ISSN 2355-3286

Hand Gesture Detection for American Sign Language using K-Nearest Neighbor with Mediapipe

Arsheldy Alvin[1], Nabila Husna Shabrina[2], Aurelius Ryo[3], Edgar Christian[4]
Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara, Teknik Komputer
Tangerang, Indonesia
[1] arsheldy.alvin@student.umn.ac.id, [2] nabila.husna@umn.ac.id, [3] aurelius.ryo@student.umn.ac.id,
[4] edgar.christian@student.umn.ac.id

Accepted 10 June 2021
Approved 27 June 2021

***Abstract*— The most popular way of interfacing with most computer systems is a mouse and keyboard. Hand gestures are an intuitive and effective touchless way to interact with computer systems. However, hand gesture-based systems have seen low adoption among end-users primarily due to numerous technical hurdles in detecting in-air gestures accurately. This paper presents Hand Gesture Detection for American Sign Language using K-Nearest Neighbor with Mediapipe, a framework developed to bridge this gap. The framework learns to detect gestures from demonstrations, it is customizable by end-users, and enables users to interact in real-time with computers having only RGB cameras, using gestures.**

***Index Terms*— hand gesture, neural network, mediapipe, image processing, touchless.**

I. Introduction

In this modern era, technology is growing rapidly. One of the goals of existing technologies is to facilitate human life. Every existing device always has an interface that allows the user to control the device. This interface is always evolving from physical buttons, touch screens, to no-touch at all like voice commands and hand gestures.

Currently, there are many devices that use voice commands, especially on smartphones that we often use, voice commands themselves are based on speech recognition algorithms, ranging from being used to type text to performing commands to AI, such as Google Assistant, Siri, Google Home, and Alexa uses voice commands to control it. However, the voice command itself has several shortcomings in its implementation, namely the sound around the environment should not be too noisy so that the commands ordered can be delivered properly and the system is not slow when processing incoming voice [1] . Therefore, another alternative to interact with the computer is to use hand gestures. Hand gestures can facilitate user mobility and flexibility in using a device or program. Coupled with the presence of the Covid-19 pandemic which requires maintaining a distance so that control with touch is enough to give its own sense of worry.

In addition to being an option for the interface of a device, hand gesture detection itself can also be applied to the communication system for friends who have hearing problems or are deaf. [16] Based on the official website of the United Nations (UN), the World Federation of the Deaf (WFD) states that there are around 72 million people who have hearing problems worldwide as of 2020. One of the communication media for deaf friends is to use sign language.

Therefore, we created a project about hand gesture detection that can detect the movement or pose of the hand to be implemented in sign language recognition. The method used to perform hand gesture detection, of course, is to use computer vision which goes through several adjustments and filters on the hand. Then, the data obtained will be processed to be able to provide the appropriate output.

We use the framework from MediaPipe to detect the hand on the camera, and the K-Nearest Neighbor (KNN) algorithm to predict the gesture on the hand. MediaPipe is a cross-platform framework for building multimodal applied machine learning pipelines that can be used to detect hands. KNN is an algorithm used to predict the hand gesture displayed on the camera. The reason we chose MediaPipe to detect hands is that it is easy to use and can provide output in the form of hand landmark coordinates [14]. Furthermore, the KNN method was chosen because it only needed to classify an alphabet using hand coordinates from the MediaPipe output plus the use of KNN which was quite easy to implement and had a fairly high accuracy value. This can be proven by various kinds of research conducted [17]. We also use a sign language dataset with the American Sign Language standard, please note that not all sign languages are the same, for example people who already know American Sign Language will not understand when communicating with people who use

British Sign Language [2]. The reason for choosing American Sign Language as the dataset is that American Sign Language is the main sign language used in the United States and has been known and adopted by several countries around the world, one of which is Indonesia, namely SIBI (Indonesian Sign Language System) which developed from ASL absorption. SIBI itself has been inaugurated in Law No. 2 of 1989 and is an introduction to communication in the Special School (SLB) curriculum [15]. However, almost every country has its own sign language. According to the European Union of the Deaf, on its website it is explained that there is no universal use of sign language in the world [18]. The method starts by creating a dataset containing the hand coordinates of the alphabet. The creation of this dataset uses the help of MediaPipe which will provide hand coordinates from hand drawings. After that the dataset will become data to train the KNN model in order to classify the alphabet.

