Illinois Institute of Technology

College of Computing

Static Hand Gesture Recognition

Author:
Norah Khalaf Alotaibi

Supervisor:
Yan Yan

1 Introduction

1.1 Problem description

Hand gestures form the main component of sign language vocabulary. Hand poses mainly constitute finger spelling of sign language vocabulary, which is used letter by letter to sign names, place names, ages, numbers, dates, years, and words that do not have predetermined signs. Hand gestures can be static or dynamic. Thus, automatic hand position recognition has been a hot research area, and there are many works on the same approach using vision-based and electronic signal-based methods. Vision-based techniques are more user-friendly and convenient than others when considering the complexity of the data acquisition process.

The difficulty of recognizing static hand gestures in the visual approach depends on many factors. First, the most important is the difficulty of detecting and segmenting hands from images taken with a complex background. Second, it is the difficulty of deriving good features that distinguish geometric differences in the appearance of the same hand position shown by different individuals and many challenges that Cause machine learning techniques to fail to recognize hand position from images / or videos.

1.2 Related research

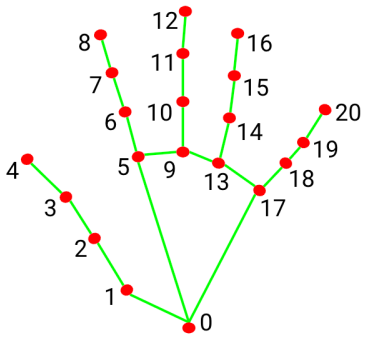
Most existing works on vision-based hand gesture recognition mainly follow the steps of classical pattern analysis that go through image/video pre-processing, feature extraction, and classification. Cao et al.[1] proposed a hand posture recognition approach with heterogeneous feature fusion and obtained a recognition accuracy of 0.9916 on the publicly available Triesh data set. They trained a multiple kernel support vector machine classifier with feature vectors obtained by combining the shape context feature, pyramidal HOG feature, and SIFT-based bag of features for recognizing the various static hand gestures. Pugeault et al.[5] presented a multiclass random forest classifier to recognize 24 static signs in ASL (American Sign Language) alphabet with the features extracted through Gabor filters in four levels. They claimed a recognition rate of 0.49

Despite promising results, the classic methods failed to recognize hand position in real-time scenarios. The main reason for this failure is the failure of traditional machine learning techniques to infer and recognize patterns from the input data accurately. The most critical challenges faced by the approach to identifying the position of the hand are the following:

- Difficulty detecting and segmentation hands from images taken with a complex background.
- Difficulty eliciting robust features that characterize geometric differences in the appearance of the same hand posture shown by different individuals.
- Having many categories of gestures with very little difference between the categories is another complex problem, especially in the automatic recognition of sign language.

2 Methods

This project aims to take advantage of the Mediapipe framework [3], provided by Google, which contains several ML solutions, including Hands Mediapipe [2], which is a high-resolution solution for real-time hand and finger tracking to infer (21) 3D landmarks of the hand, As in Figure (1).



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 1: Hand landmarks

This project proposes to build a hand descriptor that produces a set of features describing the hand, depending on the parameters of the hand 21 (joints). Such as, creating descriptors represents the relationship of a finger with other fingers (intersection, parallelism, etc.) and a relationship between a joint with a finger or with another joint, in addition to the direction of the hand and many hand descriptors. Hand descriptors will be represented through relationships and mathematical equations such as the angle between two vectors and the ratio of the intersection of two vectors, etc. What distinguishes the resulting descriptors from the proposed hand descriptor is that it is not affected by the hand's shape, size, or color because most of these descriptors are angles and proportions. Also, the proposed hand descriptor can be used in any project because the hand is not specific to sign language.

In the remaining parts of this report, the stages of the project that we carried out will be presented.

3 Feature extraction

The huge data set often contains a large number of variables that require a lot of computing resources in order to process them. In order to reduce these resources, one of the dimensions reduction processes is used. One of the processes used to reduce dimensions is feature extraction, which is concerned with finding the best features that describe the dataset accurately and authentically.

The importance of dimension reduction processes lies in building a machine model with less effort by increasing the learning speed and generalization steps. The most prominent areas that use the process of feature extraction is the field of computer vision.

