Multi-view hand-tracking and gesture detection for VR

Quincy Sproul
Supervised by Dr H. Philamore

July 21, 2024

Contents

1 Introduction

Virtual Reality (VR) has emerged as a transformative technology, reshaping the way we interact with digital environments. By immersing users in simulated worlds, VR offers exciting possibilities for engaging experiences across various domains, including gaming, education, training, and healthcare. As the field of VR continues to mature, the importance of seamless and intuitive user interactions becomes increasingly evident. Hand tracking and gesture recognition stand at the forefront of this endeavor, enabling users to navigate and manipulate virtual environments naturally.

The integration of hand tracking and gesture recognition in VR systems has the potential to greatly enhance user experience. By allowing users to interact with virtual objects using their hands, just as they would in the real world, VR applications could become more intuitive and engaging. However, despite significant advancements in VR technology, current approaches to gesture recognition still face several challenges that hinder their effectiveness and usability.

One of the primary limitations of traditional gesture recognition methods is their reliance on observing complete gesture sequences before making a classification. This introduces inherent latency, as the system must wait for the entire gesture to be performed before determining its meaning. In VR applications where real-time responsiveness is crucial, such as gaming or interactive simulations, this latency can be disruptive.

Moreover, the accuracy of gesture recognition in VR is often compromised by various factors. Occlusion, where parts of the hand are obscured from the tracking system's view, poses a significant challenge. Inconsistent hand shapes and variations in how individuals perform gestures can further hinder reliable recognition.

To address these challenges and improve the user experience in VR, this research explores the application of Transformer models for multi-view hand tracking and early classification of dynamic hand gestures. Transformer architectures, which have demonstrated remarkable success in natural language processing and computer vision tasks, possess unique properties that make them potentially well-suited for this problem domain. By leveraging Transformers' ability to model long-range dependencies and capture temporal patterns, this research aims to investigate whether accurate prediction of future frames of hand gestures can be achieved based on limited initial observations.

The proposed methodology involves encoding hand features into discrete states that capture essential information such as moving direction, palm orientation, and hand pose. By representing hand gestures as a sequence of these discrete states, the Transformer models are trained to predict future frames and enable earlier classification of dynamic gestures. This predictive capability could potentially reduce the perceived latency in gesture recognition, allowing for

more responsive and interactive VR experiences.

To thoroughly investigate the potential of Transformer models in this context, the research will explore two distinct model configurations: a multi-step prediction model and a single-step prediction model. This comparison will help evaluate the trade-offs between prediction accuracy, computational complexity, and the ability to provide early classification of gestures.

The evaluation of the proposed approach will involve a comprehensive assessment of prediction accuracy, early classification performance, and reliability. By measuring the accuracy of predicted future frames against ground truth data, the research will quantify the effectiveness of the Transformer models in capturing the temporal dynamics of hand gestures. Additionally, the study will investigate the potential impact of early classification on user experience in VR applications.

This research aims to contribute to the field of hand tracking and gesture recognition in VR systems. The findings of this study may illuminate potential gaps, pave the way for future research, and offer insights into optimising user experience within VR environments. Additionally, the research may have broader implications for other areas where real-time prediction and classification of sequential data are crucial, such as robotics, autonomous systems, and human-computer interaction.

In summary, this research aims to address the challenges of latency and accuracy in gesture recognition for VR environments by investigating the potential of Transformer models. Through multi-view hand tracking and early classification of dynamic hand gestures, the study seeks to explore possibilities for improved user experiences within interactive and engaging virtual environments.

2 Problem Statement

VR has become increasingly popular in recent years, offering immersive experiences across various domains such as gaming, education, and training. Hand tracking and gesture recognition play crucial roles in enhancing user interaction and experience within VR environments. However, current methods for gesture recognition in VR often face challenges, including:

- **Response Time:** Traditional approaches that rely on observing complete gesture sequences can introduce delays, potentially affecting the responsiveness of interactions within VR.
- **Accuracy:** Factors like occlusion, gesture complexity, and variations in hand shape can compromise recognition accuracy.
- **Data Representation:** Finding an optimal encoding for hand gestures to balance accurate prediction with model efficiency remains an area of exploration.

To address these challenges, this research investigates the use of Transformer models for multi-view hand tracking and early classification of dynamic hand gestures in VR. The proposed approach focuses on the following:

- **Discrete Encoding:** Hand features, such as moving direction, palm orientation, and hand pose, are represented by discrete states. This facilitates time-series prediction but requires careful consideration of the trade-offs between representation detail and model complexity.
- **Future Frame Prediction:** Transformer models are trained to predict future hand gesture states, enabling earlier classification compared to methods that rely on full sequences.
- **Model Comparison:** The research evaluates the effectiveness of multi-step prediction Transformers (predicting multiple frames simultaneously) against single-step prediction Transformers (iterative prediction of a single frame at a time).

Research Objectives:

1. **Dynamic Hand Feature Extraction and Encoding:** Develop a system to accurately capture and encode dynamic hand features from depth camera data into discrete states.
2. **Predictive Power of Transformers:** Investigate Transformer models for future frame prediction based on encoded hand features. Thoroughly compare multi-step vs. single-step prediction performance.
3. **Early Classification Potential:** Evaluate the potential of the proposed approach for early classification and its implications for the responsiveness and accuracy of VR gesture recognition.

3 Literature Review

3.1 Background & Motivation

Hand tracking and gesture recognition are critical to enhancing user interactions within these environments [1]. However, current methods often face challenges in real-time performance, exhibiting high latency and inaccurate classification of dynamic hand gestures. This limitation is reflected in studies where, when comparing hand tracking to handheld-controllers, "...hand-tracking [...] took longer to complete the same task and reported higher perceived mental workload scores" [2].

Deep learning techniques offer potential solutions. Convolutional neural networks (CNNs) are well-suited for image processing tasks, able to extract relevant features from depth and RGB hand images [3]. These features include edges, corners, and other texture information that

are crucial for recognising hand shapes and movements. However, CNNs often struggle with capturing temporal dependencies in sequential data, which is where Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks come in. RNNs are designed to recognise patterns in sequential data like text or speech by utilising an internal memory state. LSTM networks are capable of learning long-range dependencies by regulating memory through gate structures. Transformer models, initially developed for natural language processing, use self-attention mechanisms to weigh the importance of words in a sequence when encoding context.

According to a study [4], a proposed multi-head attention-based transformer model outperformed LSTM and GRU (a type of RNN) baselines for traffic flow forecasting. Specifically, the transformer increased mean absolute percentage error by 32.4-33.9% compared to LSTM/GRU, excelling at capturing long-range time series dependencies and enabling better long-horizon forecasting than prior BERT models. However, substantial data was required for state-of-the-art performance, and careful input window selection was crucial. Overall, the transformer showed promising results by effectively leveraging long-range information with sufficient data and tuning.

Recently, transformer-based models have excelled in various sequence modeling tasks. Their established superiority in accuracy and efficiency compared to other deep learning methods, particularly when dealing with noisy data [5], motivates their focused exploration within this research. Unlike RNNs and LSTMs, Transformers do not require sequential data processing, which allows for more parallelisation and faster training times. Moreover, previous studies [6] suggest that Transformers possess greater robustness to noise compared to RNNs; for instance, "Experimental results show that, compared to recurrent neural models, self-attentive models are more robust against adversarial perturbation." In the context of the paper, "Adversarial attack refers to applying a small perturbation on the model input to craft an adversarial example, ideally imperceptible by humans, and cause the model to make an incorrect prediction." So adversarial perturbation is a form of noise or corruption intentionally added to the input data in an attempt to fool the model, while still appearing natural. The paper examines the robustness of Transformer vs RNNs to this type of adversarial noise injection. This advantage is particularly relevant for dynamic hand gesture datasets, which are inherently noisy due to factors like sensor limitations, occlusions, and variations in hand posture between individuals. By focusing on Transformers, this research aims to leverage their superior noise handling capabilities to achieve more accurate and robust gesture recognition in VR environments.

Moreover, their self-attention mechanism, a key component of Transformer models, enables them to model long-range dependencies more effectively. The self-attention mechanism allows the model to weigh the importance of different inputs at each time step, focusing more on relevant inputs and less on irrelevant ones. This ability to 'attend' to different parts of the input sequence in a context-dependent manner makes it particularly effective for complex gesture recognition tasks. Despite progress, challenges remain. These include the need for early gesture recognition to enable responsive interactions, limitations posed by single-view hand tracking, the potential benefits of discrete hand gesture representations, and tailoring transformer models specifically for VR systems. Of particular interest is early gesture recognition. It is crucial for maintaining

the illusion of immersion in VR, as any delay in response can break the user's sense of presence. Similarly, single-view hand tracking can lead to occlusions and limited viewing angles, hindering accuracy. This research seeks to address the challenge of early recognition by investigating the ability of Transformer models to predict hand gesture sequences, enabling classification even with limited initial data.

3.2 Importance of Early Gesture Recognition in VR

Early gesture recognition in VR is essential for responsive, natural interactions. It allows the system to provide real-time feedback and reduces perceived latency [7,8]. However, early recognition is difficult due to potential ambiguity when observing only partial gestures, "...such as distinguishing genuine user intent from accidental actions amid environmental interference." [9]. This is because the initial stages of different gestures can often look similar, making it challenging for the model to make accurate predictions based on incomplete information.

Previous research explored early recognition techniques like probabilistic models for prediction [10] and reinforcement learning to optimise prediction-accuracy tradeoffs [11]. Probabilistic models, for instance, can handle the uncertainty inherent in early gesture recognition by predicting a distribution over possible gestures instead of a single gesture label. Reinforcement learning, on the other hand, can learn to make optimal predictions over time by receiving feedback on its performance and adjusting its predictions accordingly. Yet, the need remains for more efficient and robust methods for VR.

3.3 Sensors for Multi-View Hand Tracking in VR

Most VR hand tracking relies on a single viewpoint, often from an HMD-mounted camera. This approach is prone to occlusions and limited viewing angles, hindering accuracy. Multi-view hand tracking, using multiple cameras/sensors, offers the potential to improve robustness [12]. By capturing hand movements from different angles, multi-view tracking can provide a more comprehensive representation of the hand, which can lead to more accurate gesture recognition. Existing multi-view techniques include depth camera fusion [13], combining RGB and depth information, and wearable sensors [14]. Depth camera fusion, for instance, combines the depth maps from multiple cameras to create a 3D model of the hand, which can then be used for gesture recognition. RGB-D multi-view tracking, on the other hand, leverages both color and depth information to improve the robustness of hand tracking. Wearable sensors, such as gloves fitted with inertial measurement units (IMUs), can provide additional information about hand movements, such as acceleration and orientation, which can be useful for recognising more complex gestures.

3.4 Discrete Representation Learning for Hand Gestures

Discrete hand gesture representations, which could be understood as a sequence of symbols where each symbol represents a specific hand pose or movement, offer advantages over raw sensor data or continuous forms. They can boost efficiency by reducing data volume. Furthermore, studies demonstrate that discrete representations don't necessarily sacrifice accuracy, noting that "The conversion of continuous-valued sensor data to discrete representations often results in comparable activity recognition accuracy. . . '' [15]. Discrete forms can also enhance interpretability, aiding in debugging and understanding system behavior [16]. This kind of representation, being a more abstract and simplified version of the original data, can be easier to understand and analyse than raw sensor data or continuous representations.

Previous work explored learning discrete representations through various techniques. Clustering can group similar hand poses or movements, creating a set of discrete symbols for gesture representation [17]. Similarly, vector quantisation maps continuous data to a limited set of vectors, effectively discretising it [18]. Unsupervised methods like autoencoders can also learn lower-dimensional discrete representations from continuous hand data. However, there's still room to improve the expressiveness and discriminative power of these representations.

Case Study - GesturAR The GesturAR system [19] offers a practical example of the power of discrete hand gesture representations. To enhance real-time gesture recognition, GesturAR discretises dynamic gestures into sequences of states. Each state encapsulates hand pose, movement direction, and palm rotation. A Siamese neural network assists in this process by identifying similar hand poses across frames, enabling the combination of redundant information. This approach streamlines representation while preserving the essential characteristics of dynamic gestures, facilitating the design of responsive 'dynamic-provoking' interactions.

However, it's worth noting a limitation of the study: the system needs to 'see' the full gesture to recognise it. This necessitates the prediction of later time steps of the gesture for earlier detection, an area that warrants further improvement. This enhancement would allow the system to respond more quickly and accurately to user gestures, further enhancing its utility in real-time applications.

3.5 Transformer Models for Sequence Modeling

Transformer models, introduced by Vaswani et al. [20], have transformed sequence modeling. Their reliance on attention mechanisms allows efficient modeling of long-range dependencies. This is particularly useful for tasks like machine translation and language understanding, where the meaning of a word can depend on other words that are far away in the sequence. In the context of hand tracking and gesture recognition, this ability to model long-range dependencies could help the model capture complex hand movements that span several frames. This has led to breakthroughs in machine translation, language understanding, and even image captioning [21],

where models generate descriptions based on the spatial relationships within an image. For instance, a Transformer model trained on image captioning tasks can generate a description of an image that accurately reflects the objects in the image and their spatial relationships. This ability to understand and describe spatial relationships could be beneficial for hand tracking and gesture recognition in VR, where the position and orientation of the hand relative to other objects in the environment can be important for recognising certain gestures. Transformers' potential for modeling complex spatiotemporal dependencies makes them highly promising for hand tracking and gesture recognition in VR. However, the application of Transformer models in this domain is still in its early stages, and there are many challenges to overcome. For instance, the high computational cost of Transformer models can be a problem for real-time applications like VR, where low latency is crucial. Therefore, research on more efficient variants of Transformer models, or techniques for compressing Transformer models, could be an important direction for future work.

