Hand Gesture Classification

Nitin Somashekhar
somashekhar.n@northeastern.edu
Deenadayalan Dasarathan
dasarathan.d@northeastern.edu
Khoury College of Computer Sciences, Northeastern University
Boston, MA.

Abstract

Hand gestures have been a crucial part of human communication for centuries. In this paper, we present a novel two-pipeline hand gesture recognition architecture that leverages semantic key-points obtained from Mediapipe for hand detection, and various neural network designs for gesture classification. Our approach addresses the challenge of accurate and efficient hand gesture recognition, which is essential for communication with deaf and mute individuals, bridging the gap between sign language users and non-users, and facilitating human-computer interaction. We evaluate our approach on the HaGRID dataset comprising eight hand gestures and achieved an accuracy of 0.971.

1 Introduction

Hand gestures have played a significant role in human communication since ancient times. Today, they have gained even more importance in several fields, including Human-Computer Interaction (HCI), virtual reality, automation, and touchless user interfaces. Accurate and efficient hand gesture recognition is crucial for practical applications, including facilitating communication with deaf and mute individuals, bridging the communication gap between sign language users and non-users, and enabling natural interactions with computers.

In this paper, we propose a two-pipeline hand gesture recognition architecture that leverages semantic key-points obtained from Mediapipe for hand detection and tracking, and several neural network designs for gesture classification. Our approach aims to optimize the accuracy and efficiency of hand gesture recognition, providing a robust solution for practical applications.

Code Reference: Keypoints-CoEx

2 Related Works

Hand gesture classification has been a popular research topic in computer vision and human-computer interaction for many years. Many different approaches have been proposed for recognizing hand gestures in a variety of contexts, including sign language recognition, gaming, and virtual reality.

2.1 Hand Gesture Recognition using DenseNet201-Mediapipe Hybrid Modelling

One common approach to hand gesture recognition is to use a hybrid deep neural network model that combines hand tracking techniques such as Mediapipe with pre-trained deep neural network architectures. In this approach, hand landmarks or key-points are extracted using the Mediapipe

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

library and passed to a pre-trained model for classification. Several deep learning architectures, including Xception, ResNet152V2, Inception-ResNetV2, and DenseNet201, have been used in conjunction with Mediapipe for hand gesture classification. The base layer of these pre-trained models was created by removing the top layers, and the pre-trained layers with weights from the ImageNet dataset were frozen. To prevent overfitting, a dropout of 0.5 was applied to flatten the convolution output. Out of the different optimizers tested, Adam gave the best results with a learning rate of 0.001. The DenseNet architecture-based model achieved high validation accuracy of 97.55% on the HaGRID dataset.

2.2 Hand Gesture Recognition using Image Processing and Feature Extraction Techniques

Various feature extraction techniques have been proposed for hand gesture recognition. In this paper, the authors discuss feature extraction techniques based on ORB, HOG, and PCA. In this approach, strong feature edges are captured using Canny edge detection, and the resulting feature maps are extracted using ORB, HOG, or PCA. Similar descriptors are then clustered using a clustering algorithm. Finally, a multi-layer perceptron (MLP) model is used for classification. Out of the different feature extraction techniques evaluated, ORB and PCA produced the highest accuracy of 96% and 98%, respectively.

2.3 HGR-Net: A Fusion Network for Hand Gesture Segmentation and Recognition

A two-stage convolutional neural network (CNN) architecture has been proposed in this paper for robust recognition of hand gestures. The first stage performs accurate semantic segmentation to determine hand regions, and the second stage identifies the gesture. The architecture consists of three CNNs operating over two stages. In the first stage, the candidate hand regions are segmented in the RGB image and used as input for the second stage. The second stage is composed of two streams, one for the input RGB image and one for the segmentation map from the first stage. Each of the streams consists of a deep CNN which is converged in a fully-connected layer and a softmax classifier. The proposed architecture was evaluated on the OUHANDS dataset, which contains 10 different hand gestures from 23 subjects. The HGR-Net achieved an F1 score of 0.8810, indicating robust performance for hand gesture recognition.

