RESEARCH ARTICLE

SPECIAL ISSUE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
Vol. 05, Issue 05S May
www.rspsciencehub.com

http://dx.doi.org/10.47392/irjash.2023.S052

International Conference on intelligent COMPUting TEchnologies and Research (i-COMPUTER) 2023

Real-Time Computer Vision Based Hand Gesture Recognition

Rishav Nath Pati [1], Nagalakshmi Vallabhaneni [2], Prabhavathy P [3], Yash Shekhawat [1], Srijan Paria [1]

[1] *School of Information Technology and Engineering,Vellore Institute of Technology,Vellore, India.*
[2] *Assistant Professor, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India.*
[3] *Associate Professor, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India.*

Emails: rishavnath.pati2022@vitstudent.ac.in, nagalakshmi.v@vit.ac.in, pprabhavathy@vit.ac.in, yash.shekhawat2022@vitstudent.ac.in, srijan.paria2022@vitstudent.ac.in

Article History

Received: 26 February 2023
Accepted: 20 March 2023

Keywords:

Hand gesture recognition;
Palm landmark point;
Static gestures;
Dynamic gestures;
MediaPipe-Hands;
Key-Point based Classification;
Pont-History based Classification;
LSTM

Abstract

The ability to recognize the shape and movement of hands can help improve the user experience in a wide range of technical domains and platforms. It can help you understand sign language and move your hands in the right way, for example. It can also make it possible for digital information and materials to be added on top of the real world in augmented reality. Here, I talk about a real-time, on-device hand gesture recognition solution that lets us control our system's graphical user interface (GUI) with static and dynamic hand gestures that can be trained to do a set of actions that are similar to what we do with our mouse and keyboard. It is built with MediaPipe-Hands, which finds the palm landmark point. The data is then sent through a pipeline of data-preprocessing functions and trained with two models: one for static gesture recognition and one for dynamic gesture recognition. In real-time, the models are then used to detect similar gestures on-device from a video-capturing device like a webcam.

1. Introduction

To enhance the user experience in various technical fields and platforms, recognizing hand shape and motion is crucial. It can be used for augmented reality, sign language, and hand gesture control. We propose a real-time hand gesture recognition system using the MediaPipe-Hands API and a deep learning model, which eliminates the need for specialized equipment. This system works on personal computers and mobile devices, only requiring a webcam or mobile camera. Our approach addresses the previous limitations of specialized hardware or computationally intensive methods for real-time execution on mobile devices.

Our hand gesture recognition system detects the hand region from the input video stream using the MediaPipe API and tracks the hand from frame to frame to reduce performance overhead. For accuracy, a hand landmark model is provided with a properly cropped image of the hand to focus on precise coordinate prediction. To decrease data augmentation, the previously-identified hands are used to create the crops. If the landmark model is unable to recognize the hand, palm detection is used to locate it again. We implemented the hand landmark tracking as a MediaPipe graph, with a specialized Hand-Renderer subgraph handling the rendering. The palm detection module employs both a hand landmark subgraph and a palm detection subgraph.

The output from the MediaPipe API, which is in the form of landmark points, is analyzed and recorded in a CSV file for training for a specific gesture. Once trained with several gestures, the output model can detect the trained gestures from any simple video feed in real-time. The remaining sections of this paper are structured as follows. Section II presents related studies, followed by Section III providing an overview of the methodology. In subsequent sections, Section IV and Section V, we present the detailed framework and experiment results, respectively. Lastly, in Section VI, we present the conclusion summarizing the entire research.

2. Related Studies

2.1. Hand Shape based methodology

One approach to recognizing dynamic sign language involves analyzing the properties of hand forms and motion trajectory, which is a common practice. Some researchers, like Kim et al. (Kim et al.), have focused on hand shape characteristics to recognize fingerspelling. However, this method is limited to simple motions, such as alphabets and numerals, and cannot identify more complex gestures without considering hand motion. To overcome this limitation, local features from depth and intensity images are learned using the unsupervised deep learning method PCANet. A linear support vector machine classifier is then used to recognize the extracted features (S. Aly et al.). The pattern recognition system transforms an image into a feature vector and compares the feature vectors of a training set of gestures (Maung).

Other researchers, such as Haroon et al. (Haroon et al.), have proposed using artificial neural networks to recognize gestures with symmetric patterns under varying illumination conditions. Meanwhile, Mohandes et al. (Mohandes, Deriche, J. Liu, et al.) used long-short-term memory to detect hand motions based solely on hand motion trajectory. Additional research has classified hand movements using sensor technology such as the leap motion controller (Sonawane et al.), digital gloves, surface electromyography accelerometers, and gyroscopes (K. Li, Z. Zhou, Lee, et al.)– (Ma et al.). However, these methods are limited to specific hand movements like waving and gesticulating. Kumar et al. (Kumar et al.) presented a multimodal framework for detecting sign language using Kinect sensors (K. Lai, Konrad, Ishwar, et al.), while Wang et al. (H. Wang, Chai, and Chen) used sparse observation to recognize sign language based on hand postures and motions with RGB-D data. Both have successfully conducted research in this field. However, sensor-based systems are not practical or user-friendly. As the number of sign language vocabularies increases, sensor-based systems become more complicated. Recognizing the entire sign language vocabulary requires analyzing hand form details and motion trajectory, which can be challenging for users since it often requires sensors. Additionally, understanding dynamic sign language is difficult due to the complex and changeable motion trajectory driven by hand-body joint interactions.

