Available online at www.eccomasproceedia.org
Eccomas Proceedia SMART (2023) 1279-1290

ECCOMAS
Proceedia

X ECCOMAS Thematic Conference on Smart Structures and Materials
SMART 2023
D.A. Saravanos, A. Benjeddou, N. Chrysochoidis and T. Theodosiou (Eds)

HAND GESTURE RECOGNITION USING RECURRENT NEURAL NETWORKS AND SYNTHETIC DATA GENERATION

F. SABBARESE[*], L. MAGLIULO[*], P. CARRATU`[*], M. ROMANO[†]

[*] Youbiquo srl
Via de Balzico 50,
84013 Cava de' Tirreni, Italy
e-mail: info@youbiquo.eu, www.youbiquo.eu

[†] Università degli Studi Internazionali di Roma, Italy
Via Cristoforo Colombo, 200
00147 Roma, Italy
email: marco.romano@unint.eu, www.unint.eu

Abstract Interest in Extended Reality (XR) technologies has grown in recent years. Companies and researchers are focusing on the fields of applications and improving user interaction, with a particular focus on meta-user interface design. In this field, applications range from home automation to professional fields like surgery and manufacturing. Interaction can take place through various modalities, such as voice, touch, and hand gestures. Hand-gestural interaction has become more relevant in recent years, particularly due to the need for touchless interaction due to the COVID-19 pandemic. It is considered a natural interaction and allows users to feel present in the digital world through "direct manipulation". Recognizing hand gestures in real-time from video streams is difficult because it's hard to know when a gesture starts and ends. Scaling up recognition performance and the possibility of encountering unknown gestures also pose challenges. These challenges can impact the design of gestural interactions, which is closely connected to the effort needed for making XR systems recognize gestures and their precision, leading to a poor user experience. In this paper, we propose a real-time on-device Hand Gesture Recognition system that can be used in XR applications. It can handle static and dynamic gestures, with one or two hands, also considering egocentric perspective, making it usable with various devices, from expensive Smart Glasses to more affordable smartphones and laptops. The system uses well-known datasets, such as EgoGesture and Jester, and splits the gesture recognition task into two sub-tasks: identifying the hand skeleton of a human from a single RGB camera through Mediapipe Hands and then recognizing gestures using the detected hand skeleton. To extend the available datasets, we propose a procedure for generating large synthetic video datasets for hand gestures, as well as behavioural trees for generating variations of the acquired gestures. This approach saves time and effort spent on recording and annotating thousands of real videos, allowing greater flexibility in gesture design and envisioning XR applications involving intuitive and richer interactions, increasing user experience.

Key words: Hand gestures, Extended Reality, RNN, Synthetic data, Touchless interaction.

INTRODUCTION

The evolution of smart environments has significantly changed our interactions with the

Peer-review under responsibility of the organizing committee of SMART 2023.
doi: 10.7712/150123.9875.445104

surrounding environment through IoT services [1]. The design of the meta-user interface, which encompasses the interaction between humans and the environment, has become a critical research field in XR technologies, such as AR, VR, and MR. These technologies have promising applications in various fields such as surgery, manufacturing, and demotics, and hand-gestural interaction is one of the most promising interaction modalities for interacting with the digital world naturally.

Several hand gesture-based systems have been developed to control virtual objects and perform tasks [2]. This is particularly true in the fashion retail industry, where AR technology is at the forefront of the usage of fitting rooms and magic mirrors [3]. Hand gestures are considered the optimal choice to enhance the user experience and overcome issues related to the current sanitary situation. This is because users can interact with the digital world naturally and feel present through "direct manipulation" [2]. Additionally, hand gestures are critical in enabling individuals with visual impairments to access and interact with digital products [4].

The complexity and variability of human hand movements pose challenges in developing hand gesture recognition systems. In real-world scenarios, gestures are often performed continuously, making it difficult to determine when a gesture starts and ends. The diversity of hand shapes and movements makes it challenging to create a comprehensive dataset covering all possible hand gestures, impacting the design of gestural interactions, the recognition precision and user experience. Researchers have proposed various approaches for hand gestures recognition often based on machine learning or deep learning algorithms. These algorithms often require a large amount of labelled data to achieve good performances, which can be a limiting factor in many applications.