Through a group of hand joints that are detected as coordinate points. We produced 54 features that describe the shape and condition of the hand, through the use of some mathematical concepts in linear algebra.

Hand prescriptions have been classified into two categories (general hand prescriptions, and secondary hand descriptions). The following is a presentation of the descriptions for each class.

3.1 General hand descriptors

They are those descriptors that describe the general condition of the hand, which are three main characteristics (type characteristic, appearance characteristic, direction characteristic) as shown in Figure (2).

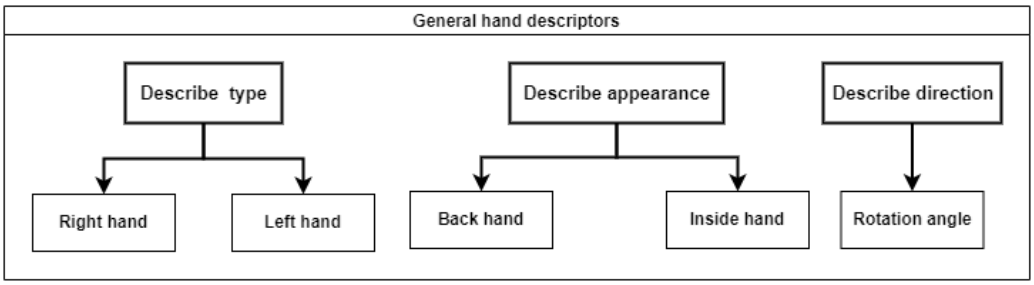


Figure 2: General hand descriptors

3.1.1 Describe type

This feature results from the algorithm used to detect (medapipe) about the hand, as it predicts the type of hand (left-right). Through it, differentiation is made in the production of other descriptors, which depend on the type of hand to change the methods of extraction, due to the different positions of the joints of the right hand from the positions of the joints of the left hand. Representation of this trait as in the relation (1).

$$f_1 = \begin{cases} 1, & \text{if } Hand \text{ is left} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

3.1.2 Describe appearance

This adjective determines the appearance of the hand, which may be (the back of the hand, the inside of the hand). Distinguish between signs based on the appearance of the hand.

Vectors consisting of joints (P_0, P_{17}, P_5) are relied upon,These points only change when the hand is turned.

Then the vectors are drawn to the origin as in relation (3)

$$\begin{aligned} \vec{A} &= \overrightarrow{P_0P_{17}} = \begin{bmatrix} P_{17,x} - P_{0,x} & -P_{17,y} + P_{0,y} \end{bmatrix} \\ \vec{B} &= \overrightarrow{P_0P_5} = \begin{bmatrix} P_{5,x} - P_{0,x} & -P_5 + P_{0,y} \end{bmatrix} \end{aligned} \tag{2}$$

Calculate the angle between each vector and the positive x-axis

$$\begin{aligned} \theta(A) &= 2\pi + arctan2(A_y, A_x) \ \ mod\ 2\pi \\ \theta(B) &= 2\pi + arctan2(B_y, B_x) \ \ mod\ 2\pi \end{aligned} \tag{3}$$

Noting the following cases:

- If the angle between vector $\vec{B}$ and the positive x-axis is greater than the angle between vector $\vec{A}$ and the positive x-axis, then the appearance of the hand is the belly of the hand, noting that if the angle between vector $\vec{B}$ and the positive x-axis lies in the first quadrant and the angle between vector $\vec{A}$ And the positive x-axis lies in the fourth quadrant, so we reflect the sign of the angle of the vector $\vec{A}$

$$\theta(A) = \begin{cases} -\theta(A), & \text{if } \theta(B) < \pi/2 \text{ and } \theta(A) > 3\pi/2 \\ \theta(A), & \text{otherwise} \end{cases} \tag{4}$$

- If the angle between vector$\vec{B}$ and the positive x-axis is smaller than the angle between vector $\vec{A}$ and the positive x-axis, then the appearance of the hand is the back of the hand, noting that if the angle between vector $\vec{A}$and the positive x-axis lies in the first quadrant and the angle between vector $\vec{B}$ And the positive x-axis is located in the fourth quadrant, so we reflect the signal of the vector angle $\vec{B}$

$$\theta(A) = \begin{cases} -\theta(B), & \text{if } \theta(A) < \pi/2 \text{ and } \theta(B) > 3\pi/2 \\ \theta(B), & \text{otherwise} \end{cases} \tag{5}$$

