Vision-based mobile sign language recognition system using Convolutional Neural Network

Harlina Harun*[a], Kek Li Yuan[a], Harlisya Harun[b]

[a] School of Computing, Faculty of Engineering, Science and Technology, Nilai University, Negeri Sembilan, Malaysia.
[b] Satellite Communication and Avionics System Integration (SCASI), Research Lab, UNIKL MIAT, Selangor, Malaysia.

**Corresponding author and e-mail: Harlina Harun, harlina@nilai.edu.my

Abstract

Sign language is a mode of communication between the hearing-impaired community and the rest of the society. Unfortunately, hearing-impaired community across the globe are still facing different obstacles within this modern society in their daily life. One such contributing factor for this saddening reality could be the lack of educational opportunity and exposure for an average hearing person towards sign language. To bridge such communication gap, researchers have been attempting to develop sign language recognition systems using a wide variety of approaches. The current state-of-art systems can be categorised into vision-based systems which are still prone to factors such as light conditions, hand-occlusion, and background noise, as well as glove-based systems that requires users to carry around the glove when the system is to be used. Little has been done on mobile platform. Therefore, to overcome this limitation, a sign language recognition system using Convolutional Neural Network (CNN) is developed. The main objective of this project is to identify signs captured through the device's camera and outputs the result in real-time. The system is written in Phyton and utilizes the MobileNet V2 model's base layers for feature extraction and several additional layers for classification. The CNN is created and trained through transfer learning and the developed system achieves an overall classification accuracy of 81.14%.

Keywords: *Convolutional Neural Network; Sign language recognition system; Vision-based data acquisition method; MobileNetV2; classification*

1. INTRODUCTION

Gestures can be used as a form of non-verbal communication, which fundamentally involves utilizing bodily action or movements to express and/or convey certain desired messages. For example, a child may express their desire to receive a hug from someone by extending and holding out their arms towards the person. Hand gestures, specifically, involves hand movements that is being utilised to achieve the same goals. For instance, a person waving their hand to another person can be interpreted as an intention to extend greetings to the receiver of the gesture. On the other hand, sign language is a more defined and discrete set of gestures, where these specific gestures correspond to some specific and predefined meaning and semantics. In this system of communication, each sign is identified with a specific type of sign language, such as Malaysian Sign Language (i.e., "Bahasa Isyarat Malaysia (BIM)"), American Sign Language (ASL), Chinese Sign Language (CSL) and to name a few. As sign languages has their own distinctive and predefined lexicon and grammar, they serve as a communication medium for the hearing-impaired community as a generic full-fledged natural language.

Sign languages are not universal, and they are not mutually intelligible with each other, although there are also striking similarities among sign languages. For instance, the thumbs-up gesture is a sign of approval in most countries. However, in several countries such as Iran, Afghanistan, and Thailand, showing someone such gesture is similar to showing them the middle finger in the United States, which is ultimately derogatory in nature. Generally, sign languages consisted of four components, where each of them is restricted within a limited range of possibilities. Excluding facial expression as a factor, a sign is composed of hand shape, hand orientation, location, and movement. To a certain extent, the meaning and semantic of a sign is dictated by each component. Due to this reason, the meaning of a sign can be drastically altered should any of these components are changed (Greenberg, 2012).

Sign languages have developed as an effective medium of communication between the hearing-impaired community, and it has become the core of their cultures. Although it is apparent that sign languages are primarily utilized by the deaf and hard of hearing communities, on several occasions, it is also used by hearing individuals. Some examples for this include individuals who have difficulties in speaking, individuals who experience challenges with spoken language due to a disability or condition, as well as the family members or communities who live with the individuals.

From the perspective of a sign language user, the process of communicating with a non-signer is an immensely challenging task. Multimedia information which can be easily perceived and understood by non-signers might not always be as easily accessible by the signers unless additional information such as subtitles are provided. An example for this sad reality can be seen when the deaf and hard of hearing community accesses the public transport facilities. They might have a hard time obtaining the important information that are being broadcasted in these places. On top of that, the lack of exposure and the educational opportunity for an average hearing person to learn about sign language is fuelling up the communication barrier between the deaf community and the rest of the society.

Therefore, to bridge such communication gap between the society, research have been performed extensively to recognise sign language within the last few decades. Sign language recognition is a collaborative research field that involves several areas such as computer vision, pattern recognition, linguistics, and others. The objective of this research is to create an amalgamation of algorithms that can identify signs and their respective meaning. The techniques and steps involved in sign language recognition systems varies according to the goals or objectives of the particular system. Generally speaking, the steps involve gesture acquisition, hand tracking, feature extraction, and classification. The first step, which is gesture acquisition will capture the required gesture data as an input for the system. The next step will be hand tracking, which involves visual-based systems in tracing the location of hand and discarding irrelevant data (such as, background). In the feature extraction stage, the goal is to extract relevant and useful data to be utilized and used for classification. Lastly, in the classification step, the meaning of the captured sign is deduced from the extracted features.