3.6 Potential Advancements and Research Gaps

Despite progress in DL-based hand tracking and gesture recognition, several gaps and opportunities remain:

- **Transformer-Specifics for VR:** Transformers in VR hand tracking/gesture recognition are still nascent. Exploring novel architectures and training methods tailored to this domain promises significant gains in accuracy and efficiency.
- **Early Gesture Recognition:** Ambiguity in initial gesture stages remains a challenge. This research hypothesises that Transformer models, with their ability to analyse temporal dependencies, have the potential to improve early gesture recognition in VR, leading to more responsive interactions.
- **Expressive Discrete Representations:** Improving the expressiveness and discriminative power of discrete gesture representations is crucial. Exploring Transformers or unsupervised learning for this purpose could enhance recognition systems.

Scope of Investigation: This research will therefore focus specifically on the use of Transformer models for early gesture recognition within a multi-view VR setup. The primary aim is to investigate how effectively Transformers can predict gesture sequences for early classification. Evaluation will center on the following:

- **Time Series Prediction:** Comparing the accuracy of single-step vs. multi-step Transformer predictions for varying output window and sequence lengths. This will shed light on the trade-offs between prediction horizon and accuracy.
- **Classification Accuracy and Confidence:** Evaluating the accuracy and confidence of gesture classification using a lookup table. Predictions generated for different output

windows will be mapped to gestures in the lookup table, and their accuracy compared against the true performed gesture sequence. This will assess whether reliable gesture classification is possible even with potentially less accurate predictions for longer output windows.

4 Methodology

4.1 Dataset Selection

The selection of an appropriate dataset is crucial for the development and evaluation of gesture recognition models. Based on the knowledge gained from the literature review, the project's focus was narrowed down to improving gesture interaction within a single-camera system. This decision was influenced by the prior work of Dr. Philamore with EmpressVR [22], which involved controlling small robots using hand gestures captured from an external camera. Thus, the decision was made to investigate whether similar results could be achieved, and potentially improved upon, with a simpler setup using only an egocentric head-mounted camera. This could potentially reduce the complexity and number of separate devices needed within the system.

To identify suitable datasets for the project, various datasets containing data captured using either a single "egocentric" camera (worn by the user) or a single "external" camera were considered. The datasets were evaluated based on their inclusion of essential features such as 2D and 3D hand joint positions, depth maps, RGB images and the type of hand gesture (e.g. 'dynamic' or 'static'). Table 8 provides a comparison of the datasets reviewed.

Among the seven datasets reviewed, the DHG2016 dataset [23] emerged as the most suitable choice. Several factors contributed to this decision:

- **Manageable Size**: With a size of 6GB, the DHG dataset allows for convenient experimentation and faster model training compared to larger datasets.
- **Dynamic Gestures**: An aim of this project was to improve gesture identification, which often involves fluid hand movements. Static gesture datasets wouldn't be as useful for training a model that needs to recognise these dynamic movements.
- **Essential Features**: DHG includes 2D and 3D hand joint positions and depth maps, which are crucial for calculating the three main features of interest: hand pose, palm movement direction, and palm orientation.
- **Focus on Depth Maps**: The lack of RGB images in DHG was considered a benefit. For a low-latency real-time system, using a stereo camera to generate depth maps directly is preferable to computationally expensive conversions from RGB images.

The selection of the DHG dataset marked a significant milestone in the project. This choice paved the way for the next phases, which involve data analysis and feature extraction from

the chosen dataset. The following sections will provide a detailed description of the DHG2016 dataset (Section 4.1.1) and the data preprocessing steps (Section 4.1.2) required for feature extraction and model development.

4.1.1 DHG2016 Dataset

Dataset Description The Dynamic Hand Gesture 14/28 (DHG2016) dataset [23] contains video sequences of 14 dynamic hand gestures performed by 20 participants. Each gesture was performed 5 times by each participant using either a single finger or the whole hand, resulting in 2800 total sequences. The gestures were captured using an Intel RealSense depth camera at 30 frames per second with 640x480 resolution.

Data Characteristics The key characteristics of the DHG2016 dataset are:

- 14 distinct dynamic hand gestures
- 20 participants (all right-handed)
- 2800 total gesture sequences
- 5 trials per gesture per participant (200 trials per gesture)
- Depth images, 2D and 3D hand joint positions for each frame
- 30 frames per second
- 640x480 depth image resolution
- Gesture sequence lengths range from 20-50 frames
- Initial frames contain open hand with palm facing towards the camera

Example Gestures The 14 hand gestures in the dataset are listed in Table 9. The gestures involve various hand motions like swiping, grabbing, tapping and drawing shapes, an example of a 'swipe right' gesture can be seen in Figure 7.

4.1.2 Data Preprocessing

Outlier Filtering: Filtering the dataset was deemed necessary to ensure the accuracy and reliability of the calculated features. These features are derived from the spatial relationships and orientations of the hand landmarks across consecutive frames. The presence of noisy data points can introduce substantial errors in the feature calculations, potentially leading to incorrect interpretations of the performed gestures. By employing a filtering algorithm that removes frames with abrupt and significant deviations from the average centroid movement, the impact

of noisy data was removed, obtaining more precise and consistent feature values. Moreover, the filtering process does not compromise the overall data representation, as the gestures are captured by a sequence of frames, and the surrounding frames can effectively convey the essential characteristics of the gesture even if a few noisy frames are removed. This approach strikes a balance between data quality and information preservation, ensuring that the filtered dataset remains representative of the original gestures while minimising the influence of noisy data points.

The algorithm:

1. Takes a list of hand landmark frames as input.
2. Calculates the centroid of the hand landmarks for each frame and then calculates the differences between consecutive centroids.
3. Finds the Euclidean distance for each differences as well as the mean and standard deviation of these distances.
4. A threshold value is defined as 2 standard deviations from the mean distance.
5. Using the threshold, it collects the indices of the frames that have a distance below the threshold, and finally returns the filtered hand landmark frames.

The effect of filtering the dataset on the centroids is visualised in Figure 8. By filtering the dataset, it was reduced from an approximate size of 6 GB to 5 GB.

4.2 Data Encoding and Feature Representation

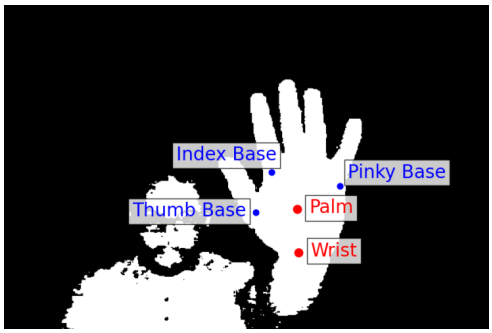
A crucial design decision in gesture recognition systems is the choice of data representation, as it profoundly influences both the feasibility of time series modeling and the potential accuracy for classification tasks. In this work, a discrete encoding scheme is adopted to prioritise time series predictability, with the conscious trade-off of potentially limiting generalisation for fine-grained gesture classification.

4.2.1 Feature Discretisation

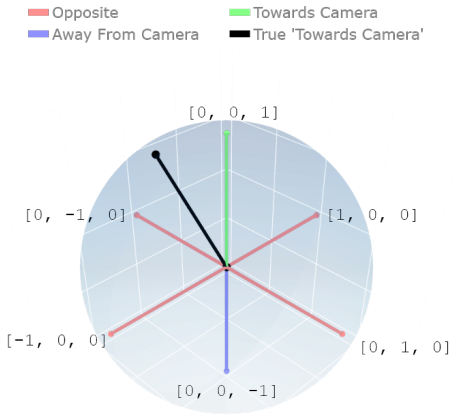
Raw, continuous hand skeleton data is transformed into a set of discrete states for the following features:

Palm Orientation: Initially, surface normal vectors between hand landmark indexes and fixed unit vectors were used for orientation mapping, resulting in inaccuracies. The number of orientations was reduced from six to three to mitigate the effect of mislabelled data. However, the main issue was that the unit vectors did not accurately describe the ground truth of the video environment. As the dataset authors stated, the first few frames of every video contained a flat

palm facing the camera (see Section 4.1.1). To address this, the average palm coordinate from the first five frames was extracted and defined as the 'Towards Camera' unit vector for each trial, resulting in a unique facing forward unit vector for each sequence as shown in Figure 1. The opposite of this vector represented the facing away from camera orientation. This approach resulted in three primary orientations: "towards camera", "away from camera", and "opposite" (encompassing all other orientations: "left", "right", "up", "down").



(a) Depth Image with Anatomical Labeling: *The palm and wrist joint are marked in red, while the thumb base, index base, and pinky base are highlighted in blue.*



(b) Orientation Unit Sphere: *Depicts 3 potential orientations of the palm, with True 'Towards Camera' representing the normal vector of the palm coordinate shown in Figure 1a.*

Figure 1: Analysis of Hand Orientation and Landmarks

Moving Direction: The initial approach of calculating difference vectors between consecutive frames to determine hand motion direction was susceptible to noise, resulting in inaccurate labels. For example, the slightest hand movement caused an inaccurate label, resulting in a sequence of moving directions that looked something like the sequence in Figure 2a, when in reality the true motion of the dynamic gesture was a lot smoother.

To mitigate this problem, a sliding window approach was employed. First, the difference vectors between each frame were calculated, and every coordinate difference for each frame was stored. Then, a sliding window was used to work through chunks of the per-frame coordinates. For each window, the average rate of change for each coordinate was calculated by summing the differences and dividing by the window size. The coordinate with the highest average rate of change over the sliding window was identified as the dominant direction of motion for that window. However, to ensure that the chosen direction was consistent and followed the expected pattern of a fluid motion, an additional criterion was applied. The algorithm checked if the coordinate with the highest average rate of change also followed the same direction (increasing or decreasing) throughout the window and appeared to increase to a maximum and decrease as expected for a fluid motion. This approach helped to filter out noise and sudden changes in direction as can be seen in Figure 2b, providing a more robust encoding of hand motion direction.

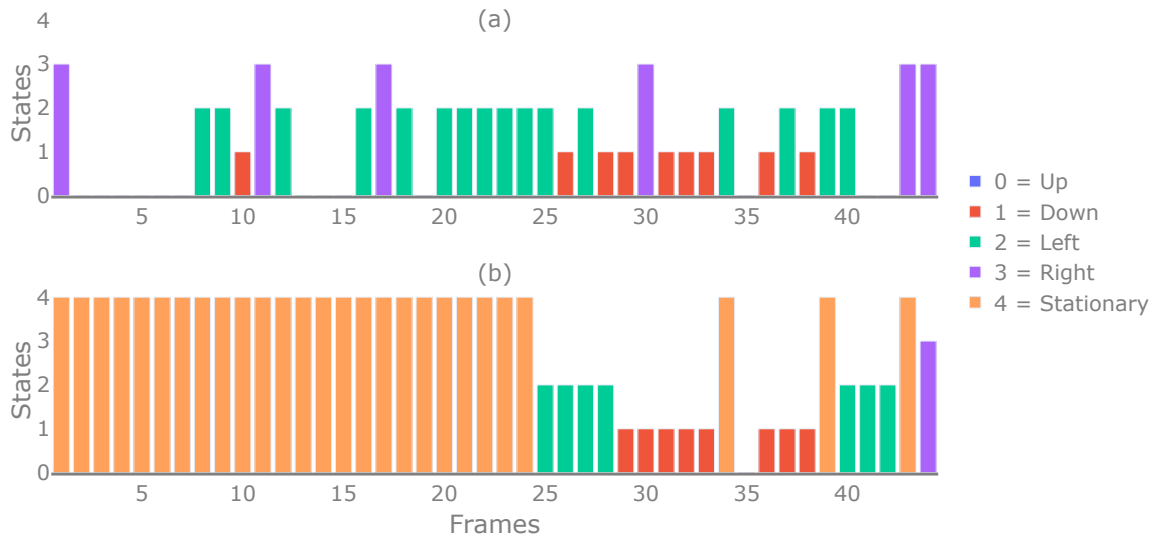


Figure 2: Before vs After sliding window applied.
a) Moving Direction Gesture Sequence, where each state is categorised by the coordinate with greatest change between consecutive frames' 3D coordinates. b) Moving Direction Gesture Sequence after applying the sliding window approach, each state is categorised by the coordinate with largest rate of change in each window.

This sliding window approach with consistency checks as well as reducing the dimension of motion (e.g. z coordinate) helped to mitigate the problem of feature imbalance when combining the three features into unique states. Previously, the top ten most frequently occurring states contained 'stationary' as the moving direction, with the top two states accounting for over 50% of all states (Figure 9a). This caused model accuracy limitations, not able to pass 60% during training, as a direct result of the model overfitting to the state with highest proportion, causing a misleading performance. By using the sliding window approach, the feature imbalance was reduced (Figure 9b), and the model was able to learn more meaningful patterns in the motion of dynamic gestures.