The representation of the property is as follows:

$$f_2 = \begin{cases} 1, & \text{if } \theta(B) < \theta(A) \text{ Xor } (Hand \text{ is left}) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

3.1.3 Describe direction

The angle of rotation determines the degree of rotation of the hand in general. This angle is produced as follows:

We deal with the first joint, the middle finger, P_9, and the wrist vertebra,P_0

Create a vector from the two previous joints and drag it to the origin

$$\vec{A} = \overrightarrow{P_0P_9} = \begin{bmatrix} P_{9,x} - P_{0,x} & -P_{9,y} + P_{0,y} \end{bmatrix} \tag{7}$$

Calculate the angle between the vector $\vec{A}$ and the positive x-axis to be the required characteristic

$$f_3 = (2\pi + arctan2(A_x, A_y)) \ mod \ 2\pi \tag{8}$$

3.2 Secondary hand descriptions

The secondary characteristics of the hand represent the relationships between all the fingers and their joints.

3.2.1 Characteristics generated from one finger

The features resulting from one finger are represented in the relationships between the finger joints. According to the detection model of the hand, four joints are detected in each finger. The following are the most important characteristics:

- **resulting features of the finger joints**

  These angles represent the state of the joints of one finger, and it was relied on to form a vector between each two successive joints in the finger, as in the figure (3). Then calculate the angle between each two successive vectors, as in figure (4), where these angles represent the characteristics that represent the state of the joints in each finger

- **Feature of closing or opening the finger** This property distinguishes the general condition of the finger, i.e. the ratio of opening and closing to the finger. This property was determined by finding the angle between the two vectors as shown in Figure (6).

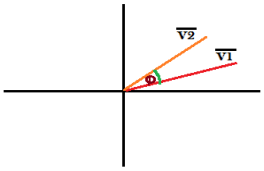


Figure 3: Finger joints

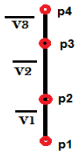


Figure 4: Finger joints vectors

3.2.2 Characteristics intersection of the fingers with thumb

A fixed point is installed, which is the midpoint of the triangle $(P_0P_5P_{17})$, and let it be the point M, as it is the point that is far from one of the arcs of the triangle emerging from the straight line passing through this point and falling perpendicular to the midpoint of the side opposite this vertex by 2/3, and the coordinates of this point are calculated as follows

- First, we must find the coordinates of the midpoint of the side $(\overrightarrow{P_0P_5})$ and let this point be C and it was found using the law of finding the coordinates of a point that divides a line segment with a certain ratio

$$C = \left[\frac{P_{5,x}+P_{0,x}}{2} \quad \frac{P_{5,y}+P_{0,y}}{2}\right] \tag{9}$$

- find the coordinates of the point M that divides the line $(P_{17}C)$ in a ratio of $1:2$

$$M = \left[\frac{2C_x+P_{17,x}}{3} \quad \frac{2C_y+P_{17,y}}{3}\right] \tag{10}$$

- After determining the midpoint of the triangle M, we calculate the resulting angles between the vectors that reach from point M to the joints of the thumb finger and the vectors that reach from point M to all the joints of the other fingers.As shown in Figure (7)

4 Classification algorithms

Training and testing were performed on an American Sign Language (ASL) character dataset [4]. Features are extracted and saved to a csv file. Figure (8) also shows which properties are the features.

After the process of exploring the data set, it was noticed that the data is somewhat balanced, with the exception of some class such as the letters (L, N). Figure (9) also represents the distribution of items in the data set.

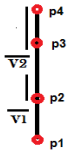


Figure 5: closing or opening vectors

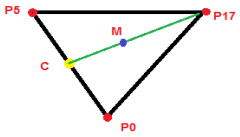


Figure 6: Find M and C points

The original data set is divided into a 70% training set and a 30% test set. The data is normalized using StandardScaler.

We now have a training dataset ready to be fed into our workbooks. I will do it as follows.