2.2. Continuous Frame based methodology

To recognize sign language, researchers have explored various approaches that do not require sensors, such as using advanced algorithms to analyze video sequence characteristics. For example, Cui et al. (Cui, H. Liu, C. Zhang, et al.) developed a recurrent convolutional neural network for continuous sign language recognition using video sequences, and Huang et al. (Huang et al.) proposed a hierarchical attention network framework for global-local video feature representations.

Some researchers have also used deep learning to achieve dynamic sign language recognition, including convolutional neural networks (CNNs) to extract features from hand gestures (Krizhevsky, Sutskever, Hinton, et al. Donahue et al.), recurrent neural networks (RNNs) to learn video sequences (Maraqa, Abu-Zaiter, et al. Murakami, Taguchi, et al.), (Srivastava, Mansimov, Salakhudinov, et al.), and combined CNNs and RNNs to learn spatiotemporal sequence features (Baccouche et al. Ng et al.). Other techniques include fuzzy classification (J. Li and D. Zhang), self-supervised contrastive pre-training (X. Zhang et al.), neural network classification (Kang, Tripathi, Nguyen, et al.), and sequence-to-sequence learning (Liao et al.). Compared to methods relying on hand forms and motion trajectories, these video-based approaches have shown higher performance in detecting dynamic sign language. However, most of these methods were not specifically designed for sign language recognition and may struggle to identify complex sign language with varied hand forms and motion trajectories.

Although some researchers have combined motion information into static photographs (Köpüklü, Köse, Rigoll, et al.), this technique has limited utility since not all hand motion information can be captured in static images. Another study (Z. Liu et al.) proposed a spotting detection-based paradigm for large-scale continuous gesture recognition.

3. Methodology

3.1. Dataset

The dataset used in this study consists of hand gestures captured using MediaPipe Hands, a technology that detects and tracks key points, or landmarks, of a hand in an image or video. The dataset comprises 21 landmark points that are identified for each hand gesture, including gestures like fist, wave, and thumbs up. Each gesture has over 1000 recorded instances, each with their respective landmark points, resulting in a significant amount of data that can be utilized for training and evaluating hand gesture recognition models. Moreover, new datasets can be created very quickly on the fly, and the model can be retrained within seconds, making it a highly flexible and efficient system.

3.2. Experimental Procedure

Our hand tracking solution employs a machine learning pipeline that consists of multiple models working in tandem. The first model, BlazePalm (Amangeldy et al.), is a palm detector that operates on the entire image and produces a bounding box around the hand. This results in a more precise image of the hand, thereby reducing the need for data augmentation techniques, such as rotation, translation, and scaling. This allows the network to focus on improving its precision.

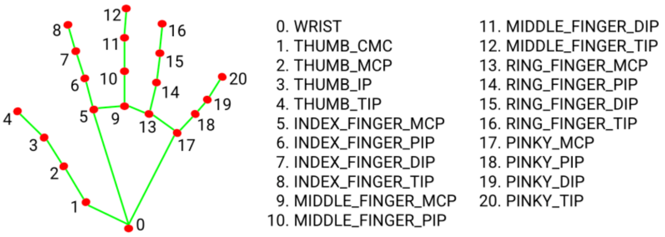


FIGURE 1. Landmark Points

The second model is a hand landmark model that works on the cropped image region defined by the palm detector and produces high-fidelity 3D hand landmark keypoints.

When the palm prediction suggests that the hand is lost, a single-shot detector model designed for mobile, real-time applications similar to BlazeFace is used to identify initial hand positions. This model can identify occluded and self-occluded hands while working with a wide range of hand sizes, which is challenging because hands do not exhibit high-contrast patterns.

The hand landmark model uses regression to locate 21 2.5D locations inside the identified hand areas. Even when the hand is partially visible or occluded, the model learns a consistent internal hand posture representation and is resilient. It has three outputs: the x and y coordinates of the landmark points, a hand flag that indicates whether or not there is a hand in the input image, and a binary classification of handedness, such as ”left” and ”right.”

Finally, there are two types of gesture recognition models used. The first model is a keypoint-based model used for static gestures, where the relative landmark points remain the same throughout. The second model is a point-history-based gesture, where the gesture is dynamic and consists of a series of changing relative landmark points.

4. Model Architecture

4.1. Keypoint-based model architecture

The described model is a feedforward neural network (FFNN) that utilizes fully connected layers. The model comprises a sequence of layers, where the output of each layer serves as the input for the subsequent layer. This neural network is also known as a dense neural network or a multilayer perceptron (MLP). The input to the model is a 1D array of length 42, and the output is a probability distribution over the output classes.