2.4 Static Hand Gesture Recognition based on Convolutional Neural Networks

In this paper, the authors have investigated the performance of CNN architectures with varying depths for static hand gesture recognition. Different CNN models with varying numbers of convolutional layers, including 2, 4, 7, and 9, were trained and evaluated. The results showed that CNN, with only two layers of convolution, achieved an accuracy rate of 94.7%, while CNN with 4, 7, and 9 achieved accuracies above 96%. The use of convolution layers was found to reduce the number of epochs necessary for the network to converge.

2.5 An Improved Hand Gesture Recognition System using Keypoints and Hand Bounding Boxes

This paper proposes an improved system for hand gesture recognition that utilizes key-points and hand-bounding boxes for feature extraction. The system is divided into three modules: feature extraction, processing, and classification. In the feature extraction module, MMDetection is used to detect whole-body bounding boxes on the WH dataset, while only hand-bounding boxes are detected on the OH dataset. Whole body and hand key-points are extracted from bounding boxes using HRNet_w48, resulting in 42 hand key-points and 1 face key-point on the nose for the processing stage. In the processing module, all extracted key-points are normalized, and the collected key-points and hand-bounding box (HBB) image are used to train the models for recognition. In the classification module, four methods are compared. Method 1 utilizes fine-tuned MobileNetV2 pre-trained on ImageNet. Method 2 employs a fully-connected neural network approach that utilizes sigmoid activation functions in the first layers. Method 3 uses MobileNetV2 with fully-connected

layers and ReLU activation functions instead of the sigmoid. Finally, Method 4 replaces MobileNetV2 with a convolutional neural network (CNN) to reduce the number of parameters.
The datasets used in this study are HANDS, SHAPE, and OUHANDS. Among the four architectures tested, the two-pipeline architecture that combines hand bounding box and key points for feature extraction is found to be the most effective.

2.6 Hand gesture recognition with convolution neural networks

In this paper, a CNN classifier was used for hand gesture recognition, with spatiotemporal data augmentation employed to address overfitting. The authors also suggested applying batch normalization before non-linearity to improve convergence. With 6 convolutional layers and 2 dense layers, the model achieved an accuracy of 98.74% on a dataset of 9 hand gestures captured using a webcam.

3 Methods

3.1 Mediapipe

MediaPipe is an open-source, cross-platform framework that Google developed for creating multimodal machine learning pipelines that can process and analyze perceptual data, such as video and audio. It provides pre-built models and pipelines for object detection, face detection, hand tracking, and pose estimation, among other tasks.

This paper uses the hand landmark detection model provided by MediaPipe [3] to extract semantic key-points from the palm. Figure 6 illustrates the pipeline's ability to extract the x, y, and z locations of 21 key-points. These key-points aid in mapping the palm's skeleton, which is useful for various hand tracking and hand pose estimation tasks.

Hand tracking and pose estimation are vital for many applications, such as augmented reality, virtual reality, and sign language recognition. The MediaPipe hand landmark detection model allows for the accurate extraction of semantic key-points from the palm, which can be used to determine hand poses and gestures. This capability presents numerous possibilities for developing efficient and innovative applications that can detect and interpret hand gestures in real-time.