The existing sign language recognition system can be classified into two types, which are sensor-based and vision-based sign language recognition systems. The former approach involves physical interaction between user and sensing devices such as gloves with cables connected. This approach usually utilizes electromyography, inertial measurement or electromagnetic to collect finger flexion, position, orientation or angle data of the sign performed. Although this method provides the advantage of acquiring much more accurate gesture feature data in comparison to vision-based approach, it comes with the obligation of wearing an additional device with decreases the user-friendliness of the system. Not only does the users have to carry the device around during the day for possible usage, the time they need to assemble the devices prolongs the duration required for the intended use, thus affecting the efficiency of the system.
On the other hand, the latter approach, such that vision-based sign language recognition system uses data collected from images or video frames captured using camera as input of the system. Since the user does not require to wear a burdensome device to achieve an acceptable accuracy in recognition and sufficient speed in computation, this option stands out as the most practical one. However, in order to achieve satisfactory results, there are many challenges that must be considered and overcome, such as light conditions, background noise, computation and energy limitations etc.
Despite many works have been done on computer platform, little has been done on mobile platform. Previous research of sign language translation on smartphone have shown that the limitation in processing power in smartphone is one of its major constraints (Joshi, et al., 2015). However, the benefits of using a smartphone platform, such as its mobility and ease of use, are a great advantage over desktop systems. It is important for the infrastructure of any gesture recognition system to be practical as we aim for the developed system to be used in real life scenarios. Based on a research done by Ghanem et al. (2017)**,** the existing smartphone sign language recognition systems are extremely limited as most existing vision-based methods only recognize not more than ten single-handed static gestures. Some proposed approaches are even impossible to run in real-time since they consist of several deep CNNs on multiple input modalities, which is forcing the limits of memory and power budget (Narayana, et al., 2018).

Therefore, in order to close the aforementioned gaps, a sign language recognition system using CNN is proposed. With the widespread availability of powerful mobile devices for the general public in this globalization era, the proposed system aims to be installed on smart phone with Android operating system. In attempt to increase the practicality and user-friendliness of the system, the system will be utilizing vision-based data acquisition method that utilizes cameras, which is generally available in most smartphone devices. This eliminates the need of additional devices to acquire data, which can be a troublesome process. In addition, the system is designed to interpret sign language performed by users and provide the results in real-time.

2. VISION-BASED MOBILE SIGN LANGUAGE RECOGNITION SYSTEM

In a vision-based mobile sign language recognition system, there are generally several main stages involved, including gesture acquisition, feature extraction, and classification. In the gesture acquisition stage, the hand movements are bring retrieved from vision-based sources such as a camera. The acquired images then go through an optional pre-processing phase, where techniques such as noise removal, dilation and erosion are applied in attempt to improve the result of recognition.

In the feature extraction stage, relevant features are being extracted, which then acts as the input for the classification stage. As the name implies, the classification stage involves the execution of classification algorithms to classify the detected sign. In attempt to achieve real-time detection, the users are to ensure that the hands of the signers are captured by the camera. There are several machine learning and deep learning algorithms that are frequently been utilized in sign language recognition systems, which include Support Vector Machines, Artificial Neural Networks, as well as Convolutional Neural Networks.

2.1 Support Vector Machine

A Support Vector Machine (SVM) is an instance of supervised machine learning algorithm. It is generally used to perform tasks such as classification and regression (Revanth & Raja, 2019). It is generally representing data points of distinct classes in a multidimensional space, separated by hyperplanes, which uniquely defines a boundary between two types of data (Fatmi, et al., 2019). This hyperplane is simply a line that separates the two types of data. It is generated by the SVM repetitively to minimize the classification error. SVM can only distinguish between two classes to perform binary classification. To illustrate this with an example, in order to create a support vector machine that can perform classification on multiple classes of animals, a binary classifier shall be created for each animal. Specifically, for a 'cat' class, there should be a binary classifier that predicts whether an instance data is a cat or not.

Amongst all of the binary classifiers, the classifier with the highest score will be chosen as the output for the classification. It is apparent that the support vector machine performs more effectively in high dimensional spaces with clear margin of separation between classes (Sahoo, et al., 2019).However, it does not perform well in large datasets, as there is a higher possibility of existence of noise in the datasets, causing the overlapping of target classes. In addition, as explained previously, the output of a support vector machine is only whether or not an instance data belongs to a specific class, in such a way that it does not directly provide estimation results in terms of possibility, which are usually desired in classification problems. In addition, the accuracy of the SVM is dependent upon the setting appropriate value for the key parameters, makes it difficult to fine-tune and achieve an optimum and satisfactory result (Jiang & Ahmad, 2019).

Figure 1 shows a plotted graph of linearly separable data (A) and non-linearly separable data (B). A linearly separable data is any data that can be separated into distinct classes by a straight line when it is being plotted in a graph. Therefore, to accurately classify non-linearly separable data, a Kernelized SVM shall be utilized. The algorithm of a Kernelized SVM involves transforming the non-linearly separable data from one dimension into two dimensions so that the data become linearly separable in the 2D (Fatmi, et al., 2019). This process involves mapping the one-dimensional data point into its respective two-dimensional ordered-pair. In simpler words, a non-linearly separable data can be converted into a linearly separable data by mapping it into a higher dimension (Jiménez, et al., 2019).

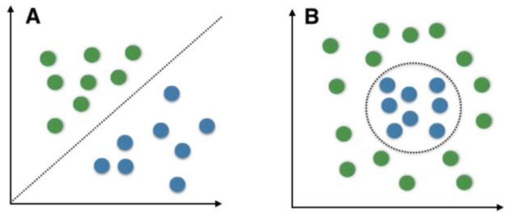


Fig. 1. Linearly Separable Data Versus Non-Linearly Separable Data

2.2 Artificial Neural Network

An artificial neural network (ANN) is a computational model or information processing paradigm that is vaguely inspired by biological neural networks of animal brains (Khan, et al., 2019). It is designed to replicate how human brain analyse and process information. Modelled after the human brain that have hundreds and billions of interconnected neurons, an ANN is comprised of an enormous number of nodes that receives input, perform some form of mathematical operations and outputs the result (Arif-Ul-Islam & Akhter, 2018). These interconnected nodes work in unison to perform complex and sophisticated tasks, such as regression and classification problems. The general topology of an ANN is feedforward, where the data flow is unidirectional (Zhang, et al., 2019).

