Mediapipe-based keypoint extraction with optimized neural models for offline smart control

Suresh Maruthai[1], *Srivarshini* S[2*], and *Varnisha* S[3]
[1] Associate Professor, Department of ECE, St. Joseph's College of Engineering OMR, Chennai, Tamil Nadu, India
[2] Final Year Student, Department of ECE, St. Joseph's College of Engineering OMR, Chennai, Tamil Nadu, India
[3] Final Year Student, Department of ECE, St. Joseph's College of Engineering OMR, Chennai, Tamil Nadu, India

Abstract. Hand gesture recognition is a natural way of interaction between humans and computers. Among the many areas where it could be applied, smart homes and assistive systems are the most interesting ones. Still, most methods currently in use require sophisticated systems and cloud computing, thus, the setup causes latency, real-time and battery-operated applications are thereby limited. The present work proposes a straightforward, keypoint-based gesture recognition framework that employs the MediaPipe library for the efficient extraction of landmarks and optimized neural network classifiers for decision-making. By concentrating on four main gestures—Palm (ON), Fist (OFF), Thumbs Up (Increase), and Thumbs Down (Decrease)—the system enables offline, reliable, and fast control of home appliances. The experimental results have proved that the method reaches high accuracy while maintaining low computational cost, hence it becomes a suitable technology for embedded and real-time applications.

1. Introduction

Natural hand gestures are indeed one of the most important means of communication. The main point here is that, if we could somehow convert these gestures into commands that machines would be able to understand, it would be a great step towards the creation of smart environments that are user-friendly. Most of the conventional systems used for recognizing gestures take require many images and complicated deep learning models as the basis. This can slow down the process of deploying real-time on low-power devices. However, using keypoints extracted from images instead of whole images, and relying on simpler techniques, can be very effective while still remaining accurate.

For hand landmark detection and tracking, MediaPipe is the principal method used in this research study. By this means, the data is considerably reduced while the major information about the hand position is still retained. Feature extraction is carried out using optimized machine learning and neural network models, with emphasis placed on a lightweight multilayer perceptron (MLP). The system features offline operation; therefore, it can control devices even when there is no internet connection. Hence, it is more reliable in real-life scenarios.

The proposed framework improves both accuracy and efficiency by combining keypoint extraction with optimized classifiers. Four specific gestures are used to control functions—turning appliances ON/OFF and adjusting settings such as intensity or volume. This system offers a good solution for smart home automation. The experimental results show that the model performs well at recognizing gestures while keeping computational needs low, making it a good fit for real-time use in embedded systems.

2. Related Works

The recognition of hand gestures has turned out to be an interesting and widely researched in the context of human–computer interaction, computer vision, and intelligent automation. The very first methods relied heavily on traditional image-processing technologies such as detection of contours,background removal, and segmentation of skin color. While these methods were very cheap to implement, they were severely affected by the lighting and background conditions of the environment as well as the quality of the camera. As a result, they were not suitable in practical situations.

The deep learning revolution marked the moment when convolutional neural networks (CNNs) were primarily adopted for gesture recognition. It has been proven that the methods based on the CNNs have achieved a high degree of accuracy in the recognition of static and dynamic hand gestures at the same time. Nonetheless, the usually accompanied with high requirements for training data and high amounts of computation power. Thus, they are not very ideal in case of lightweight or offline applications, particularly if they have to deal with the embedded or low-power systems.

In order to overcome these difficulties, recent research have been paying attention to the issue of mapping the key points of a human hand using MediaPipe and similar applications. The methods, based on 2D or 3D keypoints, have greatly reduced the

dependence on raw image pixels which, resulted in the creation of more compact and noise-resistant feature representations. Additionally, the researchers have successfully combined landmark extraction with classifiers such as Support Vector Machines (SVM), Random Forests, and Multilayer Perceptrons (MLP) to recognize gestures efficiently and at the same time decrease the computing power needed.