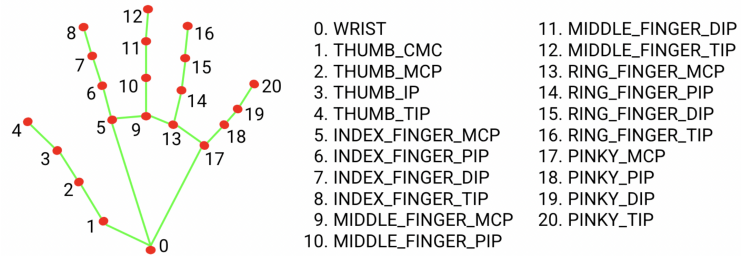

Figure 1: Hand Detection Landmarks

3.2 Dataset

The paper describes the use of the HaGRID dataset, which stands for HAnd Gesture Recognition Image Dataset, for hand gesture recognition. This dataset contains a large collection of 552,992 FullHD (1920×1080) RGB images that are categorized into 18 different classes of hand gestures. The dataset contains a wide variety of images of 34,730 unique individuals, which helps in capturing different hand gestures at varying distances and under different lighting conditions, both indoors and outdoors. This wide variety of images and conditions helps in creating a robust and generalized model for recognizing different hand gestures. In this paper, a set of 8 gestures were utilized out of the 18 classes from the HaGRID dataset for recognition purposes. Table 1 provided below outlines the specific hand gestures employed in the experiment, along with the corresponding class utilized in this study.

Figure 2: Hand Gesture Classes

Hand Sign	Class
One	0
Two	1
Three	2
Palm	3
Fist	4
Call	5
Ok	6
Like	7

Table 1: Hand Sign Classification

In this paper, each image in the HaGRID dataset is represented as an RGB image of the gesture along with 21 key-points. The Mediapipe pipeline is used to extract these 21 semantic key-points from the HaGRID dataset and store them for each image. This enables the creation of a model that can accurately recognize and classify hand gestures based on the key-points extracted and the input image. For training each gesture consists of 1000 images along with the key-points extracted for each image present in the dataset.

3.3 Methodology

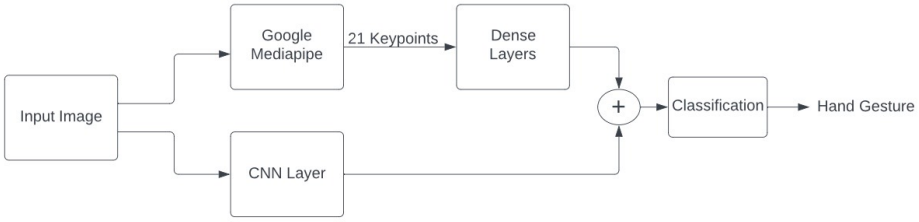

Figure 3: Two pipeline method

In this paper, we propose a two-pipeline approach for hand gesture recognition, building upon the work presented in [8]. Figure 3 depicts the two-pipeline approach, which leverages features from

both the image of the hand gesture and the 21 key-points on the palm. By incorporating information from both sources, our approach aims to improve the accuracy of hand gesture recognition.

The first pipeline involves extracting features from the image of the hand gesture using a convolutional neural network (CNN) architecture. This pipeline takes the input image and extracts relevant features, such as edges, corners, and shapes, from the image. We can use any CNN model architecture for this pipeline, depending on the task and the dataset.

The second pipeline involves extracting the 21 key-points using the Medipipe model from the same image as input. These key-points provide information about the orientation of the hand and the location of the joints in the palm. The key-points are then fed into a series of dense layers to learn and extract features related to the position of the joints and the orientation of the hand for that particular gesture. The output vectors from both pipelines are concatenated before being fed to a linear layer for classification. This linear layer takes the concatenated features as input and maps them to the eight output classes. After the linear classification, a softmax function is applied to the output of the linear layer to determine the predicted output class, which is the class with the highest probability. The softmax function is mainly used for multi-class classification problems as it is used to convert the output of a model into a probability distribution over the given classes. It is defined as

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{1}$$

where $\mathbf{z} = [z_1, z_2, \ldots, z_K]$ is the input vector of real numbers, and K is the number of classes. The output of the softmax function makes sure that the probabilities sum to 1, and that each probability is between 0 and 1. By using the linear layer and the softmax function in this approach, the input gestures are accurately classified into one of the eight output classes.

3.4 Models

For the two-pipeline approach [4], two different models are chosen: one for extracting features from the image and another consisting of dense layers that are utilized to extract features from the key-points.

3.4.1 DenseNet-201

The DenseNet architecture is based on the idea of dense connectivity, which is achieved by connecting each layer to all previous layers in a feedforward fashion. Specifically, in a DenseNet, each layer receives the feature maps of all preceding layers as input, and its own feature maps are passed on to all subsequent layers. This creates a "dense block" of layers that are densely connected to each other. The introduction of dense connectivity provides a new way to combat the problem of vanishing gradients in deep neural networks.