4.1 Support Vector Machines (SVM)

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

4.1.1 Mathematical formulation

Given training vectors $x_i \in \mathbb{R}^p$, $i = 1, \ldots, n$, in two classes, and a vector $y \in \{1, -1\}^n$, our goal is to find $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ such that the prediction given by $\text{sign}(w^T \phi(x) + b)$ is correct for most samples.

SVM solves the following primal problem:

$$\begin{aligned} \min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \\ \zeta_i \geq 0, i = 1, ..., n \end{aligned} \tag{11}$$

ntuitively, we're trying to maximize the margin (by minimizing $||w||^2 = w^T w$) , while incurring a penalty when a sample is misclassified or within the margin boundary. Ideally, the value $y_i(w^T \phi(x_i) + b)$ would be ≥ 1 for all samples, which indicates a perfect prediction. But problems are usually not always perfectly separable with a hyperplane, so we allow some samples to be at a distance ζ_i from their correct margin boundary. The penalty term C controls the strength of this penalty, and as a result, acts as an inverse regularization parameter.

The dual problem to the primal is

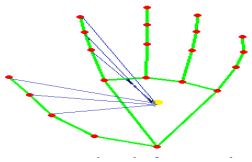


Figure 7: Intersection of the fingers with thumb

1.2 Exploratory data analysis

```
[5]: print("Dimensions of the data set = ",df.shape)
```

```
Dimensions of the data set =  (125262, 55)
```

We can see that there are 125262 image and 55 variables in the data set.

preview the dataset

```
[6]: df.sample(5)
```

t[6]:

4	5	6	7	8	9	...	45	46	47	48	49	50	51	52	53	target
3.043931	3.061419	2.416120	2.780019	2.273744	2.071630	...	1.926162	1.363408	1.007987	1.522690	0.959936	0.604516	1.309634	0.746880	0.391460	D
2.633254	1.773553	3.046755	3.098731	3.067175	3.132199	...	1.641033	1.715333	1.734753	0.702322	0.776622	0.796042	0.540370	0.466070	0.446651	U
2.428938	1.829068	0.236092	3.059619	0.197679	0.193520	...	1.179504	1.430496	1.315725	0.447653	0.698645	0.583875	0.219306	0.031685	0.083085	I
3.071958	2.726238	0.405685	2.988443	0.339305	0.133888	...	1.371885	1.546215	1.393353	1.031907	1.206237	1.053375	0.853171	1.027502	0.874640	Y
2.808614	2.586846	2.423262	2.762792	2.248462	2.695132	...	1.578974	1.492232	1.361479	1.202757	1.116015	0.985262	0.895278	0.808536	0.677783	F

Figure 8: data set Features extracted

$$\begin{aligned} \min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{subject to } y^T\alpha = 0 \\ 0 \leq \alpha_i \leq C, i = 1, ..., n \end{aligned} \tag{12}$$

where e is the vector of all ones, and Q is an n by n positive semidefinite matrix $Q_{ij} \equiv y_i y_j K(x_i, x_j)$

where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. The terms α_i are called the dual coefficients, and they are upper-bounded by C. This dual representation highlights the fact that training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ.

Once the optimization problem is solved, the output of decision function for a given sample x becomes:

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b \tag{13}$$

4.1.2 Experiments

Several experiments were carried out by implementing SVM with different hyperparameters and the following results were observed.

- **SVM with default hyperparameters**
  Default hyperparameter means C=1.0, kernel=rbf and gamma=auto among other parameters.

