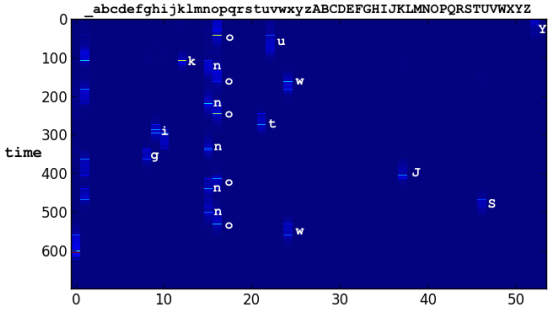


Figure 27: Soft attention window. The vertical axis is time (descending). The horizontal axis is the one-hot vectors. The first one-hot vector is _ and the next one-hot vector is *a* and so on as shown on the upper horizontal axis.

HMD in the future may have an equivalent tracking performance to OptiTrack. Therefore, a full ARGK with the machine learning-based recogniser has to be implemented in an optical-see-through HMD with a high-quality sensor in the future.

### 7.2. Future Work

First of all, synthetic data is required to tackle the difficulty of data sparseness. We have already constructed the neural network to generate synthetic trace data. I am currently working on training the neural network to achieve acceptable results. There are many difficulties in training the neural network. Even though the network constructed only has three layers of LSTM, it is still a deep neural network. Training the neural network still requires a large amount of data. We currently have over 1000 traces. This amount of data may be enough to train a three-layer neural network. However, the possibility that the data is not enough still exists. Furthermore, training this neural network requires about 3 to 4 days since I only have one Nvidia 2080 Ti which is one of the most powerful GPUs. That means the iteration of

the training process of the deep neural network can be slow. Furthermore, training the neural network requires tuning parameters and using different training techniques.

Therefore, future work includes iterating the neural networks and tuning the parameters until high-quality trace data is generated.

Then a quality check needs to be conducted to see whether the synthetic data is representative of the real data. Different methods can be used, including calculating the divergence between the synthetic data and the real data, using a classifier and comparing the micro metrics. Micro metrics can imply user behaviour. If all the micro metrics obtained from the synthetic data is similar to the micro metrics obtained from the natural data, then it suggests that the result is representative of the user behaviour.

Secondly, another neural network needs to be constructed and trained to map the combined data set consisting of both synthetic and real data to the corresponding phrases. The neural network will be constructed of LSTMs and attention mechanism.

Thirdly, the trained neural network model will be implemented into the ARGK as a recogniser to replace the simulated recogniser.

Finally, another user experiment will be carried out to assess the full ARGK with this machine learning-based recogniser. Then micro metrics will be used again to analyse this system, and they will be compared with the previously obtained micro metrics from the ARGK with the simulated recogniser.

## 8. Conclusions

AR technology augments the real world by superimposing virtual objects and cues upon the real world in real-time. AR brings virtual information not only to the user's immediate surroundings but also to any indirect view of the real-world environment. AR enhances the user's perception of and interaction with the real world. There are many possibilities for using AR in an innovative way, such as how to enter text in AR. The proposed AR Gesture Keyboard (ARGK) provides an efficient and precise way of entering text in AR.

The simulated recogniser shows the potential of a machine learning-based recogniser. The micro metrics obtained from the user experiments are important in the quality check for the synthetic gesture data. Furthermore, they also gave us a deeper understanding of user behaviours while using the ARGK. Most importantly, the simulated recogniser enabled us to collect representative gesture traces from the user experiments. Therefore, we can use a deep learning model to generate a large amount of synthetic gesture traces which are representative of the true traces.

We still need to work on tuning the neural network to synthesise data, and apply the micro metrics and classifier to check the quality of the synthetic data. Overall, the machine learning-based ARGK shows great potential in fast and accurate text entry in AR.

## Appendix A. Risk assessment

The risk assessment form I submitted to the Safety Office at the start of the Michaelmas Term reflects the hazards actually encountered during the course of the project reasonably well. There are mainly two places where I work in: in the office where I work with my computer, and in the lab where I used OptiTrack to do the user experiments. There is almost no risk related to computer work. The only risk may come from the lab where the OptiTrack system is. The risk can be tracking cameras failing off from the top. However, we double-checked whether the cameras were attached tight on the stand to minimise that risk everytime we started the user experiment. If starting the project again, I may add this point into the risk assessment form. The reason I didn't add this point to the risk assessment form is that we didn't know that we were going to use the OptiTrack system.

