future internet

MDPI

Article

Recursive Feature Elimination for Improving Learning Points on Hand-Sign Recognition

Rung-Ching Chen [1], William Eric Manongga [1] and Christine Dewi [2,*]

[1] Department of Information Management, Chaoyang University of Technology, Taichung City 413310, Taiwan
[2] Department of Information Technology, Satya Wacana Christian University, Salatiga 50715, Indonesia
* Correspondence: christine.dewi@uksw.edu

Abstract: Hand gestures and poses allow us to perform non-verbal communication. Sign language is becoming more important with the increase in the number of deaf and hard-of-hearing communities. However, learning to understand sign language is very difficult and also time consuming. Researchers are still trying to find a better way to understand sign language using the help of technology. The accuracy of most hand-sign detection methods still needs to be improved for real-life usage. In this study, Mediapipe is used for hand feature extraction. Mediapipe can extract 21 hand landmarks from a hand image. Hand-pose detection using hand landmarks is chosen since it reduces the interference from the image background and uses fewer parameters compared to traditional hand-sign classification using pixel-based features and CNN. The Recursive Feature Elimination (RFE) method, using a novel distance from the hand landmark to the palm centroid, is proposed for feature selection to improve the accuracy of digit hand-sign detection. We used three different datasets in this research to train models with a different number of features, including the original 21 features, 15 features, and 10 features. A fourth dataset was used to evaluate the performance of these trained models. The fourth dataset is not used to train any model. The result of this study shows that removing the non-essential hand landmarks can improve the accuracy of the models in detecting digit hand signs. Models trained using fewer features have higher accuracy than models trained using the original 21 features. The model trained with 10 features also shows better accuracy than other models trained using 21 features and 15 features.

Keywords: hand-sign detection; Mediapipe; feature selection; recursive feature elimination

Citation: Chen, R.-C.; Manongga, W.E.; Dewi, C. Recursive Feature Elimination for Improving Learning Points on Hand-Sign Recognition. *Future Internet* **2022**, *14*, 352. https://doi.org/10.3390/fi14120352

Academic Editor: Filipe Portela

Received: 31 August 2022
Accepted: 24 November 2022
Published: 26 November 2022

1. Introduction

Hand gestures and poses allow us to communicate non-verbally through sign language. It is the primary communication method for the deaf and hard-of-hearing communities. The importance of sign language is increasing as these groups grow. However, learning sign language is a challenging skill that requires much practice.

Due to the increasing importance of hand signs in daily life, researchers have used many methods to detect automatic hand signs. Convolutional Neural Network (CNN) [1–5] is one of the most used methods used to extract features from hand images. For the classification and detection, machine learning techniques, such as Support Vector Machine (SVM) [1,6–9], K-Nearest Neighbor (KNN) [7,10], and Light Gradient Boosting Machine (Light GBM) [9], are used.

This study used Mediapipe [11] for feature extraction of hand images. Mediapipe can extract 21 hand landmarks on each hand with minimum resource usage. Hand landmarks are used for hand-pose detection in this research, since they can reduce the interference from the image background and give better detection results by using a smaller number of parameters compared to traditional pixel-based detection using CNN. Nowadays, feature selection is becoming more important, especially in datasets containing many variables and features. Feature selection helps in eliminating irrelevant factors, which can improve

the accuracy of the classification process as well as its performance. There are several important factors that highlight the importance of feature selection. To start, it simplifies the model in such a way that fewer parameters are required. Next, it can reduce the time spent in training, reduce overfitting by increasing generalization, and avoid dimensional traps [12]. Recursive Feature Elimination (RFE) will be used for feature selection, while neural networks will be used for the classification model. The digit sign language type used in this study is American Sign Language (ASL), mainly used in English-speaking countries.

This study tries to answer three questions, which are: (1) Are all 21 hand landmarks extracted using Mediapipe useful in recognizing digits in ASL? (2) Which hand landmarks are more important in detecting ASL digits? (3) Can we improve the hand-sign detection accuracy by only using the more important hand landmarks?

The main contributions of this research are as follows: (1) To find the 10 essential hand landmarks in detecting ASL digit hand signs. (2) This shows that not all hand landmarks have the same importance in digit hand-sign detection; using only the essential landmarks can improve detection accuracy. (3) We also proposed using a novel distance from the hand landmark to the center of the palm to perform feature selection in order to address the limitation of the RFE implementation in the python library.

This research consists of five sections. Section 2 presents some research related to hand-sign detection and Section 3 describes the methodology of this research. Presentation and discussion of the experiment results are given in Section 4. Section 5 will provide the conclusion and present future works from this study.

2. Related Works

2.1. Hand-Sign Detection

In the computer vision field, hand-sign detection is a popular topic. Many researchers have been working on this topic. Different tools and methods have been utilized for hand-sign detection. Before detecting the hand sign, firstly, we need to detect the hand and extract hand features to be used for the model training for the hand-sign detection. Many methods are used to extract hand features: sensor- and vision-based modes. Sensor-based methods extract hand features using microcontrollers or specific sensors, such as data gloves [13–15], accelerometers [16,17], depth cameras [18,19], Kinect [20], leap-motion controller [19,21], etc. In contrast, vision-based methods utilize inputs from cameras, such as web cameras [7] and phone cameras.

Many machine learning and deep learning techniques have been used for hand-sign detection and classification. Machine learning techniques used include Support Vector Machine (SVM) [6–9], Random Forest [7,22,23], and K-Nearest Neighbor (KNN) [7,24,25]. One of the most popular deep learning techniques used in this topic is CNN, which is used in many studies. Ref. [4] used CNN to develop a human gesture-recognition system while also utilizing Gaussian Mixture Modal to train the CNN. The authors of [5] developed two custom CNN's recognizing 24 static ASL hand signs. In another study, [26] used CNN to classify Bangla Sign Language (BdSL) alphabets and digits.

2.2. Mediapipe for Feature Extraction

Mediapipe is an open-source framework developed by Google to build pipelines that perform inference over arbitrary sensory data, such as video or audio [11]. Since its release to the public in 2019, Mediapipe has already helped researchers and developers in prototyping by offering a lot of useful APIs.

Mediapipe is also the engine behind some innovative products and services emerging nowadays. Moreover, Mediapipe utilizes far fewer computation resources than most power-hungry machine learning frameworks. This leads to more possibility of embedding it into IoT devices, which opens many opportunities for further innovations.

2.3. Feature Selection

Nowadays, datasets are getting larger and larger. Some datasets have a lot of information, but not all this information is helpful. Some of it is redundant or even irrelevant in training our model. The redundant and irrelevant information can be called noise. The presence of irrelevant and redundant features in a dataset can lower the performance of predictive models due to over-fitting and the curse of dimensionality [27]. To produce a better model, we need to ensure that we use only the essential information when training our model. Therefore, we need to conduct feature selection to our dataset before feeding the data into our model.

Research has proved that eliminating redundant or irrelevant features can improve the model's accuracy. Research from [28] shows that the accuracy of breast cancer detection is improved after the number of features is reduced using Recursive Feature Elimination (RFE). The authors of [29] also conducted research that shows that feature selection can help improve the accuracy of predicting Type-II diabetes. In the financial sector, research from [30] used the Lasso regression method to reduce the redundant features from a bank failure prediction model, while [31] used RFE to reduce the data dimensionality to obtain more efficient stock market predictions.

