International Journal of Web Science

The Evolution of Military Operations: Artificial Intelligence to Detect Hand Gestures in Defence

Varsha Naik*

Dr. Vishwanath Karad MIT World Peace University,
Survey No: 124, Paud Rd, Kothrud, Pune, 411038, Maharashtra, India
E-mail: varsha.powar@mitwpu.edu.in
*Corresponding author

Abhishek Chebolu

Dr. Vishwanath Karad MIT World Peace University,
Survey No: 124, Paud Rd, Kothrud, Pune, 411038, Maharashtra, India
E-mail: abhishek.chebolu@gmail.com

Janhavi Chavan

Dr. Vishwanath Karad MIT World Peace University,
Survey No: 124, Paud Rd, Kothrud, Pune, 411038, Maharashtra, India
E-mail: janhavi.a.chavan@gmail.com

Prajakta Chaudhari

Dr. Vishwanath Karad MIT World Peace University,
Survey No: 124, Paud Rd, Kothrud, Pune, 411038, Maharashtra, India
E-mail: prajakta.p.chaudhari@gmail.com

Snehalraj Chugh

Dr. Vishwanath Karad MIT World Peace University,
Survey No: 124, Paud Rd, Kothrud, Pune, 411038, Maharashtra, India
E-mail: snehalchugh2016@gmail.com

Ahbaz Memon

Dr. Vishwanath Karad MIT World Peace University,
Survey No: 124, Paud Rd, Kothrud, Pune, 411038, Maharashtra, India
E-mail: ahbazmemon0@gmail.com

Abstract: Artificial Intelligence innovations hold promise for improving military operations while gestures have been simplest and most powerful mediums for communications. This idea proposes a system identifying hand movements and translating them to spoken words, where soldiers on battlegrounds may readily converse with one another. We utilize Computational Vision, Haar Cascade Classifiers, CNN, Media Pipe and other approaches for thought patterns through results. There are three techniques in this process, firstly, Hand identification system that provides borders around the hand placed to another screen fixing image magnification using OpenCV and Matplot, following with a hand- held skeleton-projected connection model using Mediapipe's mapping system libraries in real-time capturing at 30 fps, lastly, Sound element being introduced in each class improving recognition of the gestures. Dataset of around 10000 images across five different classes are built with CNN architecture used to classify. This research shows results to discover AI options for military applications.

Keywords: Military; Gesture Recognition; Object Classification; Object Tracking; Haar Cascade; MediaPipe; Artificial Neural Network; Convolutional Neural Network

Reference to this paper should be made as follows: Varsha Naik, Abhishek Chebolu, Janhavi Chavan, Prajakta Chaudhari, Snehalraj Chugh and Ahbaz Memon. (201X) 'The Evolution of Military Operations: Artificial Intelligence to Detect Hand Gestures in Defence', *International Journal of Web Science*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes:

Varsha Naik is currently pursuing PhD in Computer Engineering from Savitribai Phule Pune University. She is an Assistant Professor at the MIT World Peace University's School of Computer Engineering and has been working there since Feb 2008 having a teaching experience of almost 13 plus years. Her main areas of study interest include Artificial Intelligence and Data Science. She teaches Database Management Systems, Data Warehousing and Data mining, and Software Engineering.

Abhishek Chebolu is a student pursuing final year of Computer Science and Technology (B. Tech) at MIT World Peace University. His research interests include Computer Vision, Cloud Computing, Artificial Intelligence, Cryptography, Malware Analysis.

Janhavi Chavan is a B. Tech student in the Department of Computer Science and Engineering, at MIT World Peace University, Pune. Her research interests include Cloud Computing, AI and Big Data Analytics.

Prajakta Chaudhari is a Final Year Computer Science and Engineering student at the MIT World Peace University, Pune. Her research interest includes Applications of Artificial Intelligence, Computer Vision, Cloud Computing and the Internet of Things.

Snehalraj Chugh is attaining a Bachelor (B. Tech) in the Department of Computer Science and Engineering, at MIT World Peace University, Pune. His research interests include Artificial Intelligence, Big Data Analytics, Software Project Management and Tiny Machine Learning

Ahbaz Memon is currently studying as a final year student at MIT World Peace University, Pune. His research interests include Artificial Intelligence, Computer Vision, Machine Learning, Deep Learning and Cloud Computing.