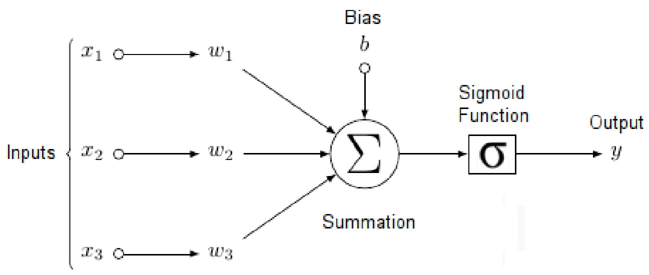


Fig. 2. Basic Sigmoidal Unit of ANN

Depending on where the node resides within the ANN, the input of a node can be either feature values of external data (raw input of the ANN), or it can be the output from other nodes. Each node is connected by edges that carry their own weights, representing their relative importance, and thus influencing the output result. The output of each node is called an activation. They are influenced by several parameters and operations that are associated with the node itself. As shown in Fig. 2., the output of a node can be deduced by calculating the summation of weighed sum of all inputs, together with the bias value (Rochez, et al., 2019). The result will then be passed through an activation function to generate an output.

In an ANN, nodes are organized into several layers. In general, nodes in one layer are connected only to its immediately preceding or following layers. The layers in an ANN can be generalized into input layers, hidden layers, and output layers in sequence. As the name implies, the input layer involves the nodes that receives external data, and the output layers produces the ultimately desired results (Fatmi, et al., 2019). In between the input and output layer, there can be zero or up to many hidden layers. Learning is the process of an ANN undergoing adaptation in order to perform intended tasks. The learning process of an ANN generally involves a repetitive process of finetuning of the nodes' weights to improve the result accuracy. In a more practical term, the goal of this process is to minimize the loss function. A loss function denotes the degree in which the network's behaviour deviates from the ideal or desired behaviour. Thus, the performance of the network can be verified by testing it against any input other than the training dataset, calculate the loss function based on the disparity between predicted value and the desired value, and the adjust the network's weights based on the loss function value (Al-Sammarraie, et al., 2018). This process is repeated until the value of loss function does not decrease anymore.

There are several methodologies to calculate the loss function. A commonly adopted supervised learning algorithm is called the back-propagation algorithm. The learning of the network starts by initializing the network's nodes with random weights (Dike, et al., 2018). For each iteration of the learning process, a two-step process, namely forward pass and backward pass, is executed. The forward pass simply involves providing a training data to the input and calculating the value of loss function, while the backward pass involves changing the weights to minimize the loss function, starting from the output layer back to the input layer (Al-Sammarraie, et al., 2018).

2.3 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a class of neural network, which is usually deployed to perform visual imagery analysis. Just like ANNs, CNNs were inspired and built to resemble the biological processes of the animal brain. As the name implies, the main operation of a CNN involves mathematical operations called convolution. Similar to ANNs, a CNN consists of input, output, and hidden layers. The common hidden layers include convolutional layers, non-linearity layers, pooling layers, flattening layers as well as fully connected layers (Suresh, et al., 2019). A CNN receives an input tensor, convolves the input and pass it to the next layers, until it reaches the output layer and generate a result (Novak, et al., 2020).

The convolutional layers, the main building block of CNNs are in charge of detecting features. This is usually done by applying filters to an image, which is capable in extracting specific attributes from the image (Alom, et al., 2019). Filters usually take the form of multi-dimensional array of pixel values. To perform convolution, filters are being applied from top-left to bottom-right of the image iteratively, creating an output matrix called the feature map. A feature map simply indicates the location of the feature in an image. There are two additional parameters that are essential in adjusting the convolution process, which is stride and padding (Rao, et al., 2018). A stride dictates how a filter is applied to an image by controlling the distance between pixel values whereby the filter is applied, while padding is to add zero values surrounding the feature map to preserve its size.

Fig. 3 illustrates the convolution process. Imagine positioning the filter on the starting location of the image, as shown in Fig. 3(a). Then, the element-wise multiplication between the filter and the image pixel value is calculated and stored in the output matrix. Next, the filter is moved to the right by a pre-defined stride value, and iterate this process, until there is no more image pixel value on the right of the same row. Then, the filter will be placed all the way back to the left in the next row. This process is repeated until the filter is being applied to every pixel value of the image. In short, a convolutional layer detects the location of features in an image by applying filter through the image, using simple element-wise multiplication (as in Fig. 3(b)).

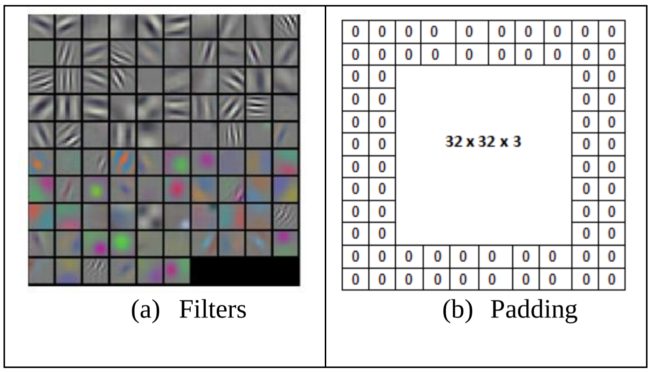


Fig. 3. Convolution Process

Non-linearity is a mathematical term used to describe indirect or unpredictable relationship between variables. To plot the relationship between the variables on a graph, non-linear relationships create a curve, as opposed to linear relationships that creates a mere straight line on the graph. Non-linearity is a common problem during the examination of cause and effect relationships. Based on this definition, it is apparent that non-linearity layers are used to introduce non-linearity into the system. Generally, the non-linearity layer is added after every convolutional layers. This layer is sometimes called an activation layer due to its utilization of activation functions. Some frequently adopted non-linear functions include sigmoid, hyperbolic tangent (tanh), and the Rectifier function (also known as the "ReLU layer") (Rao, et al., 2018). This is illustrated in Fig. 4.