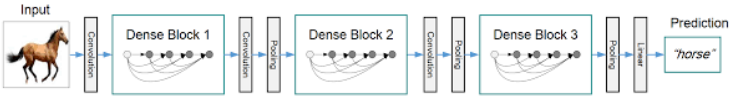

Figure 4: A deep DenseNet with three dense blocks

To facilitate the integration of these dense blocks into a full CNN architecture, the authors introduce a "transition layer" that consists of a batch normalization layer, a 1x1 convolutional layer, and a pooling layer. The transition layer reduces the dimensionality of the feature maps and compresses information before passing it on to the next dense block.

The growth rate determines the number of feature maps that are added to the input of each layer in a dense block. This growth rate is a hyperparameter that can be adjusted to control the number of parameters in the network.

The DenseNet-201 architecture consists of multiple dense blocks, each containing several convolutional layers that are densely connected to each other. The growth rate in DenseNet-201 is set to 32, which means that each layer adds 32 feature maps to the input of the next layer. The network also includes several transition layers that reduce the spatial dimensionality of the feature maps and compress the information before passing it on to the next dense block.

Overall, DenseNet-201 has 201 layers, which makes it a relatively deep network compared to other CNN architectures. However, because of the dense connectivity and efficient use of parameters, DenseNet-201 has a much smaller number of parameters than other deep networks, such as ResNet-200. This allows DenseNet-201 to achieve state-of-the-art performance on several benchmark datasets, including ImageNet while using significantly fewer parameters than other CNN architectures.

DenseNet-201 has been shown to be particularly effective for tasks that require fine-grained visual recognition, such as object detection and segmentation. The pre-trained model can be fine-tuned for these tasks using transfer learning, where the initial layers of the network are frozen and the later layers are fine-tuned on a specific dataset.

3.4.2 Dense Layers

The input image is fed to a Mediapipe model to extract the coordinates of 21 points on the palm which represents the location of the prominent joints in the arm. This results in a vector of size 42, which consists of x, and y image coordinates for each of the 21 points. These coordinates are then fed into the dense layers which are made up of three linear layers as mentioned in [8], each with an output size of 64. These dense layers are used to extract information on the orientation of the hand with respect to the 21 points extracted. To add non-linearity to the model, a Rectified Linear Unit (ReLU) activation function is used after each linear layer. The ReLU function is defined as

$$Relu(z) = max(0, z) \tag{2}$$

where all negative values are set to zero and positive values remain unchanged. The use of ReLU activation functions in the dense layers allows the model to learn and model complex relationships between the input vector and the output classes. To prevent overfitting, a dropout layer was added after each ReLU activation function. Dropout layers randomly drop out a certain percentage of neurons during training, which helps prevent the model from relying too heavily on any one feature. This improves the model's ability to generalize to new data and improves its overall performance.

Figure 4 below shows the components of the dense layers. As seen, the input vector of size 42 is first passed through a linear layer with an output size of 64. The output of this layer is then passed through a ReLU activation function and a dropout layer. This process is repeated two more times, resulting in a final output vector of size 64. Fig 4 below shows the components of the dense layers

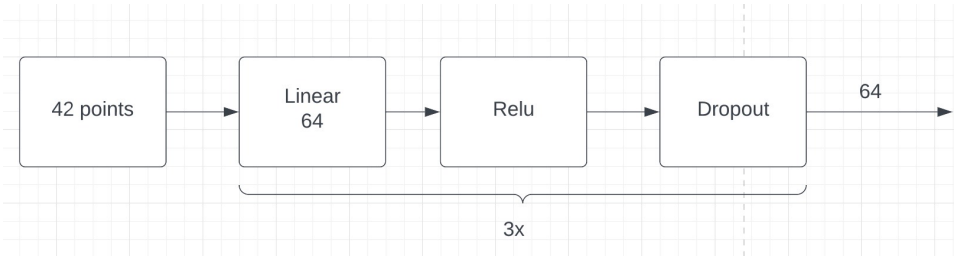

Figure 5: Dense layer pipeline

4 Experiments

This paper proposes a two-pipeline approach for classifying hand gestures. The first pipeline extracts feature from the hand key-points, using a model architecture with three linear layers followed