RFE is different from the single-run feature selection method because of its recursive nature. A normal feature selection in a single run will fit the learning model and calculate the feature importance based on the fitted model. Then, it will show the feature importance of each feature and a threshold should be selected to decide which feature is important. Because it is only run one time, the feature's importance could be influenced by some other features that are not important. Meanwhile, RFE will select the best n features based on how many features we want. RFE also works by fitting the model and then calculating the feature importance of each feature. After that, it will prune the feature set by removing the least important feature and start another model fitting and calculation of feature importance based on the remaining features and remove the least important feature. This process will be repeated until no feature remains in the feature set. Finally, RFE will show the top-n most important features. From this, RFE removes the correlation between the remaining features and the eliminated features in each recursion.

2.4. Performance Evaluation

Conventional performance indicators are used in this research to evaluate the models' performance in detecting digit hand signs. Accuracy is used as the evaluation metric in this research. *Accuracy* can be denoted by Equation (1):

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (1)$$

where TP = True Positive; TN = True Negative; FP = False Positive; FN = False Negative.

A confusion matrix is used to show the models' performance. The confusion matrix can offer a detailed breakdown of the correct and incorrect predictions from the model to further analyze the model's performance and then find a way to improve the model's accuracy.

3. Methodology

3.1. Research Design

This section will explain the general design of how this research is conducted, as seen in Figure 1. The system will use an input in the form of a hand image from the dataset. The input will then be preprocessed. The data preprocessing step includes data cleaning to remove images that cannot be used, data augmentation to enrich the training and testing data, and splitting the data into the training and testing sets. The next step will be the feature extraction process using Mediapipe. Then feature selection using RFE will select the 15 and 10 most important features to detect ASL digit signs. Three models will be trained and tested using a different number of features, the original 21 features, 15 features, and

10 features. The output from the training will then be validated to assess the performance. The following sections will explain more in detail about the datasets used in this study and the process performed at every step.

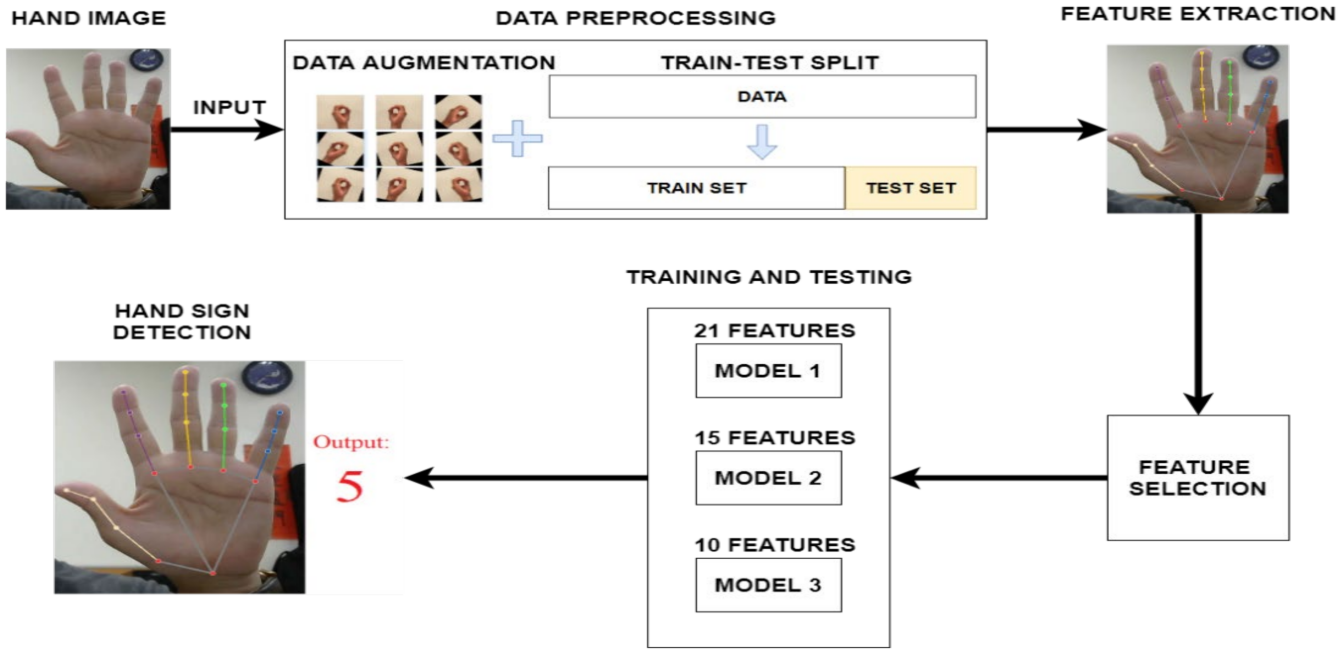


Figure 1. The general research design.

3.2. Datasets

Four datasets are used in this research. The first dataset is the "Sign Language Digits Dataset" [32] and the second dataset is the "American Sign Language Digit Dataset" [33]. The third dataset is the "American Sign Language Dataset" [34]. The last dataset is the validation dataset. Table 1 exhibits the list of datasets that we use in our research.

Table 1. List of datasets.

Dataset	Image Dimension	Total Number of Images
Sign Language Digits Dataset (Dataset 1)	100 × 100 pixels	2062
American Sign Language Digit Dataset (Dataset 2)	400 × 400 pixels	5000
American Sign Language Dataset-Digits Only (Dataset 3)	422 × 422 pixels	12,000
Validation Dataset (Dataset 4)	1280 × 720 pixels and 640 × 480 pixels (full, uncropped image) 100 × 100 pixels (cropped hand image)	300

The first dataset consists of hand images for the hand signs 0–9 in American Sign Language (ASL). Each image has a dimension of 100 × 100 pixels. Each class has more than 200 images, comprising 2062 images in 10 categories. Images in this dataset have good quality and are all taken with a uniform white-grey background color. Figure 2 shows some example images from the dataset.

The second dataset contains 5000 raw images of hand signs 0–9 in ASL, 500 images per class. It also has the corresponding output image files, which were processed using Mediapipe to detect and show the hand landmarks. In addition, this dataset also includes the hand landmark detection results in the form of CSV files. In this research, the CSV files are used to train and test the model.

Figure 2. The sample images from Dataset 1.

The third dataset consists of the whole digit and alphabet hand signs in ASL and some hand signs for simple sentences, but only the digit hand signs are used in this research. Images in this dataset have a dimension of 422 × 422 pixels. Similar to the first dataset, the images have good quality and are also taken in a uniform background color. The dataset's author has already split the images into a train set and a test set. The train set contains 1000 images per class for 10,000 images, while the test set contains 200 images per class for a total of 2000 images. Figure 3 shows some examples of the images in this dataset.

Figure 3. The sample image from Dataset 3.