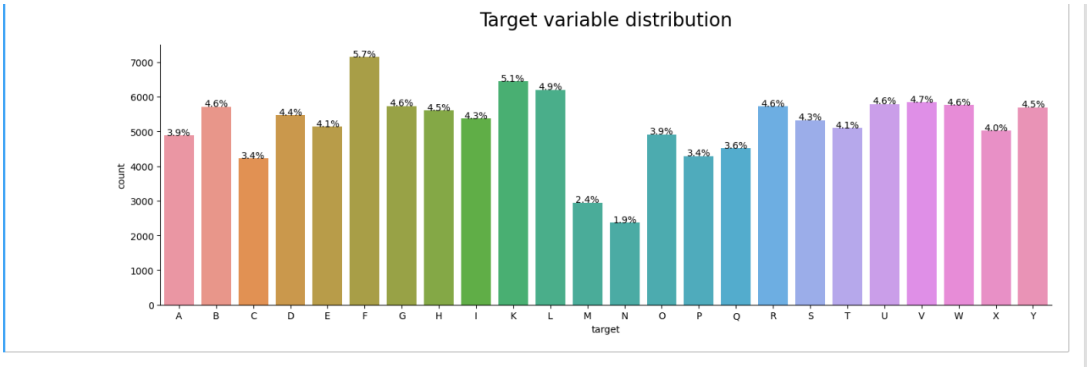


Figure 9: Target variable distribution

We obtained that the accuracy of the test data is 0.9859 and the accuracy of the training process is 0.9866.

here may be outliers in our dataset due to different representations of the same signal. So, we should increase the value of C because higher C means fewer outliers. So, I'm run an SVM with kernel=rbf and C=100.0.

- **SVM with rbf kernel & C=100.0**
  We obtained that the accuracy of the test data is 0.9951 and the accuracy of the training process is 0.9995.

  We can see that we obtain a higher accuracy with C=100.0 as higher C means less outliers.Now, I further increase the value of C=1000.0 and check accuracy.

- **SVM with rbf kernel & C=1000.0**
  We obtained that the accuracy of the test data is 0.9950 and the accuracy of the training process is 0.9998.

  In this case, we can see that the accuracy had decreased with C=1000.0

And I did many experiments with many different kernel functions (polynomial, linear) and the results were observed as in the figure (10).

```
Training set svm with rbf kernel & C= 1 score: 0.9866
Test set svm with rbf kernel & C= 1 score: 0.9859
Training set svm with rbf kernel & C= 100 score: 0.9995
Test set svm with rbf kernel & C= 100 score: 0.9951
Training set svm with rbf kernel & C= 1000 score: 0.9998
Test set svm with rbf kernel & C= 1000 score: 0.9950
Training set svm with linear kernel & C= 1 score: 0.9787
Test set svm with linear kernel & C= 1 score: 0.9760
Training set svm with linear kernel & C= 100 score: 0.9835
Test set svm with linear kernel & C= 100 score: 0.9787
Training set svm with linear kernel & C= 1000 score: 0.9839
Test set svm with linear kernel & C= 1000 score: 0.9784
Training set svm with poly kernel & C= 1 score: 0.9873
Test set svm with poly kernel & C= 1 score: 0.9850
Training set svm with poly kernel & C= 100 score: 0.9993
Test set svm with poly kernel & C= 100 score: 0.9940
Training set svm with poly kernel & C= 1000 score: 0.9998
Test set svm with poly kernel & C= 1000 score: 0.9934
```

Figure 10: Experiments Results (SVM)

The accuracy of the training set and the test set's accuracy in each kernel are quite similar. Therefore, there is no question of overfitting. Undoubtedly, we will choose the best hyperparameters that achieves higher accuracy in the test data and also the training data, which is less

prone to overfitting and underfitting.It is hyperparameters kernel=rbf & C= 100, The training and testing accuracy of these parameters is 0.9995,0.9951.We will take care of these parameters in the rest of the steps

Comments

- We get maximum accuracy with rbf and polynomial kernel with C=100.0. and the accuracy is 0.9951,0.9940. Based on the above analysis we can conclude that our classification model accuracy is very good. Our model is doing a very good job in terms of predicting the class labels.
- But this is not true. Here, we have some classes in the data set that are unbalanced as (M,N) as shown earlier when we presented the distribution of classes in the data set. The problem is that accuracy is an insufficient measure of predictive performance in the unbalanced dataset problem.
- So, we must explore alternative met rices that provide better guidance in selecting models. In particular, we would like to know the underlying distribution of values and the type of errors our classifier is making.

  One such metric to analyze the model performance in imbalanced classes problem is Confusion matrix.