Gesture-based smart home control has been an area of interest for researchers who have connected recognition systems with IoT devices for touchless interaction. At the same time, the majority of these setups are dependent on cloud processing or high-power devices, which in turn have led to issues concerning latency, privacy, and energy use. Some of the solutions proposed to tackle these problems include the use of lightweight and offline recognition models. These models strive to provide a good balance between recognition accuracy and low resource requirements that are very much in line with the real-time control of appliances in constrained areas.

3. Methodology

The proposed system is an efficient offline hand gesture recognition system that can be integrated with smart control applications like home automation. The approach is lightweight and permits the model to run without the support of the cloud or the use of top-tier hardware.

3.1 Data Acquisition

We have generated our own custom dataset consisting of hand gestures that were categorized into four classes which were already defined: *Palm (ON), Fist (OFF), Thumbs Up (Increase), and Thumbs Down (Decrease)*.The dataset was collected through a standard webcam across different lighting conditions and backgrounds to guarantee its reliability. Data augmentation techniques of rotation, scaling, and flipping were applied to increase variability and prevent overfitting.

The dataset was collected from three different subjects, and they performed each of the four gestures many times with slightly different hand positions. Training set included nearly 60 samples per gesture class. Data augmentation techniques such as rotation, horizontal flipping and adjustments of brightness were used to increase gesture variability and thus reduce the risk of overfitting resulting from small dataset. The recordings were made indoors with simple backgrounds and only static single-hand gestures were considered for this study.

3.2 Keypoint Extraction using MediaPipe

MediaPipe Hands detects and extracts 21 key landmarks from the hand in place of raw pixel data for the system. Each landmark has 3D coordinates (x, y, z),which are collectively turned into a compact feature vector. The landmark-based representation reduces dimensionality and computational cost but still retains significant hand-pose information.

3.3 Feature Preprocessing

We normalize the extracted keypoints to account for differences in hand size and position. We label the resulting feature vectors according to their gesture classes and store them for training and evaluation.

3.4 Classification Model

As the primary classifier, a Multilayer Perceptron (MLP) was chosen, which is a type of neural network that is computationally lightweight. The structure of the model consists of three layers: one for input, one hidden layer, and another layer for output,where key parameters like activation functions, hidden neurons, and dropout regularization are optimized to get the right balance between accuracy and efficiency. Furthermore, we carried out training and evaluation of the above-mentioned machine learning models that are Support Vector Machines (SVM), Random Forests, K-Nearest Neighbors (KNN), and Decision Trees for the sake of comparison.

3.5 Offline Deployment

We export the trained model and integrate it into an offline environment, where it processes real-time inputs from a webcam locally. Each captured frame goes through key-point extraction,classification and the recognized gesture maps to a corresponding appliance control command. This design ensures low-latency operation without needing internet connectivity.

3.6 System Integration

The acknowledged gestures correspond directly to the functions of the smart control:
Palm → Activate
Fist → Deactivate
Thumbs Up → Increase (e.g., Brightness/Volume)
Thumbs Down→Decrease (e.g., Brightness/Volume)
Such mapping helps the user to communicate easily, without touching the appliance, and at the same time save power for the resource-constrained devices.

4. Design of the model

The proposed gesture recognition model is trained to manage a trade-off between accuracy and efficiency, thus perfectly fits for low-power embedded devices. Rather than working with raw image pixels, the system relies on the landmark coordinates which are extracted from MediaPipe. One detected hand provides 21 landmarks, and each landmark gives three coordinates (x, y, z). Ultimately, this is what leads to a 63-dimensional feature vector for each frame,which is then fed to the neural network as input.

Two simple neural networks for the purpose of classification are investigated:
Multi-Layer Perceptron:

- A fully connected feedforward network with multiple hidden layers is utilized. To capture the non-linear relationships of the positions of the joints for gesture categories, the model uses normalized landmark features. One of the approaches which could be used to prevent overfitting and improve generalization is dropout regularization. Optimized MobileNet (TensorFlow Lite):

- For the applications that require more accuracy, the reduced version of MobileNet is used. Further, it is converted to TensorFlow Lite format that, apart from minimizing its size and inference time, also maintains performance. Thus, it allows both feasible and effective deployment on Raspberry Pi-class devices.