1 Introduction

A gesture is described as a muscle action of the wrists, fingertips, forearms, and other elements of the human body that allows people to communicate. Hand signs are among the most reasonable methods to create a user-friendly and adaptable interface between products and people (Kang and Tversky, 2016). For several years, vision-based palm position estimation has been explored. Hand gestures may be recognized using one of two techniques: postural (a steady hand form ratio without arm movements) or gesture (a dynamical hand movement with or without hand motions). Based on a theory, palm tracking addresses three key parts of computer vision: hand segmentation, tracking, & its detection, where the data gloves-based method, as well as the vision-based technique, are two methodologies for human & machine communication. One of the really common instances of a gesture recognition system is sign language and Three-Dimensional (3D) hand monitoring is a topic of great interest in the gaming industry.

This gesture recognition system is highly beneficial for soldiers wishing to connect with the various technologies that are located mainly in field operations. The requirement for movement recognition systems to function in much fewer settings when speech is not a feasible or optimal option motivates this study (e.g., loud environments, firing sounds, bombs, quiet places, etc). Nonetheless, motions are inherently intricate and ambiguous; so, we choose to concentrate on military palm gestures because they constitute a collection of motions with precise meanings.

The approach suggested in this study makes use of a camera to collect and identify hand movements given by the user. Hand motions offer a wide array of uses. Google's Mediapipe aids in improving recognition and accuracy. This study describes a unique method by making use of MediaPipe and CNN together as a hands-free hand gesture detection tool for the military (Zhang et al., 2020). Using this technique, it is feasible to show that motions can be recognized in real-time using regular webcams or ordinary 720p cameras available to the general public, whereas, before that, which was possible only with a backdrop earlier with several cameras, all focused on a hand at a time.

2 Data Generation

To come up with a solution to the problem, the statement works around the palm gestures in real-time which detects the hand signs and gestures the commander's hand movements, which uses picture analysis and image computing techniques. Until recently, some hand recognition datasets were accessible having static hand gestures available to the public which were designed to meet our problem statement, but the hand was not properly aligned in class and had problems with flickering and blurrier motions of the hand (Suresh and Niranjanamurthy, 2021). Having the above in regard, we decided to record a hand posture dataset containing hand gestures of the subject with steady and flicker images, i.e., images which are steady and also ones with flickers concerning each class, using OpenCV aiding to perform deep learning. We add flicker classes to make sure that the model learns about hand movements which aren't any gestures but just movement (Ranawat et al., 2021; Chandan et al., 2018), all of this helps the model to restrain itself from falsely detecting the hand. To our full knowledge, there is no other publicly accessible, large dataset with well-aligned hands for training CNNs on real-time military applications.

Our dataset was collected via a laptop's camera and the resulting picture is saved as a jpg file. We utilise five gestures in our algorithm: palm, fist, thumbs up, gun, and call. These individual gestures are often employed during large-scale military operations, where each of them is captured with a certain number of images by providing a delay between each input type. We set up each angle with various rotations of both the subject's hand in all three axes and laterally. We utilise OpenCV and Matplotlib in our data collecting procedure (Wright et al., 2010; Ari and Ustazhanov, 2014). All the required libraries such as Keras and TensorFlow were imported, TensorFlow is Google's proprietary machine learning framework for rapid computation. The OpenCV computer vision library is a fully accessible image processing library that's used for real-time picture capture, reading, and showing. A wide variety of Open Object Detection & Artificial Intelligence methods is offered by OpenCV (Chandan et al., 2018). To showcase and generate stationary, animated, and interactive visualisations, we use the full library of matplotlib, whereas OpenCV is focused on input, helping us showcase and analyse our data.