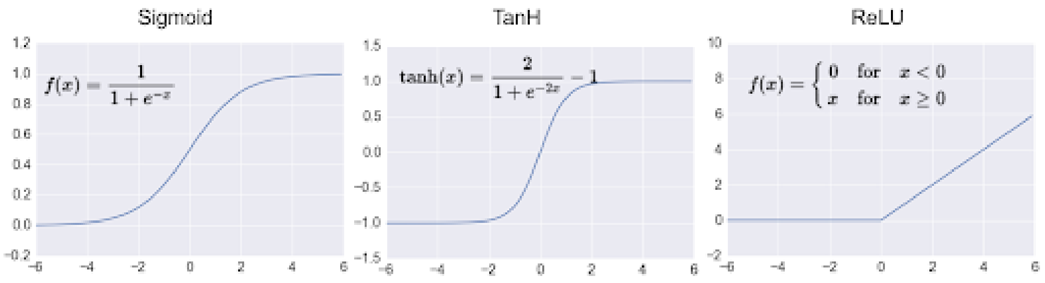


Fig. 4. Sigmoid, Tanh and ReLU Activation Function Graphs

The pooling layer is usually placed between successive convolutional layers in a CNN. The main functionality of this layer is to reduce the dimensionality of matrices in the CNN, and in turn reducing the computation required, as shown in Fig. 5. This pooling process can control the over-fitting of the network. There are several pooling operations, where the popular ones include max pooling and average pooling. To perform the pooling operation, the input feature map is divided into patches. Then, the max pooling layer extracts the most activated presence of a feature, while the average pooling layer extracts the average presence of a feature in each particular patch (Rahim, et al., 2019).

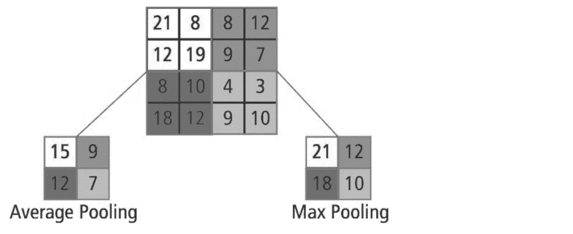


Fig.5. Average Pooling and Max Pooling

The flattening layer is a very simple and intuitive layer that is used to prepare and process the data to be the input of the Fully Connected Layer. This step is vital because the Fully Connected Layer usually receives data in only one dimension. This layer takes in the input data and transform them into the required mono-dimensional tensor (Suri & Gupta, 2019). In a CNN, the final layer and the exact layer that actually performs the classification task is called the Fully Connected Layer. This layer takes the mono-dimensional tensor from the previous flattening layer and generate the final classification (Suri & Gupta, 2019).

2.4 Rational behind-Convolutional Neural Network Approach

In terms of functionality, both SVM and NN are able to perform non-linear decision classification. An SVM does this through the usage of kernels, while the NN handles non-linearity through application of activation functions. An SVM learns and performs classification by representing a data of two different classes in a multidimensional space, separated by hyperplanes. To perform classification tasks that involves more than two classes, we need to create an SVM for each of the classes. In contrary, a single neural network can be trained to perform classification of several classes in one setting. The output of a NN will be the probability (i.e., confidence) of classification for each class, which is usually desired in classification tasks. On the other hand, an SVM does not directly provide the classification result in probability form.

An SVM generally performs effectively given only relatively small dataset as the training data (Alom, et al., 2019). This is because SVM identifies the optimum hyperplanes based on the support vectors. Therefore, with the pre-condition that a dataset with well-separated classes is provided, the SVM only needs less training to perform the classification well. On the other hand, the training for a neural network is based on the batches of data it was provided. This implies that the decision boundary learnt by the neural network is highly dependent upon the amount of data being fed. The size of neural networks is fixed. A neural network consists of an input layer, several hidden layers and an output layer. The size of a neural network model, depending on the number of layers, the number of nodes in each layer are fixed and defined during the process of model creation. In contrast, an SVM consists of a collection of support vectors, which are being selected from the training set. In the worst-case scenario, the number of support vectors will be the same as the number of training samples, which causes the size of the model to increase linearly.

From the viewpoint of model output, it is apparent that a neural network is more suitable to be utilized in this project. As mentioned previously, a neural network is capable of performing classification of multiple classes in one training. On top of that, the output result is in the form of probability, which is usually desired in classification tasks. Although we can handle multi-class problems by creating multiple SVMs for the different output classes, the training time and effort required, as well as the classification performance of the model might not be efficient.

An ANN is capable of learning non-linear relationships between an input and output by using activation functions. In order for an ANN to perform image classification tasks, the very first step involves converting the two-dimensional image into a one-dimensional vector before passing it into the input layer. This process itself is inefficient and computationally expensive due to the gigantic number of trainable parameters. To put numbers into perspective, consider an ANN that receives images with size 240×240 as an input. The number of trainable parameters at the first hidden layer with only five neurons will be 240×240×3×5 (width × height × RGB channel ×

number of neurons), which is 864,000. Not only that, but this process also makes ANN loses the spatial features of a picture. On the other hand, a CNN captures relevant features from the input data by automatically learning appropriate filters. Not only that, an CNN is able to retain spatial features of an image, which is vital for image classification tasks. A spatial feature refers to the relationship between the adjacent of pixels in an image. Spatial features are essential as they help the CNN to recognise the identity, location and other attributes of an object in the image.