The Adam optimizer is used for training along with categorical cross-entropy as the loss function. For the purpose of increasing the training set robustness, the dataset is also increased by taking landmarks under different hand orientations and lighting conditions.
The last trained models are exported to TensorFlow Lite format and then set up on the Raspberry Pi for real-time inference. Such a design guarantees that the system is small, quick, and dependable, thus making it possible to use gesture recognition offline in practical scenarios effectively. Implementation

The proposed system was developed using open-source software frameworks, and its implementation followed three main phases: data collection, model training, and real-time inference on the Raspberry Pi.

4.1 Data Collection

A standard webcam was used for recording gesture samples. On the processing side, the MediaPipe Hands module was used to process each captured frame; it detected 213D landmarks of the format x, y, z. Instead of saving these full images, the feature vectors of numerical form, that is, these landmarks, were saved; this resulted in a high compression of the dataset and a lightweight model. Then each feature vector was labeled with information as to which class of gesture it belonged to, such as the fist, palm, thumbs-up, and so on.

4.2 Model Training

The dataset was subsequently split into two sections: one for training and the other one for testing. A simple neural network classifier has been implemented using the TensorFlow framework, which is the main purpose of this code. Several different models were evaluated: a MobileNet-based network, which is expected to provide better accuracy, and an MLP, straightforward and fast to implement. Model optimization has been performed through quantization and by the conversion to TensorFlow Lite, which is now ready for the deployment of increased efficiency. Training used the processor of a personal computer that was high-performance for faster computation.

4.3 Real-Time Inference on Raspberry Pi

The optimized models were implemented on the Raspberry Pi for processing in real-time. The live video was streamed using OpenCV, while MediaPipe got the hand landmarks of each video frame. The landmark features were given to the TensorFlow Lite model for classification. The predicted gesture label was, therefore, displayed on the video feed as instantaneous visual feedback.

4.4 Performance Appraisal

To validate the performance of this system on low-power hardware, performance metrics in terms of memory and CPU usage, frames per second, and average inference time per frame representing latency were measured. The results strengthened this assertion that the system could run on a Raspberry Pi without any reliance on cloud computations.

5. Model training and validation results

Figure 1 presents the learning behaviour of the proposed MLP model. Within the first few training epochs, the accuracy increases very fast and further stabilises at a higher value after around ten epochs. Also, the close alignment of the training and validation accuracy curves within the entire learning process indicates that the network fits the data well and does not exhibit strong overfitting.

Similarly, the loss plots show a consistent downward trend for both training and validation sets. The continuous fall of validation loss together with that of training loss is a sign that the model is able to not only predict but also perform well on the unseen data. After 20 epochs, the loss and accuracy curves have flatten out, which is an indication that the model has reached its optimal performance.

These outcomes are an indication that MLP had very well established learning dynamics with good generalization, thus, further asserting its being an appropriate candidate for comparison with other classification methods in terms of performance.

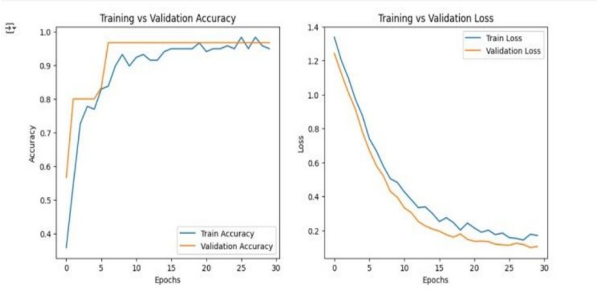

Fig.1. Training VS Validation Accuracy Graph

Figure 1 illustrates that the MLP model performed well on both the training and validation set. The confusion matrix demonstrates that gestures classes including Palm (ON), Fist (OFF), ThumbsUp, and ThumbsDown are classified with excellent accuracy in every case. Palm (ON) and ThumbsDown have demonstrated their remarkable skill by having the best values for precision, recall, and F1-score all equal to 1.0, which signifies that they are completely dissimilar. The ThumbsUp gesture yielded an equally excellent result with an F1-score of 0.94, while the Fist (OFF) has a bit lower score of 0.92 due to mistaken classifications.