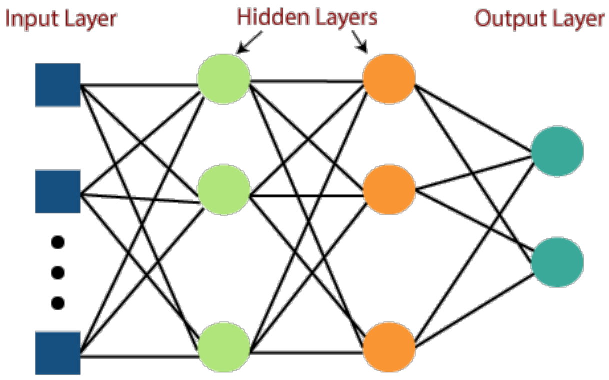


FIGURE 2. A Multilayer Perceptron

The architecture of the model is sequential and takes preprocessed data as input. The first layer is an

Input layer that takes input of shape (21 * 2), representing 21 landmark points with (x, y) coordinates. The second layer is a Dropout layer with a rate of 0.2, which randomly sets 20% of the input units to 0 during each training batch to avoid overfitting.

The third layer is a Dense layer with 20 units and a rectified linear unit (ReLU) activation function that applies a ReLU activation function to the output of the previous layer. The fourth layer is another Dropout layer with a rate of 0.4, which randomly sets 40% of the input units to 0 during each training batch.

The fifth layer is another Dense layer with 10 units and a ReLU activation function. Finally, the output layer is a Dense layer with N units and a softmax activation function.

The softmax activation function in the output layer is particularly suitable for multi-class classification problems. It transforms the output of the previous layer, which is a vector of arbitrary values, into a probability distribution over the classes. The softmax function normalizes the output values such that they add up to 1, ensuring that the output values can be interpreted as probabilities. The highest probability value corresponds to the predicted class for a given input sequence. Therefore, the softmax activation function is used to produce a probability distribution over the output classes and to make the final prediction. The model architecture is designed for a classification problem with N output classes.

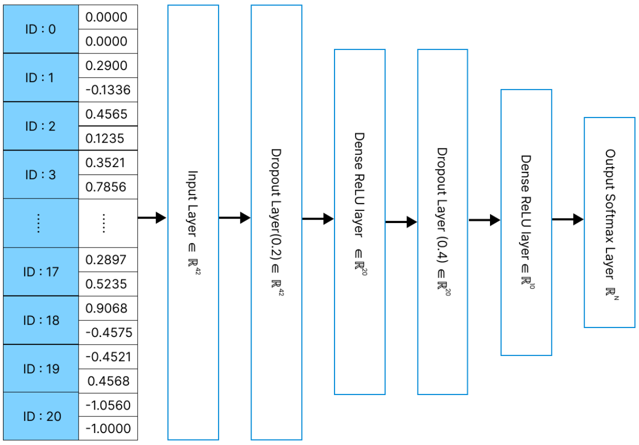


FIGURE 3. **Keypoint based model architecture**

In this model, the MediaPipe Hands API is used to obtain landmark coordinates of a hand from a video sequence. Each frame of the video contains a set of 21 landmark coordinates, which correspond to key points on the hand, such as fingertips, knuckles, and the wrist. The raw landmark coordinates are then processed using a 4-stage preprocessing pipeline to transform them into a suitable format for the model to learn from.

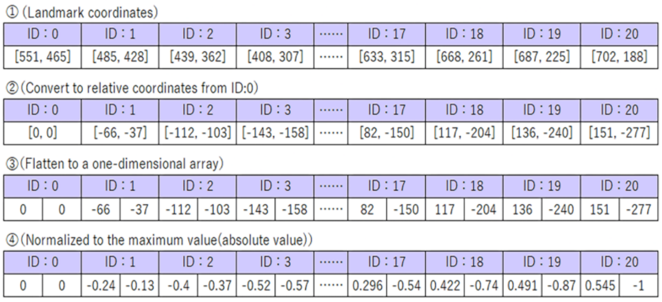
In the first stage of preprocessing, the landmark coordinates of each frame are recorded and stored in a 2D array of shape (21, 2). Each row of the array represents the x and y coordinates of a single landmark point. This step helps to capture the spatial information of the hand gestures and represents the hand pose in a structured format that can be used as input to the model.

In the second stage of pre-processing, the coordinates are converted into relative coordinates with respect to a reference point, which is typically the wrist landmark point with index value 0. This transformation helps to eliminate the effect of translation of the hand in the video. Specifically, the x and y coordinates of each landmark point are subtracted from the x and y coordinates of the wrist landmark point, respectively. This step helps to normalize the position of the hand and makes the model more invariant to hand movement in the video.

In the third stage of pre-processing, the relative coordinates are flattened into a 1D array of length 21 * 2, where each element in the array represents a single coordinate value. This step helps to create a fixed-length input representation for the model to learn from. By flattening the 2D array, the model can treat each coordinate value as a separate input feature, which makes it easier to learn complex relationships between the input and output.

In the final stage of preprocessing, the values in the flattened array are normalized to the maximum absolute value. Normalizing the data helps to reduce the effect of scaling in the input data, which can improve the performance of the model. Specifically, by scaling the values to lie between -1 and 1, the model can learn more efficiently and avoid issues such as vanishing gradients.