In this paper, we propose a real-time Hand Gesture Recognition system for XR applications that can recognize static and dynamic gestures from one or two hands, also considering egocentric perspective, usable in "frugal" devices such as smartphones and laptops with RGB cameras. The system improves upon existing ones by splitting the task into two sub-tasks, identifying hand skeletons with Mediapipe Hands and recognizing gestures from those skeletons using well-known datasets such as EgoGesture and Jester. Additionally, a procedure for generating synthetic video datasets for hand gestures is proposed, reducing the time and effort required to record and annotate real videos. The proposed method has been tested within the HandyTrack project, experiment 1107 of the FF4EuroHPC programme, a European initiative that helps facilitate access to all high-performance computing-related technologies for SMEs and thus increases the innovation potential of European industry.

The remainder of the paper is structured as follows. In the next section, we provide an overview of the literature on hand gesture recognition and the involved challenges. In the third section, we describe the proposed hand gesture recognition system and its various components. We then evaluate the proposed system performances both offline and online and we present an online benchmark procedure to assess the system's recognition accuracy. Finally, we conclude the article by summarizing our contributions and discussing future research directions in this area.

RELATED WORKS

In recent years, the growing use of deep learning has led to the use of Convolutional Neural Networks (CNN) for hand gesture recognition feature extraction. 2D-CNNs are limited in their

ability to extract only spatial information. To handle temporal information, 3D-CNNs or Recurrent Neural Networks (RNN), particularly Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU), can be used, which are capable of handling even long-term temporal dependencies. In [5], the responses of a 3D-CNN used on short video clips are averaged. In [6], LSTMs are used to handle the gesture recognition problem. A system based on the combination of CNNs and LSTMs is presented in [7], starting from a Leap Motion Controller. Instead, researchers at NVIDIA in [8] experimented with a multimodal system extending the concept of early detection in this context. The study [9] introduces a large-scale dataset and a challenge to which many researchers have contributed by producing many approaches to tackle the problem of gesture recognition starting from RGB images, including the one presented by the authors of the study. In other studies, [10], the problem of on-air writing is addressed using depth information and finger trajectories. Quantized depth is used in [11] to create high contrast between different key regions of the hands. The literature is also rich in skeleton-based approaches. In [12], authors proposed a spatio-temporal graph convolutional network (ST-GCN) for gesture recognition using skeletons.[13] utilizes a skeleton-based approach that combines self-attention mechanism. [14] addresses gesture recognition from 3D hand skeleton sequences using 3D-CNNs and LSTMs. An approach based on point cloud sequences is presented in [15], where a PointLSTM architecture is used to capture long-term spatial correlations, propagating information from the past to the future.

PROPOSED METHOD

To use the solution across a wide range of devices, the algorithm must run continuously, and consequently, it must be lightweight and energy efficient. While many solutions in the literature are based on complex and heavy architectures, our approach involves decomposing the problem into two simple tasks: i) *hand tracking over time, identifying landmarks;* ii) *gesture recognition based on landmarks*. The advantage is the ability to leverage not only the predicted gestures, but also the continuous tracking of the hands. In this section, we detail our system components and analyse the algorithm training procedure and datasets used.

Hand Tracking

Real-time and robust hand tracking is a challenging task in computer vision due to self-occlusion and lack high-contrast patterns. Most existing approaches rely on powerful computing environments for inference. Instead, we need a solution that can achieve real-time performance even on lightweight devices. To address this challenge, a ML-based solution such as Mediapipe Hands [16] is used to infer hand landmarks from a single frame or video sequence. The solution consists of a pipeline of algorithms: i) a *Palm Detector Model* that operates on the input images to detect a hand bounding box; ii) a *Hands Landmark Model* that operates on the region defined by the Palm Detector, to predict 21 3D landmarks (x, y, z) in screen space or world/metric space for each hand and a float scalar representing the handedness probability of the hand.