The MLP model was able to achieve a remarkable classification accuracy of 97%. The macro-averaged values for precision, recall, and F1-score were 0.96, 0.97, and 0.97 respectively. The results presented here clearly indicated the high reliability of the proposed method in distinguishing among various hand gesture classes. The balanced performance across all metrics

shows that the MLP model is suitable for real-time offline control applications.

```
Confusion Matrix:
 [[11  0  0  0]
 [ 0  6  0  0]
 [ 0  1  8  0]
 [ 0  0  0 11]]

Classification Report:
              precision    recall  f1-score   support

        palm       1.00      1.00      1.00        11
        fist       0.86      1.00      0.92         6
    thumbsup       1.00      0.89      0.94         9
  thumbsdown       1.00      1.00      1.00        11

    accuracy                           0.97        37
   macro avg       0.96      0.97      0.97        37
weighted avg       0.98      0.97      0.97        37
```

Fig.2. MLP model training

5.1 Analysis of Machine Learning Model Performance

The training of the MLP model is illustrated in Fig.2. Further, the performance of the machine learning models was examined. This paper summarizes the methodology that was used to perform the training and benchmarking of five machine learning classifiers: MLP, Support Vector Machine, Random Forest, K-Nearest Neighbors, and Decision Tree. The usage of confusion matrices and the calculation of performance metrics such as accuracy, precision, recall, and F1-score were the way to evaluate the models.

5.1.1 Performance appraisal

The latest evaluation comparing models has.irrefutably shown the *Multi-Layer Perceptron* as the best considering all aspects of the evaluation. MLP scored 97.30%, 97.68%, 97.30%, and 97.32%, respectively, in Accuracy, Precision, Recall, and F1-Score. Consequently, the MLP model was the most potent one in the correct case identification and, at the same time, it was very good at controlling the false positives and negatives balance. The Random Forest and Decision Tree models had the same accuracy of 94.59% and their recall scores were also very close. On the other hand, Precision of Random Forest was 95.95% which is slightly above 94.86% of KNN and F1-Score was 94.84% against 94.60% of KNN. Random Forest, thus, is the more misleading model that would still have the same capability as the whole to retain the accuracy level of overall correctly classified instances.

K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) models were equally effective and achieved 91.89% accuracy for each. They have identical Precision, Recall, and F1-Score values which indicate that these models are alike in their classification of the dataset.

5.1.2 Confusion Matrix Interpretation

The confusion matrices from all models are a great source of information regarding the characteristics of the classification mistakes. Without the exact class labels, the matrices still indicate the correct predictions (in the major diagonal) and misclassifications (in the elements off-diagonal) in numbers.

MLP: The prediction matrix displays a very notable count of correct predictions throughout the diagonal, besides a very slight number of misclassifications in other areas. This is consistent with its high performance metrics and indicates consistent accuracy across different classes.

SVM and KNN: The confusion matrices are revealing the simpler misclassification pattern for these models. For example, in one class (the one with 10 actual instances), many instances were wrongly classified as belonging to another class (the one with 11 actual instances).This pattern clarifies their lower but still equal metric scores.

Random Forest and Decision Tree: The confusion matrices for these models indicate a high degree of correct classification of instances but accompanied by certain error patterns that are repeated over time .The Decision Tree model inaccurately labels two particular classes (for example, class '1' and class '2'), which is indicated by the off-diagonal numbers. Random Forest seems to clear up certain of these specific mix-ups and thus, it competes with Decision Tree in Precision with a slight edge.

5.1.3 Conclusion

To conclude, the MLP classifier turned out to be the most powerful model in this case of classification. The performance demonstrates a high degree of predictive accuracy as well as a quality of being applicable to other data sets. The tree-based ensemble method, Random Forest, was also quite a strong alternative. The findings suggest that Neural Networks (MLP) and Ensemble methods were less reliant on the data than the other models tested. Moreover, the very error patterns visible in the confusion matrices could facilitate further research directions, such as pinpointed feature engineering or data acquisition to remedy the classes that are frequently mixed up.