The last dataset is the validation dataset. The validation dataset is a collection of 300 images taken directly by researchers to test the models using images not part of the training and testing sets. These images are taken using webcams and are taken in good lighting conditions. The images in this dataset have some background variation, making them better when used to evaluate the accuracy of the trained models. The uncropped image is a half-body image with one hand showing a digit hand sign, while the cropped image is a 100x100 pixels image of the hand, cropped from the full image. The cropping is carried out programmatically by utilizing the hand-detection feature from Mediapipe. After

adding some padding on the detected hand's top, bottom, left, and right sides, the hand will be cropped. This dataset is only used to evaluate the models that have already been trained and tested using the other three datasets and will not be used for any model training.

3.3. Data Pre-Processing

Data pre-processing is used for training and testing the models in this research. We use different pre-processing techniques based on the datasets, since the first and third datasets are images, while the second dataset already provided the landmark detection result in a CSV file.

3.3.1. Data Augmentation

Image augmentation techniques are used for the first and third datasets to increase the variation and number of images in the dataset. Since the hand images are all right-hand, the image is flipped horizontally to obtain a left-hand image. The images will also undergo clockwise and counter-clockwise rotation to gain more variation in the hand position. Figure 4 below shows an example of the image augmentation.

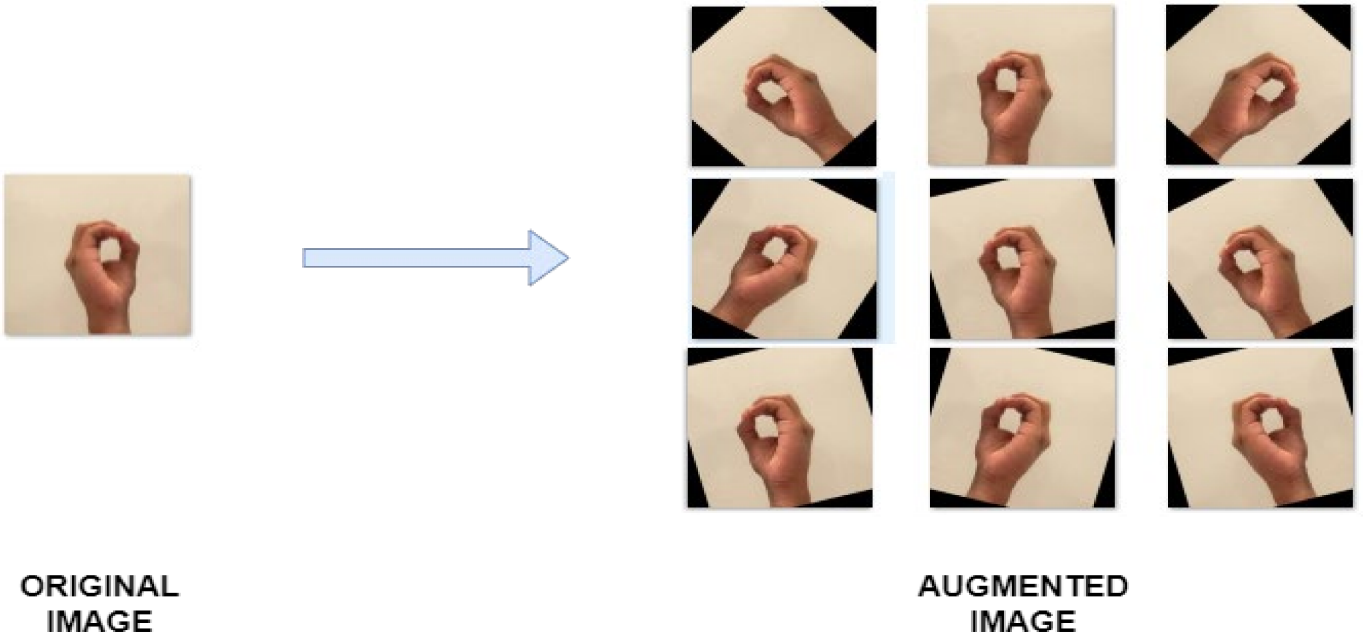


Figure 4. The process of image augmentation.

For the second dataset, we use the CSV file provided by the dataset author. The CSV file contains the image name, the x and y coordinates of the hand landmarks, and the class name. All the hand images in this dataset are also right hands, but they already have some rotation applied. Further, to have the same data condition as the first and third datasets, we need to have the left-hand-side data of the hand landmarks. To achieve this, we need to reflect the position of the landmark points based on the line x = image width/2. The formula applied to the x coordinate on each landmark point to convert pn (xn, yn) into pn' (xn,' yn) is denoted by Equation (2):

$$x' = w - x \tag{2}$$

where x' is the new x coordinate of the hand landmarks, w is the image width, and x is the original x coordinate in the hand landmarks.

3.3.2. Train–Test Split

Before the datasets are used to train and test the models, they should be split into the training and testing sets. The first and second datasets will have 30% of the total data as the testing set and the remaining 70% is used as the training set. The author already separated the images into training and testing sets for the second dataset. The ratio between the images in the training and testing dataset is 5:1.

3.4. Hand Landmark Detection and Extraction

Mediapipe is utilized in this research to detect the hand landmark and obtain the landmark position from an image. Mediapipe can detect and extract 21 3D hand landmarks from a hand image. It can also detect left and right hands and supports multi-hand detection. Figure 5 shows how Mediapipe will detect a hand and the hand landmark coordinates. The output from Mediapipe is then stored as a Numpy file. This file will be loaded and used in the feature selection process, model training, testing, and validation.

Figure 5. Hand landmarks detected by Mediapipe Hands.

3.5. Feature Selection Method

Recursive Feature Elimination (RFE) is implemented to reduce the number of features used to train the models. RFE is chosen since we can decide the number of features to be selected from the original features. RFE fits a learning model as a wrapper method and removes the less critical features [35]. Random Forest (RF) is the learning model chosen for this research. Random Forest uses Gini importance to calculate the feature importance. The feature importance of a binary tree decision tree can be denoted by Equation (3):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \quad (3)$$

where ni_j is the importance of node j; w_j is the weighted number of samples reaching node j; C_j is the impurity value of node j; *left (j)* is the child node from left split on node j; *right (j)* is the child node from right split on node j.

Then the importance of each feature on a decision tree is based on Equation (4):

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_j} \quad (4)$$

where fi_i is the importance of feature I; ni_j is the importance of node j.

These values then can be normalized by dividing them by the sum of all feature importance values with Equation (5):

$$norm\ fi_i = \frac{fi_i}{\sum_{j \in all\ features} fi_j} \tag{5}$$

Finally, the feature importance of a feature in the random forest level is the average over all the trees as in Equation (6):

$$RF\ fi_i = \frac{\sum_{j:all\ trees} norm\ fi_{ij}}{T} \tag{6}$$

$RF\ fi_i$ is the importance of a feature I in the random forest model; $norm\ fi_{ij}$ is the normalized feature importance of feature I in tree j; T is the total number of trees in the random forest.