For the screen-space landmarks, the x and y are estimated with respect to the image plane and are normalized in range $[0.0, 1.0]$ according to the image size. The z coordinate represents the landmark depth with respect to the wrist landmark. A positive value indicates that the

landmark is located closer to the acquisition device, while a negative value indicates the opposite. Instead, for world/metric landmarks the coordinates are in meters with the origin at the hand's approximative geometric centre.

To use this solution, several configurations are required, including specifying whether to work with images or video sequences, the maximum number of detectable hands k, the complexity of the model (lite or full), and the minimum detection threshold th_d and tracking confidence th_t, which indicate respectively, when a hand detection or tracking is considered successful. These are hyperparameters in the design process.

Gesture Recognition

In this section, we formalize the approach followed for Gesture Recognition, analysing the procedure for determining the input to the model, its architecture and training.

Model Input

Dynamic gestures are executed over time, and therefore, there are n frames over which the gestures are performed. If the Hand Tracking module is applied to each frame for every time instant t, a series of landmarks is obtained. In our approach, we set m as the number of time instances on which a gesture can be performed, resulting in a temporal window where $m <= n$. By fixing the length of the temporal window, once m instances of the gesture are captured, the least recent ones are discarded, ensuring that there are always m instances of the gesture within the window. This allows for multiple samples to be extracted from the same performed gesture and used as input to the model, as depicted in Figure 1.

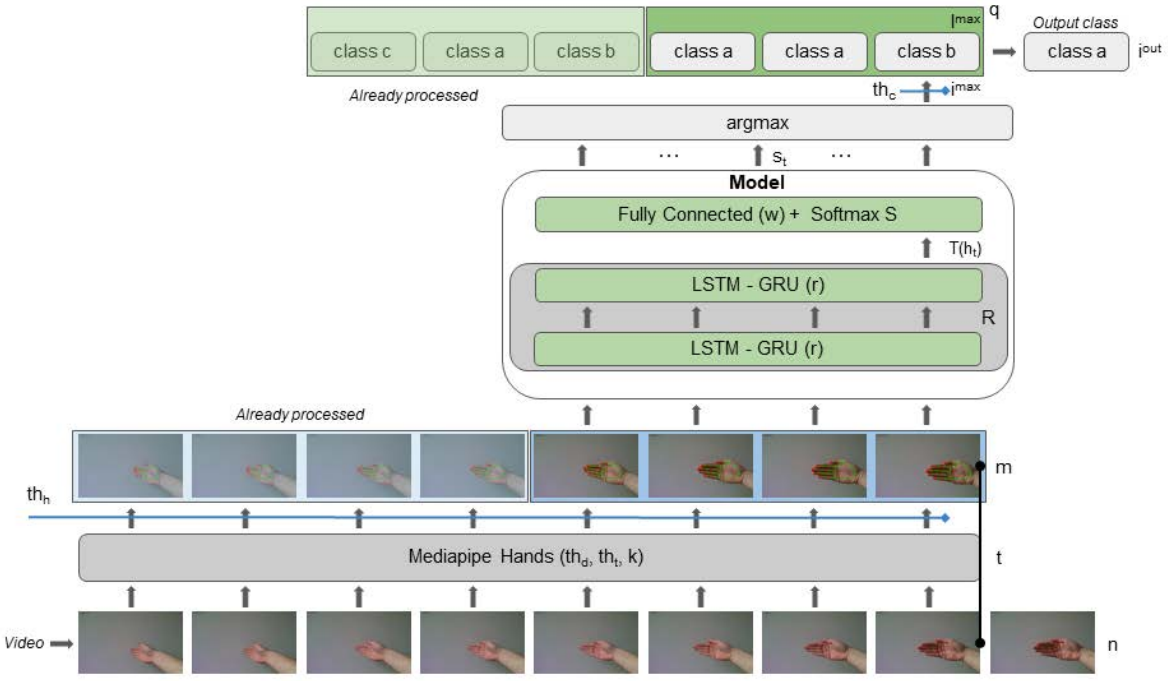

Figure 1 - The proposed architecture for Hand tracking and Gesture Recognition.