Hand Pose: Hand pose representation heavily relies on hand joint bending angles, which are calculated using the relative 3D positions of adjacent hand landmarks (see Figure 11). Experimentation with clustering techniques for pose identification was conducted. However, a more intuitive and effective method was chosen. This method involves defining a threshold for natural changes in hand pose based on differences in joint bending angles between consecutive frames. Through trial and error, it was determined that a good threshold value would cause approximately 10% of frames to have state changes or dissimilar hand poses, which seemed reasonable given the average frame count of around 50 frames per video.

The process for determining hand pose similarity is as follows:

1. Calculate the difference between each pair of consecutive frames of hand joint bending angles.

2. Determine a threshold for similarity based on the interquartile range (IQR) of the absolute

differences between all pairs of consecutive frames, using the 5% and 95% percentile range (Q1 and Q3, respectively).

3. Determine if each pair of consecutive frames is similar by checking if the absolute difference between them is below the threshold.

After labelling all the data, a Siamese network model was trained on these pairs of similar and dissimilar bending angles. The Siamese network model was chosen for its ability to generalise across new unseen examples more effectively. However, ultimately, the Siamese network labelling had little to no impact on the method used. This could be due to several factors, such as the similarity based on IQR threshold was sufficiently robust, or the dataset potentially being too small for the Siamese network to demonstrate its advantages fully.

4.2.2 Implications of Discretisation

Discrete feature encoding offers the distinct advantage of enabling effective time series modeling. By representing gestures as sequences of well-defined states, strong predictive accuracy was achieved with the Transformer-based models. However, a key limitation of discretisation is reduced flexibility for distinguishing between nuanced gestures that might be visually similar but differ subtly in their continuous representations. This limitation is likely to affect the generalisation of classification methods, as explored in Section 5.

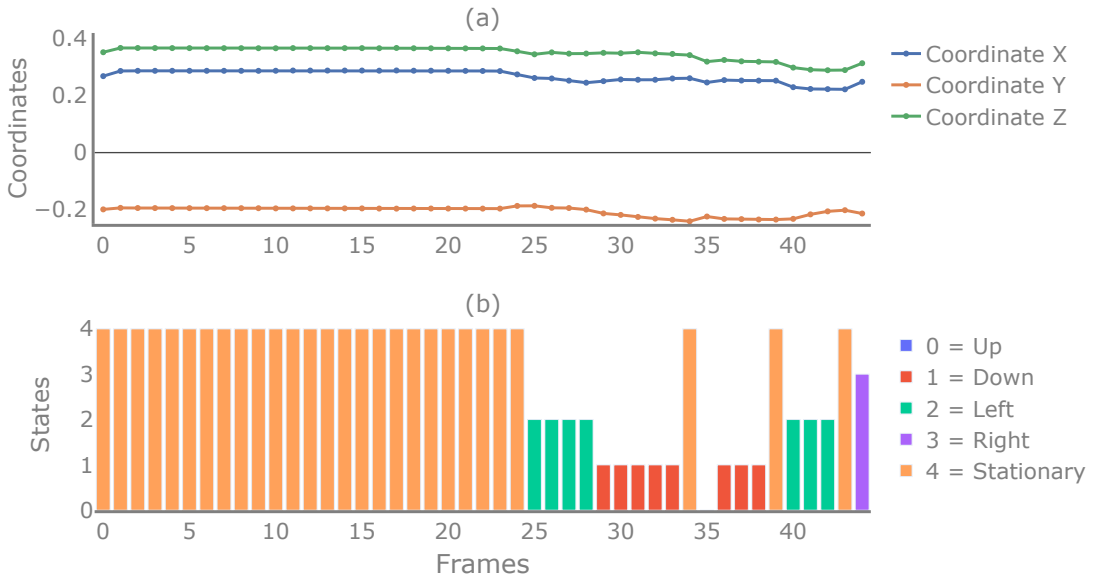


Figure 3: Discrete vs. Continuous Encoding Gesture Sequence Example.
a) 3D Coordinates of the Centroid of all Hand Landmarks. b) The Equivalent Gesture Sequence with Discretised Moving Direction States

4.3 Transformer Model for Gesture Sequence Prediction

The Transformer architecture incorporates an embedding layer for each input feature, facilitating the learning of meaningful representations. The embedded features undergo concatenation and pass through a linear encoder layer before being fed into the Transformer encoder. The encoder comprises multiple layers of self-attention and feedforward neural networks, empowering the model to capture long-range dependencies and temporal patterns within the input sequence. To inject temporal information into the permutation-invariant Transformer architecture, positional encoding is applied to the input sequence. The decoder layer subsequently maps the encoded representations to the output space, which is split into separate features corresponding to the predicted gesture attributes.

4.3.1 Single-Step / Multi-Step Model

The single-step prediction model operates with an input sequence length set to the desired number of historical frames, while the output window is fixed at 1, enabling the model to predict only the immediate next frame. Conversely, the multi-step prediction model accommodates a longer input sequence and generates predictions for multiple future frames within a specified output window. An illustration of the this is shown in Figure 12.

4.4 Early Classification with Limited Observation

4.4.1 Autoregressive Forecasting

In the context of gesture recognition, autoregressive forecasting is a technique used to iteratively predict future hand gesture states based on a combination of the initial observed sequence and the model's own predictions. This approach is particularly relevant for enhancing early classification, where the goal is to recognise gestures as soon as possible, even before the complete gesture sequence is available (see Figure 14 for a visual representation). The process can be broken down as follows:

- **Initial Prediction:** The process begins with an initial input sequence of hand gesture states. The Transformer model, whether single-step or multi-step, is used to make an initial prediction of a specified number of future states (the output window).
- **Input Update:** The key idea in autoregressive forecasting is to then incorporate these predicted states back into the input sequence. The model essentially treats its own predictions as if they were true observations. A new input sequence is formed by appending the predicted states to the end of the original observed sequence (removing the oldest states to maintain the desired sequence length).
- **Iterative Prediction:** The model generates another prediction based on the updated

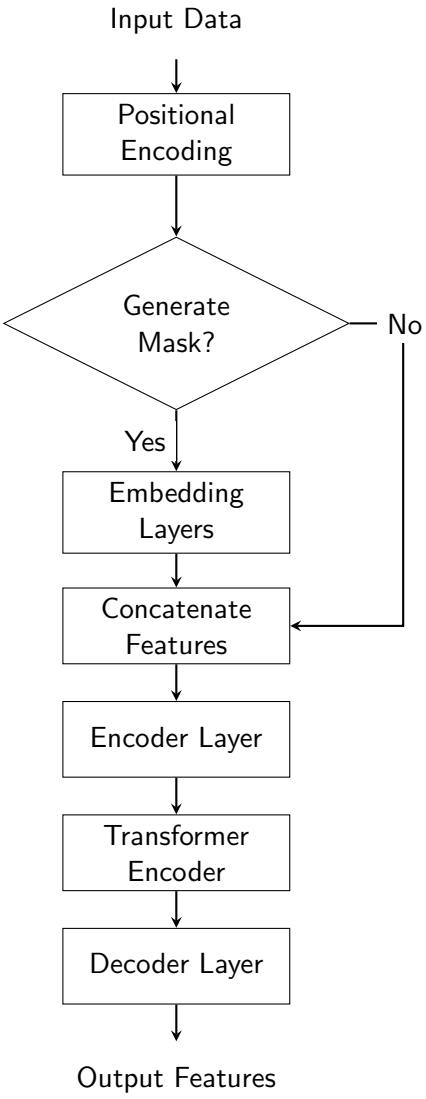


Figure 4: Diagram of the adapted Transformer Forward Pass (Encoding and Decoding), Emphasising Feature Concatenation. *Input data comprising various hand features encoded into individual discrete states. In contrast to methods pre-encoding all features into a single representation, here the discrete features are concatenated with each other during the forward pass. This allows the model to learn individual as well as collective representations based on the combination of the discrete features.*

input sequence. This process of appending predictions and regenerating the next output window is repeated for a specified number of iterations.

Limited Observation: Autoregressive forecasting extends the model's visibility beyond the initial observation window (see Figure 13). This approach may prove particularly beneficial in cases where gestures share similar initial segments. For example, trying to identify the gestures for "stop" and "slow down". Both might begin with a raised hand, making differentiation impossible based solely on initial data. However, with autoregressive forecasting, the model could anticipate if the follow-up motion remains flat (stop) or starts to curve downwards (slow down). Importantly, the effectiveness of autoregressive forecasting in these situations is still contingent on how accurately the model's own predictions represent the true, underlying patterns of the gestures.

Choosing Between Autoregressive & Multi-step: Multi-step prediction models and autoregressive models differ in their approach to forecasting future states. Multi-step models predict multiple future states directly, making them naturally suited for early classification tasks. In contrast, autoregressive models predict one step at a time, using the predicted output as input for the next prediction. This iterative process can potentially lead to more accurate long-term predictions, but at the cost of increased computational complexity.

In this study, performance of both multi-step and autoregressive models for early classification are compared. The results show that while a well-tuned multi-step model can provide a strong baseline performance, autoregressive forecasting can yield additional accuracy gains in certain scenarios. However, the choice between the two approaches depends on the specific requirements of the early classification task and the computational resources available. Careful analysis is necessary to determine whether the improved accuracy of autoregressive models justifies their higher computational cost compared to the baseline performance of a multi-step model.

4.5 Gesture Classification

4.5.1 Lookup Table

Construction: To facilitate efficient gesture classification, a lookup table approach was employed. The lookup table was constructed by extracting each trial sequence. The resulting lookup table structure consisted of a list of sequences for each gesture, with each sequence having varied lengths.

Preparation: To ensure the quality and uniformity of the gesture recognition process, the use of several techniques for data preprocessing and augmentation were considered in order to create consistent gesture sequences within the lookup table:

1. **Sequence Length Normalisation:** Gesture sequences were normalised to a fixed target

Gesture	Data
Gesture 1	$\begin{bmatrix} [4,2,0], & [4,2,1], & \dots \\ [4,1,1], & [4,1,1], & \dots \\ & \vdots & \end{bmatrix}$
Gesture 2	$\begin{bmatrix} [3,2,1], & [3,1,1], & \dots \\ [3,1,1], & [3,2,1], & \dots \\ & \vdots & \end{bmatrix}$
	$\vdots$

Table 1: Lookup Table Structure. Each gesture is associated with multiple sequences of discretised features (i.e., moving direction, palm orientation, hand pose).

length of 64 using linear interpolation. This ensured uniformity and compatibility across all comparisons, which was essential for accurate classification. Normalisation allowed each gesture sequence in the lookup table to be compared to each time series predicted sequence in a consistent manner.

2. **Discrete Sequence Augmentation:** Sequence augmentation specifically for discrete data was investigated. The goal was to decrease temporal similarity between gesture sequences that were not from the same gesture (i.e., for gestures that differ very subtly through time) and increase temporal similarity between gesture sequences from the same gesture. Various researched and experimentally created methods were attempted. However, none of these methods yielded benefits in separating the temporal similarity between gesture sequences. This remains an area for future research to investigate more sequence augmentation methods specifically for discrete data.

4.5.2 Sequence Similarity Metrics

In the process of selecting the most suitable metric for gesture sequence classification, several options were considered:

1. **String Similarity Ratio Metrics (RapidFuzz library [24]):** The string similarity metrics were considered due to their computational efficiency. However, the metric was not used as the package required sequences of strings, which was not directly applicable to the data. This could be a potential area for future research.

2. **Optimal Sequence Alignment and Longest Common Subsequence:** These methods were considered both individually as well as togther in combination as many dynamic gesture sequences contain small descriptive sub-sequences within the larger gesture sequence. That said, a key objective of the study is to work with similar dynamic gestures, such as swipe up and swipe down, and subtle variations within these sub-sequences can be critical for accurate classification.

3. **Dynamic Time Warping (DTW):** The selected classification metric was DTW as it allowed for the classification of gesture sequences with varying lengths and had no constraints on the type of data, making it a more general and versatile distance metric.

Classification Metric Selection: Dynamic Time Warping (DTW) is a technique used to measure the similarity between two time series or sequences that may vary in speed or timing. It finds the optimal alignment between the sequences by warping the time axis, allowing for non-linear mapping between the points. DTW calculates the minimum distance or cost required to align one sequence with another, considering the possibility of stretching or compressing the time axis.

In this study, DTW was applied to the data by using it to classify gesture sequences consisting of three discrete features for each frame: moving direction, palm orientation, and hand pose. To classify a new gesture sequence, the DTW distance between the sequence to classify and each gesture sequence in a lookup table was calculated and stored in a results dictionary. The distances were then sorted, and the average of the 'Top K' smallest distances was taken for each gesture in the lookup table. The classified gesture was the one with the smallest average distance. A value of $K = 20$ was chosen due to:

- Each gesture in the lookup table containing a total of 200 performed trials (Section 4.1.1). So a value of $K = 20$ ensured that the average of the top 10% of gesture sequences was accounted for, rather than just using the minimum distance from each gesture.
- The 'Top K' approach was adopted to mitigate the issue of misclassifications that arose when classifying gestures using a 'Top 1' minimum distance. The misclassifications were caused by the presence of gesture sequences in the lookup table that were very similar to sequences of other gestures, resulting in very low distances. To address this, a larger sample size was used for averaging, and a 10% value (i.e., Top-K of 20) was chosen as a good starting point.

It is important to note that further research is required to determine a more empirical value for K. The choice of K may depend on various factors such as the complexity of the gestures, the size of the dataset, and the desired balance between accuracy and computational efficiency.