Before making the feature selection, the hand landmark data need to be converted from coordinates to distance between points because the feature selection methods currently available in Python cannot accept a 2D array as the input feature. We choose not to flatten the input into 1D because doing so will destroy the relationship between the x and y coordinate of each point and RFE will treat each x and y value of a point as a separate feature. In this research, we propose using a novel distance between the hand landmark to the hand palm centroid. The distance is calculated as a Euclidean distance from the hand landmarks to an anchor point Pa which is in the center of the palm. The point Pa is calculated as a centroid of a triangle formed by the coordinates $P0$, $P5$, and $P17$. This coordinate can be represented by Equations (7) and (8):

$$Pa_x = \frac{P0_x + P5_x + P17_x}{3} \tag{7}$$

$$Pa_y = \frac{P0_y + P5_y + P17_y}{3} \tag{8}$$

Pa_x is the x coordinate for Pa; Pn_x is the x coordinate of the n landmark point; Pa_y is Pa's y coordinate; and Pn_y is the y coordinate of the n landmark point.

Euclidean distance between Pa and Pn is indicated in Equation (9):

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{9}$$

where (x_1, y_1) is the coordinate of the first point, which is the anchor point; (x_2, y_2) is the coordinate of the second point; and d is the Euclidean distance between the 2 points.

Using RFE, the number of features is reduced from the original 21 features to 15 and 10 features. The algorithm of RFE can be described in Algorithm 1:

RFE will take an input of the number of desired features and also the dataset containing the original number of features. Then it will build and fit a random forest model and calculate the feature importance. The next step is to remove the least important feature from the dataset. If there is still a feature in the dataset, it will recursively repeat this until there are no more features in the dataset. Finally, it will return the top-n most important features for digit hand-sign classification.

Algorithm 1: Random Forest-Recursive Feature Elimination.

```
    input: training dataset, n number of desired features
    output: feature set of top-n most important features
1   Fs ← all features in the original dataset
2   Data ← training dataset
3   function RFE(Fs, Data)
4       build and fit Random Forest model using feature set Fs
5       evaluate feature importance and rank the features
6       remove the least important feature from Fs
7       if count(Fs) ! = 0
8           RFE(Fs, Data)
9       end
        return list of top-n most important features
10  end
```

3.6. Training Steps

Neural networks are used as the classifier in this research. The type of neural network used is a sequential neural network. The number of input nodes will be the number of features used for training, while the number of output nodes will be ten, corresponding to the digit 0–9. Dataset 1, Dataset 2, and Dataset 3 are used to train three models, totaling nine trained models. Dataset 4 is used to evaluate the performance of the models and not for training. This study's training and evaluation algorithm is depicted in Algorithm 2.

Algorithm 2: Model Training Steps.

```
    inputs: training, testing, and validation set
            The selected feature sets Fs15 and Fs10
    Initialization: initialize untrained model m1, m2, and m3
    output: trained model m1, m2, and m3
1   train m1 using the original training and testing set
2   evaluate m1 performance using the original validation set
3   reduce the number of features in the dataset to follow Fs15
4   train m2 using the Fs15 training and testing set
5   evaluate m2 performance using the Fs15 validation set
6   reduce the number of features in the dataset to follow Fs10
7   train m3 using the Fs10 training and testing set
8   evaluate m3 performance using the Fs10 validation set
```

For the model training and evaluation, it will need the training, testing, and validation set. We will also need the feature selection result. **Fs15** is the top 15 important features and **Fs10** is the top 10 important features. First 3 untrained models **m1**, **m2**, and **m3** will be initialized. Model **m1** will be trained using the original 21 features, **m2** will use 15 features in **Fs15**, while **m3** will use 10 features in **Fs10**. First, we will train and test the model **m1** using the original training and testing data. Then model **m1** will be evaluated using the validation dataset. Next, we will reduce the number of features in the training, testing and validation sets to follow **Fs15** features. Then model **m2** will be trained and tested using the already feature-reduced dataset. After that, model **m2** will be evaluated for its performance. The features in the training, testing, and validation will then be reduced once more to conform with **Fs10**. Finally, model **m3** will be trained, tested, and also validated using the dataset having 10 features only. This procedure will be conducted for Dataset 1, Dataset 2, and Dataset 3 to produce three trained models per dataset.

4. Results and Discussion

4.1. Experiment Results

This section will present the results of the feature selection and the results of the model training and testing using different numbers of features. This experiment used an Nvidia RTX2080 Ti GPU accelerator with 11 GB memory and an Intel i7-8700 Central Processing Unit (CPU) with 32 GB DDR4-2666 memory.

4.1.1. Feature Selection Results

Table 2 shows the feature selection results using RFE to reduce the number of features to 15 features and 10 features. The numbers shown in the selected feature columns are the number of the hand landmark points, which can be observed from Figure 5. From the results, the features selected by RFE with ten features are the subset from the result of RFE with 15 features.

Table 2. Feature selection results.

Number of Selected Features	Selected Features (Landmark Points)
15 Features	1, 2, 3, 4, 7, 8, 11, 12, 14, 15, 16, 17, 18, 19, 20
10 Features	2, 3, 4, 8, 11, 12, 15, 16, 19, 20

4.1.2. Model Accuracy Results

After conducting the training, testing, and validating for each dataset and model, the results can be seen in Table 3. The models' accuracy improved by reducing the number of features used in the training and detection. The improvement can be seen while reducing features from the original 21 features to 15. The validation accuracy improves significantly for Dataset 1, from 85% to 96%; Dataset 2, from 96.30% to 97%; Dataset 3, from 72.30% to 84.70%. Another improvement also happens when reducing the number of features further from 15 features to 10 features. While the improvement is not that significant, it can still be seen that there is a small increase in validation accuracy for Dataset 1, from 96% to 96.30%, and for Dataset 3, from 84.70% to 88.30%. At the same time, there is no improvement for Dataset 2.

Table 3. Training, testing, and validation accuracy for all datasets and models.

Dataset	Scenario	Features		
		Original 21 Features	15 Features	10 Features
Dataset 1	Training	95.05%	**98.41%**	98.30%
	Testing	91.70%	**93.40%**	**93.40%**
	Validation	85.00%	96.00%	**96.30%**
Dataset 2	Training	99.70%	**100%**	**100%**
	Testing	**100%**	**100%**	**100%**
	Validation	96.30%	**97.00%**	**97.00%**
Dataset 3	Training	99.70%	**99.96%**	99.93%
	Testing	98.60%	98.60%	**98.80%**
	Validation	72.30%	84.7%	**88.30%**

Figure 6 shows the confusion matrix for the models trained using ten features. For the model trained by Dataset 1, the most error in detection occurred on digit 7, which detects the hand sign as an 8. Furthermore, the validation result is very good for the model trained using Dataset 2. There is only a slight mistake when detecting some digit hand signs. The model trained using Dataset 3 has worse accuracy than the other two models. False detections mostly occurred on digits 0, 2, and 7.