We check handedness score against a threshold th_h and label the landmarks accordingly, ensuring that the input to the model is ordered based on handedness at each instant of the

gesture. For gestures involving both hands, we choose to provide the model with the landmarks of the left hand first. If th_h is not exceeded or the landmarks are not detected by the Hand Tracking module at a particular instant, the input to the model is zero-filled to indicate the absence of that hand. Only when the landmarks of at least one hand are available, the input to the model at that instant is filled. Both th_h and m are hyperparameters of the system along with the number of hands.

Model Architecture

For Gesture Recognition, we propose a RNN illustrated in Figure 1. The architecture comprises two layers of either LSTM [17] or GRU [18] and a Fully Connected output layer with a Softmax activation to predict gestures probabilities. We found that adding additional layers or Bidirectional mode for recurrent layers made the model prone to overfitting or not lead to substantial increases in the evaluated metrics despite the added complexity.

Below we formalize the operations carried out by our model. We define the input to the model at time t as $C_t \in \Re^{m\,x\,k\,x\,l\,x\,d}$, with $m > 1$ time steps, $k = 1,2$ if the gestures are performed respectively with one hand or two hands, with $l = 21$ landmarks per hand, each with $d = 2,3$ dimensions (i.e., considering only x and y or also z). The first recurrent layer is configured to output a sequence, i.e. an output for each input time step, while the second to return the last output in the output sequence. Thus, starting from the input C_t the two stacked recurrent layers compute the output vector $h_t \in \Re^r$ as

$$h_t = T\big(R(C_t)\big) \tag{1}$$

with R: $\Re^{m\,x\,k\,x\,l\,x\,d} \rightarrow \Re^r$, which summarizes the operations performed by the recurrent layers, T the hyperbolic tangent activation function for which $T: \Re^r \rightarrow \Re^r$ and r that represents the number of recurrent units. Finally, the output and softmax layers transform h_t into class probabilities s_t of w classes:

$$s_t = S(W_s h_t + b) \tag{2}$$

with $W_s \in \Re^{w\,x\,r}$ representing the output weights, bias b and the Softmax function $S: \Re^w \rightarrow \Re^w_{[0,1]}$, where $[S(x)]_i = \frac{e^{x_i}}{\sum_k e^{x_k}}$ for each class in I.

Inference

Since $m <= n$, from one video sequence of a gesture we can determine multiple inputs to the model. This implies that p predictions of the same gesture can be obtained, with $p < n - m$, since the Hand Tracking module may not determine landmarks in some frame of the video sequence. To output a gesture, we first select from the set of classes I the most probable class i^{max} predicted by the Softmax output of the model as

$$i^{max} = \arg\max_{i \in I}(s_t) \tag{3}$$

then we compare the probability of the selected class against a confidence threshold th_c and output the null gesture if the threshold is not exceeded. The last q predictions are stored in I^{max}, with $q <= p$. We output the class i^{out} if this is the most present inside I^{max}, that is

$$i^{out} = argmax_{i \in I^{max}}(f(i)) \tag{4}$$

with $f(i) = \sum_{y \in I^{max}} (y == i)$.

For real-time prediction, the input to the system is a continuous video stream, and the Hand Tracking module is always enabled to detect hand landmarks. Gesture recognition is activated only when a series of m landmarks is detected, and therefore, only in these cases a gesture can be predicted. In all other cases, the null gesture is added among the last q predictions, so that it is output as i^{out}.

The hyperparameters q and th_c are optimized during the model deployment phase to ensure the robustness of gesture recognition.