4.6 Training and Testing

4.6.1 Data Collection for Testing

In order to test the Transformer model and lookup table, an Ultraleap Leap 2 camera [25] was utilised to capture new hand gesture sequences. The focus was on four specific gestures found in the dataset: 'Swipe right', 'Swipe left', 'Swipe up' and 'Swipe down'.

These gestures were chosen due to their similar nuanced dynamics, allowing for an evaluation of the discretised features' ability to describe the differences between each gesture and classify them with confidence.

The data collection process involved the following steps:

1. Adapting code from the Gemini LeapC Python Bindings **GitHub repository** to capture hand landmarks with the Ultraleap camera and manipulate them in Python.
2. For every frame captured by the Ultraleap camera, the adapted code stored the hand landmarks in an object and extracted the relevant features (moving direction, palm orientation, and hand pose).
3. Five trials were performed for each of the four gestures (Swipe right, Swipe left, Swipe up, Swipe down).
4. Each trial consisted of a varying number of frames, to ensure data was preprocessed consistently, each performed gesture sequence was normalised to a fixed length of 64.

The gesture sequence data, containing the extracted features for each frame across multiple trials, are prepared in the same way as the lookup table gesture sequences Section 4.5.1. This data will be used to test the different model variations performance in classifying the four gestures based on the discretised features.

4.6.2 Evaluation Metrics

Transformer Prediction Accuracy:

- **Feature-wise:** This metric assesses the Transformer model's ability to correctly predict individual features of the hand gesture representation. These features are moving direction; palm orientation; and hand pose. Analysing feature-wise accuracy helps identify which specific aspects of the gesture representation the model is particularly good (or less good) at predicting.
- **State-wise:** This metric focuses on the model's ability to correctly predict the combined state representation of the hand gesture at each time step. A state comprises the combination of the discretised values for the features: moving direction; palm orientation; and hand pose. State-wise accuracy provides an overall picture of how well the model understands the sequential nature of the gestures.

Cross-Entropy Loss: The loss function employed is the Cross-Entropy Loss, which is well-suited for multi-class classification tasks. Cross-Entropy Loss quantifies the dissimilarity between the predicted probability distribution and the true probability distribution of the target classes. By minimising this loss, the model learns to assign higher probabilities to the correct gesture attributes, thereby improving its classification accuracy.

Classification Performance Metrics:

- **Precision:** Precision is the ratio of true positive predictions to the total number of positive predictions (true positives + false positives). It measures the model's ability to correctly identify positive instances while minimising false positives. A high precision indicates that when the model predicts a particular gesture, it is highly likely to be correct.
- **Recall:** Recall, also known as sensitivity or true positive rate, is the ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). It measures the model's ability to identify all the positive instances in the dataset. A high recall indicates that the model successfully captures a large proportion of the relevant gesture.
- **F1-score:** The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, taking into account both precision and recall. The F1-score is particularly useful when the classes are imbalanced, as it gives equal importance to both false positives and false negatives. A high F1-score indicates a good balance between precision and recall.
- **F2-score:** The F2-score is a variant of the F-score that places more emphasis on recall than precision. It is calculated by assigning a higher weight (2) to recall in the harmonic mean formula. The F2-score is useful in scenarios where false negatives are more costly than false positives, and capturing as many positive instances as possible is a priority.

4.6.3 Training Procedure

The dataset is split into train, validation, and test sets to commence the training procedure. Processing steps, including stationary dropout and cutoff filtering, are applied to enhance the model's generalisation ability and address class imbalance. A percentage of stationary moving direction states is removed from the sequences by stationary dropout, while states with a total count below a specified percentile threshold are discarded by cutoff filtering. Each gesture sequence is normalised to a fixed length of 64 using linear interpolation before extracting discretised gesture sub-sequences of length equal to the sequence length. This normalisation step ensures that all gesture sequences have the same length, allowing the entire gesture sequences to be extracted without losing any information, as the different model variations trained were for sequence lengths of 16 and 32. Duplicate sequences are removed to ensure data balance and prevent overfitting. The Train/Test distributions of the three features (moving direction, palm orientation, hand pose) for the Transformer models are shown in Figure 15.

The training loop employs the Adam optimiser with a learning rate of 0.01 and a warmup schedule based on the learning rate decay formula described in the *"Attention is all you need"* paper [20]. The models are trained for 100 epochs with a batch size of 64. Table 11 summarises the key hyperparameters used in the training process.

4.7 Code Implementation

The code implementation for the methodology discussed above can be found in this **GitHub repository.**

5 Results and Discussion

5.1 Transformer Models

5.1.1 Training Evaluation:

Analysis of Figure 16 reveals that models trained with larger output window sizes exhibit decreased performance compared to those trained on smaller output windows. This likely stems from the larger output windows requiring the models to process a greater number of input states before prediction. Consequently, models with smaller output windows have the advantage of considering a more focused, recent portion of the sequence. There's no clear difference in accuracy between Figure 16.1a and Figure 16.2e, implying no clear difference in performance between the two sequence lengths. This suggests that increasing the number of input states doesn't necessarily lead to an improved accuracy. Therefore, the remaining investigation will focus on models trained with a smaller output window size as these models have the potential to provide earlier gesture recognition. By focusing on models that predict fewer future frames, the system can make predictions earlier in the gesture sequence, enabling quicker response times. This is particularly important for real-time applications where early gesture recognition can greatly enhance user experience and system responsiveness. Furthermore, using models with smaller output windows reduces computational complexity and resource requirements, making them more suitable for deployment on resource-constrained devices. Given the similar performance between models with different sequence lengths, prioritising early gesture recognition through smaller output windows is a pragmatic approach that balances accuracy and efficiency.

Model	Model Type	Sequence Length	Output Window
Model A	Single-Step	16	1
Model B	Single-Step	32	1
Model C	Multi-Step	16	4
Model D	Multi-Step	32	8

Table 2: Model variations selected for further investigation

5.1.2 Performance Comparison of Selected Models

To test the performance of the selected models, the data collected from Section 4.6.1 to see how well the models generalise to unseen data. The collected data consists of five trials for each of the four gestures: 'Swipe right', 'Swipe left', 'Swipe up', and 'Swipe down'. Therefore the comparison will evaluate the different model's capabilities in predicting nuanced sequences of discretised features.

The performance comparison will be conducted in three stages:

1. **Single-Step Models:** The performance of Model A and Model B will be compared. These models are trained with an output window of 1 but have different sequence lengths

(16 and 32, respectively). By evaluating their performance on the collected data, we can determine how well these models generalise to unseen data and if the difference in sequence length has any significant impact on their prediction accuracy.

2. **Multi-Step Models:** The performance of Model C and Model D will be compared. These models are multi-step models with different sequence lengths (16 and 32) and output windows (4 and 8). Due to Model C's smaller output window, this comparison will focus predicting 4 time steps. It will help to understand how well multi-step models perform on unseen data and if the increased output window size affects their ability to predict discrete features accurately.

3. **Autoregressive Forecasting Performance:** The autoregressive forecasting performance of Model A and Model C will be compared with the normal forecasting performance of Model D. Model A and Model C will be used to autoregressively forecast the output window of Model D (8), while Model D will directly predict its entire output window in a normal manner. This comparison will provide insights into the effectiveness of autoregressive forecasting compared to non-autoregressive forecasting for longer horizon predictions.

Single-Step Models: Table 3 shows the average state-wise accuracy and average feature-wise accuracy for the two single-step models (Model A & Model B). **Feature 1** is the 'Moving Direction', **Feature 2** is the 'Palm Orientation' and **Feature 3** is the 'Hand Pose'.

Model	**Average State-wise Accuracy**	**Average Feature-wise Accuracy**		
		Feature 1	**Feature 2**	**Feature 3**
Model A	84.928%	68.982%	91.609%	94.192%
Model B	87.267%	77.321%	90.665%	93.818%

Table 3: Average Accuracy Comparison of Single-Step Models (Table 1). Model B demonstrates slightly higher state-wise accuracy than Model A, potentially due to its longer sequence length. Both models struggle most with predicting Feature 1 (moving direction), suggesting inherent challenges in predicting this feature.

Model B exhibits a slightly higher state-wise accuracy compared to Model A. This difference might be attributed to Model B's longer sequence length, potentially allowing it to better learn patterns of motion and therefore improve its predictions of Feature 1 (moving direction). The results suggest that increasing the sequence length can lead to better generalisation and prediction accuracy, especially for features with higher variability like moving direction.

Interestingly, both models demonstrate lower accuracy when predicting Feature 1 compared to Features 2 and 3. This could indicate that moving direction is inherently more challenging to predict, possibly due to greater variability in motion paths. The higher accuracy on Features 2 (palm orientation) and 3 (hand pose) might be partially explained by the limited number of states in these features, making them easier to predict accurately.

Multi-Step Models: Figures 5,?? show how the multi-step models' feature-wise and state-wise accuracy vary over time.

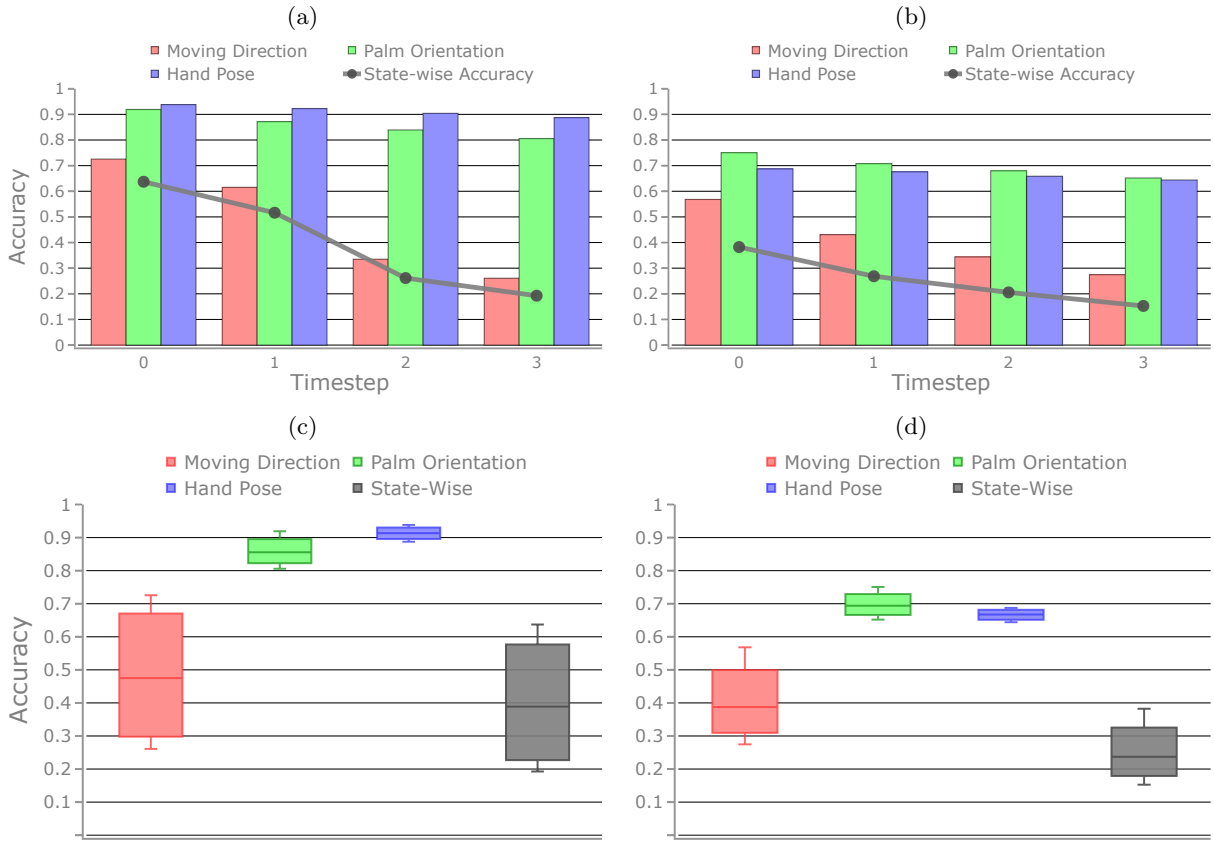


Figure 5: Multi-Step Model Performance over 4 Timesteps: ***(a) Model C (sequence length 16):*** *Demonstrates a sharp decline in moving direction accuracy at timestep 2 and has higher variability in state-wise accuracy compared to Model D. Despite this, Model C outperforms Model D on Features 2 and 3. (b)* ***Model D (sequence length 32):*** *Exhibits more consistent state-wise accuracy, as well as feature-wise accuracy for Features 2 and 3.*

The results illustrate that Model C has a slightly higher state-wise accuracy than Model D when predicting over a 4-timestep horizon. However, the box plots reveal greater variability in Model C's state-wise accuracy, suggesting that Model D maintains more consistent performance across different trials. This finding indicates that increasing the sequence length or number of inputs states (as in Model D) might lead to more stable predictions, even if the average accuracy is slightly lower.

Both models struggle with predicting Feature 1 (moving direction), with Model C showing a particularly sharp accuracy drop-off at timestep 2. This decline could contribute to the less stable state-wise performance of Model C and suggests that predicting moving direction becomes increasingly challenging as the prediction horizon extends.

Model C outperforms Model D on Features 2 and 3, particularly on Feature 3. However, Model D demonstrates more consistent accuracy on Feature 3. These results imply that increasing the output window size (as in Model D) might lead to more stable predictions for certain features, even if the average accuracy is slightly lower.