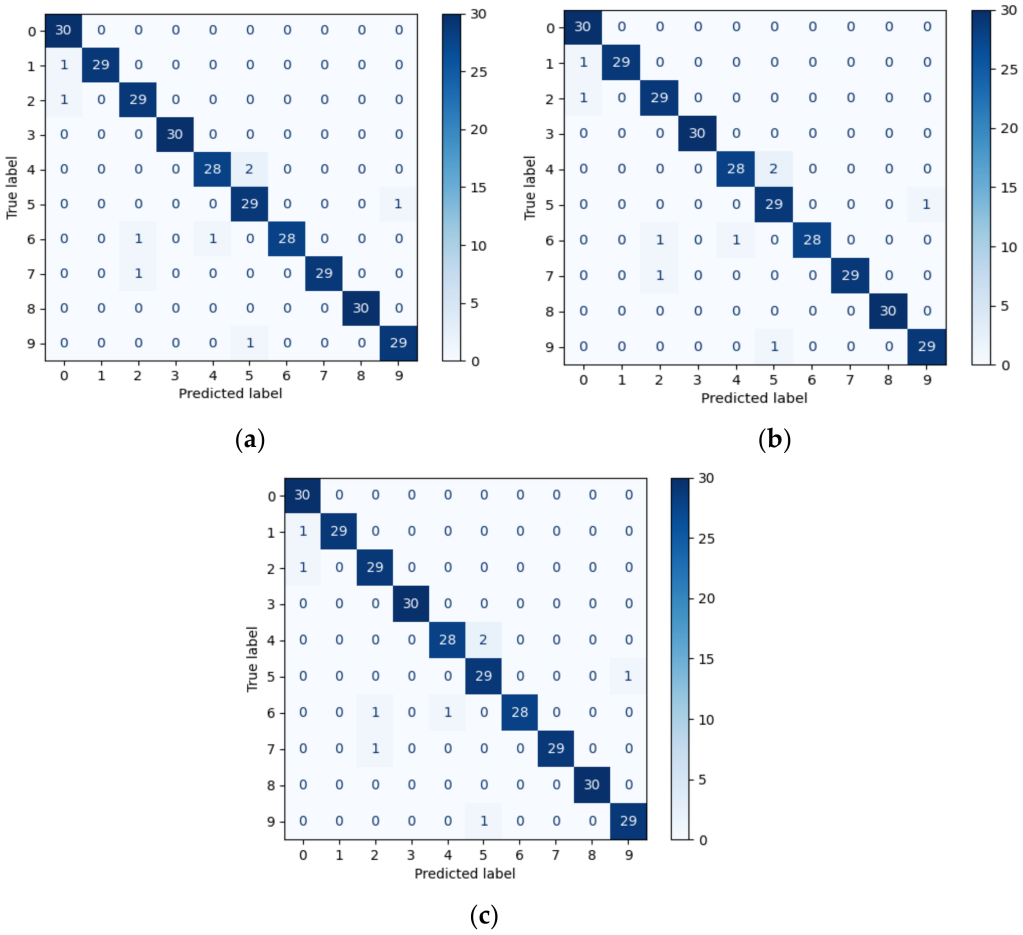


Figure 6. Confusion matrix of the models introduced with ten features: (**a**) trained used Dataset 1; (**b**) trained using Dataset 2; (**c**) trained using Dataset 3.

Figures 7–9 show some sample digit hand-sign detection results from the models trained using 10 features. From the examples, the angle where the hand is tilted affects the detection accuracy. This could result from the distortions in the training data when Mediapipe gave a distorted hand detection result for training images that are rotated to a certain degree. These insufficient data may mislead the model to learn the wrong patterns, which results in a decrease in detection accuracy.

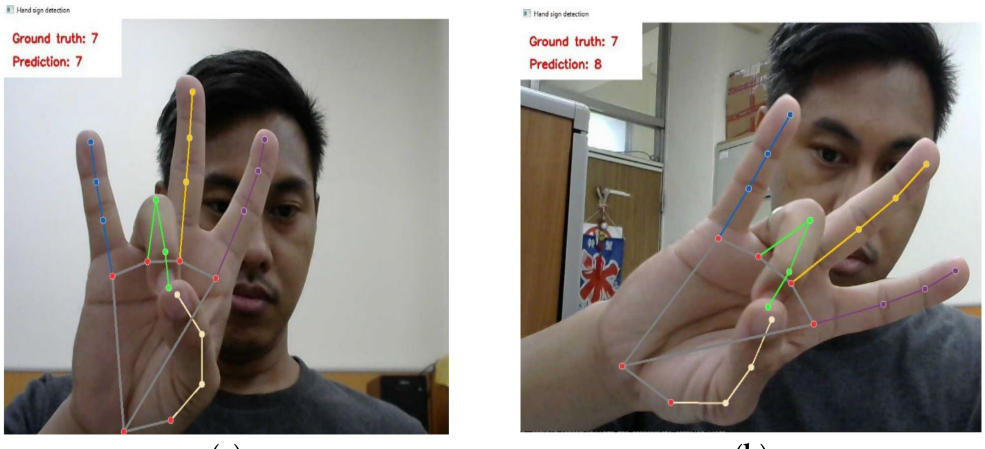
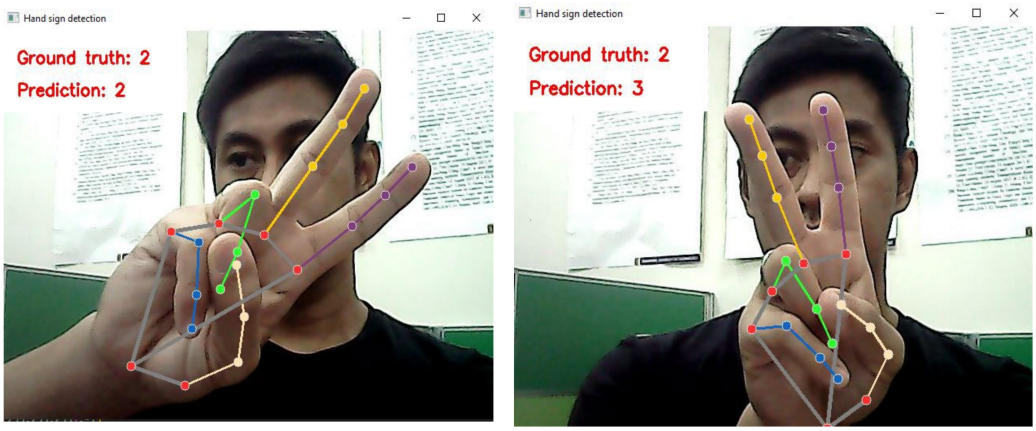


Figure 7. Prediction result using Dataset 1 and 10 features: (**a**) correct prediction; (**b**) incorrect prediction.

(**a**) (**b**)

Figure 8. Prediction result using Dataset 2 and 10 features: (**a**) correct prediction; (**b**) incorrect prediction.



(**a**) (**b**)

Figure 9. Prediction result using Dataset 3 and 10 features: (**a**) correct prediction; (**b**) incorrect prediction.

4.2. Discussion

After seeing the results from the feature selection and also the models' accuracy, there are two important findings to look into further. The first one is about the result from the feature selection and the second is about the gap in accuracy for models trained with different datasets. We will also talk about the advantage of using our method compared to using CNN.

4.2.1. Feature Selection Result

Using RFE, we reduced the number of features used for the detection of ASL digit hand signs from the original 21 features to 15 features and 10 features. The result of the feature selection can be seen in Table 2 in Section 4.1.1. Figure 10 shows the visualization of the hand landmarks selected using RFE.