Based on this comparison, it is apparent that CNN is the ideal neural network to be applied in this project due to its characteristics and efficiency in image classification tasks. Unlike SVMs that only produces binary outputs without probability, a single CNN model is able to perform multi-class classification task. In addition, complicated real-world data is likely to cause SVMs to produce a large amount of support vectors and thus increase the model size, which might be not conducive to be incorporated and installed in mobile systems. In contrary to ANNs that is computationally expensive in performing image classification tasks, pooling layers in CNNs reduces the number of training parameters of the model, and thus lessen the required computational power. Lastly, convolutional layers in CNNs allows them to recognize spatial relationship between pixels, making them more conducive in performing image classification tasks.

3. RESEARCH METHODOLOGY

The creation and training of a high accuracy and performance machine learning model from scratch is definitely not an easy task. It requires massive datasets to train the model in order to prevent issues such as overfitting, while the process itself consumes a huge amount of time as well as computing power. Therefore, to achieve a maximum performance by utilizing a minimum amount of training time and computing power, the proposed model will be created and trained by utilizing transfer learning. Through this method, the proposed model can capitalize on the performance and capability of the pre-trained model layers with a minimal amount of training time, computational resource and data. In this project, the pre-trained model used is MobileNetV2. It is a CNN model developed and targeted to be operated on mobile devices, which is being trained with 1.4 million images of one thousand classes.
The proposed training methodology for the model consist of five stages:

1. Stage one: Collect and Pre-process Datasets
2. Stage two: Load and Freeze Base Model
3. Stage three: Build and Compile Model
4. Stage four: Train Model
5. Stage five: Save Model

The first stage involves collecting the required dataset. that include images of hand performing specific signs. The dataset should be split into performing training, validation and testing respectively. The ratio of splitting of dataset into the training and validation will be in the ratio of 80:20. In addition, the dataset will be pre-processed by performing augmentation and rescaling. Dataset augmentation is performed by repeatedly applying random rotation to images. This practise is done to introduce diversity of the data in the dataset, preventing issues such as overfitting of the model. Rescaling of the dataset is done to ensure all the images are in the same shape, scale and pixel value.

The next stage (i.e., second stage) includes writing codes to declare and load the layers of MobileNet V2 that will be utilized in the project. The loaded layers of the MobileNet V2 model will be frozen by declaring the model as untrainable so as to retain the pre-trained weights of the MobileNet V2 layers. This is followed by stage three, in which codes need to be written to append customised layers after the loaded MobileNet V2 layers. These layers will be combined, built and compiled as a single coherent CNN model to be used in the proposed system. When the model is being fed with an image, the image will act as the input to the retained initial layers of MobileNet V2. The output of this process will be the internal activations from the MobileNet V2 layers. These outputs will in turn act as the input for the appended layers to perform further computation and classifications.

Utilizing the processed datasets to train the compiled model lies in stage four. This process will cause the appended layers of the models to fine-tune their weights. Finally, in stage five, the trained model can be saved into desired format after obtaining the desired results. The model will be saved into tflite format, which is mobile-friendly.

Fig. 6. shows the flowchart of the proposed system which include eight stages:

1. Stage one: Request Camera Permission
2. Stage two: Choose and Open Camera
3. Stage three: Instantiate Classifier
4. Stage four: Camera Frame Acquisition
5. Stage five: Pre-process Data
6. Stage six: Classify Frame
7. Stage seven: Sort Result
8. Stage eight: Output Result

As the system relies on using the device's camera to perform necessary operations, the system should request camera permission from the user if the permission is yet to be granted (i.e, first stage). The system will not be able to proceed to any further stages if the user chooses not to grant the required permission and will continuously request permission from the user until it is granted. The next stage (i.e., stage two) involves retrieving and identifying the list of available cameras of the device, and to open an appropriate camera to proceed with the process. Then, in the third stage of the processes, necessary operations need to be performed to instantiate the classifier, such as loading the model and label files from the asset folder etc. After gaining access and opening the camera, this stage (such that, stage four) involves acquiring the camera frames from the opened camera. Stage five includes applying necessary pre-processes to the acquired camera frames before utilizing it to perform classification, i.e. converting the acquired frame to Byte Buffer format. This stage involves using the pre-processed data as the input to perform classification. After obtaining the classification result from the previous stage, results need to be sorted in descending confidence level of the classification. The last stage involves displaying the top three sorted results (three classification result with the highest confidence level), as well as the time cost for classification on the user interface.