Autoregressive Forecasting Performance: Figure 6 provides insights into the comparative performance of autoregressive forecasting and multi-step forecasting.

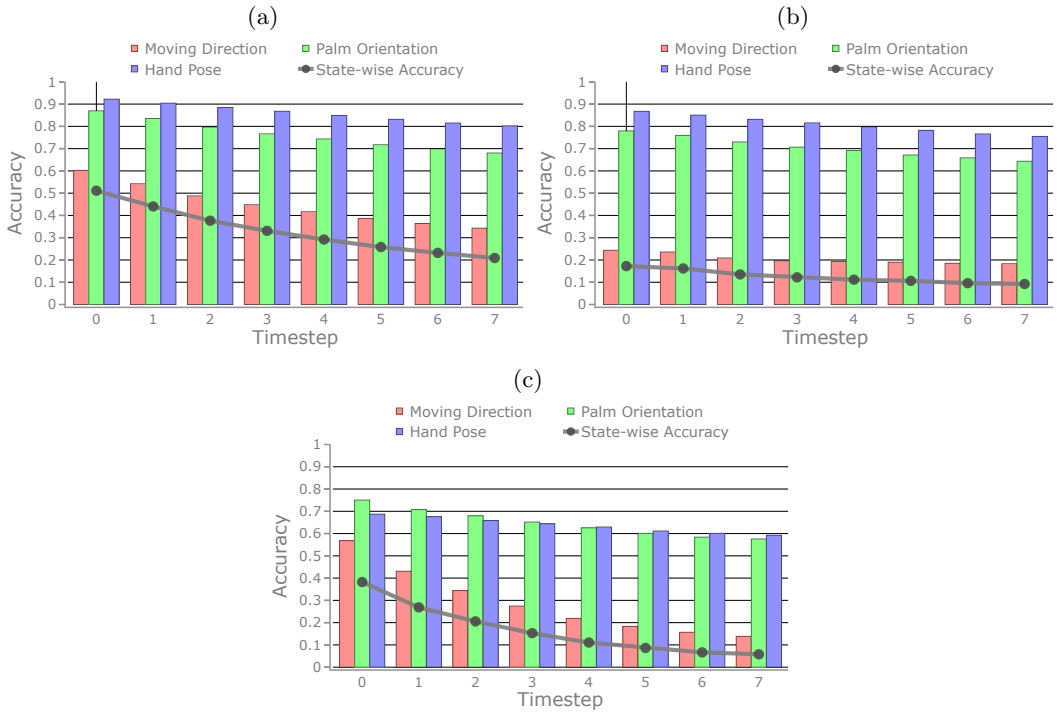


Figure 6: Autoregressive vs. Multi-Step Forecasting Performance over 8 Timesteps. *(a)* ***Model A Autoregressive:*** *Shows slightly higher state-wise accuracy than multi-step Models C and D, but struggles with Feature 1 (moving direction). (b)* ***Model C Autoregressive:*** *Demonstrates significantly poorer performance, especially in predicting Feature 1, impacting overall accuracy.* ***(c) Model D Multi-Step:*** *The decline in state-wise accuracy over time mirrors Model A's autoregressive trend, suggesting potential advantages of the autoregressive approach in this context.*

Model A, employing an autoregressive approach, exhibits slightly higher state-wise accuracy compared to both Model C and Model D, which utilise multi-step forecasting. A notable observation is the significantly lower performance of Model C's autoregressive forecasting for Feature 1 (moving direction) compared to the other models. This deficiency negatively impacts its overall accuracy.

Interestingly, the decline in state-wise accuracy over time for Model D's multi-step forecasts mirrors the trend observed in Model A's autoregressive predictions. This suggests that, in this context, the autoregressive approach with the single-step Model A might offer advantages in terms of accuracy.

As with the single-step and multi-step models, all three models exhibit the lowest performance on Feature 1, with Model C again demonstrating the most pronounced challenges. This reinforces the inherent difficulty of predicting moving direction using autoregressive forecasting methods. Across all models, Feature 3 (hand pose) consistently emerges as the most predictable feature among the three.

5.2 Model Suitability: Balancing Latency and Reliability Considerations

The selection of the most suitable model for gesture recognition applications necessitates a careful evaluation of the trade-off between prediction latency and prediction reliability. This trade-off arises due to the inherent characteristics of the models under consideration, with smaller output windows favoring reduced latency and larger output windows prioritising reliability over extended prediction horizons.

5.2.1 Latency-Critical Applications

In scenarios where minimising prediction delay is of paramount importance, models with smaller output windows should be the preferred choice. These models generate predictions with lower latency due to the reduced number of time steps that require computation. Consequently, the following metrics become crucial for latency-sensitive applications:

- **Early Prediction Accuracy:** Evaluating the model's accuracy within the initial few time steps is essential, as these predictions will be utilised in real-time applications.
- **Autoregressive Prediction Accuracy:** Assessing the model's performance when using its own predictions as inputs for subsequent time steps is critical. If the autoregressive accuracy degrades rapidly, the model may not be suitable for latency-critical applications.

Potential Candidates:

- **Model A:** This model demonstrates high accuracy within the first time step and exhibits reasonable autoregressive performance, making it a strong contender for latency-sensitive scenarios.
- **Model B:** With its longer sequence length, Model B potentially provides a larger context for predictions, which could improve its early prediction accuracy. The trade-offs between its sequence length and computational cost would need to be carefully evaluated.
- **Model C:** While offering slightly higher overall accuracy than Model A, its autoregressive performance for Feature 1 (moving direction) is significantly worse. The importance of accurately predicting moving direction in the specific application should be carefully considered when evaluating this model.

5.2.2 Reliability-Focused Applications

In applications where prediction reliability is of utmost importance, prioritising models with larger output windows and potentially longer sequence lengths (e.g. 32) becomes crucial. These models offer the advantage of providing predictions further into the future, allowing for more informed and proactive decision-making. The key metric to focus on in this context is:

- **Sustained Prediction Accuracy:** Evaluating the maximum accuracy achieved by the model and its ability to maintain a high level of performance across the entire prediction horizon is essential.

Potential Candidates:

- **Model D:** Despite exhibiting slightly lower average accuracy, Model D demonstrates more consistent performance across trials and a slower decline in accuracy over time compared to Model C. This characteristic makes it a more reliable choice for applications where consistent predictions are essential for maintaining reliability.
- **Model B:** With its longer sequence length, Model B demonstrates better performance in predicting Feature 1 (moving direction) compared to Model A. This could be advantageous in situations where accurately predicting the direction of movement is crucial for maintaining reliability.

In summary, the selection of the most suitable model should be guided by the specific requirements of the application, carefully weighing the trade-off between prediction latency and prediction reliability. Latency-sensitive applications may benefit from models with smaller output windows, while reliability-focused applications may favor models with larger output windows and longer sequence lengths.

Table 4: Model Suitability for Latency-Critical and Reliability-Focused Applications

Application Category	**Suitable Models**	**Key Performance Metrics**
Latency-Critical	Model A, Model C, Model D	• Early Timestep Accuracy (e.g., Accuracy at timestep 1 and 2) • Autoregressive Accuracy
Reliability-Focused	Model B, Model D	• Maximum Achieved State-wise Accuracy • Sustained State-wise Accuracy over Prediction Horizon

5.3 Early Gesture Classification

To assess the efficacy of early gesture classification, each model was evaluated on its ability to predict a standardised sequence. This sequence was normalised to a length equivalent to the the model's specific sequence length. This normalisation procedure ensured a fair and application-oriented evaluation by providing each model with an input sequence commensurate with its designed prediction scope.

The classification performance achieved using each model's predicted sequence was then compared to the baseline performance obtained by classifying the ground truth gesture sequence

directly. This comparison allows for an evaluation of whether the potential accuracy improvement offered by time-series prediction outweighs the added complexity and latency introduced by the prediction models. Ultimately, this analysis provides insights into the feasibility and effectiveness of employing time-series models for early gesture recognition applications.

Table 5: Performance Metrics for Different Models.
The first row "True Sequence" represents the actual, unpredicted sequence that serves as the benchmark for evaluation. Red values indicate a metric lower than the baseline "True Sequence" case, green values indicate a metric higher than the baseline, and gray values are equal to the baseline.

Model	Precision	Recall	F1 Score	F2 Score
True Sequence	46.071%	40%	42.5%	40.88%
Model A	41.04%	35%	36.90%	35.54%
Model B	52.5%	45%	47%	45.44%
Model C	49.65%	30%	30.59%	29.34%
Model D	53.57%	40%	39.77%	39.43%

Overall Performance: Analysis of Table 5 reveals that Model B demonstrates the best overall performance, with improvements across precision, recall, F1, and F2 scores compared to the baseline "True Sequence". This suggests that the longer sequence length effectively captures temporal patterns and leads to better predictions, ultimately improving classification accuracy. Model D achieves the highest precision but suffers in recall and F1/F2 scores, likely due to accumulated errors over its longer prediction horizon. Model C, despite having good precision, has the lowest recall, leading to the worst F1 and F2 scores among all models. This highlights the trade-off between precision and recall, emphasising the importance of balancing these metrics for optimal performance. Model A underperforms compared to the baseline across all metrics, indicating that the shorter sequence length and output window might not be sufficient for accurate predictions and subsequent classification.

Table 6: 'True Sequence' Performance Metrics for Different Gestures

Gesture	Precision	Recall	F1 Score	F2 Score
Swipe Right	14.29%	20%	16.67%	18.52%
Swipe Left	75%	60%	66.67%	62.5%
Swipe Up	20%	20%	20%	20%
Swipe Down	75 %	60%	66.67%	62.5%

(a)

Gesture	Precision	Recall	F1 Score	F2 Score
Swipe Right	12.5%	20%	15.38%	17.86%
Swipe Left	60%	60%	60%	60%
Swipe Up	25%	20%	22.22%	20.83%
Swipe Down	66.67%	40%	50%	43.48%

(b)

Gesture	Precision	Recall	F1 Score	F2 Score
Swipe Right	25%	40%	30.77%	35.71%
Swipe Left	60%	60%	60%	60%
Swipe Up	25%	20%	22.22%	20.83%
Swipe Down	75%	60%	75%	62.22%

(c)

Gesture	Precision	Recall	F1 Score	F2 Score
Swipe Right	11.11%	20%	14.29%	17.24%
Swipe Left	37.5%	60%	46.15%	53.57%
Swipe Up	50%	20%	28.57%	22.73%
Swipe Down	100%	20%	33.33%	23.81%

(d)

Gesture	Precision	Recall	F1 Score	F2 Score
Swipe Right	14.29%	20%	16.67%	18.52%
Swipe Left	50%	60%	54.55%	57.69%
Swipe Up	50%	60%	54.55%	57.69%
Swipe Down	100%	20%	33.33%	23.81%

Table 7: Performance Comparison of Models for Different Gestures. Red values indicate a metric lower than the 'True Sequence' baseline, green values indicate a metric higher than the baseline, and gray values are equal to the baseline. See Figure 19 for the corresponding confusion matrices. *(a)* ***Model A:*** *Struggles significantly with 'Swipe Right' and 'Swipe Down'. (b)* ***Model B:*** *Demonstrates improved 'Swipe Right' prediction compared to Model A, but still performs below baseline for 'Swipe Left'. (c)* ***Model C:*** *Exhibits difficulty predicting 'Swipe Left' and particularly 'Swipe Right'. (d)* ***Model D:*** *Achieves perfect precision on 'Swipe Down', but at the cost of very low recall.*

Gesture-Specific Performance: The performance for individual gestures is analysed, revealing interesting insights. "Swipe Right" remains the most challenging gesture to classify across all models, aligning with the observations from Section 5.1.2, where Feature 1 (moving direction) consistently exhibited the lowest prediction accuracy which would imply the models struggle to predict nuanced motion. Model B shows significant improvements for "Swipe Right" compared to the baseline and other models, further supporting the advantage of longer sequence lengths in capturing the complexities of movement direction. "Swipe Left" and "Swipe Down" show mixed results across models, with some improvements and some regressions compared to the baseline, suggesting that the effectiveness of prediction and subsequent classification depends on the specific characteristics of each gesture. "Swipe Up" sees improvements in precision and F1/F2 scores for Models B and C, while Model D maintains similar performance to the baseline.

Key Takeaways and Considerations: The results indicate that early gesture classification using time-series prediction shows promising results, especially with models like B that effectively capture temporal information. The choice of model and sequence length significantly impacts performance, with longer sequence lengths generally leading to better accuracy. Balancing precision and recall is crucial for optimal classification performance. However, predicting and classifying gestures with complex movement patterns, like "Swipe Right", remains a challenge. Further investigation into feature engineering and model architectures could improve performance for specific gestures and address the limitations observed.

Future Directions: To further enhance the performance of early gesture classification, several future directions are proposed. Exploring alternative model architectures, such as recurrent neural networks with attention mechanisms, could better capture long-term dependencies and improve prediction accuracy. Investigating the impact of different feature representations and dimensionality reduction techniques on classification performance could provide insights into optimal data preprocessing strategies. Evaluating the feasibility of incorporating additional sensor data, such as gyroscope or accelerometer readings, could provide richer context for gesture recognition. Developing adaptive models that can adjust their prediction horizon based on the observed gesture dynamics could optimise the trade-off between latency and reliability.