From Figure 10, we can see that RFE first eliminates the hand landmarks that are less likely to change position or, in this case, the landmarks on the finger base. This can be seen in Figure 10a. When we move our fingers to perform the ASL digit hand signs, these eliminated landmark positions do not change much or even do not change. When these hand landmarks are all eliminated, the next landmarks to be eliminated will be those that are closer to the finger base, while none of the landmarks on the fingertips are eliminated. This is shown in Figure 10b. From this observation, we can see that to detect ASL digit hand signs, the hand landmarks closer to the fingertips are more important.

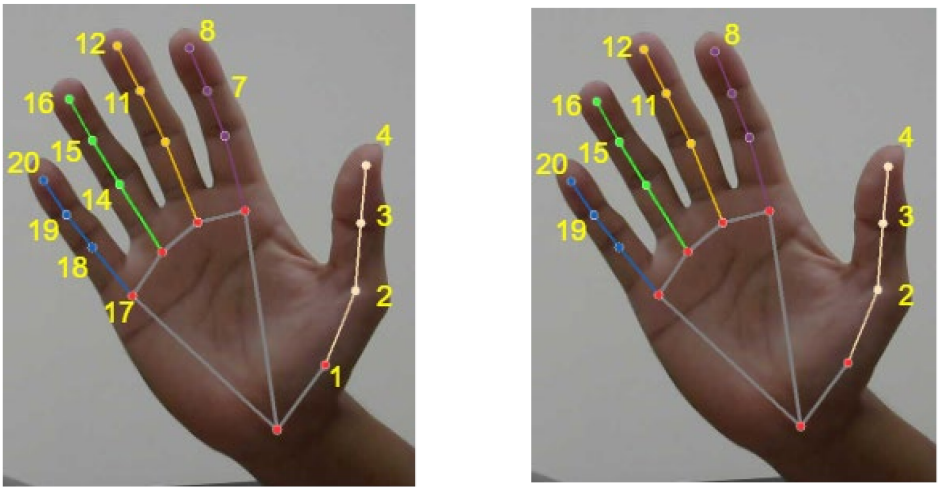


Figure 10. Hand landmarks selected using RFE. Eliminated landmarks are not shown in the figure. (**a**) 15 features, (**b**) 10 features.

4.2.2. The Gap between the Validation Accuracy for Dataset 3 and the Other Datasets

From the models' accuracy shown in Table 3, we can see that the model trained using Dataset 2 has the highest validation accuracy, while trained using Dataset 3 has the lowest validation accuracy. The gap between the two models is quite huge.

From the investigation, we found out that, in some cases, Mediapipe failed to detect the hand landmark position correctly when the images are augmented. Figure 11a shows a good detection result using Mediapipe, while Figure 11b shows a distorted landmark detection as an effect of image rotation.

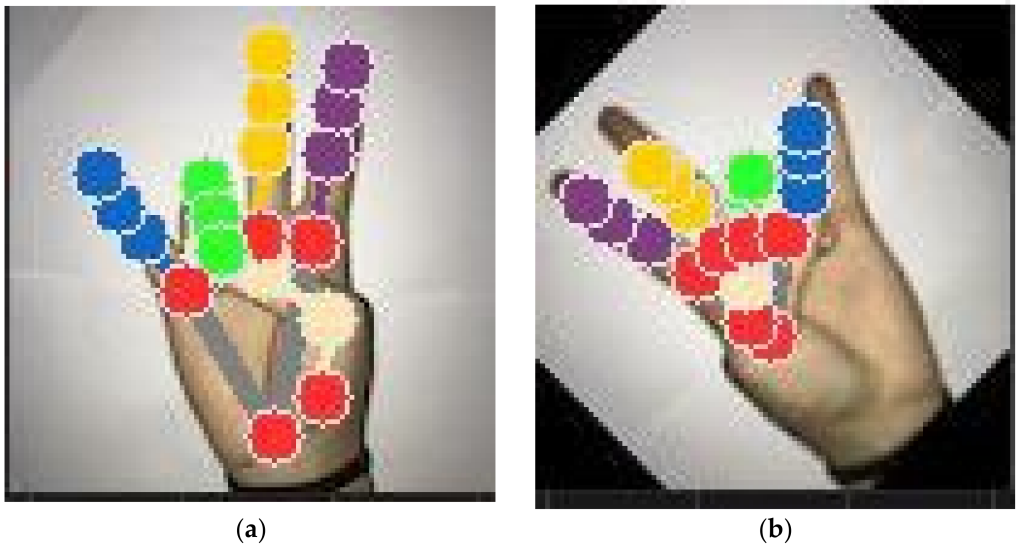


Figure 11. Detection and extraction of hand landmarks using Mediapipe. (**a**) Good result, (**b**) distorted result.

These data with incorrect information will become noise when used to train the model and may cause the accuracy to become lower. As already mentioned, Datasets 1 and 3 only provide hand-sign images and the feature extraction process is performed using Mediapipe. Dataset 2 provides a CSV file containing the hand landmark information as an output from Mediapipe, which is used in this study. This makes the data in Dataset 2 more stable and robust. This can explain why the accuracy of the models trained using Dataset 2 is better than for those trained using other datasets.

While it has not been investigated deeply yet, there are some suspicions on why the accuracy of the model trained using Dataset 3 has a very huge gap compared to Dataset 1, which also only provides hand images. This could be attributed to the huge number of images available in Dataset 3 compared to Dataset 1. These images are still being augmented for mirroring and also rotation, making the number of images in Dataset 3 even larger. This may cause Dataset 3 to have more images with wrongly detected hand landmark positions, as shown in Figure 11b. This means more noise is coming to the model during training that reduces the models' accuracy.

4.2.3. Advantages of Our Method Compared to Using CNN

In our proposed methods, we chose the combination of Mediapipe for feature extraction and neural networks for classification instead of using CNN, which can conduct feature extraction and classification by itself. As seen in Figure 12, a simple CNN with three convolutional layers, three batch normalization layers, three max-pooling layers, and three dense layers has more than 5 million parameters, while in our proposed method, we used a simple neural network with three dense layers only, as shown in Figure 13.

```
Model: "CNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 98, 98, 64)        1792

 batch_norm_1 (BatchNormaliz  (None, 98, 98, 64)       256
 ation)

 max_pool2d_1 (MaxPooling2D)  (None, 48, 48, 64)       0

 conv2d_2 (Conv2D)           (None, 46, 46, 128)       73856

 batch_norm_2 (BatchNormaliz  (None, 46, 46, 128)      512
 ation)

 max_pool2d_2 (MaxPooling2D)  (None, 22, 22, 128)      0

 conv2d_3 (Conv2D)           (None, 20, 20, 256)       295168

 batch_norm_3 (BatchNormaliz  (None, 20, 20, 256)      1024
 ation)

 max_pool2d_3 (MaxPooling2D)  (None, 9, 9, 256)        0

 dense_1 (Dense)             (None, 9, 9, 4096)        1052672

 dense_2 (Dense)             (None, 9, 9, 1024)        4195328

 output (Dense)              (None, 9, 9, 10)          10250

=================================================================
Total params: 5,630,858
Trainable params: 5,629,962
Non-trainable params: 896
_________________________________________________________________
```

Figure 12. Example of a simple CNN architecture.

```
Model: "proposed_approach"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 21, 1024)          3072

 dense_2 (Dense)             (None, 21, 512)           524800

 dense_3 (Dense)             (None, 21, 256)           131328

 output (Dense)              (None, 21, 10)            2570

=================================================================
Total params: 661,770
Trainable params: 661,770
Non-trainable params: 0
_________________________________________________________________
```

Figure 13. The classification model in our proposed method.