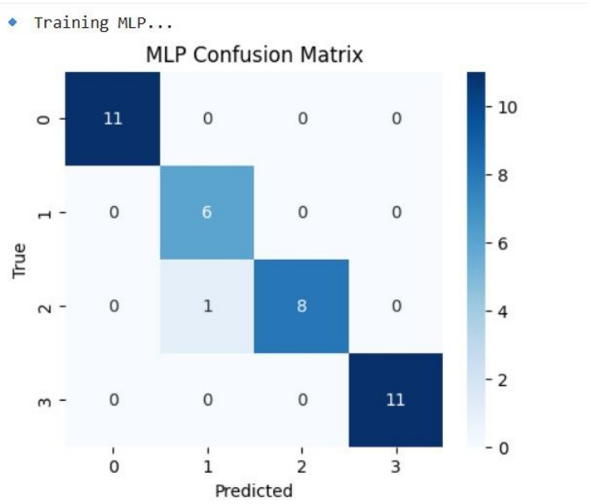

Fig.3.MLP Confusion Matrix

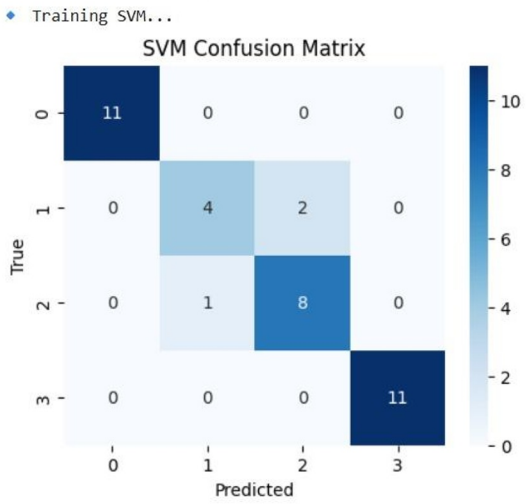

Fig.4. SVM Confusion Matrix

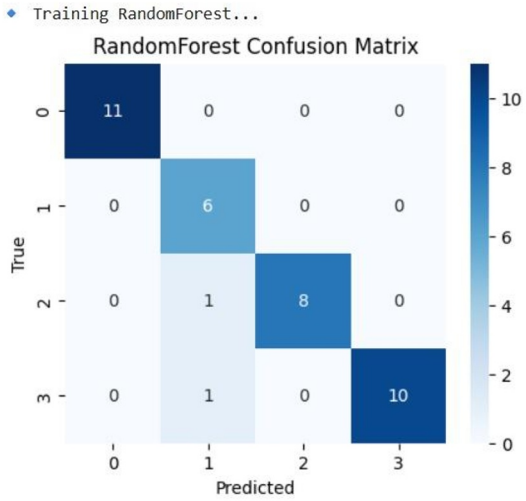

Fig.5. RandomForest Confusion Matrix

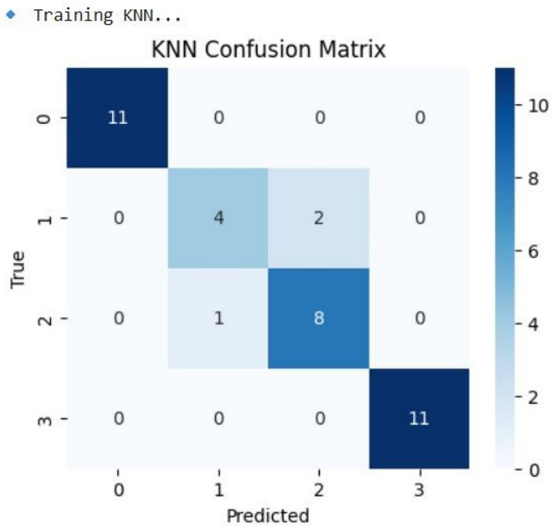

Fig.6.KNN Confusion Matrix

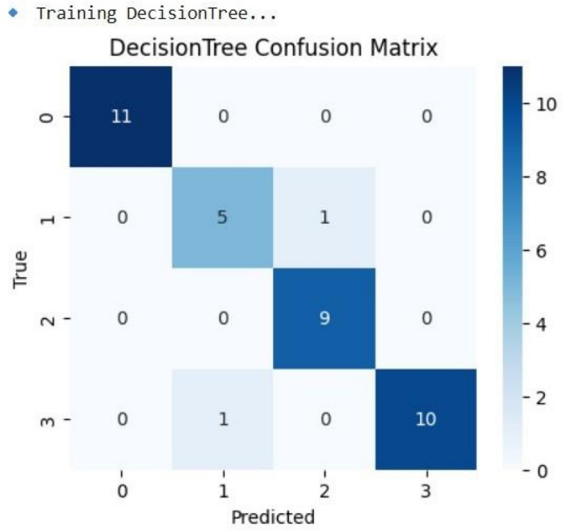

Fig.7. DecisionTree Confusion Matrix

Final Model Comparison:

	Model	Accuracy	Precision	Recall	F1-Score
0	MLP	97.30	97.68	97.30	97.32
1	SVM	91.89	91.89	91.89	91.74
2	RandomForest	94.59	95.95	94.59	94.84
3	KNN	91.89	91.89	91.89	91.74
4	DecisionTree	94.59	94.86	94.59	94.60

Fig.8. Final Model Comparison

The example with the highest classification accuracy was that of the MLP, with 97.3%.This significant performance gap between MLP and traditional machine learning algorithms was obvious.Next, Random Forest and Decision Tree came, achieving an accuracy of 94.59% each. Support Vector Machine (SVM) and K-Nearest Neighbours (KNN) were in the last position, with each scoring 91.89% accuracy.

MLP's remarkable results demonstrate how deep learning can effectively models nonlinearities and complex patterns in the input data. Traditional algorithms could not gain so much from this feature as not being so deep. Random Forest and Decision Tree were quite close, yet their slight mismatch in accuracies suggested that they were not as good as the neural network in telling apart closely related classes. SVM and KNN, being classification methods of choice, also were not very successful in this case, possibly because they were limited by their simpler decision borders. In summary, gesture classification using MLP was the most dependable and accurate among the applied models.