II. Literature Review

A. Hand Tracking

There are various kinds of literature that discuss how to do hand tracking via video or static images. To achieve this, a camera that can produce RGB images is needed [10]. The camera will provide output in the form of images or videos to the MediaPipe framework [3]. MediaPipe will first run the palm detector on the image. Palm detector on MediaPipe works using a model that works like BlazeFace [4]. To detect the hand, it is very difficult to detect the fingers at once, therefore, MediaPipe uses a model that will detect the palm first. This is because the palm is a fairly small object and can be modeled easily using the Bounding Box method which will not be affected by the aspect ratio [5]. The model will then be run by mapping 21 coordinates on the detected palms. This model will provide output in the form of 21 coordinates of the fingers, a marker of the presence of a hand or not, and a marker of the left or right hand.

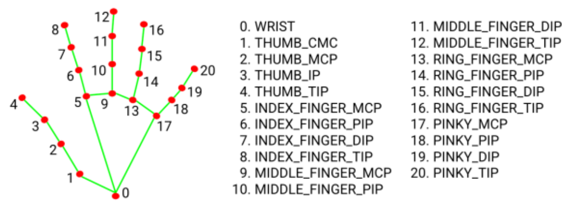


Fig. 1. Hand Landmark

B. K Nearest Neighbor Classifier

K Nearest Neighbor or abbreviated KNN, is a supervised learning method that classifies classes based on the k closest neighbors. Where the purpose of this method is to classify new objects based on previous data. KNN classification is a well-known method used for image classification [12]. KNN itself first works by determining the value of k which determines the number of neighbors to be used. Then the distance from the k neighbors will be calculated using the Euclidean distance.

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}. \quad (1)$$

After calculating the distance, k neighbors will be taken according to the results of the distance calculation. Then from the k neighbors, the number of data points for each class that is included in the calculation distance will be calculated. After that, to determine the new data class, the largest number of neighbors from the class will be seen. In KNN itself, there are various ways to determine distance and similarity [6]. Although this KNN method is easy to implement, there are various weaknesses, namely poor performance in overcoming data that has very large dimensions and performance that is influenced by the magnitude of the value of k neighbors [8].

C. Data Scaling

Data scaling is a process that is carried out before classifying data. This process will transform the existing value into the existing scale. To achieve this process can be done normalization or standardization of data [9]. This normalization process is carried out with the aim of distributing data evenly and increasing the value of system accuracy. There are several normalization techniques including min-max normalization, z-score normalization, decimal scaling and sigmoidal normalization.

$$Z-SCORE = \frac{X_i - \overline{X}}{SD} \quad (2)$$

III. Methodology and Implementation

A. Design Stage

Hand Gesture Detection for Sign Language is intended to detect hands and provide output in the form of letters from the sign language demonstrated by the user's hand. To achieve this, the researcher uses the MediaPipe framework to detect hands and cv2 to do hand capture via a webcam as well as several libraries that support data processing, namely matplotlib, pandas, numpy, and sklearn. Matplotlib, pandas, and numpy are used for reading dataset files to plotting data, while sklearn is used for training data. The following is an architectural design for alphabet gesture recognition.