Table 4 shows the comparison between CNN and the classification model used in our proposed method when trained using Dataset 1. As we can see, CNN has more parameters, which results in longer training time. This could be attributed to the feature extraction ability of CNN, which causes the number of parameters to be much larger. In our proposed method, the training time recorded is purely the time to train the model. The feature extraction step is performed during the data preprocessing. For Dataset 1, which has more than 30,000 images after augmentation, the data preprocessing took 2885 s. Though the data preprocessing took a long time to finish, it is a one-time process for each dataset, as there is no need to conduct preprocessing every time we want to train our model. The data preprocessing is not only for feature extraction, it also cleans the data by removing images that cannot be used for training, labeling, one-hot encoding for the classes, and also saving the result into a .npy file to be used in the future processes. By separating the feature extraction process from the actual training process, our approach benefits from faster model training time. This will be very beneficial when conducting experiments by adjusting training hyperparameters and other configurations. We can perform more experiments with shorter training time and adjust the hyperparameters and model configuration to obtain the best model.

Table 4. Comparison between CNN and our proposed approach.

	CNN	**Proposed Approach**	
Input	100 × 100 pixels image	Ten hand landmarks extracted using Mediapipe	
Epoch	30	30	100
Batch size	128	512	512
Training accuracy	92.79%	94.3%	**98.3%**
Testing accuracy	100%	90.8%	**93.40%**
Training time	1607 s	62.1 s *	198.5 s *
Validation accuracy	57.7%	95.3%	**96.3%**
Evaluation time	2.4 s	**1 s**	**1 s**

* does not include the time needed for feature extraction.

As seen in Table 4, trained using the same dataset, CNN shows good accuracy on the train–test data with 100% accuracy in testing. However, it only achieves 57.7% accuracy when tested on Dataset 4, while our proposed method can achieve 96.3% accuracy. This shows that in training a CNN, more images with many variations and different backgrounds are needed before it can generalize well. Suppose the model is only trained using images

with light-colored and monotone backgrounds. In that case, it will not perform well in a complex background, which is more likely to happen in real-life situations. At the same time, most publicly available hand-sign datasets are taken in uniform settings with no complex background. By using Mediapipe for hand detection and feature extraction, our classification model can focus on learning the pattern of the hand landmarks for ASL digit hand-sign classification and achieve far better accuracy.

5. Conclusions

This study compares models trained using several different datasets and a different number of features. Four datasets are used in this study; three datasets are used to train the model, while the fourth is used for evaluation purposes only. The original dataset has 21 features to train the model. Feature selection using RFE is made to reduce the number of features to 15 elements and 10 features. A novel hand-crafted feature, the distance from hand landmark to palm centroid, is proposed to address the limitation of the RFE implementation in python's sklearn library.

Based on the experiments and results, we can conclude that not all 21 hand landmarks have the same importance in detecting ASL digit hand signs. Removing less important hand landmarks from the set of features can improve the model's detection accuracy. This conclusion is supported by the experiment results shown in Table 3, where the model's accuracy improved when the number of features was reduced to 15 features and 10 features.

The quality of the dataset also has an impact on the model accuracy. Dataset 2 has the best performance compared to the two other datasets. Dataset 2 already provided the coordinates of the hand landmarks in CSV format. These detection results are of excellent quality and stability. The model has a better and more stable feature as an input in training. The other two datasets only provided hand images; feature extraction is performed using Mediapipe. During the feature extraction stage, some hand landmark detections are distorted when the images are rotated during image augmentation. The data quality becomes worse compared to Dataset 2, making the models less accurate.

In the future, we will try to expand the number of classes by including the alphabet of the ASL hand sign. We will try to explore the possibility of using the palm centroid distance as the input to our classification network. Feature-reduction methods using neural networks, such as autoencoders, will also be explored to improve the hand-sign detection performance.

Author Contributions: Conceptualization, W.E.M.; methodology, R.-C.C. and W.E.M.; software, W.E.M.; investigation, W.E.M.; resources, R.-C.C.; writing—original draft preparation, W.E.M. and C.D.; writing—review and editing, R.-C.C. and C.D.; visualization, W.E.M.; supervision, R.-C.C. and C.D.; project administration, R.-C.C. and C.D.; funding acquisition, R.-C.C. and C.D.; All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by the Ministry of Science and Technology, Taiwan. The Nos are MOST-110-2927-I-324-50, MOST-110-2221-E-324-010, and MOST-111-2221-E-324-020, Taiwan.

Institutional Review Board Statement: Ethical review and approval were waived for this study, due to the reason that we use the public and free datasets.

Informed Consent Statement: All subjects gave their informed consent for inclusion before they participated in the study.

Data Availability Statement: Sign Language Dataset (https://www.kaggle.com/datasets/ardamavi/sign-language-digits-dataset, accessed on 22 November 2022) American Sign Language Digit Dataset (https://www.kaggle.com/datasets/rayeed045/american-sign-language-digit-dataset, accessed on 22 November 2022) American Sign Language Dataset (https://www.kaggle.com/datasets/joannracheljacob/american-sign-language-dataset, accessed on 22 November 2022) Validation Dataset (https://drive.google.com/drive/folders/1o6BvphcDqT-rCjlIyQwpav2q5_E3Goso?usp=sharing, accessed on 22 November 2022).