5.4 Demo Video

A demonstration of how the Ultraleap camera extracts discretised features for gesture classification can be viewed here: **Demo Video**

6 Conclusion

6.1 Summary of Findings

This research investigated the use of Transformer models for early classification of dynamic hand gestures in virtual reality (VR) environments. The primary objective was to explore the effectiveness of Transformer architectures in predicting future hand gesture states, enabling earlier recognition compared to traditional methods that require observing the complete gesture sequence. The study adopted a discrete representation of hand gestures, encoding features such as moving direction, palm orientation, and hand pose into discrete states. This discrete encoding facilitated time series modeling while introducing a trade-off between expressiveness and generalisation capabilities.

Through comprehensive experimentation and evaluation, several key findings emerged regarding the performance of the Transformer models. The results demonstrated that the choice of model architecture and sequence length significantly impacted the classification performance, with longer sequence lengths generally leading to better accuracy. The single-step model trained on

sequences of length 32 (Model B), effectively captured temporal information, showed promising results compared to the baseline and other models. However, accurately distinguishing between similar gesture pairs (e.g., 'Swipe Right/Left', 'Swipe Up/Down') remained a challenge across all models.

6.2 Limitations and Future Work

It is important to note that the overall classification accuracy achieved using the lookup table approach was relatively low, indicating potential limitations in this methodology. Attempts were also made to train LSTM and RNN models for sequence classification, but these did not yield improved performance compared to the lookup table. These outcomes suggest that the discrete encoding of gesture features and/or the presence of noise in the dataset may have contributed to the suboptimal classification results. The discrete representation, while enabling effective time series prediction, could have compromised the ability to distinguish between nuanced gestures with subtle variations. Additionally, the inherent noise in the depth camera data, arising from factors such as occlusions and varying hand postures, may have introduced inconsistencies that hindered accurate classification.

Consequently, a crucial lesson learned from this research is the paramount importance of high-quality data for successful gesture recognition. Future work should prioritise efforts to improve data quality, either through advanced preprocessing techniques or the acquisition of cleaner datasets. Techniques such as data augmentation, denoising, and outlier removal could potentially enhance the consistency of the training data.

Another potential direction for future work would be to investigate creating multi-dimensional hand movement state tracking. By defining stationary thresholds for individual coordinates (x, y, and z), one could track whether movement veers in specific paths, encompassing not only 2D but also potential 3D trajectories. This would address the current method's limitation of calculating a single direction of motion from 2D coordinates. The Transformer's ability to efficiently learn multiple features, as evidenced by the high accuracy in predicting hand pose and palm orientation, suggests its potential effectiveness in learning all three coordinates' motion data.

While the current study explored the use of Transformer models for time series prediction and early gesture recognition, future research could investigate the application of deep learning techniques for direct sequence classification. However, the effectiveness of these approaches will ultimately depend on the quality and representativeness of the underlying data. Ensuring that the training data accurately captures the nuances and variations of different gestures is a critical prerequisite for achieving high classification accuracy.

6.3 Contributions and Outlook

Despite the limitations encountered, the findings of this research contribute to the advancement of hand tracking and gesture recognition techniques in VR environments. By leveraging the predictive capabilities of Transformer models and exploring early gesture classification, this study has laid a foundation for future work in this domain. The insights gained from the performance analysis and the identification of key challenges, such as the complexity of certain gestures and the impact of data quality, provide valuable guidance for subsequent research efforts.

To further enhance the performance of early gesture classification, several future directions are proposed. Exploring alternative model architectures, such as recurrent neural networks with attention mechanisms, could better capture long-term dependencies and improve prediction accuracy. Investigating the impact of different feature representations and dimensionality reduction techniques on classification performance could provide insights into optimal data preprocessing strategies. Evaluating the feasibility of incorporating additional sensor data, such as gyroscope or accelerometer readings, could provide richer context for gesture recognition. Developing adaptive models that can adjust their prediction horizon based on the observed gesture dynamics could optimise the trade-off between latency and reliability.

In conclusion, this research has demonstrated the potential of early gesture classification using time-series prediction with Transformer models in VR environments. While challenges remain, particularly in accurately classifying complex gestures and addressing data quality issues, the findings provide a solid foundation for future research in this domain. Continued efforts to refine data quality, explore alternative representation learning techniques, and investigate more advanced model architectures could pave the way for more responsive and immersive user interactions within VR applications.

7 References

[1] G. Buckingham, "Hand tracking for immersive virtual reality: Opportunities and challenges," *Frontiers in Virtual Reality*, vol. 2, 2021.

[2] A. Hameed, S. Möller, and A. Perkis, "How good are virtual hands? influences of input modality on motor tasks in virtual reality," *Journal of Environmental Psychology*, vol. 92, p. 102137, 2023.

[3] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt, "Real-time hand tracking under occlusion from an egocentric rgb-d sensor," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[4] S. Reza, M. C. Ferreira, J. Machado, and J. M. R. Tavares, "A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks," *Expert Systems with Applications*, vol. 202, p. 117275, 2022.

[5] B. Khanal, P. Shrestha, S. Amgain, B. Khanal, B. Bhattarai, and C. A. Linte, "Investigating the robustness of vision transformers against label noise in medical image classification," 2024.

[6] Y.-L. Hsieh, M. Cheng, D.-C. Juan, W. Wei, W.-L. Hsu, and C.-J. Hsieh, "On the robustness of self-attentive models," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1520–1529, 2019.

[7] X. Lou, X. A. Li, P. Hansen, and P. Du, "Hand-adaptive user interface: improved gestural interaction in virtual reality," *Virtual Reality*, vol. 25, 2021.

[8] A. Barten, "Gesture recognition: Enhancing virtual reality video game simulation." `https://alfredbarten.com/gesture-recognition/`, 2023.

[9] S. Li, L. Zhou, M. Fan, and Y. Xiong, "A comprehensive analysis of gesture recognition systems: Advancements, challenges, and future direct," *Applied and Computational Engineering*, vol. 43, pp. 68–75, 2 2024.

[10] A. van Diepen, M. Cox, and B. de Vries, "A probabilistic modeling approach to one-shot gesture recognition," *arXiv preprint arXiv:1806.11408*, 2018.

[11] J. P. Vásconez, L. I. B. López, Á. L. V. Caraguay, P. J. Cruz, R. Álvarez, and M. E. Benalcázar, "A hand gesture recognition system using emg and reinforcement learning: A q-learning approach," in *Artificial Neural Networks and Machine Learning – ICANN 2021* (I. Farkaš, P. Masulli, S. Otte, and S. Wermter, eds.), (Cham), pp. 580–591, Springer International Publishing, 2021.

[12] J. Paulo, P. Girão, and P. Peixoto, *Multi-view Robust Gesture Recognition for Assistive Interfaces*, pp. 1685–1695. 2020.

[13] G. Poon, K. C. Kwan, and W.-M. Pang, "Occlusion-robust bimanual gesture recognition by fusing multi-views," *Multimedia tools and applications*, vol. 78, pp. 23469–23488, 2019.

[14] C. Pacchierotti, G. Salvietti, I. Hussain, L. Meli, and D. Prattichizzo, "The hring: A wearable haptic device to avoid occlusions in hand tracking," in *2016 IEEE haptics symposium (HAPTICS)*, pp. 134–139, IEEE, 2016.

[15] H. Haresamudram, I. Essa, and T. Ploetz, "Towards learning discrete representations via self-supervision for wearables-based human activity recognition," 2023.

[16] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch, "Som-vae: Interpretable discrete representation learning on time series," 2019.

[17] G. I. Parisi, D. Jirak, and S. Wermter, "Handsom - neural clustering of hand motion for gesture recognition in real time," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pp. 981–986, 2014.

[18] C.-Y. Chen, "Accelerometer-based hand gesture recognition using fuzzy learning vector quantization," *Advanced Science Letters*, vol. 9, pp. 38–44, 4 2012.

[19] T. Wang, X. Qian, F. He, X. Hu, Y. Cao, and K. Ramani, "Gesturar: An authoring system for creating freehand interactive augmented reality applications," pp. 552–567, Association for Computing Machinery, Inc, 10 2021.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara, "Meshed-memory transformer for image captioning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10578–10587, 2020.

[22] "Robotic echoes." Retrieved December 10, 2023, from `https://www.empressvr.com/what-we-do/robotic-echoes`. Accessed: 10-December-2023.

[23] Q. De Smedt, H. Wannous, and J.-P. Vandeborre, "Skeleton-based dynamic hand gesture recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1–9, 2016.

[24] Max Bachmann and others, "Rapidfuzz," 2023. GitHub Repository.

[25] Ultraleap, "Leap motion controller 2," April 2024. Accessed April 27, 2024.

A Appendix

Table 8: Dataset comparison.

Dataset	**2D Joint Positions**	**3D Joint Positions**	**Depth Maps**	**RGB**	**Gesture Type**
Ego4D	No	No	Yes	Yes	Dynamic
ContactPose	No	Yes	Yes	Yes	Static
First-Person-Action	No	Yes	Yes	Yes	Dynamic
SCUT-DHGA	No	No	Yes	Yes	Dynamic
HandDB	Yes	Yes	Yes	Yes	Both
Briareo	No	Yes	Yes	Yes	Dynamic
DHG Dataset	Yes	Yes	Yes	No	Dynamic

Table 9: List of the 14 gestures in the DHG2016 dataset

#	**Name**	**Type**
1	Grab	Fine
2	Tap	Coarse
3	Expand	Fine
4	Pinch	Fine
5	Rotation CW	Fine
6	Rotation CCW	Fine
7	Swipe Right	Coarse
8	Swipe Left	Coarse
9	Swipe Up	Coarse
10	Swipe Down	Coarse
11	Swipe X	Coarse
12	Swipe V	Coarse
13	Swipe +	Coarse
14	Shake	Coarse

(a) Frame 0 (b) Frame 25 (c) Frame 40 (d) Frame 50

Figure 7: Example of the "Swipe Right" gesture from the DHG2016 dataset.

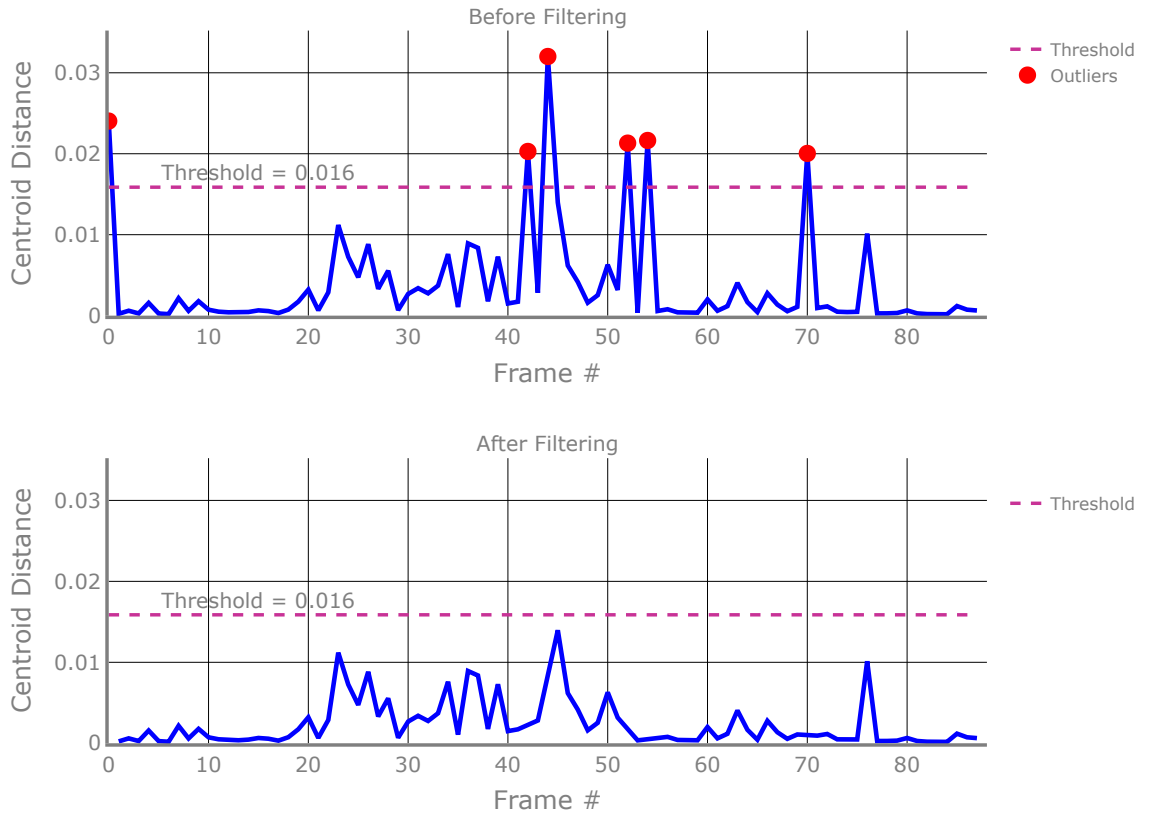


Figure 8: Hand Gesture Centroid Distances Before and After Outlier Removal.
The top subplot shows original distances between consecutive centroids with the threshold for significant movement. Points above this threshold are marked as outliers. The bottom subplot displays distances after outlier removal, demonstrating the effectiveness of the filtering process.