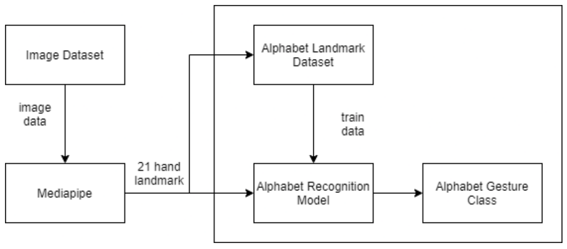


Fig. 2. Project Architecture

B. *Implementation Stage*

The programming language used in this project is Python. To facilitate the design, the researcher used the Jupyter Notebook coding environment which was run on the 2019 ASUS Vivobook A412FL laptop with the following specifications :

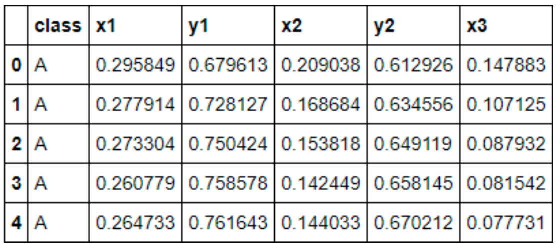
CPU : Intel Core i5-8265U CPU @ 1.60GHz
GPU : NVIDIA GeForce MX250 2GB
RAM : 8GB

To implement hand gesture detection for sign language, several stages are needed, namely:

1. *Creating a sign language dataset in the form of coordinates*

This is due to the unavailability of a sign language dataset that has the coordinates of 21 landmarks belonging to Mediapipe. The dataset used is an image dataset which contains a total of 87.000 images, with each image having a size of 200 x 200 pixels. There are 29 classes in the dataset consisting of 26 alphabets and 3 classes of space, delete, and nothing. After finding the dataset, each image in the dataset needs to be modified using the MediaPipe Hands API [7] which can output hand landmark coordinates from static images. Each landmark output from MediaPipe will be assigned a class according to the file name and inserted into the csv table. [13] Every 21 landmarks produced by MediaPipe will be entered in the form of values x[i] and y[i] where i denotes the landmark number in Fig. 1. So that each alphabet will have 42 features consisting of x1 and y1 to x21 and y21. Referring to the weakness of KNN, namely KNN is prone to high dimensionality where this can make the space that can be occupied by each instance bigger so that there is a possibility that the nearest neighbor of an instance can no longer be said to be "near" because the dimensions of the instance space increase [11], the researchers choose to use only the landmark x[i] and y[i] even though there are up to z[i] where x[i] and y[i] represent the width and height, respectively. While z[i] represents the depth, whose value will decrease when the landmark is getting closer to the camera. When using the z value, each alphabet will produce 63 features. This can cause a decrease in performance in the KNN model, because KNN cannot handle very large dimensional data. So the researchers decided that only using x[i] and y[i] would provide more optimal performance.

TABLE 1. DISPLAY OF OUTPUT IN CSV

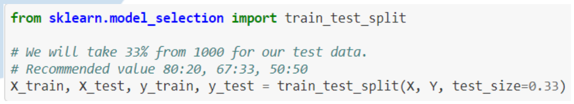


	class	x1	y1	x2	y2	x3
0	A	0.295849	0.679613	0.209038	0.612926	0.147883
1	A	0.277914	0.728127	0.168684	0.634556	0.107125
2	A	0.273304	0.750424	0.153818	0.649119	0.087932
3	A	0.260779	0.758578	0.142449	0.658145	0.081542
4	A	0.264733	0.761643	0.144033	0.670212	0.077731

In this implementation, the researcher created 1000 training datasets with 24 alphabets. The making of 1000 training datasets is because not all data in the image dataset is readable, then the researcher only uses 24 alphabets, because the J and Z alphabets use movement so further implementation is needed to read movement.

2. *Machine learning model validation*

The dataset used in this project is a dataset that has just been previously created so that researchers do not have a test dataset to validate or evaluate machine learning models. Therefore, the researcher uses the train/ test split method which is used to validate the model with each data being processed and divided into training and testing data. This split is done in a 67:33 ratio with the help of sklearn's train_test_split library.