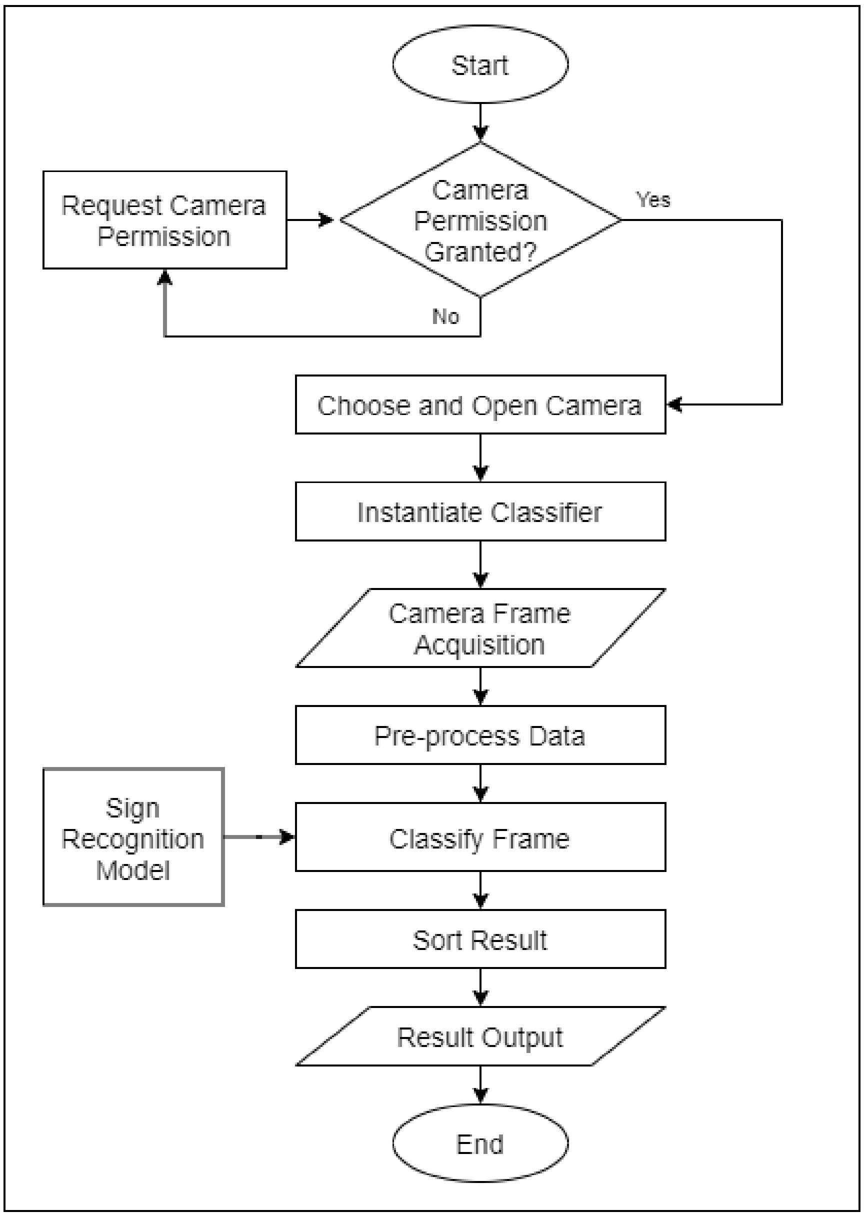
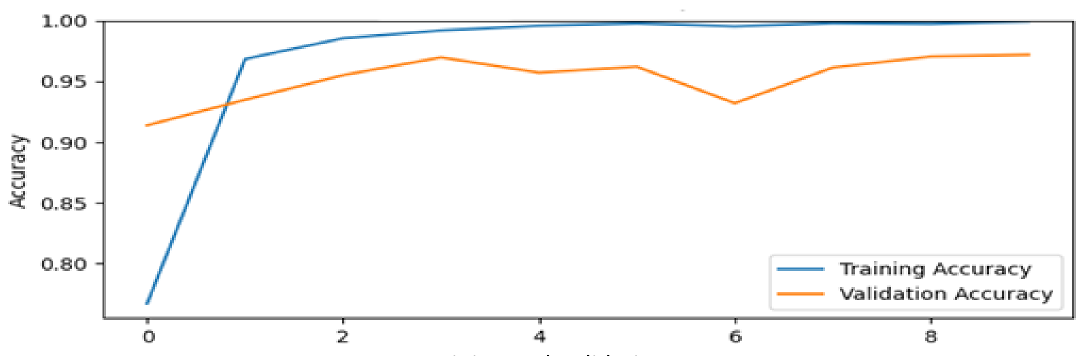


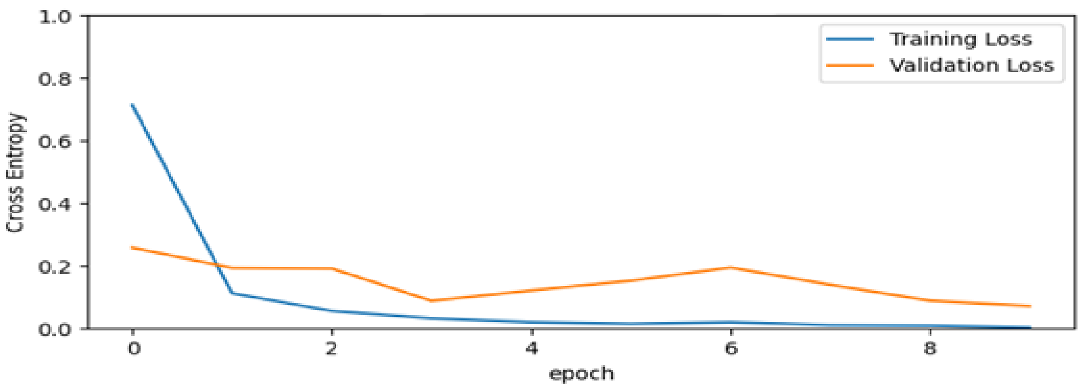
Fig.6. Flowchart of the Proposed System

4. RESULTS AND DISCUSSION

The performance of CNN is tested by applying two methods. The first method involves observing the accuracy and loss function of the CNN model during training and validation stage, while the second method involves manually testing the output produced by the CNN when it is being presented with brand new testing data.



a. Training and Validation Accuracy



b. Training and Validation Loss

Fig. 7. Learning Curves of the CNN Model

Fig. 7 shows the learning curves of the CNN model, where the accuracy and the loss of the model during the training and validation phase is plotted. Fig 7(a) illustrates the training and validation accuracy of the model, which is being logged after every epoch (iteration of training batch). The blue line (denotes the training accuracy) starts with a steep increment in gradient, indicating a high learning rate. The orange line (denotes the validation accuracy) however shows much less learning rate, but with a high average value. The gap between these two lines indicates the occurrence and extent of overfitting, which in this case, the model shows some overfitting towards the training data.