Training Details

To train the model we use Categorical Cross Entropy [19] as loss function and either the Adam [20] or RMSProp [21] optimizer. In our experiments, we observed that the RMSProp optimizer allowed for faster convergence and lower loss values with a batch size of 256. The starting learning rate is tuned during experiments and adjusted using ReduceLROnPlateau, a scheduling technique that decreases the learning rate when validation set loss stops improving for longer than the patience number allows. To prevent overfitting, we employed regularization techniques such as adding Dropout [22] after recurrent layers with a rate of 50%, EarlyStopping [23] to training when no improvement is observed on the validation set loss and shuffling the data before each training epoch. We evaluate regularization with $L1$ and $L2$ norms, as well as their combination, but did not observe a positive impact. To address class imbalance, we consider sample weights, computing weights to use alongside the loss function based on the number of samples for each class in the training set. However, this technique did not yield substantial benefits. No pre-training is used to initialize model weights. Glorot/Xavier Uniform [24] initialization is used for input weights of recurrent layers and the output layer, while Orthogonal one is used for recurrent states.

Knowledge Distillation and Pruning

To reduce the size of the model, preserving its structure, we use the Knowledge Distillation [25]. This enables us to transfer the knowledge from a more complex model to a lighter one. To achieve this, we use the Kullback-Leibler Divergence to minimize the distance between the output distribution of the student model and of the teacher model. We employ a factor denoted as α to balance the distillation loss and the one derived from the training data. α is set to values in $[0.1, 0.3]$, indicating low dependence on the data.

To further improve latency, we applied weight pruning by gradually setting the weights to zero based on their magnitude to obtain model sparsity. This method involves an additional training phase during where the model is trained to achieve the desired sparsity over a certain number of epochs.

Implementation Details

Although other input modes are available for some of the dataset, in our work we use only the RGB images in input to the system.

The model is trained in two different environments: i) an ordinary laptop configured with an Intel Core i7-1065G7 CPU and 16 GB of RAM, without a GPU; ii) the High-Performance Computing (HPC) Marconi100 by CINECA, configured with an IBM POWER9 AC922, 128 GB of RAM and a NVIDIA Volta V100 GPU with 16 GB of VRAM. In the laptop case, we use the TensorFlow and Keras frameworks, along with Horovod to distribute the training over the HPC. To deploy models on low-computational-capability devices, we perform a conversion

to TensorFlow Lite format and apply Dynamic Range post-training quantization with a $8-bits$ precision, to reduce the model size and improve latency. After quantization and conversion, the model is re-evaluated on the test data, typically showing a reduced impact on its predictive capabilities in most cases.

Dataset

Several datasets of dynamic gestures are available, varying in terms of gesture complexity, labels, actors, acquisition sensors, and perspective (egocentric or third person). However, as highlighted in [9], large-scale datasets from an egocentric perspective are limited. Our work is evaluated on three different datasets with both perspectives: Jester [9], EgoGesture [26] and Synthetic. The datasets are split into 3 parts: i) *training set* for model fitting; ii) *validation set* for unbiased model evaluation and hyperparameter tuning during the training; iii) *test set* for final unbiased evaluation of the trained model. Details of the datasets are provided in the following sections.

Jester

This is a real-world large-scale dataset. It includes $148{,}092$ RGB clips with a third-person perspective, a duration of 3 seconds at 12 frames per seconds with the height fixed at 100 pixels and a variable width. Each clip contains a gesture annotation from a set of 27 gestures. The dataset consists of $1{,}376$ actors performing the gestures in their home environment. The average number of clips for each person is 43. We adopt the $80{:}\,10{:}\,10$ split as in [9].

EgoGesture

The dataset comprises $24{,}161$ samples acquired from the top of the subject's head in an egocentric perspective using a RealSense SR300 sensor in RGB and depth modes with a resolution of 640 x 480 and 30 frames per seconds. It includes 83 classes of gestures performed by 50 distinct subjects in 6 diverse scenes, with the aim to cover most kinds of manipulation and communication operations on wearable devices. As in [26], the sample split is $60{:}\,20{:}\,20$ and the division is random and based on subjects.