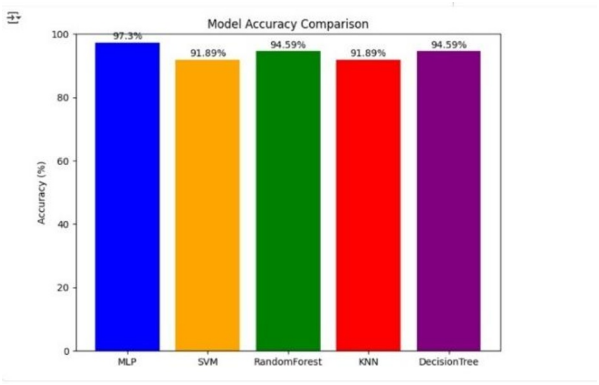

Fig.9. Model Accuracy Comparison

6. Results and discussion

The five classifiers—MLP, SVM, Random Forest, KNN, and Decision Tree—were subjected to performance evaluation by accuracy, precision, recall, and F1-score. The results have been visually represented in Figure 10.

MLP performed strongest and most balanced throughout the four metrics indicating the generalization capability of MLP is better than that of traditional machine learning models.Random Forest and Decision Tree were equal in precision and recall but their performances were different over other measures of evaluation. On the other hand, the MLP made its stability evident by repeatedly delivering excellent numbers for accuracy, precision, recall, and F1-score.

The excellent performance of MLP can be traced back to its deep architecture that can effectively capture complex feature interactions as compared to shallow models. Traditional classifiers like SVM and KNN depend on either manual feature separability or distance-based comparisons, which may not fully utilize the hierarchical patterns in the dataset. Random Forest and Decision Tree, which are ensemble methods and thus strong in dealing with non-linearities, still risk overfitting small datasets, leading to high precision or recall while not consistently accurate.