The training and validation loss of the model is shown in Figure 7(b). Similar to the training accuracy, the training loss starts with a steep decrement in gradient, indicating a high initial learning rate. The validation loss in turn shows more consistent and lower value of loss function. In average, the difference between training and validation loss decreases across each epoch, showing that the model is able to fine-tune itself to minimize errors in predictions.

Table 1. Overall Accuracy of the CNN Model

Sign	Number of Test	Number of Correct Detection	Accuracy (%)
A	100	93	93%
B	100	87	87%
C	100	92	92%
D	100	83	83%
E	100	89	89%
F	100	61	61%
G	100	63	63%
		Overall Accuracy:	81.14%

As shown in Table 1., the CNN model has achieved an overall accuracy of 81.14%. Based on the results, it is apparent that the model is able to recognize signs from letter A to letter E with the accuracy above eighty percent. Unfortunately for letter F and G, the model tends to under-perform with the average accuracy of sixty-two percent. The causation for such result might be factors such as bad lightning conditions, similar background colour etc. An extracted sample of some tested images, along with their predicted results are plotted in Fig. 9.

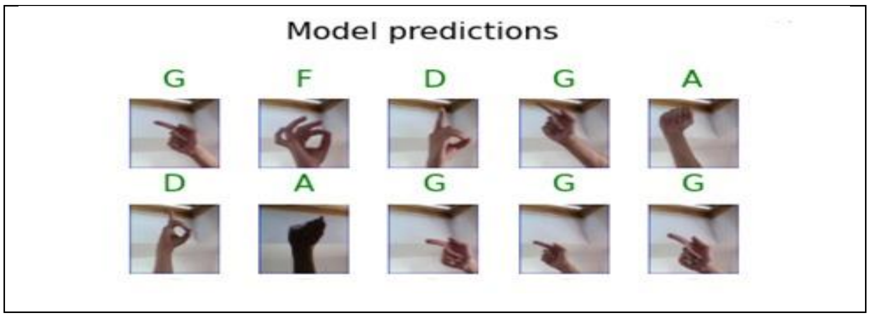


Fig. 9: Testing Results

In summary, the performance of CNN is evaluated by plotting the learning curves of the CNN model as well as testing it against a new dataset. By plotting the learning curve, the accuracy and loss of the model during the training and validation phase on each epoch can be visually inspected and evaluated. On the other hand, the proposed CNN model used to classify sign language in the system has achieved an overall accuracy of 81.14% when it was being tested against a new dataset.

5. CONCLUSIONS

In a nutshell, a vision-based mobile sign language recognition system using CNN which can be used to classify sign language captured through the mobile device camera in real-time is developed. The application is developed using Android Studio IDE using Java programming language, while the codes used to create, train and test the CNN model are developed using Python programming language in text editors. While the current state-of-art system are mainly focused on sensor-based and computer-based system which can be inefficient and impractical for daily uses, this project focuses on creating a vision-based mobile sign language recognition system using CNN which is capable of produce the classification result in real-time. By applying the proposed methodology, the developed system has achieved an overall accuracy of 81.14%.

Although the developed system is able to perform classification on the captured signs and display the results in real-time, the quality of the system is drastically degraded due to its inability to determine the presence or absence of hand during classification. Therefore, the hand detection stage is highly recommended to be added into the system to increase the quality and accuracy of classification results. The presence of the hand detection stage can be followed by a hand segmentation phase, which can be used to separate the background data. This process can greatly reduce redundant computation due to background noise as well as improving the efficiency of the system.

In addition, as a visual-gestural language that heavily involves bodily movements, a system that only considers the visual shape of the hand is inherently limited. Thus, future works are encouraged to consider factors such as the expressions of the face, actions produced by upper body parts other than the hands (head, eyebrows, mouth, shoulders) etc. so as to fully capture other subtle cues that can greatly alter the meaning of signs. On the other hand, the capability of the proposed system is only capable of identifying static signs one at a time. In other words, the recognized signs involve no movement, and the meanings are separated with the preceding and following sign. As sign language involves static and dynamic signs, future works are encouraged to work on identifying continuous signs (that involve both static and dynamic signs). A continuous sign is simply a set or group of signs which are performed continuously to form a sentence. Identifying continuous signs (segmentation of signs) itself is an enormous challenge as it is difficult to accurately determine the precise start and end time of each sign in the sentence, let alone the task to accurately classify the signs.

In conclusion, the proposed vision-based mobile sign language recognition system is successfully developed by using Java in Android Studio IDE. With the proposed CNN model developed using Python scripts to classify signs, the project objectives which involves classifying signs on mobile using vision-based methods have been fulfilled.

REFERENCES

Alom, M. S., Hasan, M. J. & Wahid, M. F., (2019). Digit Recognition in Sign Language Based on Convolutional Neural Network and Support Vector Machine. *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pp. 1-5.

Alom, M. S., Hasan, M. J. & Wahid, M. F., (2019). Digit Recognition in Sign Language Based on Convolutional Neural Network and Support Vector Machine. *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pp. 1-5.

Al-Sammarraie, N. A., Al-Mayali, Y. M. H. & El-Ebiary, Y. A. B., (2018). Classification and diagnosis using back propagation Artificial Neural Networks (ANN). *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, pp. 1-5.