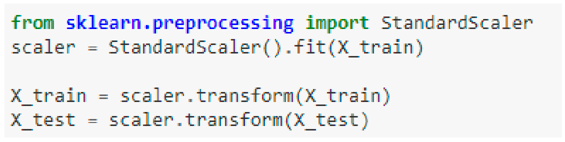
```
from sklearn.model_selection import train_test_split

# We will take 33% from 1000 for our test data.
# Recommended value 80:20, 67:33, 50:50
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33)
```

Fig. 3. Display of the train/test split process for model validation

3. *Scaling the dataset*

The divided dataset is scaled with the StandardScaler method from the sklearn library. This method works like z-score normalization. In accordance with the previous chapter, this process serves to improve the performance of the KNN algorithm.



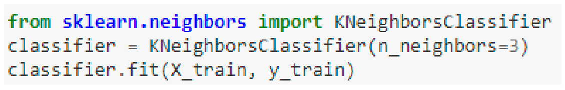
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Fig. 4. Display of scaling data process

4. *Generating classification model*

In making the classification model, the researcher uses the KNeighborsClassifier library belonging to sklearn. This library makes it easy to create KNN methods. In this model, the researcher uses 3 neighbors with an automatic algorithm from the library, the default weight function is uniform, the default metric is minkowski.



```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(X_train, y_train)
```

Fig. 5. Display of fitting model classifier process

IV. Result

A. Testing Scenario

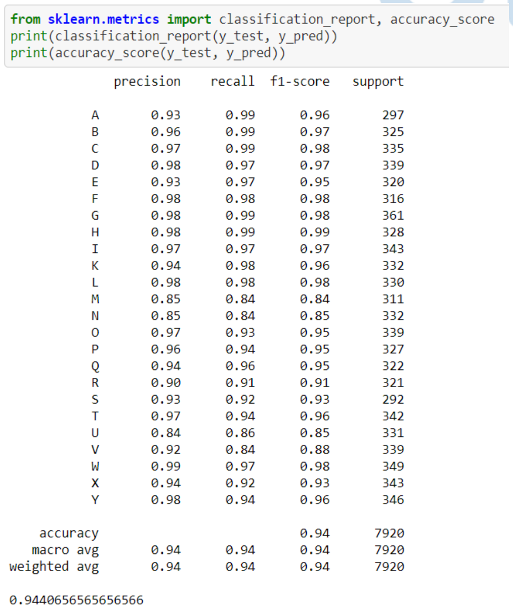
Researchers conducted 2 types of testing, namely testing with the help of a library and direct testing. For testing with the help of the library, the researcher aims to determine the accuracy value of the existing model. Meanwhile, in direct testing, the researcher aims to try this project in real-time.

Direct testing was carried out by 2 examiners with 2 significantly different conditions. The following is an explanation of the test conditions :

1. Examiner A
   Device : Laptop Lenovo G470 i5-2410M (2011 released)
   Brightness : dim
   Amount of testing : 2 times for every alphabets
2. Examiner B
   Device : Laptop ASUS VivoBook A412 FL i5-8265U (2019 released)
   Brightness : bright
   Amount of testing : 2 times for every alphabets

B. Result with Library Assistance

After the fitting process with the KNN method was successful, the researcher calculated the accuracy of the existing model with the help of sklearn's classification_report and accuracy_score. From this model, the researcher obtained an accuracy of 0.944 from a scale of 0-1, which means that it shows good accuracy.



```
from sklearn.metrics import classification_report, accuracy_score
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           A       0.93      0.99      0.96       297
           B       0.96      0.99      0.97       325
           C       0.97      0.99      0.98       335
           D       0.98      0.97      0.97       339
           E       0.93      0.97      0.95       320
           F       0.98      0.98      0.98       316
           G       0.98      0.99      0.98       361
           H       0.98      0.99      0.99       328
           I       0.97      0.97      0.97       343
           K       0.94      0.98      0.96       332
           L       0.98      0.98      0.98       330
           M       0.85      0.84      0.84       311
           N       0.85      0.84      0.85       332
           O       0.97      0.93      0.95       339
           P       0.96      0.94      0.95       327
           Q       0.94      0.96      0.95       322
           R       0.90      0.91      0.91       321
           S       0.93      0.92      0.93       292
           T       0.97      0.94      0.96       342
           U       0.84      0.86      0.85       331
           V       0.92      0.84      0.88       339
           W       0.99      0.97      0.98       349
           X       0.94      0.92      0.93       343
           Y       0.98      0.94      0.96       346

    accuracy                           0.94      7920
   macro avg       0.94      0.94      0.94      7920
weighted avg       0.94      0.94      0.94      7920

0.9440656565656566
```