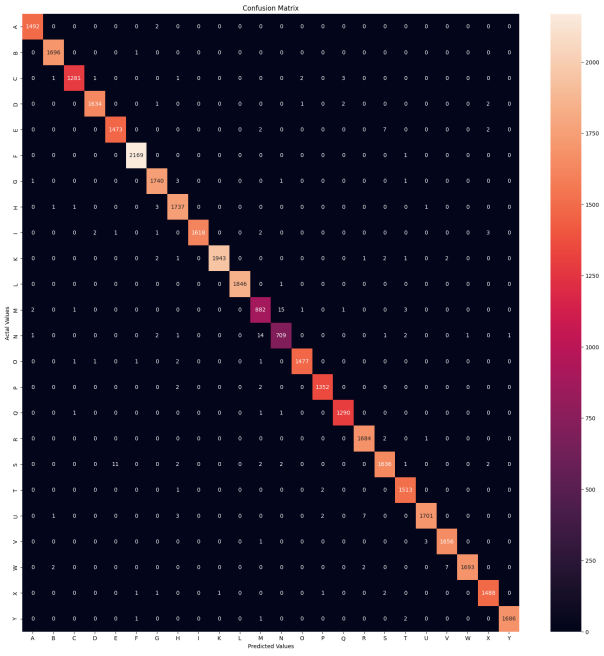


Figure 11: Confusion matrix SVM(rbf,C=100)

Classification report is another way to evaluate the classification model performance. It displays the precision, recall, f1 and support scores for the model.

We notice from previous reports that the model has been trained very well.

	precision	recall	f1-score	support
A	1.00	1.00	1.00	1494
B	1.00	1.00	1.00	1697
C	1.00	0.99	1.00	1289
D	1.00	1.00	1.00	1640
E	0.99	0.99	0.99	1484
F	1.00	1.00	1.00	2170
G	0.99	1.00	0.99	1746
H	0.99	1.00	0.99	1743
I	1.00	0.99	1.00	1627
K	1.00	1.00	1.00	1952
L	1.00	1.00	1.00	1847
M	0.97	0.97	0.97	905
N	0.97	0.97	0.97	732
O	1.00	1.00	1.00	1483
P	1.00	1.00	1.00	1356
Q	1.00	1.00	1.00	1293
R	0.99	1.00	1.00	1687
S	0.99	0.99	0.99	1656
T	0.99	1.00	1.00	1516
U	1.00	0.99	0.99	1714
V	0.99	1.00	1.00	1660
W	1.00	0.99	1.00	1704
X	0.99	1.00	0.99	1494
Y	1.00	1.00	1.00	1690
accuracy			1.00	37579
macro avg	0.99	0.99	0.99	37579
weighted avg	1.00	1.00	1.00	37579

Figure 12: Classification report SVM(rbf,C=100)

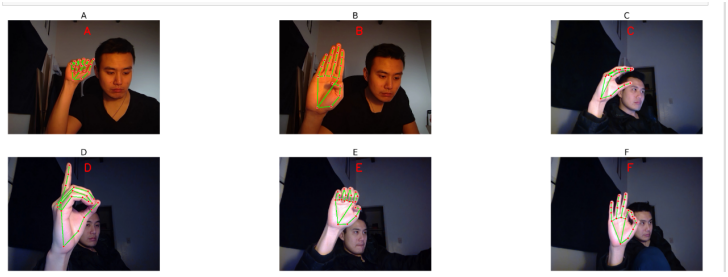


Figure 13: prediction SVM(rbf,C=100)

4.2 Multlayer Perceptron (MLP)

A neural network was also built as another experiment for the classification process. The structure of the neural network consists of several layers, as shown in the figure(14). The

```
]: model.summary()
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 2, 1)]       0           []

 input_2 (InputLayer)           [(None, 16, 1)]      0           []

 input_3 (InputLayer)           [(None, 36, 1)]      0           []

 dense (Dense)                  (None, 2, 4)         8           ['input_1[0][0]']

 dense_1 (Dense)                (None, 16, 4)        8           ['input_2[0][0]']

 dense_2 (Dense)                (None, 36, 4)        8           ['input_3[0][0]']

 flatten (Flatten)              (None, 8)            0           ['dense[0][0]']

 flatten_1 (Flatten)            (None, 64)           0           ['dense_1[0][0]']

 flatten_2 (Flatten)            (None, 144)          0           ['dense_2[0][0]']

 concatenate (Concatenate)      (None, 216)          0           ['flatten[0][0]',
                                                                  'flatten_1[0][0]',
                                                                  'flatten_2[0][0]']

 dense_3 (Dense)                (None, 30)           6510        ['concatenate[0][0]']

 dense_4 (Dense)                (None, 24)           744         ['dense_3[0][0]']

==================================================================================================
Total params: 7,278
Trainable params: 7,278
Non-trainable params: 0
```