Ananya, C., Anjan, T. & Kandarpa, S., (2015). A Review on Vision-Based Hand Gesture Recognition and Applications. pp. 261-286.

Anon., (2019). *developers.* [Online]
Available at: https://developer.android.com/guide/components/fragments
[Accessed 30 August 2020].

Anon., (2020). *developers.* [Online]
Available at: https://developer.android.com/reference/kotlin/androidx/appcompat/app/AppCompatActivity
[Accessed 30 August 2020].

Arif-Ul-Islam & Akhter, S., (2018). Orientation Hashcode and Articial Neural Network Based Combined Approach to Recognize Sign Language. *2018 21st International Conference of Computer and Information Technology (ICCIT)*, pp. 1-5.

Bora, A., (2020). *GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-support-vector-machines-svm/
[Accessed 25 August 2020].

Dike, H. U., Zhou, Y., Deveerasetty, K. K. & Wu, Q., (2018). nsupervised Learning Based On Artificial Neural Network: A Review. *2018 IEEE International Conference on Cyborg and Bionic Systems*, pp. 322-327.

Fatmi, R., Rashad, S. & Integlia, R., (2019). Comparing ANN, SVM, and HMM based Machine Learning Methods for American Sign Language Recognition using Wearable Motion Sensors. *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0290-0297.

Ghanem, S., Conly, C. & Athitsos, V., (2017). *A Survey on Sign Language Recognition Using Smartphones.* s.l., s.n., pp. 171-176.

Greenberg, P., (2012). *Sign Can You.* s.l.:s.n.

Ibraheem, N. & Khan, R., (2018). Vision based gesture recognition using neural networks approaches: A review. *International Journal of human Computer Interaction (IJHCI)*, 3(1), pp. 1-14.

Jiang, X. & Ahmad, W., (2019). Hand Gesture Detection Based Real-Time American Sign Language Letters Recognition using Support Vector Machine. *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 380-385.

Jiménez, G., Moreno, E., Guzman, R. & Barrero, J., (2019). Automatic method for Recognition of Colombian Sign Language for vowels and numbers from zero to five by using SVM and KNN. *2019 Congreso Internacional de Innovación y Tendencias en Ingenieria (CONIITI)*, pp. 1-6.

Jin, C. M., Omar, Z. & Jaward, M. H., (2016). *mobile application of American sign language translation via image processing algorithms.* s.l., s.n., pp. 104-109.

Joshi, T., Kumar, S., Tarapore, N. & Mohile, V., (2015). Static Hand Gesture Recognition using an Android Device. *International Journal of Computer Applications 120*, Volume 21.

Khan, S., Ali, M. E., Das, S. & Rahman, M. M., (2019). Real Time Hand Gesture Recognition by Skin Color Detection for American Sign Language. *2019 4th International Conference on Electrical Information and Communication Technology (EICT),* pp. 1-6.

Lahoti, S. et al., (2018). Android Based American Sign Language Recognition System with Skin Segmentation and SVM. *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT),* pp. 1-6.

Murthy, G. & Jadon, R., (2019). A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management,* 2(2), pp. 405-410.

Narayana, P., Beveridge, J. R. & Draper, B. A., (2018). *Gesture recognition: Focus on the hands..* s.l., 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Novak, B., Ilić, V. & Pavković, B., (2020). YOLOv3 Algorithm with additional convolutional neural network trained for traffic sign recognition. *2020 Zooming Innovation in Consumer Technologies Conference (ZINC),* pp. 165-168.

Rahim, M. A., Shin, J. & Islam, M. R., (2019). Dynamic Hand Gesture Based Sign Word Recognition Using Convolutional Neural Network with Feature Fusion. *2019 IEEE 2nd International Conference on Knowledge Innovation and Invention (ICKII),* pp. 221-224.

Rao, G. A., Syamala, K., Kishore, P. V. V. & Sastry, A. S. C. S., (2018). Deep convolutional neural networks for sign language recognition. *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES),* pp. 194-197.

Rautaray, S. & Agrawal, A., (2015). Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review,* 43(1), pp. 1-54.

Revanth, K. & Raja, N. S. M., (2019). *Comprehensive SVM based Indian Sign Language Recognition.* India, s.n., pp. 1-4.

Rochez, J., Woodruff, I., Rogers, M. & Alba-Flores, R., (2019). *Classifying Hand Gestures using Artificial Neural Networks for a Robotic Application.* s.l., s.n.

Sahoo, J. P., Ari, S. & Patra, S. K., (2019). Hand Gesture Recognition Using PCA Based Deep CNN Reduced Features and SVM Classifier. *2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS),* pp. 221-224.

Suresh, S., Mithun, H. T. & Supriya, M., (2019). Sign Language Recognition System Using Deep Neural Network. *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS),* pp. 614-618.

Suri, K. & Gupta, R., (2019). Convolutional Neural Network Array for Sign Language Recognition Using Wearable IMUs. *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN).*

Wario, R. & Nyaga, C., (2019). A Survey of the Constraints Encountered in Dynamic Vision-Based Sign Language Hand Gesture Recognition. *Universal Access in Human-Computer Interaction. Multimodality and Assistive Environments,* Volume 11573, pp. 373-382.

Zhang, Z., Su, Z. & Yang, G., (2019). Real-Time Chinese Sign Language Recognition Based on Artificial Neural Networks. *019 IEEE International Conference on Robotics and Biomimetics (ROBIO),* pp. 1413-1417.

Zhang, Y. a. Z. W. a. W. Y. a. X. L., (2020). A real-time recognition method of static gesture based on DSSD. *Multimedia Tools and Applications,* February.