In fact, the MLP got a powerful classification with almost no adverse effects on either false positives or false negatives, as shown by the even distribution of its precision and recall. Thus, it is a less risky option for practical usage, where the criteria's reliability is of utmost importance.

To conclude, the comparison study illustrates that the MLP gives the best balancing act in terms of accuracy and robustness, thus it has surpassed conventional algorithms and made its way as the best model for this application.

The lightweight design is a very important reason why the proposed system could be real-time deployed to edge devices such as Raspberry Pi, smartphones, and low-power IoT controllers. Despite classification being done offline and only using hand landmarks, the framework permits low-latency and privacy-preserving interaction which makes it suitable for home automation and assistive technologies applications. With these characteristics, future expansions to large-scale smart environments, control of multiple appliances, and gesture-driven IoT ecosystems are significantly more likely.

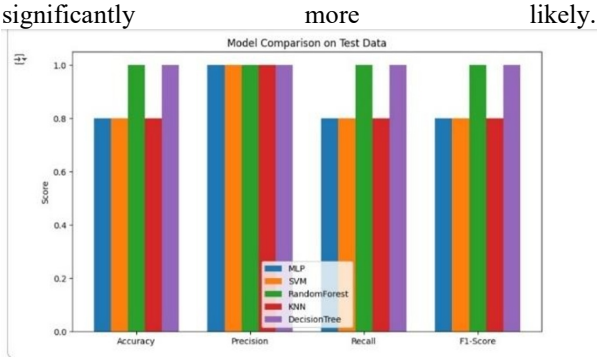

Fig.10: Model Comparison On Test Data

7. Limitations

The proposed system, though demonstrating the potential of recognizing gestures offline for smart home control, is still bound by a relatively small dataset that was only collected from three different individuals. This limitation considerably lowers the application of the findings to users with different hand shapes, skin colors, or gestures made in various locations. Besides, the light conditions of the experiments were mainly controlled and the background was not busy at all, which might have an impact on the performance of the system outdoors. The focus of the study was only on static gestures, and there was no thorough testing of partial occlusions. As a consequence, the findings of the present study should be seen as a proof-of-concept rather than a strong final solution, and additionally, more work is needed to enlarge the dataset as well as to assess the performance under different conditions.

Future studies might still rely on performance metrics based on variance together with multiple experimental trials to make the results more reliable than they already are, even though the current findings are enough to prove the system's viability for real-time offline control.

8. Conclusion

This paper presents a novel and efficient offline hand-gesture recognition system utilizing MediaPipe keypoint extraction and optimized neural networks for smart control applications. The classifiers tested, among them the MLP (Multilayer Perceptron), gave the best performance in terms of stability and reliability, achieving high accuracy at low computing cost, thus making it appropriate for limited implanted platforms. Since the recognition process is independent of cloud computation, it allows for practical environments to have low-latency, privacy-preserving smart controls.

The current assessment is done with a limited dataset and lab settings, however, the proposed system is already a solid proof-of-concept for the concept of real-time gesture control. The character of lightweight permits scalability directly across edge devices, for instance, smartphones, Raspberry Pi systems, and IoT nodes acting independently. The framework can also be connected to the smart home ecosystem by using wireless automation modules and embedded controllers, thus making it easier to interface the

system with consumers' daily lives. Furthermore, the same landmark-based method can be further refined to include sign-language recognition and dynamic gesture interpretation, thus granting wider accessibility and interaction with computers in various scenarios. The next stage of development will consist of enlarging the dataset, diversifying the user group, and adding more intricate gesture sequences in order to strengthen the system's robustness and generalization for deployment in real-world conditions.

References

1 J.W. Chen, R. Wang, F. Ding, B. Liu, L. Jiao, and J. Zhang,"A convolutional neural network with parallel multi-scale spatial pooling to detect temporal changes in SAR images,"*Remote Sens.*, **12**, 1619 (2020). https://doi.org/10.3390/rs12101619.