Synthetic

The Synthetic Dataset is generated through a flexible application developed at Youbiquo. The application can reproduce not only the same gestures but also variations of them through behavioural trees. The landmarks are mapped onto 3D hand models, rendered with different backgrounds, variations of position, distance, and viewpoint rotations. The rendered frames can be exported in different modes: RGB, depth, silhouette, json (which contains the position and rotation of each joint of the hands). The sensor used in the acquisition phase is a RealSense D455 with a resolution of $640\ x\ 480$, at 60 frames per second. The same resolution and frames per second are used for the generation phase. Ten acquisitions are made for each gesture, varying the position in space and handedness, with the option to configure the number of reproductions per acquisition. Four gestures from the EgoGesture dataset are selected for the generated dataset in this experiment, with a reproduction factor of 50. A random split of $70{:}\,20{:}\,10$ is applied for each label.

Figure 2 – Sample of gestures from Jester, Ego Gesture, Synthetic dataset.

EXPERIMENTAL RESULTS

We evaluate our system's performance in both offline and online scenarios. For the latter, we introduce an online benchmark procedure to assess the system's recognition accuracy in real-world contexts.

Offline Evaluation

The offline evaluation of the system is performed on the test set. In Table 1, we compare the results obtained from our models with those obtained by some of the 59 participants in the Jester Challenge [9], where accuracy values range from 68% to 97%. For all experiments, d is fixed at 3. Some gestures can be recognized even if they are made with two hands at the same time. Therefore, k could be fixed at 2.We also experimented with a subset of the classes available in the dataset. w is set to the number of classes used.

Table 1: Accuracy of Jester Challenge models vs models of the experiment.

Model	Acc
RFEEN, 20 Crops	97.06%
3D ResNet 101	85.99%
Our, $w = 27, m = 16, k = 2, th_d = 0.72, th_t = 0.72, th_h = 0.85,$ $layers = LSTM, units = 16, lr = 0.001$	85.63%
Our, $w = 27, m = 16, k = 2, th_d = 0.72, th_t = 0.72, th_h = 0.85,$ $layers = LSTM, units = 64, lr = 0.0001$	87.84%
Our, $w = 27, m = 16, k = 2, th_d = 0.72, th_t = 0.72, th_h = 0.85,$ $layers = GRU, units = 64, lr = 0.0001$	88.65%
Our, $w = 15, m = 21, k = 2, th_d = 0.70, th_t = 0.65, th_h = 0.85,$ $layers = LSTM, units = 16, lr = 0.0001$	92.78%
Our, $w = 10, m = 16, k = 2, th_d = 0.80, th_t = 0.75, th_h = 0.90,$ $layers = LSTM, units = 32, lr = 0.0001$	93.00%

In the case of EgoGesture dataset, we compare, in Table 2**,** our models and those in [26]. Specifically, we consider only the cross-subject case, since the test set and the training set are disjoint and using only RGB frames.

Table 2: Accuracy of EgoGesture models vs models of the experiment.

Model	**Acc**
VGG16 fc6	62.50%
C3D fc6, 16 frames	86.40%
Our, $w = 83$, $m = 16$, $k = 1$, $th_d = 0.60$, $th_t = 0.60$, $th_h = 0.71$, $layers = LSTM$, $units = 64$, $lr = 0.0005$	71.31%
Our, $w = 83$, $m = 16$, $k = 1$, $th_d = 0.60$, $th_t = 0.60$, $th_h = 0.71$, $layers = LSTM$, $units = 128$, $lr = 0.0005$	72.90%
Our, $w = 18$, $m = 11$, $k = 1$, $th_d = 0.65$, $th_t = 0.65$, $th_h = 0.75$, $layers = LSTM$, $units = 22$, $lr = 0.0005$	90.92%

We present a subset of the experiments conducted on both datasets, achieving results comparable to the literature, but with a lighter architecture. Reducing the number of classes in the training set can increase accuracy, while the number of units in the layers needs to be increased when the dataset becomes more complex. Time windows ranging from 15 to 25 are tested on the Jester dataset, while values between 10 and 18 are used for the EgoGesture and Synthetic datasets. Higher values for th_d, th_t and th_h can improve accuracy, but for the more complex EgoGesture, these values must be kept low. In most cases, switching from LSTM to GRU layers results in a slight performance variation. By applying Knowledge Distillation and then Pruning to the same trained models, the result is a smaller model, but approximately the same precision.