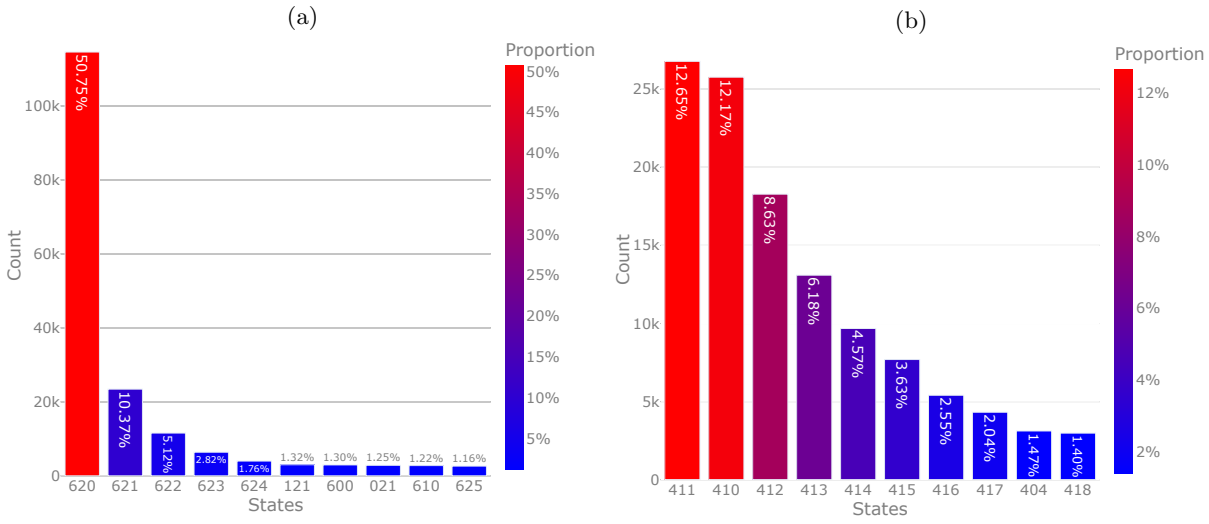
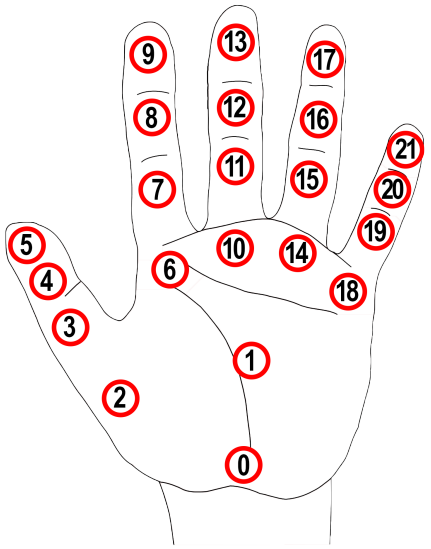


Figure 9

(a)



(b)

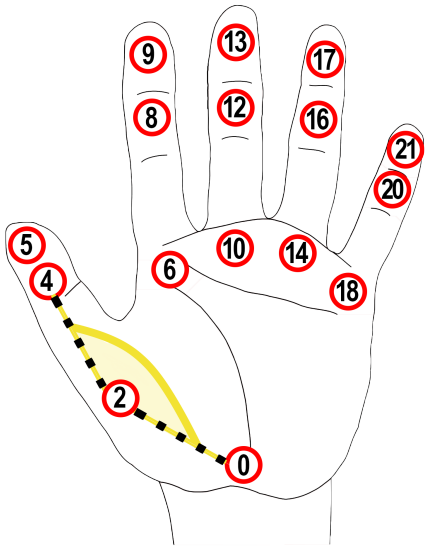
Index	Region
0	Wrist
1	Palm
2	Thumb Base
3	Thenar
4	Thenar
5	Thumb Tip
6	Index Base
7	Index
8	Index
9	Index Tip

(c)

Index	Region
10	Middle Base
11	Middle
12	Middle
13	Middle Tip
14	Ring Base
15	Ring
16	Ring
17	Ring Tip
18	Pinky Base
19	Pinky
20	Pinky
21	Pinky Tip

Figure 10: Landmarks and Regions of the DHG 2016 Hand Dataset. (a) Annotated hand landmarks on a sample image. (b) Table of hand region indexes and descriptions (part 1). (c) Table of hand region indexes and descriptions (part 2).

(a)



(b)

Finger	Joint Indexes	Joint Type
Thumb	(0, 2, 4)	MCP joint
	(2, 4, 5)	IP joint
Index	(0, 6, 8)	MCP joint
	(6, 8, 9)	PIP joint
Middle	(0, 10, 12)	MCP joint
	(10, 12, 13)	PIP joint
Ring	(0, 14, 16)	MCP joint
	(14, 16, 17)	PIP joint
Little	(0, 18, 20)	MCP joint
	(18, 20, 21)	PIP joint

Figure 11: Hand Joint Representation. (a) Diagram of a hand with the MCP joint of the thumb (yellow) and the PIP joint of the little finger (green). (b) Table listing hand joint indexes and types. *MCP joint: Metacarpophalangeal joint, the joint between the metacarpal bones and the proximal phalanges. IP joint: Interphalangeal joint, the joint between the phalanges (only applicable to the thumb). PIP joint: Proximal interphalangeal joint, the joint between the proximal and intermediate phalanges (applicable to fingers other than the thumb).*

Table 10: Preprocessing Parameters

Parameter	Value
Stationary Dropout	0.4
Cutoff Percentile (%)	25%

Table 11: Training Hyperparameters

Hyperparameter	Value
Learning Rate	0.01
Batch Size	64
Epochs	100
Optimiser	Adam
Warmup Steps	700
Dropout	0.1

Table 12: Transformer Hyperparameters

Hyperparameter	Value
Input Dimension	512
Hidden Dimension	512
Attention Heads	8
Encoder Layers	6

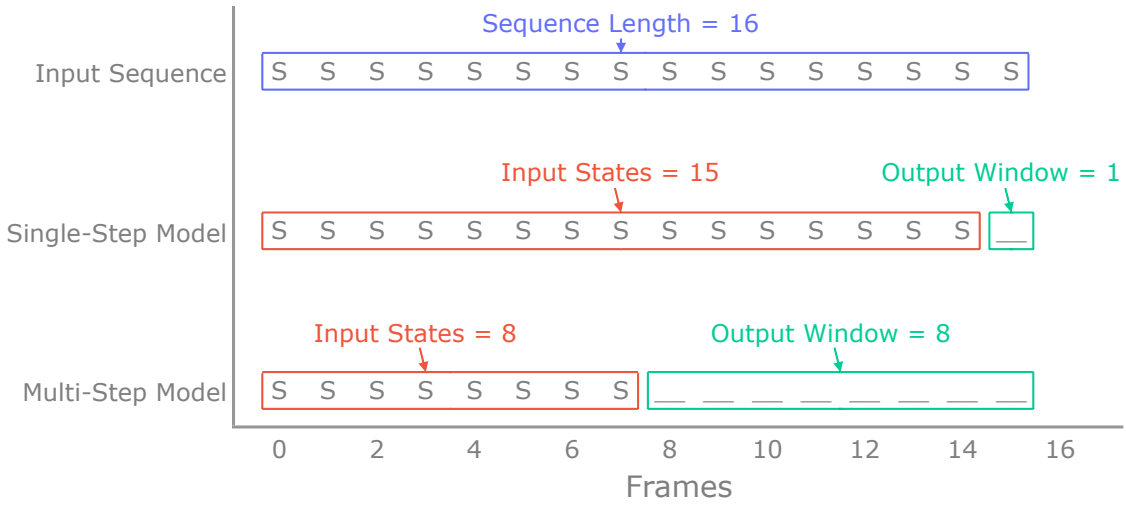


Figure 12: Illustration of Single-Step prediction vs Multi-Step prediction:
The input sequence is displayed at the top, where 'S' represents some encoded state. The example shown is for a single-step prediction model with a sequence length of 16, output window of 1; and a multi-step prediction model with a sequence length of 16, output window of 8. The green boxes highlight the model output window, where each model will make a prediction.

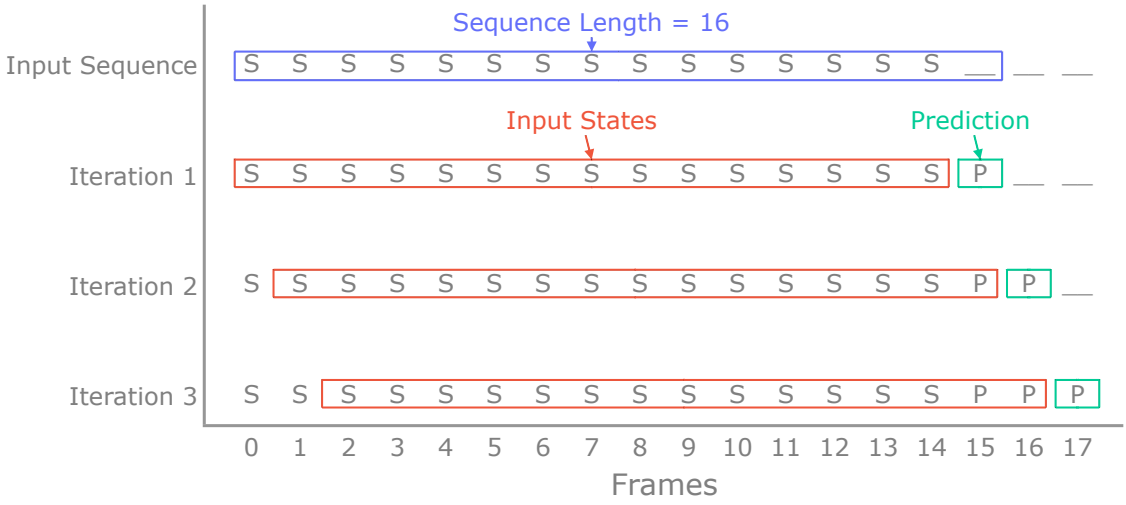


Figure 13: Illustration of Autoregressive Forecasting:
The input sequence is displayed at the top, where 'S' represents some encoded state and 'P' is some predicted state. The example shown is for a single-step prediction model with a sequence length of 16. The green box represents the forecasted value for the next time step, refined iteratively using historical and the predicted data.

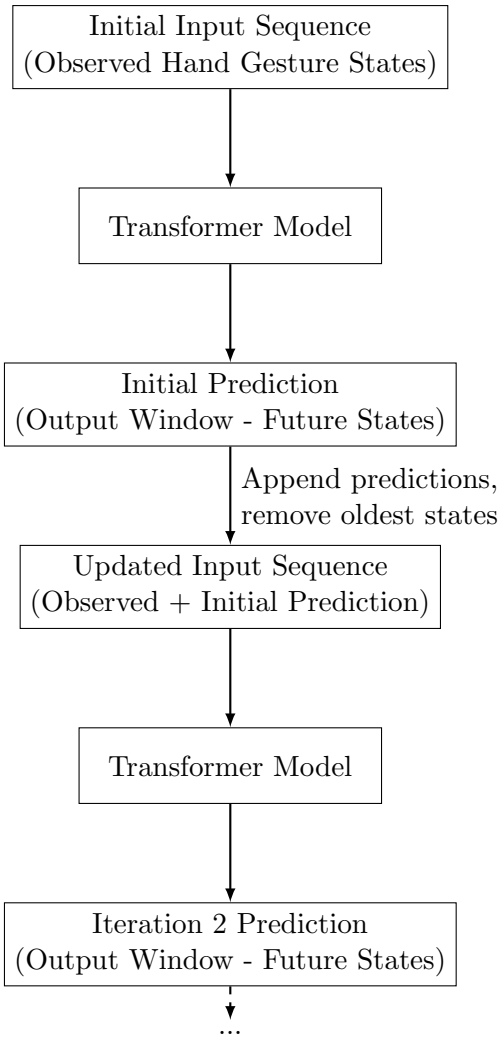
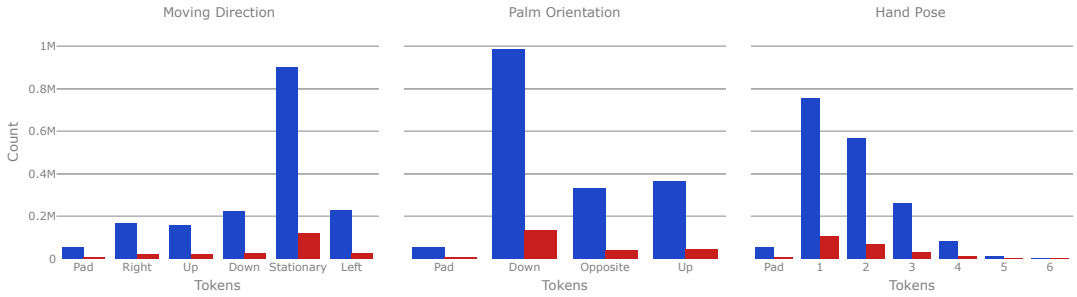
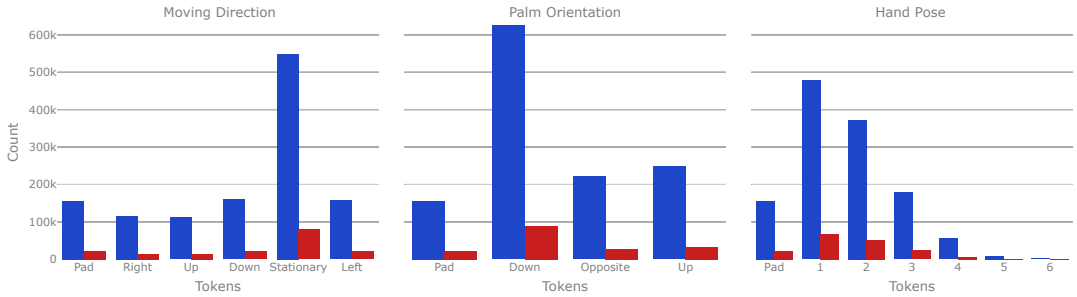