Overall, the data preprocessing steps help to transform the raw landmark coordinates into a suitable format for the model to learn from. The resulting input representation captures the key features of the hand gestures and provides a suitable input for the model to learn from. This approach has been shown to be effective in improving the performance of hand gesture recognition models, and it can be extended to other applications such as sign language recognition and human-robot interaction.



① (Landmark coordinates)

ID : 0	ID : 1	ID : 2	ID : 3	......	ID : 17	ID : 18	ID : 19	ID : 20
[551, 465]	[485, 428]	[439, 362]	[408, 307]	......	[633, 315]	[668, 261]	[687, 225]	[702, 188]

②(Convert to relative coordinates from ID:0)

ID : 0	ID : 1	ID : 2	ID : 3	......	ID : 17	ID : 18	ID : 19	ID : 20
[0, 0]	[-66, -37]	[-112, -103]	[-143, -158]	......	[82, -150]	[117, -204]	[136, -240]	[151, -277]

③(Flatten to a one-dimensional array)

ID : 0		ID : 1		ID : 2		ID : 3		......	ID : 17		ID : 18		ID : 19		ID : 20	
0	0	-66	-37	-112	-103	-143	-158	......	82	-150	117	-204	136	-240	151	-277

④(Normalized to the maximum value(absolute value))

ID : 0		ID : 1		ID : 2		ID : 3		......	ID : 17		ID : 18		ID : 19		ID : 20	
0	0	-0.24	-0.13	-0.4	-0.37	-0.52	-0.57	......	0.296	-0.54	0.422	-0.74	0.491	-0.87	0.545	-1

FIGURE 4. Point-History based model architecture

The model described below is a type of neural network known as a Sequential model. It is commonly used for processing sequential data, such as time series, natural language text, and speech recognition. The architecture of this model includes several layers, each with a specific function in processing the input data.

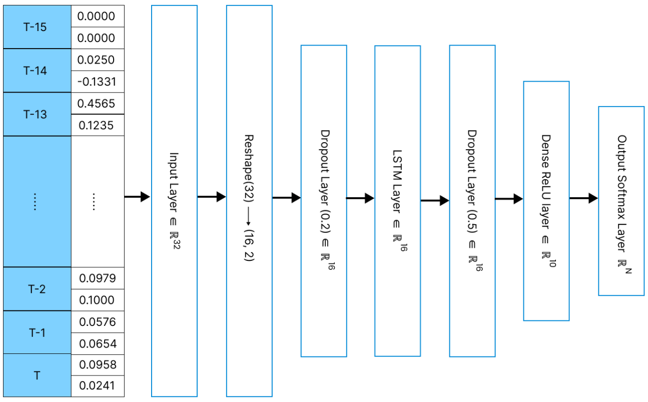


FIGURE 5. Point history based model architecture

The Input layer is the first layer of the model, which expects an input of shape (TIME_STEPS * DIMENSION,). Here, TIME_STEPS represents the number of time steps or intervals over which data is collected and processed for each instance, while DIMENSION represents the number of features in the input data. In this model, DIMENSION is set to 2, which corresponds to the x and y coordinates of each landmark point in the input data.

The Reshape layer is used to convert the 1D input array into a 3D tensor of shape (TIME_STEPS, DIMENSION). This reshaping is necessary for the input to be compatible with the LSTM layer, which requires a 3D input shape. The first Dropout layer is added to the model to prevent overfitting, which is a common problem in neural networks. Overfitting occurs when the model learns the training data too well and fails to generalize to new, unseen data. The Dropout layer randomly sets 20% of the input units to 0 during each training batch, which helps to prevent the model from overfitting.

The LSTM layer is a type of recurrent neural network that is well-suited for processing sequential data. It has 16 units and an input shape of [TIME_STEPS, DIMENSION]. This layer is particularly useful for modeling sequences of data, as it is capable of capturing long-term dependencies in the input sequence.

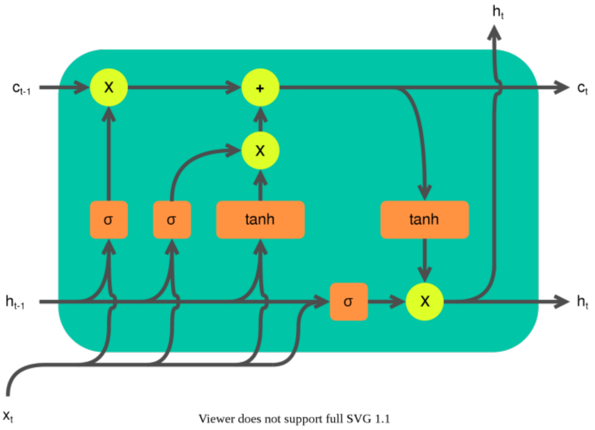


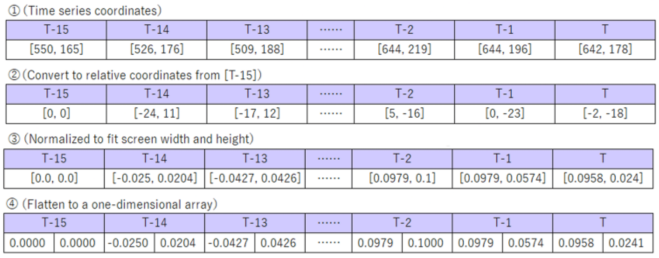
FIGURE 6. General scheme of a LSTM cell

The second Dropout layer is added after the LSTM layer to further prevent overfitting. This Dropout layer randomly sets 50% of the input units to 0 during each training batch.