We compare training times and accuracy for the Synthetic dataset on a laptop and Marconi100 HPC to highlight the need for increased computing power when dealing with complex models and large batch sizes, especially considering the possibility of generating a potentially unlimited dataset through our procedure. The Table 3 shows the results.

Table 3: Accuracy and training times for the same model trained on the synthetic data and different computers.

Model	**Acc**	**Time**
Laptop, $w = 4$, $m = 10$, $k = 1$, $th_d = 0.70$, $th_t = 0.70$, $th_h = 0.70$, $layers = LSTM$, $units = 128$, $batch\ size = 512$, $lr = 0.00005$	95.00%	$2h, 35m, 14s$
HPC, $w = 4$, $m = 10$, $k = 1$, $th_d = 0.70$, $th_t = 0.70$, $th_h = 0.70$, $layers = LSTM$, $units = 128$, $batch\ size = 512$, $lr = 0.00005$	95.70%	$0h, 30m, 56s$

Online Evaluation

Benchmark Evaluation

The benchmark involves evaluating the accuracy of 10 inferences for a set of 5 gestures selected from the Jester dataset and performed 5 times with both the hands. The experiments are conducted on two different devices: the same laptop used for training and the Smartglass Leonardo supplied by Youbiquo. Measurements are taken for 3 different models trained on the same dataset ($w = 10$, $m = 10$, $k = 2$, $d = 3$, $th_d = 0.70$, $th_t = 0.65$, $th_h = 0.90$), all with 32 units in hidden layers 256 as batch size, 0.0001 as learning rate and different initialization. The offline accuracy for the first model is approximately 93%, around 91% for the second, and 92% for the third. Furthermore, the confidence threshold for gesture prediction th_c and the size of the set of last q predictions are set at 0.80 and 15, respectively. The Table 4 shows the measurements in terms of accuracy on the presented benchmark.

Table 4: Accuracy of three different models on two different devices.

Device	Model 1 - LSTM	Model 2 - GRU	Model 3 - LSTM	Avg – Std
Ordinary Laptop	90%	94%	84%	$89.33 \pm 4.16\%$
Leonardo	94%	88%	96%	$92.66 \pm 5.03\%$

Fps Evaluation

Below we provide an estimation of the frames per second (fps) at which the system can run on 3 different devices: the laptop used for training, the SmartGlass Leonardo provided by Youbiquo and the OnePlus 9 Pro 5G smartphone. Results are reported under two different conditions, when only Mediapipe Hands is running and when gesture recognition implemented using Model 1 - LSTM is overlaid. The Table 5 reports all obtained results.

Table 5: FPS of a trained model running on different devices.

Device	Mediapipe Hands	Mediapipe Hands + Gesture Recognition
Ordinary Laptop	24.13 ± 2.46	21.32 ± 2.57
Leonardo	21.01 ± 2.85	20.06 ± 2.35
OnePlus 9 Pro 5G	30.13 ± 3.38	29.63 ± 4.07

CONCLUSIONS

In this paper, we present our solution for hand tracking and gesture recognition that can be seamlessly integrated into XR systems. One of the key strengths of our approach is its flexibility. The ability to generate datasets and extend existing ones allows for further improvement of the solution and enables its deployment in contexts where a dedicated set of gestures may not yet be available. Furthermore, the potential to integrate other input modalities,

such as depth images, serves as a foundation for future studies, expanding the possibilities of our solution.

REFERENCES

[1] J. Coutaz. Meta-User Interfaces for Ambient Spaces. *In International Workshop on Task Models and Diagrams for Users Interface Design; Springer: Singapore*, 2007; pp. 1–15.

[2] Battistoni, P., Di Gregorio, M., Romano, M., Sebillo, M., Vitiello, G., & Brancaccio, A. (2022). Interaction design patterns for augmented reality fitting rooms. *Sensors, 22(3),* 982.