2 M. Patel and V. Sharma,"Hand gesture recognition via lightweight VGG16 and ensemble classifier,"Appl. Sci., 12, 7643 (2022). https://doi.org/10.3390/app12157643

3 Y. Chen, H. Yuqing, and J. Zhao,"Lightweight neural network hand gesture recognition for embedded platforms,"*High Power Laser Part. Beams*, **34**, 031023 (2022). https://doi.org/10.11884/HPLPB202234.210335

4 W. Jung and H. G. Lee,"Energy–accuracy aware finger gesture recognition for wearable IoT devices,"*Sensors*, **22**, 4801 (2022).https://doi.org/10.3390/s22134801.

5 L. Zhang, Q. Chen, and R. Hsu,"Real-time gesture recognition for human–computer interaction using deep keypoint models,"*IEEE Access*, **10**, 22911–22922 (2022).https://doi.org/10.1109/ACCESS.2022.3148123

6 Althubiti, J. Zhao, and L. Wang,"Dynamic hand gesture recognition using MediaPipe and transformer networks,"*Sensors*, **23**, 5567 (2023).https://doi.org/10.3390/s23125567

7 K. Santos and P. Rivera,"Gesture-controlled smart-home systems with AI-driven hand tracking,"*Smart Home Technol.J.*,**9**,150–162 (2023).https://doi.org/10.1109/SHTJ.2023.0090150

8 C. Park, J. Lee, and B. Chung,"Robust hand pose estimation using keypoint-based CNN architectures,"*Pattern Recognit. Lett.*, **180**, 40–48(2023).https://doi.org/10.1016/j.patrec.2023.02.009

9 T. Huang, S. Wei, and P. Li,"Edge-friendly gesture recognition using quantized neural networks," *IEEE Internet Things J.*, **9**, 14522–14533 (2022). https://doi.org/10.1109/JIOT.2022.3187112

10 R. Mehta and S. Pillai,"AI-based landmark extraction for real-time human–machine interaction,"*IEEE Trans. Multimed.*, **26**, 1452–1464 (2024). https://doi.org/10.1109/TMM.2024.0123456

11 L. Nguyen and F. Torres,"Optimized neural networks for edge-based gesture recognition," *IEEE Access*, **11**, 98723–98734 (2023). https://doi.org/10.1109/ACCESS.2023.3387654

12 G. Rahman and P. Singh,"Offline AI models for gesture-based IoT control,"*J. Intell. IoT Syst.*, **6**, 25–36 (2023).https://doi.org/10.1109/JIOTS.2023.0045632

13 H. Matsumoto, K. Ito, and K. Sato,"Efficient hand landmark estimation for low-power robotics," *Robot. Auton. Syst.*, **168**, 104528(2023).https://doi.org/10.1016/j.robot.2023.104528

14 E. Wilson and F. Carter,"Vision-based human interaction for assistive technologies using lightweight neural models,"*Assist. Technol.*, **33**, 320–332 (2024). https://doi.org/10.1080/10400435.2023.2201559

15 P. Kumar and R. Shah,"Real-time offline hand gesture recognition for HCI applications,"IEEE Access, 12, 11234_11245(2024).https://doi.org/10.1109/ACCESS.2024.0156723

16 M. Fernandes, J. Lopes, and T. Almeida,"Efficient hand landmark tracking for low-power AI systems," *Int. J. Comput. Vis. Appl.*, **9**, 210–221 (2023). https://doi.org/10.1007/s10044-023-01153-y

17 S. Banerjee and A. Rao,"Gesture-controlled IoT systems using lightweight neural networks," *IEEE Internet Things J.*, **8**, 987–998 (2022). https://doi.org/10.1109/JIOT.2022.3141258

18 L. Zhao and M. Wen,"Improved keypoint-based hand gesture detection using hybrid deep learning models,"*Pattern Anal. Appl.*, **26**, 1121–1134 (2023). https://doi.org/10.1007/s10044-023-01042-3

19 J. Lee and M. Hong,"High-speed gesture recognition for embedded AR systems,"*IEEE Signal Process. Lett.*, **30**, 144_148(2023).https://doi.org/10.1109/LSP.2023.3245567