With the implementation of OpenCV and a laptop's 720p webcam, we capture the video input in OpenCV, where webcam is a parameter that gets passed on being initialised to 0, which means we currently are in use of one camera, so for example if we have two web cameras it would be initialised to 2. Using a loop, we collect 2,000 pictures, which is the optimal number to utilise for training a model for images at 30 frames per second. We had also passed blur images as sometimes there may be unbalanced scenarios in real-time where the objects are not visible, which aids in avoiding the issue (Figure 1).

Using the Frames Capture function, we can extract all the frames. It contains a boolean value which we initialise to 0, It helps capture the frame until true and until and unless a key is pressed the frame keeps capturing or the loop reaches a set number, i.e., in our case 2000 images for each class. This process continues for 5 classes, collecting 10,000 pictures, with a delay of 50 seconds after every class is collected.

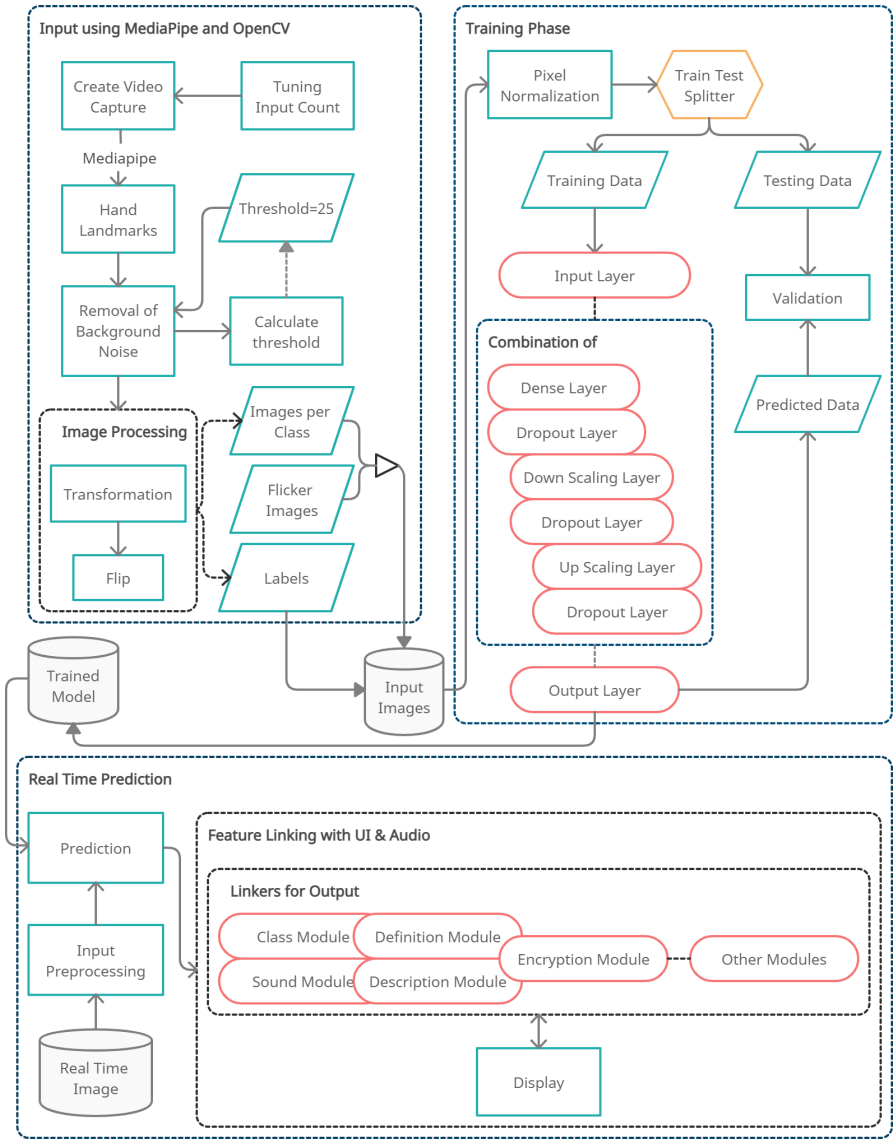
2.1 Object Tracking

Object tracking techniques and Image classification are used in tandem to locate the presence of hands and military gestures. The primary goal of image classification is to identify everything about a picture. For example, one may predict the class or kind of an item in a picture using a metric (probability, loss, accuracy, etc). Using picture input, our classifier produces class labels. While object detection is concerned with recognising and locating all elements in a scene, attention is directed on the particulars of a particular object.