The Dense layer is added to the model after the Dropout layer. It has 10 units and a ReLU activation function, which applies a rectified linear unit (ReLU) activation function to the output of the previous layer. The ReLU activation function is commonly used in neural networks and helps to introduce non-linearity into the model.

Finally, the output layer is a Dense layer with NUM_CLASSES units and a softmax activation function. This layer is responsible for producing the output of the model, which in this case is a classification of the input data into one of the NUM_CLASSES output classes.

The data preprocessing in this model is a crucial step that transforms the raw landmark coordinates into a suitable input format for the LSTM layer to



① (Time series coordinates)

T-15	T-14	T-13	......	T-2	T-1	T
[550, 165]	[526, 176]	[509, 188]	......	[644, 219]	[644, 196]	[642, 178]

② (Convert to relative coordinates from [T-15])

T-15	T-14	T-13	......	T-2	T-1	T
[0, 0]	[-24, 11]	[-17, 12]	......	[5, -16]	[0, -23]	[-2, -18]

③ (Normalized to fit screen width and height)

T-15	T-14	T-13	......	T-2	T-1	T
[0.0, 0.0]	[-0.025, 0.0204]	[-0.0427, 0.0426]	......	[0.0979, 0.1]	[0.0979, 0.0574]	[0.0958, 0.024]

④ (Flatten to a one-dimensional array)

T-15		T-14		T-13		......	T-2		T-1		T	
0.0000	0.0000	-0.0250	0.0204	-0.0427	0.0426	......	0.0979	0.1000	0.0979	0.0574	0.0958	0.0241

FIGURE 7. Stages of point history pre-processing

learn from. Since the data is a time series of landmark coordinates, it is essential to select the appropriate frames that capture the motion of the hand over time. Therefore, the second stage involves selecting the last 16 frames for each gesture, which are referred to as the keyframes. These frames serve as the input for the model, and they are used to capture the temporal evolution of the hand gestures.

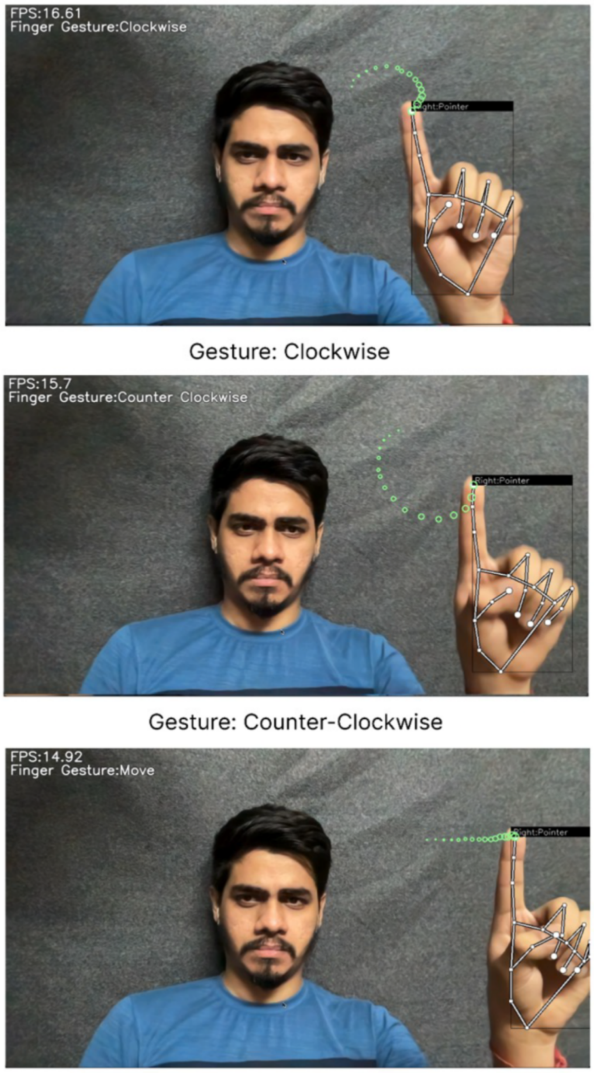
In the third stage, the relative coordinates of each landmark point are calculated with respect to the position of the first landmark point. This step helps to eliminate the effect of translation of the hand in the video, which can affect the accuracy of the model. The fourth stage involves normalizing the relative coordinates to the range of [0, 1]. Normalizing the data helps to reduce the effect of scaling in the input data, which can improve the performance of the model. The model gets its information from the normalized and flattened list of landmark coordinates that was made. The length of the array is determined by the number of keyframes (16), the number of landmark points per frame (21), and the number of coordinates per landmark point (2). This input representation captures the temporal evolution of the hand gestures, providing a suitable input for the LSTM layer to learn from.