Fig. 6. Classification Report & Accuracy Score

In addition to the accuracy score, the researcher also plots the Error Rate K Value with a K value range from 1-40 which can be used to adjust what is the best K value to use.

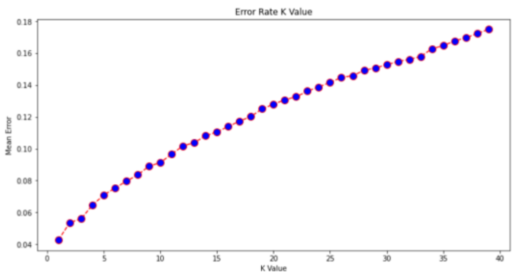


Fig. 7. Error Rate K Value Graph

C. Result of Direct Testing

In the final stage, the researcher conducted a trial on the existing model directly by detecting the hand and obtaining an alphabet that was in accordance with the American Sign Language that had been included in the dataset. Researchers use the help of the cv2 library to do hand capture via webcam and the Mediapipe framework to detect hands and obtain the coordinates of hand landmarks captured by the webcam and then predict what alphabet is being demonstrated based on existing models.

The test results are as follows :

1. Examiner A :

   The alphabet can be predicted when the webcam with the help of the Mediapipe framework detects the presence of a hand so that in low light conditions, the distance between the hand and the webcam must be closer to be readable by the webcam. As long as the webcam reads the presence of a hand, the expected output of the alphabetical prediction will appear. From 2 trials for all alphabets, all alphabets can be predicted with some adjustment of hand distance with the webcam. So, it can be concluded that even in dim conditions as long as the hand can still be read by the webcam, the sign language will still be predictable.

2. Examiner B :

   With good lighting, the entire alphabet can be predicted well because the webcam is able to capture a clear hand image so that the Mediapipe framework is able to recognize every part of the hand that practices sign language well.