Object detection defines where numerous items are located in a picture, and object tracking is always about monitoring a moving object(s) in real-time (Chandan et al., 2018). Although it doesn't have to specifically identify things since the major benefit of it is that motion characteristics can follow objects. Using the temporal connection between picture frames, object tracking is quite efficient. Locating the landmarks in each successive frame enables the process of locating the hands in the frames.

To construct the object bounding boxes, we use object detection models. The shapes of the objects in the enclosing boxes are rectangular or square, therefore the object information isn't clear. Thus, we use Haar Cascade and Mediapipe (Wang and Lee, 2007; Leibe et al., 2008), which enable us to position our hands at the optimal level for maximum visualisation. By using Deep Learning, this method reduces the use of data and prioritises the use of landmark localization techniques.

Figure 1 Steps taken for retrieving and training of images and prediction of gestures.



Notes: The diagram depicts all of the processes involved in collecting data, preprocessing images, passing them through the CNN model, and resulting in refined images with projections. All these images are predicted in real-time with additional modules linked for better output.

2.1.1 Haarcascade

The Haar Cascade classifier is an efficient object identification method. The idea here is that a cascade feature is trained using many pictures from both the positive and negative categories. The Haar feature captures how the grayscale value changes from one picture to the next. To create the feature model, the black and white rectangles are utilised. To get the Haar feature value, four angular locations of the matrix region in the integral picture are identified, and the sum of pixel values of those 4 samples will be calculated (Yustiawati, 2018).

A formula for the constituents of an integral image is as follows [equation (1)]:

$$ii(x, y) = \sum_{k \le x,\ l \le y} I(k, l) \tag{1}$$

Every pixel in the left upper corner of its source picture contributes to the total of the components in the integral image. The preceding iterative formula is used once the picture has been scanned [equations (2) and (3)]:

$$s(i, j) = s(i, j - 1) + I(i, j) \tag{2}$$

$$ii(i, j) = ii(i - 1, j) + s(i, j) \tag{3}$$

To quickly compute the Haar feature value, scan each picture row by row before moving on to the next row.

Once the object recognition training is complete, it is utilised to identify the items in the rest of the pictures. Together with OpenCV, which was imported from cv2, and matplotlib, this programme works.

2.1.2 Mediapipe

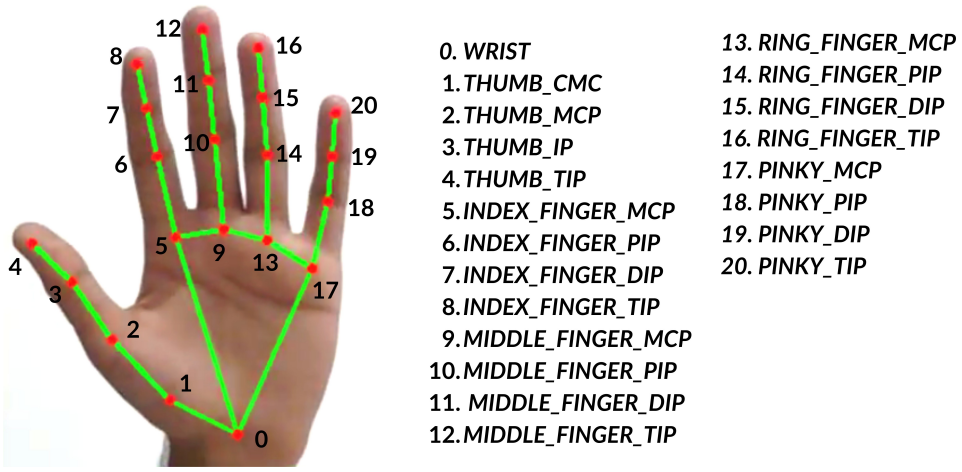
Open-source Google MediaPipe tracking technology is being used to help track the item in this instance (Lugaresi et al., 2019; Leibe et al., 2008). The most advanced and comprehensive human body recognition and tracking algorithms available on Google's most extensive and diverse dataset are all included in MediaPipe. Real-time finger monitoring with several hands is provided via the Google MediaPipe technology. As per the documentation section of mediapipe, the palm detection accuracy is typically 95.7% accurate. We only have a baseline score of 86.22% with our cross-entropy loss and no decoder.