Thus, the data preprocessing steps in this model are crucial for transforming the raw landmark coordinates into a suitable input format that captures the temporal evolution of the hand gestures. The resulting input representation is used to train the LSTM layer of the model, which is well-suited for processing sequential data and capturing long-term dependencies in the input sequence.

5. Results

5.1. Dynamic Gesture Recognition

Examples of Dynamic Gesture Recognition is shown in the Figure 8.



Gesture: Clockwise

Gesture: Counter-Clockwise

Gesture: Move

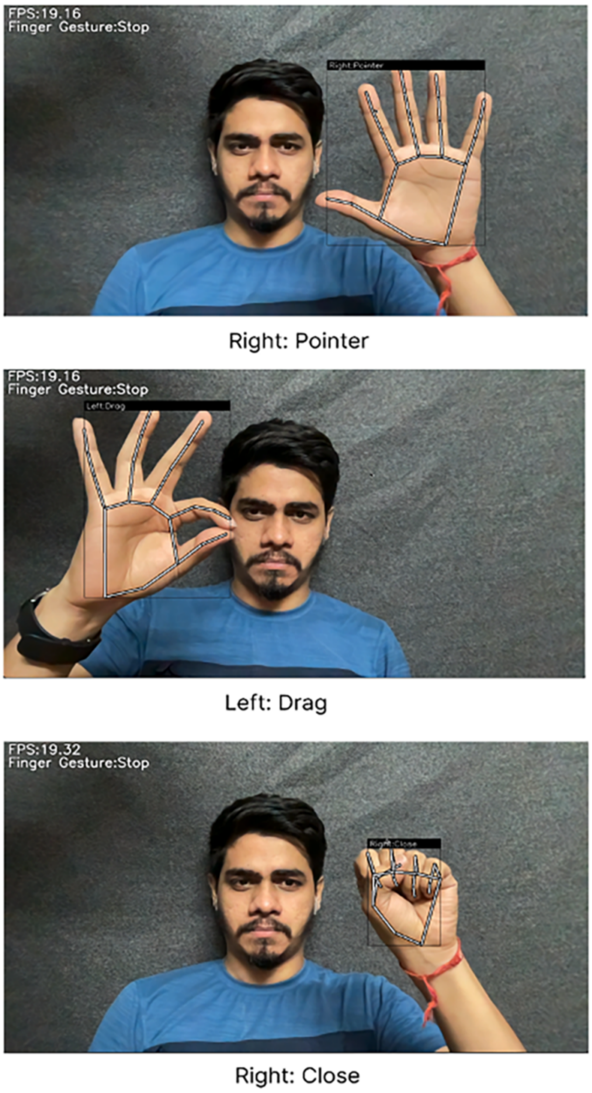
FIGURE 8. Examples of Dynamic Gesture Recognition

5.2. Static Gesture Recognition

Examples of Static Gesture Recognition is shown in Figure 9.

5.3. Confusion Matrix for Key-point based Model

This is a classification report that shows the performance of a classification model on a test set. The report shows the precision, recall, and F1-score for each class, as well as the accuracy and weighted average across all classes. Precision is the proportion of true positive predictions (correctly classified instances) out of all positive predictions (total instances predicted as positive). Recall is the proportion of true positive predictions out of all actual positive instances in the test set. The F1-score is the harmonic mean of precision and recall, and provides



Right: Pointer

Left: Drag

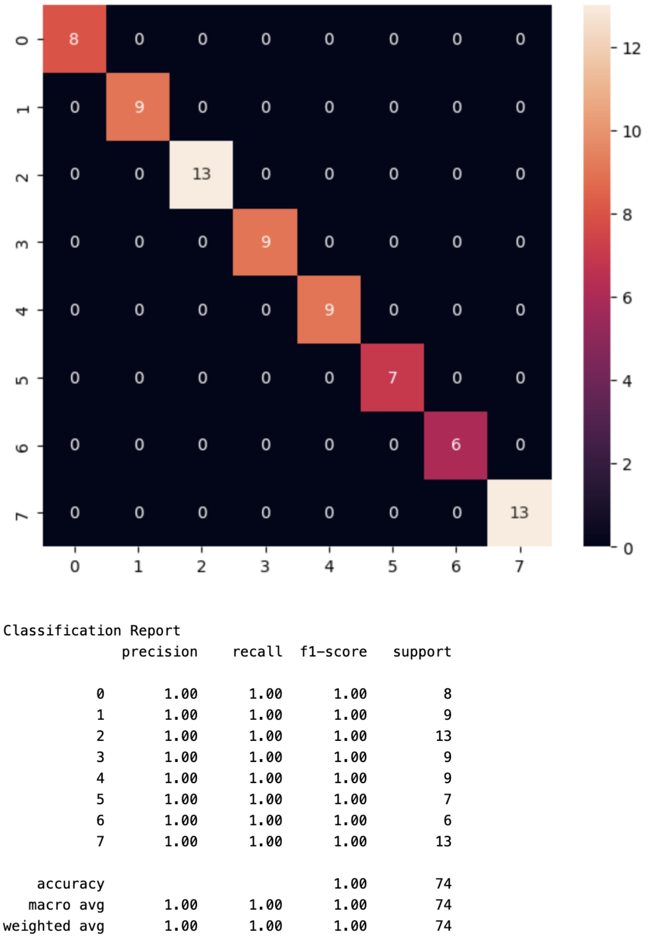
Right: Close

FIGURE 9. Examples of Static Gesture Recognition

a balanced measure of the model's performance. In this report, we see that the precision, recall, and F1-score for all classes are 1.00, indicating that the model achieved perfect classification performance on the test set. This means that the model correctly classified all instances in the test set for each class. The accuracy of the model is also 1.00, which further confirms the perfect performance of the model on the test set. The macro average and weighted average F1-scores are also 1.00, which indicates that the model has excellent overall performance across all classes.