Figure 14: Structure neural network

model is trained with the following parameters loss=categorical_crossentropy, optimizer=Adam, epochs=50 Training accuracy is obtained as in Figure (fig:40).

The model achieves an accuracy for the test set of 0.9844328165054321. We conclude that the neural network model is also good, but. But the accuracy of SVM is bigger and better.

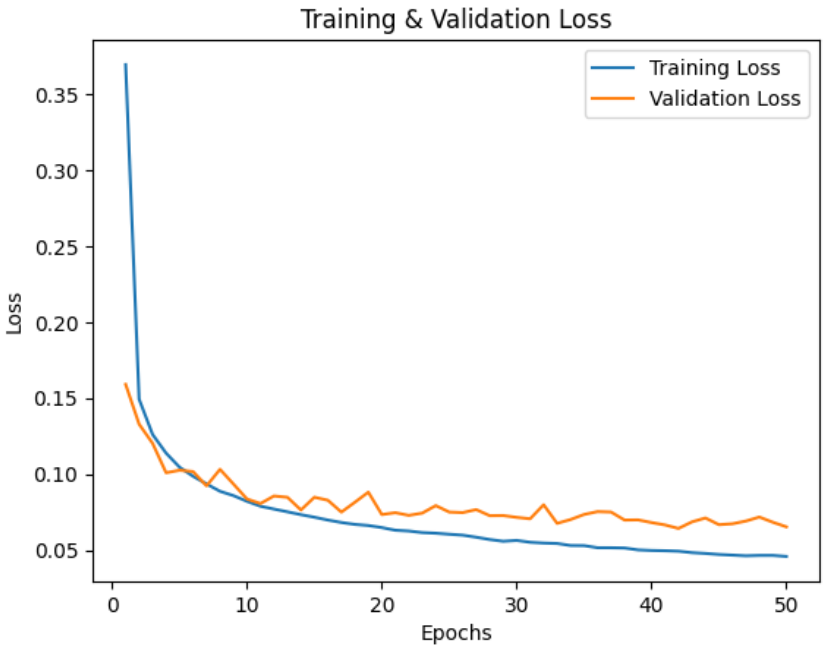


Figure 15: Training & Validation Loss neural network

5 Conclusions & future work

We conclude that the features that were produced were good, for the process of distinguishing between classes. That is, the quality of the quality of the trained models depends mainly on the quality of the features. The better the features, the better the performance of the model. In our experience, the best accuracy in SVM was reached with an accuracy of 0.9951, while the neural network model was 0.9844328165054321.

In our project, only 54 features were produced. In the future, we may create more relationships and features Also, we can describe features that describe the relationship of the hand with the rest of the body, such as the head and chest,Also, in the future, we may produce movement descriptors.

I benefited from this project well, as I got to know how to produce features from raw data as well as knowing the effect of the quality of features on machine learning models and a lot of things.

References

[1] Jiangtao Cao et al. "Hand posture recognition based on heterogeneous features fusion of multiple kernels learning". In: *Multimedia Tools and Applications* 75.19 (2016), pp. 11909–11928.

[2] Google. *MediaPipe Hands solution*. URL: https://google.github.io/mediapipe/solutions/hands.html.

[3] Google. *MediaPipe Main*. URL: https://google.github.io/mediapipe/.

[4] kaggle. *ASL alphabets images for classification*. URL: https://www.kaggle.com/datasets/debashishsau/aslamerican-sign-language-aplhabet-dataset.

[5] Nicolas Pugeault and Richard Bowden. “Spelling it out: Real-time ASL fingerspelling recognition”. In: *2011 IEEE International conference on computer vision workshops (ICCV workshops)*. IEEE. 2011, pp. 1114–1119.