First MediaPipe first checks to see whether the palm is still in place. If the palm is still in place, it eliminates the need for a palm detection check, thereby saving time. In order to prevent erroneous finger detection, the finger detection is performed only on the palm region that was previously detected. To identify an image of a specific hand motion, we employ OpenCV for our image detection technique, which includes taking a picture of the frame, then finding the key points in the image that indicate the particular hand move. The method that the Mediapipe library uses for recognising motions and computing outcomes is termed holistic. The components work together to help recognise a hand. We access a module from mediapipe that helps in drawing the landmarks on the image and also connect the dots when a hand is recognised (Figure 1). In the hand and palm modelling, the points are rendered using drawing.utils and then merged using holistics.

The holistic class has five parameters, all of which are listed below:

1. **Static Image Mode:** By default, it is set to False, this approach treats the input pictures as a video stream by default. It will try to detect hands in the first input images, and upon a successful detection further localizes the hand landmarks.
2. **Upper Body Only:** In holistics, we have different models like face detection, iris, pose, hands etc. This parameter is set to false by default, where we only use hands for gesture recognition. Hand and Pose models are embedded in the holistics model and have similar functionality.
3. **Smooth Landmarks:** By default, this parameter is set to true which reduces the jitter.
4. **Min Detection Confidence:** The value ranges from [0.0, 1.0]. By default, it is set to 0.5, which yields successful detection of a particular object. If the value of confidence is decreased than 0.5 it may detect any random object (Lugaresi et al., 2019), which is not the intent of our application.
5. **Min tracking confidence:** By default, the value is set to 0.5.

Figure 2 All the hand landmarks of holistic utility which are mapped through MediaPipe



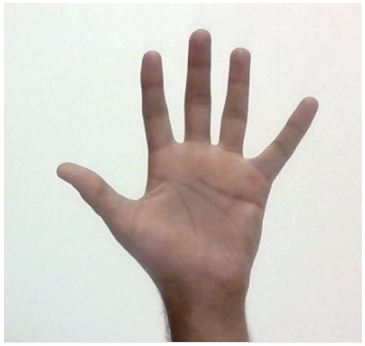
Notes: They are from dot 0 to dot 20, in all having 21 dot projections over a hand.

In order to depict various military hand motions, we require hand landmarks and hand-me-down utility using small, moving dots to help with object tracking. We utilise Right-Hand Landmarks in our issue description because the holistic model includes hand linkages. These hand landmarks consist of 21 points. For example, 0 that is Wrist, that will be connected to point 5 that is Index_Finger_MCP, 1 that is THUMB_CMC and point 17 that is Pinky_MCP (Figure 2). It depicts the same thing with regard to every other point on the diagram. While not having to fuss about colour or reflective properties, our method eliminates the complexity of the picture since the whole identification is based on the points, we detected using our camera while taking the images. We calculate 21 handshake coordinates in 3D after the hand identification has completed scanning the whole region (i.e., x, y, z-axis). Because of its excellent coverage and rigorous hand-detection, this system identifies places on maps to partially visible hands.

2.2 Preprocessing of Images

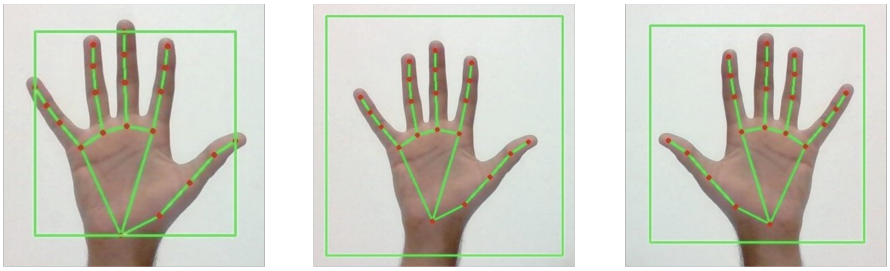
The pictures are first collected and processed using OpenCV, and then the processed results are sent on for additional processing. Both Matplot and MediaPipe utilise the RGB colour stream, while OpenCV uses the BGR colour stream (Wright et al., 2010). We do the conversion from BGR to RGB, using OpenCV's internal module. Results are directly anticipated due to OpenCV being a one-shot model. To process, we need two frames: one for collecting pictures (Figure 3) and the other for plotting the projections of the dots from MediaPipe.