## Appendix B. Covid-19 disruption

Due to the breakout of COVID-19, I had to fly back to China in mid-March. However, I was unable to take my desktop, which is the computer that I train my neural network on. Therefore, It took me over a week to set up remote SSH login to that remote desktop in the UK. However, the connection is unstable, and the internet speed is slow. When the computer system crashed due to software reasons, I will need to find someone to go to that desktop and reboot it. This has caused a lot of inconvenience to my project since I will require access to that desktop constantly to train my neural network. Thus I have spent more time on the literature review of the artificial neural networks. Furthermore, Google is blocked in China, and sometimes Cambridge VPN does not work. Thus many research papers are inaccessible. Therefore, the breakout of COVID-19 has slowed down my overall progress to train the recurrent neural network and write the literature review by over 40%.

### *References*


*[1] P. O. Kristensson, Discrete and continuous shape writing for text entry and control, Ph.D. thesis (2007).*

*[2] H. Benko, E. W. Ishak, S. Feiner, Collaborative mixed reality visualization of an archaeological excavation, in: Third IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE, 2004, pp. 132–140.*

*[3] X. Wang, Augmented reality in architecture and design: potentials and challenges for application, International Journal of Architectural Computing 7 (2) (2009) 309–326.*

*[4] H. Kaufmann, Geometry education with augmented reality, 2004.*

*[5] H.-K. Wu, S. W.-Y. Lee, H.-Y. Chang, J.-C. Liang, Current status, opportunities and challenges of augmented reality in education, Computers & education 62 (2013) 41–49.*

*[6] A. Boccaccio, G. Cascella, M. Fiorentino, M. Gattullo, V. Manghisi, G. Monno, A. Uva, Exploiting augmented reality to display technical information on industry 4.0 p&id, in: Advances on Mechanics, Design Engineering and Manufacturing II, Springer, 2019, pp. 282–291.*

*[7] J. J. Dudley, K. Vertanen, P. O. Kristensson, Fast and precise touch-based text entry for head-mounted augmented reality with variable occlusion, ACM Trans. Comput.-Hum. Interact. 25 (6).*

*[8] S. Zhai, P. Kristensson, The word-gesture keyboard: Reimagining keyboard interaction, Communications of The ACM - CACM 55.*

*[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Processing Magazine 29 (6) (2012) 82–97.*

*[10] J. Ha, K. Cho, F. A. Rojas, H. S. Yang, Real-time scalable recognition and tracking based on the server-client model for mobile augmented reality, in: 2011 IEEE International Symposium on VR Innovation, IEEE, 2011, pp. 267–272.*

*[11] J. P. Rolland, R. L. Holloway, H. Fuchs, Comparison of optical and video see-through, head-mounted displays, in: Telemanipulator and Telepresence Technologies, Vol. 2351, International Society for Optics and Photonics, 1995, pp. 293–307.*

*[12] A. Takagi, S. Yamazaki, Y. Saito, N. Taniguchi, Development of a stereo video see-through hmd for ar systems, in: Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000), IEEE, 2000, pp. 68–77.*

[13] *R. Azuma, G. Bishop, Improving static and dynamic registration in an optical see-through hmd, in: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, 1994, pp. 197–204.*

[14] *S. Liu, H. Hua, D. Cheng, A novel prototype for an optical see-through head-mounted display with addressable focus cues, IEEE transactions on visualization and computer graphics 16 (3) (2009) 381–393.*

[15] *S. Mehra, P. Werkhoven, M. Worring, Navigating on handheld displays: Dynamic versus static peephole navigation, ACM Transactions on Computer-Human Interaction (TOCHI) 13 (4) (2006) 448–457.*

[16] *D. Wagner, D. Schmalstieg, Handheld augmented reality displays, in: IEEE Virtual Reality Conference (VR 2006), IEEE, 2006, pp. 321–321.*

[17] *O. Bimber, R. Raskar, Spatial augmented reality: merging real and virtual worlds, CRC press, 2005.*

[18] *C. Y. Wang, W. C. Chu, P. T. Chiu, M. C. Hsiu, Y. H. Chiang, M. Y. Chen, Palmtype: Using palms as keyboards for smart glasses, in: Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services, Association for Computing Machinery, 2015, p. 153–160.*

[19] *A. Feit, S. Sridhar, C. Theboalt, A. Oulasvirta, Investigating the dexterity of multi-finger input for mid-air text entry, 2015.*