by a ReLU and dropout layer. The second pipeline uses three different pre-trained CNN model architectures as feature extractors for the input image: MobileNet V3-tiny, ResNet 101, and DenseNet 201. The reason for evaluating these models was twofold: MobileNet V3-tiny is a lightweight model, whereas ResNet 101 is a deeper model. Additionally, DenseNet 201 was chosen as the baseline model, as it is a state-of-the-art architecture for object classification and achieved an accuracy of 97.55 [4] on the Hagrid Dataset. An Adam optimizer with a learning rate of 0.0002 was used for training, and the models were trained for 30 epochs. Since this is a multi-class classification problem, the cross-entropy loss was used as the loss function to measure the difference between the predicted and actual probability distributions of the target classes. The cross-entropy loss for multi-class classification is calculated using the following formula:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{3}$$

Where M is the number of classes, y is the one-hot encoded ground truth label, p is the predicted probability distribution over the classes, and o is the index of the correct class. Three primary metrics were used for evaluating the models, namely: accuracy, precision, and recall, which are defined as follows:

1. Accuracy: The percentage of correctly classified instances among all instances. Where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$
2. Precision: The proportion of correctly predicted positive instances among all predicted positive instances.
$$Precision = \frac{TP}{TP + FP} \tag{5}$$
3. Recall: The proportion of correctly predicted positive instances among all actual positive instances.
$$Recall = \frac{TP}{TP + FN} \tag{6}$$

In Table 2, the results obtained from our two-pipeline method using different feature extractor models for hand gesture classification are presented.

The results indicate that Resnet-101 outperforms MobileNetV3-tiny in terms of metrics. This is because Resnet-101 is a deeper model with a higher number of parameters, allowing it to extract more features from the images. In contrast, MobileNetV3-tiny may not have been able to learn enough features to achieve comparable performance. DenseNet-201, which is currently considered the state-of-the-art model for image classification, outperformed Resnet-101 and MobileNetV3-tiny, achieving an accuracy of 0.971. One possible explanation for the superior performance of DenseNet-201 is its dense connectivity pattern, which enables the model to extract more relevant features from the images. The dense connections between layers allow for better information flow and reuse of features, leading to more efficient learning and higher accuracy. Additionally, the high number of parameters in DenseNet-201 provides the model with more capacity to learn complex features from the images.

Model	Accuracy	Precision	Recall
DenseNet-201	0.971	0.96	0.965
ResNet-101	0.938	0.937	0.92
MobileNetV3-tiny	0.828	0.826	0.815

Table 2: Results of different models

In Figure 6, we present the confusion matrix generated for the 8 classes in the train set using DenseNet-201 as the feature extractor model for the images. The confusion matrix is a visualization tool that

helps evaluate the performance of the model by showing the number of correctly and incorrectly classified samples for each class.

Our results indicate that the model performed well in identifying most of the hand key-points accurately. The diagonal cells of the confusion matrix represent the correctly classified samples, indicating that the model achieved high accuracy for most classes. The confusion matrix depicts that the model had some difficulty accurately identifying certain classes. For instance, the classes "two," "three," and "one" were sometimes confused with the number of fingers shown in the image, leading to misclassifications. Moreover, the class "ok" had the lowest accuracy among all the classes due to the presence of false positives corresponding to other classes, such as "palm," "fist," and "call."