Figure 3 Frame 1, before preprocessing, without any conversions, projections and changes.



Notes: This figure portrays all the images captures into the dataset as frame 1, without any conversions and projections.

Figure 4 Reduction of unnecessary space around hand in leftmost and middle figure, processed flipped version in rightmost figure.



Notes: The figure contains three images, right hand with iterations and trials that were applied over the dataset images to get a perfect optimised rectangle around the hand and a mirrored image. The leftmost figure signifies the smallest rectangle around the hand over the projections and the middle figure signifies the largest rectangle around the hand projections and the rightmost image contains the flipped/mirrored image showcasing the right-hand properly.

For the second frame, we build a rectangular frame around the hand's borders (Figure 4), reducing additional unneeded space. The hand landmarks variable is created inside the try block. It contains all the hand landmarks coordinates that we are providing it with. This is done by iterating all possible rectangles around the hand model and finding the smallest one that contains all the points while ensuring that the minimum number of points that have to be projected are achieved. Once the optimized rectangle has been found, we initialise it to 30. Cropping the picture is made easier using it. Those images are mapped using the MediaPipe, which we get via OpenCV. Some computations are necessary to trace these landmarks on the second frame.

We obtain the right-hand markers in the result after the processing of the frame. We may also reverse the image by flipping as the image captured through the webcam is mirrored. Finally, the frame is converted back to BGR for display, using OpenCV, since we just needed RGB for Matplot and Mediapipe's processing.

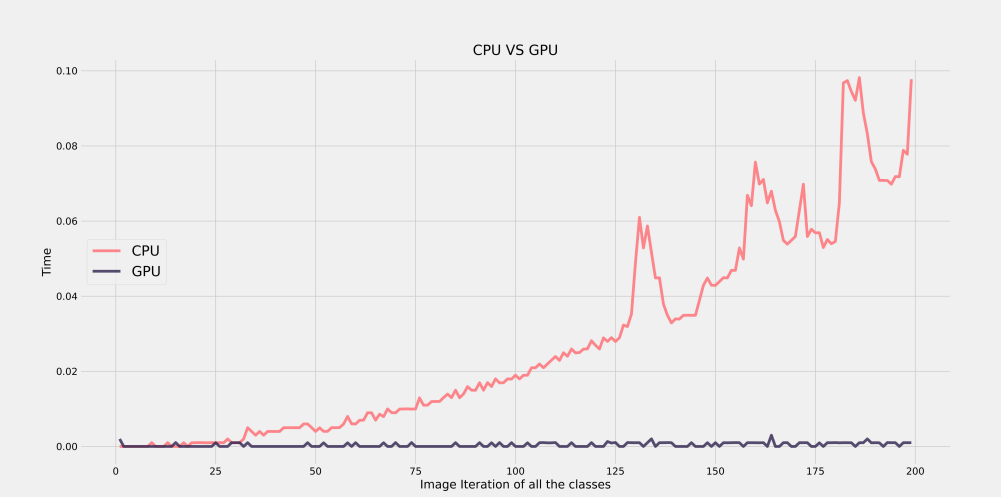
The pictures are transformed to a 96*96-pixel dimension when a gesture is detected (Figure 1), which helps to analyse the images more quickly. We attempted to take pictures of various pixels in size (Pranoto et al., 2018), i.e., 28*28 pixels, but this size is too small, and it may recognize any object in the image which we are not supposed to. With a size greater than 96 pixels i.e 128*128 the computational work is more. We provide a black background with only the hand projections (Rajnoha et al., 2018; Ruslan et al., 2018), all of this computation helps in reducing the size of the jpg and removing unwanted objects from the image. The pictures on the black screen are subsequently saved to our training and test model in a specific directory. After collecting and saving pictures, all resources utilised are released and all windows are destroyed.