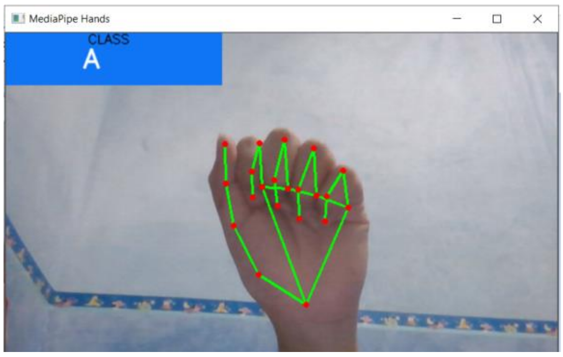


Fig. 8. *Hand Gesture* Alphabet A

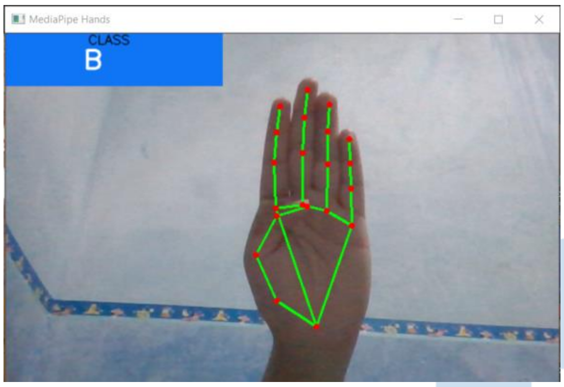


Fig. 9. *Hand Gesture* Alphabet B

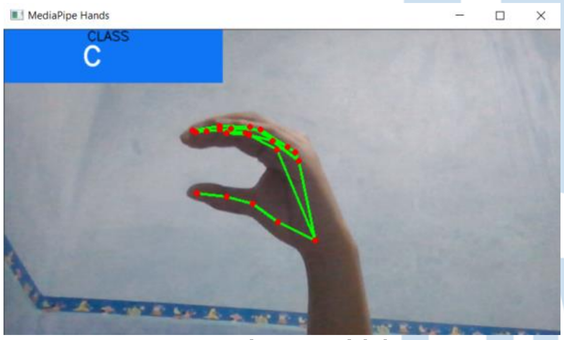


Fig. 11. *Hand Gesture* Alphabet C

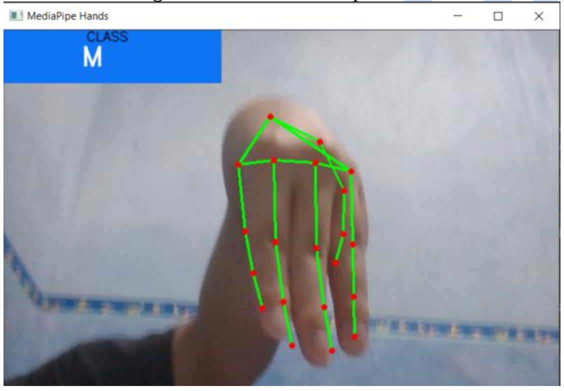


Fig. 10. *Hand Gesture* Alphabet M

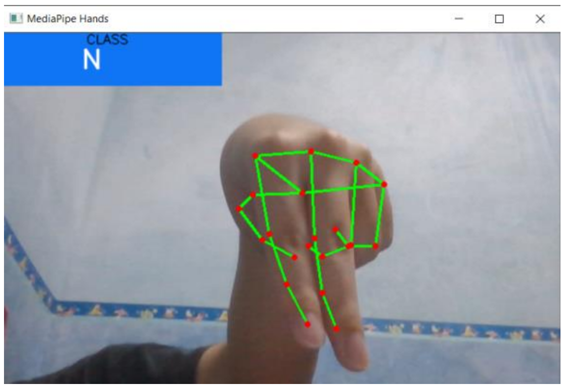


Fig. 11. *Hand Gesture* Alphabet N

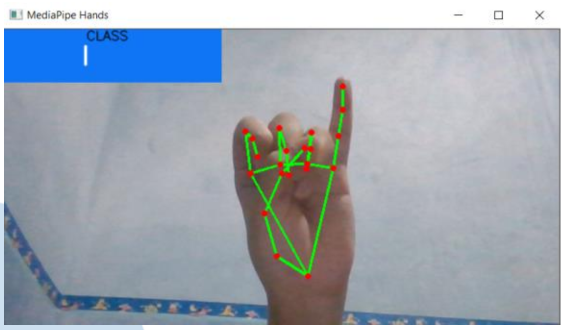


Fig. 12. *Hand Gesture* Alphabet I

Above are some of the results of the American Sign Language demonstration trials which were then detected and translated into the alphabet. Almost all alphabets can be read and translated well, except for some alphabets that may be a bit difficult to detect such as J and Z. However, apart from that, some alphabets that have similarities to their coordinate points, such as the M and N alphabets can still be distinguished and legible even though almost similar.