5.4. Confusion Matrix for Point history based Model

This is a classification report that provides various evaluation metrics such as precision, recall, and f1-score for a multi-class classification problem. The report shows the performance of the model on a dataset of 1341 samples, where the model achieved an accuracy of 0.95. The macro average and weighted average for precision, recall, and f1-score are also provided

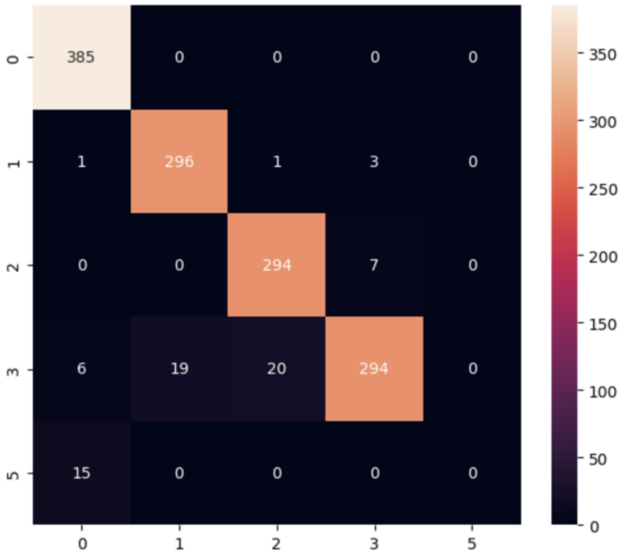


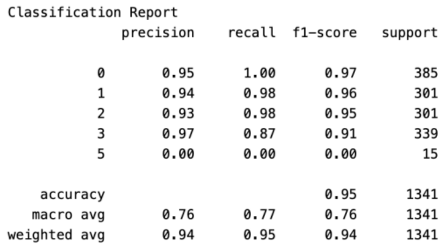
```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        13
           3       1.00      1.00      1.00         9
           4       1.00      1.00      1.00         9
           5       1.00      1.00      1.00         7
           6       1.00      1.00      1.00         6
           7       1.00      1.00      1.00        13

    accuracy                           1.00        74
   macro avg       1.00      1.00      1.00        74
weighted avg       1.00      1.00      1.00        74
```

FIGURE 10. Confusion Matrix for Key-point model

6. Conclusion

In conclusion, this paper presents a real-time on-device hand gesture recognition solution that utilizes the MediaPipe-Hands API and deep learning models, implemented as TensorFlow Lite models, for static and dynamic gesture recognition. This approach is able to run in real-time on both PC and mobile devices without the need for specialized hardware, and the tf-lite models make it easy to deploy on mobile hardware. The system works by detecting the hand region from the input video stream, tracking the hand from frame to frame, and utilizing a hand landmark model to increase accuracy. The palm detection module is also utilized





```
Classification Report
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       385
           1       0.94      0.98      0.96       301
           2       0.93      0.98      0.95       301
           3       0.97      0.87      0.91       339
           5       0.00      0.00      0.00        15

    accuracy                           0.95      1341
   macro avg       0.76      0.77      0.76      1341
weighted avg       0.94      0.95      0.94      1341
```

FIGURE 11. **ConfusionMatrix for Point-History model**

to help locate the hand when the landmark model is unable to recognize it. The output from the MediaPipe API is used to train the model for different gestures, and once trained, the model can detect these gestures from any simple video feed. This technology has potential applications in a wide range of fields, including but not limited to: sign language interpretation, hand gesture control for virtual and augmented reality, gaming, and assistive technology for people with mobility impairments. Future scope of work includes:a. Incorporating more complex gestures: The current model is trained on a limited set of hand gestures. Future work can involve expanding the gesture vocabulary and training the model on a more diverse set of gestures.b. Improving speed and accuracy: While the proposed solution is real-time, there is still room for improvement in terms of speed and accuracy. This can be achieved through optimizing the model architecture and hyperparameters, as well as using more powerful hardware.c. Multi-user support: Currently, the proposed solution is designed to recognize gestures from a single user. Future work can involve extending the system to support multiple users simultaneously.d. Integrating with other computer vision tasks: Hand gesture recognition can be combined with other computer vision tasks such as object detection and facial recognition to create more sophisticated applications. This can be achieved by integrating multiple models and developing more complex algorithms.

References

Aly, Saleh, et al. “Arabic sign language fingerspelling recognition from depth and intensity images”. *2016 12th International Computer Engineering Conference (ICENCO)* (2016): 99–104.