[3] Javornik, A.; Rogers, Y.; Moutinho, A.M.; Freeman, R. Revealing the shopper experience of using a 'magic mirror' augmented reality make-up application. *In Proceedings of the ACM Conference on Designing Interactive Systems, Brisbane, Australia, 4–8 June 2016*; Volume 2016, pp. 871–882.

[4] Romano, M., Bellucci, A., & Aedo, I. (2015). Understanding touch and motion gestures for blind people on mobile devices. In *Human-Computer Interaction–INTERACT 2015: 15th IFIP TC 13 International Conference,* Bamberg, Germany, September 14-18, 2015, Proceedings, Part I 15 (pp. 38-46). Springer International Publishing.

[5] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri; Learning Spatiotemporal Features with 3D Convolutional Networks; In *Proceedings of the 2015 IEEE International Conference on Computer Vision;* 2015; pp. 4489-4497.

[6] Girdhar R., Ramanan D., Gupta A., Sivic J., Russell B. Actionvlad: Learning spatio-temporal aggregation for action classification . In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* . 2017 . pp. 971–980.

[7] Wang L., Xiong Y., Wang Z., Qiao Y., Lin D., Tang X., Van Gool L. Temporal segment networks . Towards good practices for deep action recognition . *Proceedings of the European Conference on Computer Vision* . 2016 . pp. 20–36

[8] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, J. Kautz . Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. 2016. pp. 4207-4215.

[9] J. Materzynska, G. Berger, I. Bax, R. Memisevic. The jester dataset: A large-scale video dataset of human gestures. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision,* pp. 2874-2882.

[10] Mahmud, H., Islam, R. & Hasan, M.K. On-air English Capital Alphabet (ECA) recognition using depth information. The Visual Computer 38, 1015–1025 (2022).

[11] H. Mahmud, M. M. Morshed, M. Hasan. A deep-learning–based multimodal depth-aware dynamic hand gesture recognition system. *Computer Vision and Pattern Recognition*. https://doi.org/10.48550/arXiv.2107.02543

[12] S.Yan, Y. Xiong, D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *Thirty-second AAAI conference on artificial intelligence* (2018)

[13] Y. Chen, L. Zhao, X. Peng, J. Yuan, D.N. Metaxas (2020). Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. *Paper presented at 30th British Machine Vision Conference*, BMVC 2019, Cardiff, United Kingdom.

[14] J.C. Núñez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, J. F. Vélez. Convolutional Neural Networks and Long Short-Term Memory for skeleton-based human activity and hand gesture recognition. *Pattern Recognition* (Volume 76). April 2018. pp 80-94

[15] Y. Min, Y. Zhang, X. Chai and X. Chen. An Efficient PointLSTM for Point Clouds Based Gesture Recognition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. pp. 5760-5769.

[16] F. Zhang. MediaPipe Hands: On-device Real-time Hand Tracking. *arXiv e-prints*, 2020. doi:10.48550/arXiv.2006.10214.

[17] S. Hochreiter , J. Schmidhuber. LONG SHORT-TERM MEMORY. Neural Computation 9(8):1735-1780, 1997.

[18] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Doi:10.48550/arXiv.1406.1078

[19] S. Mannor, D. Peleg, R. Rubinstein. The cross entropy method for classification. In *Proceedings of the 22nd international conference on Machine learning*. (2005, August). pp. 561-568.

[20] D.P. Kingma, J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. (2014).

[21] T. Tieleman, G. Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. University of Toronto, Technical Report, 2012.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15(1)*. 2014. Pp. 1929-1958.

[23] L. Prechelt. Early Stopping — But When?. *In: G. Montavon, G.B. Orr, K.R. Müller, (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. (2012)*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_5

[24] X. Glorot, Y. Bengio, (2010, March). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256). JMLR Workshop and Conference Proceedings.

[25] G. Hinton, O. Vinyals, J. Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[26] Y. Zhang, C. Cao, J. Cheng, H.Lu. Egogesture: a new dataset and benchmark for egocentric hand gesture recognition. *IEEE Transactions on Multimedia, 20(5)*. 2018. 1038-1050.