V. Conclusion

In this research, we propose a system that can detect sign language with the American Sign Language standard that can achieve real-time detection performance on desktop platforms, the workings of the researcher's detection system is based on the coordinates of 21 hand landmarks generated by MediaPipe. From these coordinates there are x, y, and z values for mapping hand gestures. The researcher suggests using only x and y values to improve the performance of the KNN model. This is due to the weakness of KNN in dealing with large-dimensional data. Then there are weaknesses in the system that the researcher proposes, namely the lack of datasets. To achieve an optimal system, a large number of datasets are needed so that it can detect the alphabet from various angles. Later, this system can be improved by using the Modified KNN model to obtain better results.

REFERENCES

[1] Yoshua Constantin, Ucuk Darusalam, and Novi Dian Nathasia. (2020). Aplikasi Personal Assistant Berbasis Voice Command Pada Sistem Operasi Android Dengan NLP.https://www.researchgate.net/publication/341791181_Aplikasi_Personal_Assistant_Berbasis_Voice_Command_Pada_Sistem_Operasi_Android_Dengan_NLP/fulltext/5ed51350299bf1c67d323d04/Aplikasi-Personal-Assistant-Berbasis-Voice-Command-Pada-Sistem-Operasi-Android-Dengan-NLP.pdf

[2] Niki's Int'l Ltd. (2017). The Different Types of Sign Language.https://nilservices.com/different-types-sign-language/

[3] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., &amp; Grundmann, M. (n.d.). MediaPipe Hands: On-device Real-time Hand Tracking. 2006.10214.pdf. https://arxiv.org/pdf/2006.10214.pdf.

[4] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus.

[5] Tomas Simon, Hanbyul Joo, Iain A. Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. CoRR, abs/1704.07809, 2017.

[6] Padraig Cunningham, Sarah Jane Delany: "k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples)", 2020; arXiv:2004.04523.

[7] MediaPipe. (n.d.). MediaPipe Hands. mediapipe. https://google.github.io/mediapipe/solutions/hands.html#python-solution-api.

[8] Hassanat, Ahmad & Abbadi, Mohammad & Altarawneh, Ghada & Alhasanat, Ahmad. (2014). Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach. International Journal of Computer Science and Information Security. 12. 33-39.

[9] Muhammad Ali, Peshawa & Faraj, Rezhna. (2014). Data Normalization and Standardization: A Technical Report. 10.13140/RG.2.2.28948.04489

[10] Utaminingrum, F., Somawirata, I. K., &amp; Naviri, G. D. (2019). Alphabet Sign Language Recognition Using K-Nearest Neighbor Optimization. Journal of Computers. https://doi.org/10.17706/jcp.14.1.

[11] Puspadini, Ratih. (2020). Seleksi Atribut Pada Algoritma K-Nearest Neighbor Menggunakan Similarity Distance Measures.http://repositori.usu.ac.id/bitstream/handle/123456789/24587/157038008.pdf

[12] Mufarroha, Fifin & Utaminingrum, Fitri. (2017). Hand Gesture Recognition using Adaptive Network Based Fuzzy Inference System and K-Nearest Neighbor. International Journal of Technology. 8. 559. 10.14716/ijtech.v8i3.3146.

[13] S K, Sriram & Sinha, Nishant. (2021). Gestop: Customizable Gesture Control of Computer Systems. https://arxiv.org/pdf/2010.13197.pdf

[14] MediaPipe. (n.d.). MediaPipe Hands. mediapipe. https://google.github.io/mediapipe/solutions/hands.html#output

[15] Taufik, Miskudin. (2020). Bahasa Isyarat Menyatukan Dunia.https://itjen.kemdikbud.go.id/public/post/detail/bahasa-isyarat-menyatukan-dunia

[16] United Nations. (2020). Sign Languages Are for Everyone. https://www.un.org/en/observances/sign-languages-day

[17] Trigueiros, Paulo & Ribeiro, Fernando & Reis, Luís. (2012). A comparison of machine learning algorithms applied to hand gesture recognition. Iberian Conference on Information Systems and Technologies, CISTI. 41-46.

[18] European Union of the Deaf. (2012). International Sign. https://www.eud.eu/about-us/eud-position-paper/international-sign-guidelines/

UMN