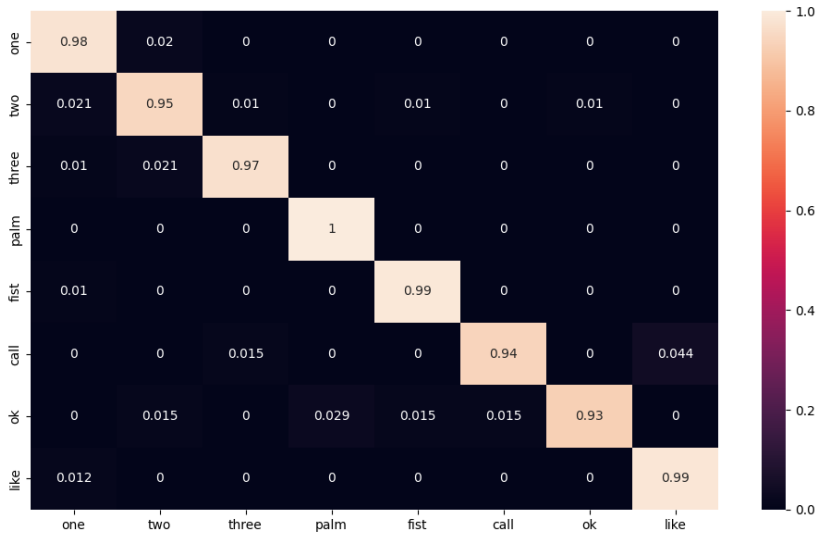

Figure 6: DenseNet-201 Confusion Matrix

5 Conclusion

The experimental results provide compelling evidence that our optimized two-pipeline model architecture achieves almost the same accuracy as the baseline model which is 0.975, but with significantly fewer epochs of training. Specifically, we were able to achieve comparable results using only 30 epochs of training, as opposed to the 100 epochs required for the baseline model. This finding suggests that the model was able to converge faster and was able to optimize the model training process to achieve maximum performance with fewer computational resources and less time.

6 Future Scope

The field of hand gesture recognition and sign language interpretation is a rapidly evolving area with numerous opportunities for future research and development. One potential avenue is the development of an end-to-end pipeline for hand gesture recognition that incorporates both key-point extraction and gesture classification. This pipeline could have far-reaching implications for a range of applications, including virtual reality, gaming, and human-computer interaction.

In addition, there is a need to expand training models to incorporate sign language interpretation. This will require researchers to address technical challenges such as temporal variations in gesture

appearance and tracking the movement of hands over time. Another promising direction is to explore gesture classification for continuous movements, which poses significant challenges that need to be overcome to achieve accurate results.

Further research can also explore the integration of other modalities, such as voice recognition and facial expression analysis, to improve the accuracy and robustness of gesture recognition and interpretation systems. Overall, the field of hand gesture recognition and sign language interpretation has tremendous potential for future advancements, and there is a need for continued research in this area to develop more efficient and accurate systems.

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Kapitanov, Alexander, Andrew Makhlyarchuk, and Karina Kvanchiani. "HaGRID-HAnd Gesture Recognition Image Dataset." arXiv preprint arXiv:2206.08219 (2022)

[3] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." arXiv preprint arXiv:1906.08172 (2019)

[4] Padhi, P. and Das, M., 2022, December. Hand Gesture Recognition using DenseNet201-Mediapipe Hybrid Modelling. In 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS) (pp. 995-999). IEEE.

[5] Sharma, A., Mittal, A., Singh, S. and Awatramani, V., 2020. Hand gesture recognition using image processing and feature extraction techniques. Procedia Computer Science, 173, pp.181-190.

[6] Dadashzadeh, A., Targhi, A.T., Tahmasbi, M. and Mirmehdi, M., 2019. HGR-Net: a fusion network for hand gesture segmentation and recognition. IET Computer Vision, 13(8), pp.700-707.

[7] Pinto, R.F., Borges, C.D., Almeida, A.M. and Paula, I.C., 2019. Static hand gesture recognition based on convolutional neural networks. Journal of Electrical and Computer Engineering, 2019, pp.1-12.

[8] Dang, T.L., Tran, S.D., Nguyen, T.H., Kim, S. and Monet, N., 2022. An improved hand gesture recognition system using keypoints and hand bounding boxes. Array, 16, p.100251.

[9] F. Zhan, “Hand gesture recognition with convolution neural networks,” in 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), 2019, pp. 295–298.

[10] Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700-4708. 2017.