Acknowledgments: The authors would like to thank Chaoyang University of Technology, Satya Wacana Christian University, and others that took part in this work for the support and help.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alom, M.S.; Hasan, M.J.; Wahid, M.F. Digit recognition in sign language based on convolutional neural network and support vector machine. In Proceedings of the 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), Dhaka, Bangladesh, 24–25 December 2019; pp. 1–5.
2. Hossain, M.B.; Adhikary, A.; Soheli, S.J. Sign language digit recognition using different convolutional neural network model. *Asian J. Res. Comput. Sci.* **2020**, *6*, 16–24. [CrossRef]
3. Kalam, M.A.; Mondal, M.N.I.; Ahmed, B. Rotation independent digit recognition in sign language. In Proceedings of the 2nd International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox's Bazar, Bangladesh, 7–9 February 2019; pp. 1–5.
4. Lin, H.I.; Hsu, M.H.; Chen, W.K. Human hand gesture recognition using a convolution neural network. In Proceedings of the IEEE International Conference on Automation Science and Engineering, New Taipei, Taiwan, 18–22 August 2014; pp. 1038–1043.
5. Paul, P.; Bhuiya, M.A.U.A.; Ullah, M.A.; Saqib, M.N.; Mohammed, N.; Momen, S. A modern approach for sign language interpretation using convolutional neural network. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Porto, Portugal, 8–11 October 2019; Volume 11672 LNAI, pp. 431–444.
6. Abiyev, R.H.; Arslan, M.; Idoko, J.B. Sign language translation using deep convolutional neural networks. *KSII Trans. Internet Inf. Syst.* **2020**, *14*, 631–653. [CrossRef]
7. Chakraborty, S.; Bandyopadhyay, N.; Chakraverty, P.; Banerjee, S.; Sarkar, Z.; Ghosh, S. Indian sign language classification (ISL) using machine learning. *Am. J. Electron. Commun.* **2021**, *1*, 17–21. [CrossRef]
8. Rajan, R.G.; Judith Leo, M. American sign language alphabets recognition using hand crafted and deep learning features. In Proceedings of the 5th International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–28 February 2020; pp. 430–434.
9. Shin, J.; Matsuoka, A.; Hasan, M.A.M.; Srizon, A.Y. American sign language alphabet recognition by extracting feature from hand pose estimation. *Sensors* **2021**, *21*, 5856. [CrossRef] [PubMed]
10. Alvin, A.; Shabrina, N.H.; Ryo, A.; Christian, E. Hand gesture detection for sign language using K-nearest neighbor with mediapipe. *Ultim. Comput. J. Sist. Komput.* **2021**, *13*, 57–62. [CrossRef]
11. Lugaresi, C.; Tang, J.; Nash, H.; McClanahan, C.; Uboweja, E.; Hays, M.; Zhang, F.; Chang, C.-L.; Yong, M.G.; Lee, J.; et al. MediaPipe: A Framework for Building Perception Pipelines. 2019. Available online: http://arxiv.org/abs/1906.08172 (accessed on 18 May 2022).
12. Chen, R.C.; Dewi, C.; Huang, S.W.; Caraka, R.E. Selecting critical features for data classification based on machine learning methods. *J. Big Data* **2020**, *7*, 52. [CrossRef]
13. Assaleh, K.; Shanableh, T.; Zourob, M. Low complexity classification system for glove-based arabic sign language recognition. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Paphos, Cyprus, 23–27 September 2012; Volume 7665 LNCS, pp. 262–268.
14. Shukor, A.Z.; Miskon, M.F.; Jamaluddin, M.H.; Ali Ibrahim, F.B.; Asyraf, M.F.; Bahar, M.B. Bin a new data glove approach for malaysian sign language detection. In Proceedings of the Procedia Computer Science, Sousse, Tunisia, 5–7 October 2015; Volume 76, pp. 60–67.
15. Tubaiz, N.; Shanableh, T.; Assaleh, K. Glove-based continuous arabic sign language recognition in user-dependent mode. *IEEE Trans. Hum.-Mach. Syst.* **2015**, *45*, 526–533. [CrossRef]
16. Pan, T.Y.; Tsai, W.L.; Chang, C.Y.; Yeh, C.W.; Hu, M.C. A hierarchical hand gesture recognition framework for sports referee training-based emg and accelerometer sensors. *IEEE Trans. Cybern.* **2022**, *52*, 3172–3183. [CrossRef]
17. Zhang, X.; Chen, X.; Li, Y.; Lantz, V.; Wang, K.; Yang, J. A framework for hand gesture recognition based on accelerometer and emg sensors. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2011**, *41*, 1064–1076. [CrossRef]
18. Almeida, S.G.M.; Guimarães, F.G.; Ramírez, J.A. Feature extraction in brazilian sign language recognition based on phonological structure and using RGB-D sensors. *Expert Syst. Appl.* **2014**, *41*, 7259–7271. [CrossRef]
19. Chophuk, P.; Pattanaworapan, K.; Chamnongthai, K. Fist american sign language recognition using leap motion sensor. In Proceedings of the 2018 International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, Thailand, 7–9 January 2018; pp. 1–4.
20. Lai, K.; Konrad, J.; Ishwar, P. A Gesture-driven computer interface using kinect. In Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation, Santa Fe, NM, USA, 22–24 April 2012; pp. 185–188.
21. Avola, D.; Bernardi, M.; Cinque, L.; Foresti, G.L.; Massaroni, C. Exploiting recurrent neural networks and leap motion controller for the recognition of sign language and semaphoric hand gestures. *IEEE Trans. Multimed.* **2019**, *21*, 234–245. [CrossRef]
22. Bajaj, Y.; Malhotra, P. American sign language identification using hand trackpoint analysis. In Proceedings of the International Conference on Innovative Computing and Communications (Advances in Intelligent Systems and Computing), Delhi, India, 19–20 February 2022; pp. 159–171.

23. Nai, W.; Liu, Y.; Rempel, D.; Wang, Y. Fast hand posture classification using depth features extracted from random line segments. *Pattern Recognit.* **2017**, *65*, 1–10. [CrossRef]
24. Tharwat, G.; Ahmed, A.M.; Bouallegue, B. Arabic Sign Language Recognition System for Alphabets Using Machine Learning Techniques. *J. Electr. Comput. Eng.* **2021**, *2021*, 2995851. [CrossRef]
25. Gunji, B.M.; Bhargav, N.M.; Dey, A.; Zeeshan Mohammed, I.K.; Sathyajith, S. Recognition of sign language based on hand gestures. *J. Adv. Appl. Comput. Math.* **2022**, *8*, 21–32. [CrossRef]
26. Podder, K.K.; Chowdhury, M.E.H.; Tahir, A.M.; Mahbub, Z.B.; Khandakar, A.; Hossain, M.S.; Kadir, M.A. Bangla sign language (BdSL) alphabets and numerals classification using a deep learning model. *Sensors* **2022**, *22*, 574. [CrossRef]
27. Alsahaf, A.; Petkov, N.; Shenoy, V.; Azzopardi, G. A framework for feature selection through boosting. *Expert Syst. Appl.* **2022**, *187*, 115895. [CrossRef]
28. Mathew, T.E. A logistic regression with recursive feature elimination model for breast cancer diagnosis. *Int. J. Emerg. Technol.* **2019**, *10*, 55–63.
29. Misra, P.; Yadav, A.S. Improving the classification accuracy using recursive feature elimination with cross-validation. *Int. J. Emerg. Technol.* **2020**, *11*, 659–665.
30. Shrivastava, S.; Jeyanthi, P.M.; Singh, S. Failure prediction of Indian Banks using SMOTE, Lasso regression, bagging and boosting. *Cogent Econ. Financ.* **2020**, *8*, 1729569. [CrossRef]
31. Gunduz, H. An efficient stock market prediction model using hybrid feature reduction method based on variational autoencoders and recursive feature elimination. *Financ. Innov.* **2021**, *7*, 28. [CrossRef]
32. Mavi, A. A New dataset and proposed convolutional neural network architecture for classification of american sign language digits. *arXiv* **2020**, arXiv:2011.08927.
33. Barczak, A.L.C.; Reyes, N.H.; Abastillas, M.; Piccio, A.; Susnjak, T. A new 2D static hand gesture colour image dataset for ASL gestures. *Res. Lett. Inf. Math. Sci* **2011**, *15*, 12–20.
34. Jacob, J. American Sign Language Dataset. 2022. Available online: https://www.kaggle.com/datasets/joannracheljacob/american-sign-language-dataset (accessed on 18 July 2022).
35. Priscilla, C.V.; Prabha, D.P. A two-phase feature selection technique using mutual information and XGB-RFE for credit card fraud detection. *Int. J. Adv. Technol. Eng. Explor.* **2021**, *8*, 1656–1668. [CrossRef]