Amangeldy, Nurzada, et al. “Sign Language Recognition Method Based on Palm Definition Model and Multiple Classification”. *Sensors* 22.17 (2022): 6621–6621.

Baccouche, Moez, et al. “Sequential Deep Learning for Human Action Recognition”. *Lecture Notes in Computer Science* (2011): 29–39.

Cui, Runpeng, Hu Liu, Changshui Zhang, et al. “Recurrent Convolutional Neural Networks for Continuous Sign Language Recognition by Staged Optimization”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017): 7361–7369.

Donahue, Jeff, et al. “Long-term Recurrent Convolutional Networks for Visual Recognition and Description”. *IEEE Trans. Pattern Anal. Mach. Intell* 39.4 (2014): 677–691.

Haroon, M, et al. “Hand Gesture Recognition with Symmetric Pattern under Diverse Illuminated Conditions Using Artificial Neural Network”. (2022).

Huang, Jie, et al. “Video-Based Sign Language Recognition Without Temporal Segmentation”. *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018): 1–8.

Kang, Byeongkeun, Subarna Tripathi, Truong Q Nguyen, et al. “Real-time sign language fingerspelling recognition using convolutional neural networks from depth map”. *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (2015): 136–140.

Kim, Taehwan, et al. “Lexicon-free fingerspelling recognition from video: Data, models, and signer adaptation". *Computer Speech & Language* 46 (2017): 209–232.

Köpüklü, O, N Köse, G Rigoll, et al. "Motion fused frames: Data level fusion strategy for hand gesture recognition". *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)* (2018): 2184–21848.

Krizhevsky, Alex, Ilya Sutskever, Geoffrey E Hinton, et al. "ImageNet classification with deep convolutional neural networks". *Communications of the ACM* 60.6 (2017): 84–90.

Kumar, Pradeep, et al. "A multimodal framework for sensor based sign language recognition". *Neurocomputing* 259 (2017): 21–38.

Lai, Kam, Janusz Konrad, Prakash Ishwar, et al. "A gesture-driven computer interface using Kinect". *2012 IEEE Southwest Symposium on Image Analysis and Interpretation* (2012): 185–188.

Li, Jinyu and De Zhang. "Face gesture recognition based on clustering algorithm". *2019 Chinese Control And Decision Conference (CCDC)* (2019).

Li, Kehuang, Zhengyu Zhou, Chin-Hui Lee, et al. "Sign Transition Modeling and a Scalable Solution to Continuous Sign Language Recognition for Real-World Applications". *ACM Transactions on Accessible Computing* 8.2 (2016): 1–23.

Liao, Yanqiu, et al. "Dynamic Sign Language Recognition Based on Video Sequence With BLSTM-3D Residual Networks". *IEEE Access* 7 (2019): 38044–38054.

Liu, Zhipeng, et al. "Continuous Gesture Recognition with Hand-Oriented Spatiotemporal Feature". *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* (2017): 3056–3064.

Ma, Zhanyu, et al. "Variational Bayesian Learning for Dirichlet Process Mixture of Inverted Dirichlet Distributions in Non-Gaussian Image Feature Modeling". *IEEE Transactions on Neural Networks and Learning Systems* 30.2 (2019): 449–463.

Maraqa, Manar, Raed Abu-Zaiter, et al. "Recognition of Arabic Sign Language (ArSL) using recurrent neural networks". *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)* (2008).

Maung, T H H. "Real-Time Hand Tracking and Gesture Recognition System Using Neural Networks". *World Academy of Science, Engineering and Technology* 50 (2009): 466–470.

Mohandes, M, M Deriche, J Liu, et al. "Image-Based and Sensor-Based Approaches to Arabic Sign Language Recognition". *IEEE Transactions on Human-Machine Systems* 44.4 (2014): 551–557.

Murakami, K, H Taguchi, et al. "Gesture Recognition using Recurrent Neural Networks". (2022): 237–242.

Ng, J Y, et al. "Beyond short snippets: Deep networks for video classification". *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* (2015): 4694–4702.

Sonawane, T, et al. "Sign language recognition using leap motion controller". *Internation J. Advance Res. Innov. Ideas Edu* 3.2 (2017): 1878–1883.

Srivastava, N, E Mansimov, R Salakhudinov, et al. "'Unsupervised learning of video representations using lstms". *Proc. 32th Int. Conf* (2015): 843–852.

Wang, Hanjie, Xiujuan Chai, and Xilin Chen. "Sparse Observation (SO) Alignment for Sign Language Recognition". *Neurocomputing* 175.29 (2016): 674–685.

Zhang, X, et al. "Self-Supervised Contrastive Pre-Training For Time Series via Time-Frequency Consistency". (2022).

Embargo period: The article has no embargo period.

To cite this Article: , Rishav Nath Pati, Nagalakshmi Vallabhaneni , Prabhavathy P , Yash Shekhawat , and Srijan Paria . "**Real-Time Computer Vision Based Hand Gesture Recognition**." International Research Journal on Advanced Science Hub 05.05S May (2023): 382–391. http://dx.doi.org/10.47392/irjash.2023.S052