[20] *T. Grossman, X. A. Chen, G. Fitzmaurice, Typing on glasses: Adapting text entry to smart eyewear, in: Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services, Association for Computing Machinery, 2015, p. 144–152.*

[21] *P. Isokoski, R. Raisamo, Device independent text input: A rationale and an example, in: Proceedings of the Working Conference on Advanced Visual Interfaces, Association for Computing Machinery, 2000, p. 76–83.*

[22] *S. J. Castellucci, I. S. MacKenzie, Graffiti vs. unistrokes: An empirical comparison, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, 2008, p. 305–308.*

[23] *J. O. Wobbrock, B. A. Myers, J. A. Kembel, Edgewrite: A stylus-based text entry method designed for high accuracy and stability of motion, in: Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology, Association for Computing Machinery, 2003, p. 61–70.*

[24] *S. Zhai, P. O. Kristensson, The word-gesture keyboard: Reimagining keyboard interaction, Commun. ACM 55 (9) (2012) 91–101.*

[25] *P. O. Kristensson, S. Zhai, Shark2:a large vocabulary shorthand writing system for pen-based computers, Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology.*

[26] *S. Zhai, P.-O. Kristensson, Shorthand writing on stylus keyboard, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '03, Association for Computing Machinery, New York, NY, USA, 2003, p. 97–104.*

[27] *I. S. MacKenzie, R. W. Soukoreff, J. Helga, 1 thumb, 4 buttons, 20 words per minute: Design and evaluation of h4-writer, in: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Association for Computing Machinery, 2011, p. 471–480.*

[28] K. Perlin, Quikwriting: continuous stylus-based text entry, in: Proceedings of the 11th annual ACM symposium on User interface software and technology, 1998, pp. 215–216.

[29] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics 5 (4) (1943) 115–133.

[30] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychological review 65 (6) (1958) 386.

[31] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biological cybernetics 36 (4) (1980) 193–202.

[32] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536.

[33] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[35] D. L. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, J. J. DiCarlo, Performance-optimized hierarchical models predict neural responses in higher visual cortex, Proceedings of the National Academy of Sciences 111 (23) (2014) 8619–8624.

[36] H. Sak, A. W. Senior, F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

[37] A. M. Schäfer, H. G. Zimmermann, Recurrent neural networks are universal approximators, in: International Conference on Artificial Neural Networks, Springer, 2006, pp. 632–640.

[38] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems 2 (4) (1989) 303–314.

[39] M. Leshno, V. Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural networks 6 (6) (1993) 861–867.

[40] Y. Bengio, et al., Learning deep architectures for ai, Foundations and trends® in Machine Learning 2 (1) (2009) 1–127.

[41] S. El Hihi, Y. Bengio, Hierarchical recurrent neural networks for long-term dependencies, in: Advances in neural information processing systems, 1996, pp. 493–499.

[42] J. Schmidhuber, Learning complex, extended sequences using the principle of history compression, Neural Computation 4 (2) (1992) 234–242.

[43] R. Pascanu, C. Gulcehre, K. Cho, Y. Bengio, How to construct deep recurrent neural networks, arXiv preprint arXiv:1312.6026.

[44] O. Alsharif, T. Ouyang, F. Beaufays, S. Zhai, T. Breuel, J. Schalkwyk, Long short term memory neural network for keyboard gesture decoding, 2015, pp. 2076–2080.

[45] B. Klimt, Y. Yang, The enron corpus: A new dataset for email classification research, in: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 217–226.

*[46] I. S. MacKenzie, R. W. Soukoreff, Phrase sets for evaluating text entry techniques, in: CHI '03 Extended Abstracts on Human Factors in Computing Systems, CHI EA '03, Association for Computing Machinery, New York, NY, USA, 2003, p. 754–755.*

*[47] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Networks, arXiv e-prints.*

*[48] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, WaveNet: A Generative Model for Raw Audio, arXiv e-prints.*

*[49] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (2008) 2579–2605.*

*[50] A. Graves, Supervised Sequence Labelling with Recurrent Neural Networks, Vol. 385, 2012.*

*[51] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.*

*[52] O. Alsharif, T. Ouyang, F. Beaufays, S. Zhai, T. Breuel, J. Schalkwyk, Long short term memory neural network for keyboard gesture decoding, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 2076–2080.*

*[53] A. Graves, Generating sequences with recurrent neural networks (2013).*

*[54] P. J. Werbos, Backpropagation through time: what it does and how to do it, Proceedings of the IEEE 78 (10) (1990) 1550–1560.*