Figure 14: Autoregressive Forecasting Framework for Hand Gesture Recognition

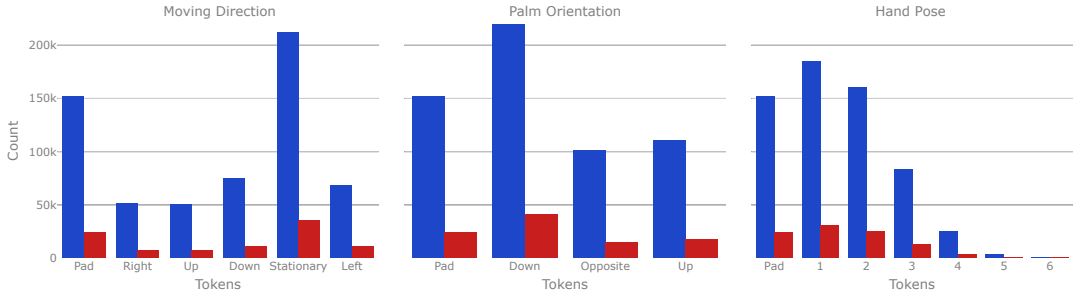
(a) Ratios: (16 : 1, 32 : 1)



(b) Ratios: (16 : 4, 32 : 8)



(c) Ratios: (16 : 8, 32 : 16)



(d) Ratios: (16 : 12, 32 : 24)

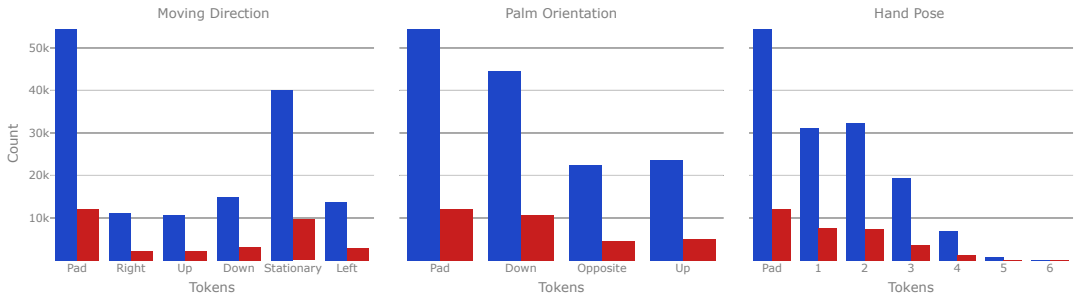
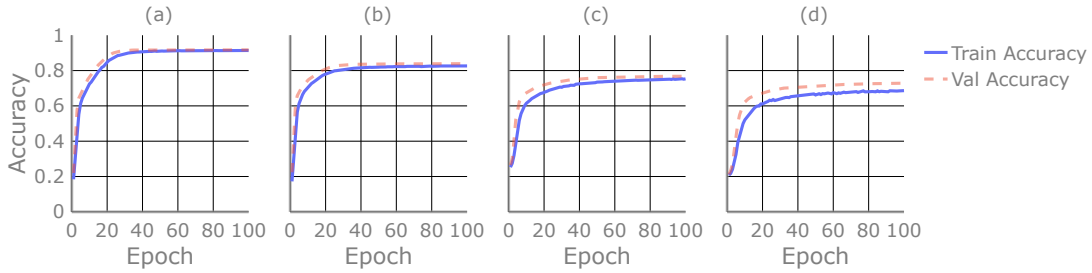


Figure 15: Comparison of Train/Test distributions for the different Transformer models, with a test size of 10%, the 'Train' counts are in blue and 'Test' counts in red for the three discrete features. 'Pad' is the mask token, the position where the model needs to predict the values based on the input sequence and output window size. The 'Ratios' are the ratio between the sequence length and output window for the various models.

(1) Sequence Length: 16



(2) Sequence Length: 32

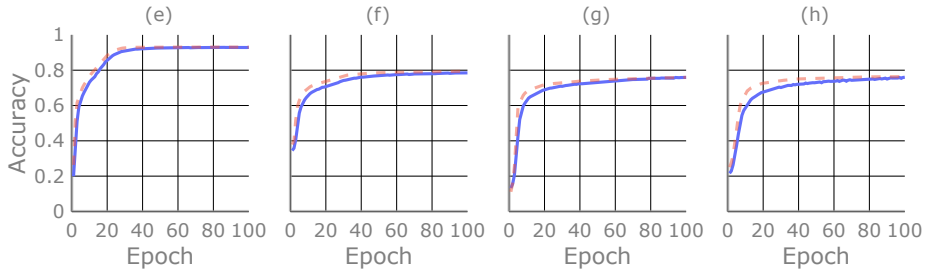
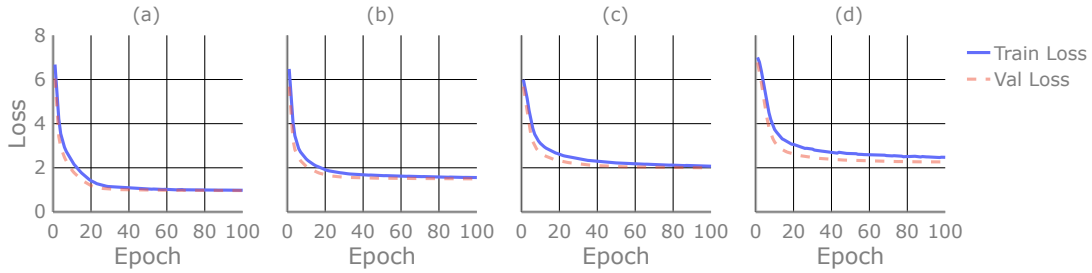


Figure 16: Transformer accuracy on Train/Validation set for different Sequence length and Output window variations.
a) Output window: 1, b) Output window: 4, c) Output window: 8, d) Output window: 12, e) Output window: 1, f) Output window: 8, g) Output window: 16, h) Output window: 24.

(1) Sequence Length: 16



(2) Sequence Length: 32

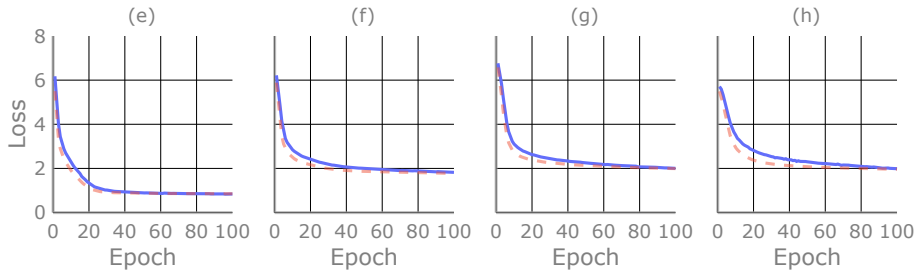


Figure 17: Transformer loss on Train/Validation set.
a) Output window: 1, b) Output window: 4, c) Output window: 8, d) Output window: 12, e) Output window: 1, f) Output window: 8, g) Output window: 16, h) Output window: 24.

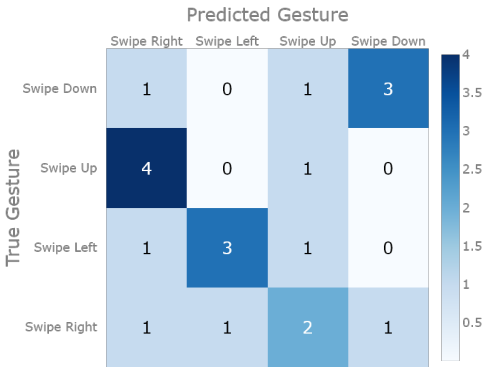


Figure 18: Classification Confusion Matrix for 'True Sequence'.

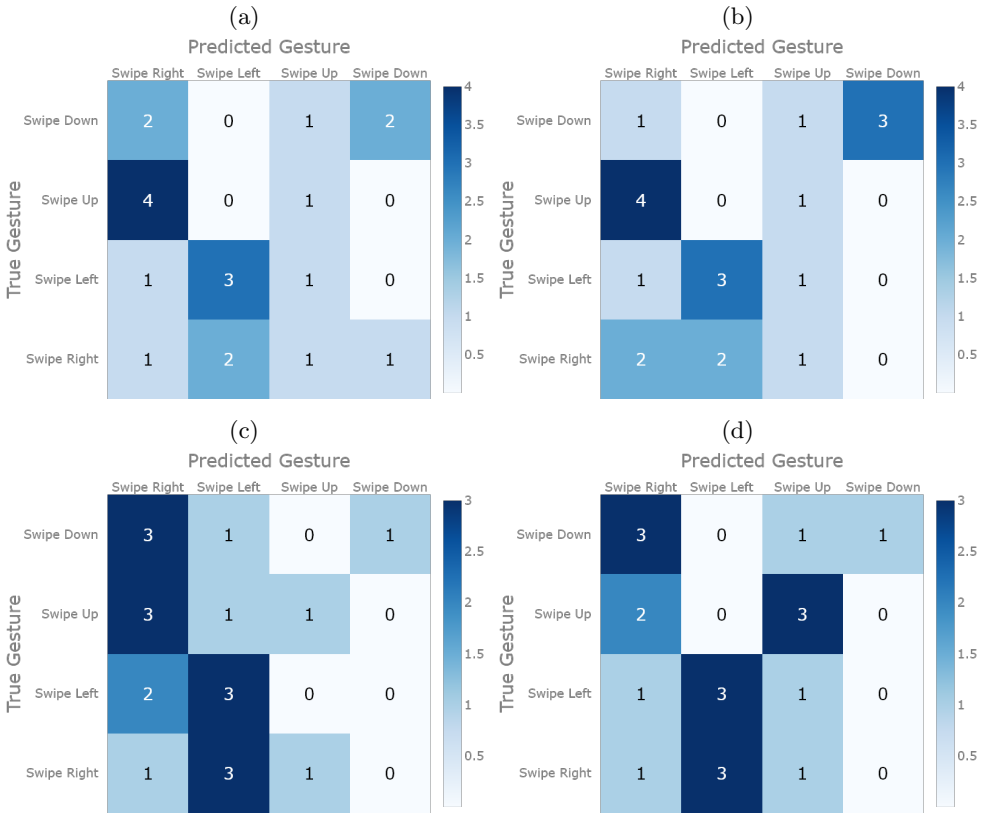
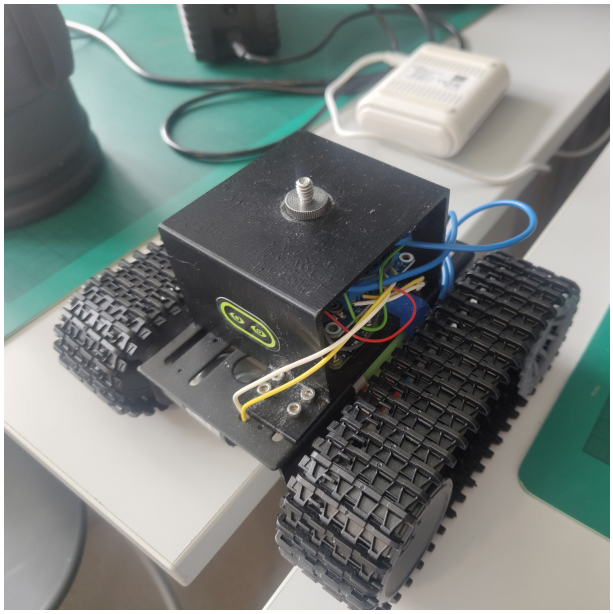


Figure 19: Classification Confusion Matrices for Different Models.
a) Model A, b) Model B, c) Model C, d) Model D

(a)



(b)

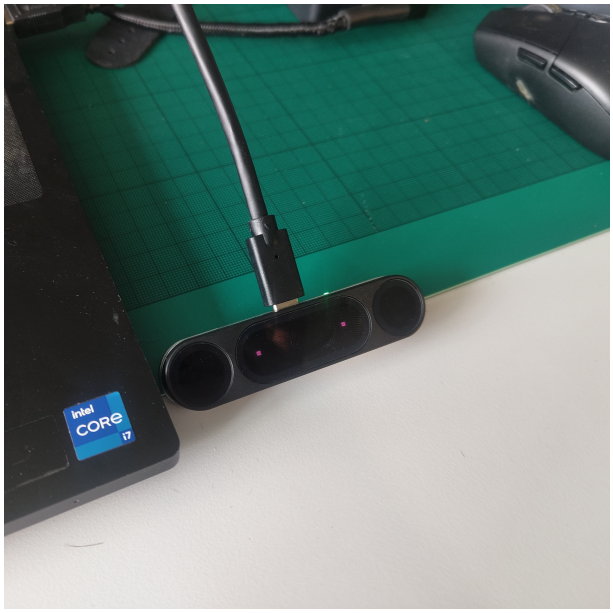


Figure 20: Pictures of resources used for demo. (a): Robot to be controlled. (b) ultraleap camera.