3 Data Preparation

GPU is utilised for quicker processing in our calculation while not CPU. It is possible for a CPU to handle just one job at a time; GPUs split the work into many tasks, and as a result, throughput is much higher. Because of this (Cengil et al., 2017), GPUs are better than CPUs in our scenario and hence preferable over a CPU (Figure 5). The overall performance is based on GPU. The GPU which we utilize in our calculations is Nvidia GTX 1660 TI with 6 GB graphics, though the performance may vary from GPU to GPU. While Accuracy and Performance are both different terms as performance is how good the model gets trained and accuracy is the limit at which an algorithm could work (Strigl et al., 2010). We try using different GPUs of lower ranges like Nvidia's GTX 1650 Tl with 4 GB graphics but the latter performs much better with 6 GB with better performance. We use TensorFlow's version 2.5.0, but to use them for our model we installed some packages out of which the most important ones were:

- Nvidia Graphics.
- Cuda Toolkit - 11.0
- cuDNN - 8.2
- Tensorflow -2.5.0

Figure 5 Performance analysis between CPU and GPU is made with respect to time.

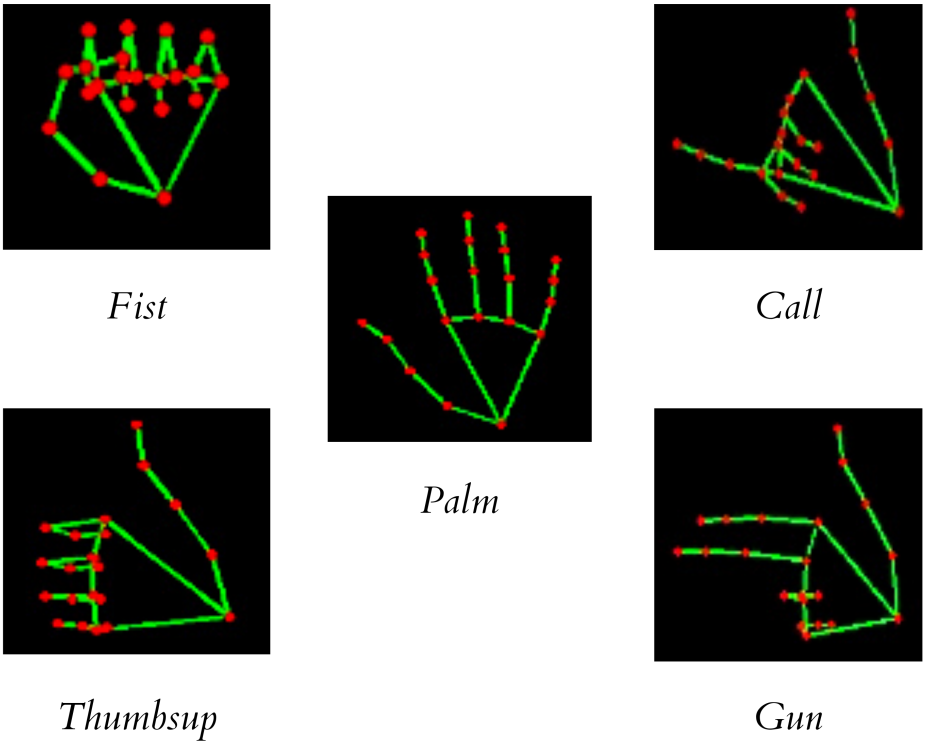


Notes: The graph above shows that GPU outperforms CPU.

As TensorFlow's version needs to match with the Cuda package's version as well, thus version 2.5.0 just yields the perfect and balanced performance together. Our final output includes five different input classes, including palm, fist, thumbs up, gun, and call. After gathering all of the picture files in JPG format, the results are stored in a specific class list, and the whole process is preserved in the data. We particularly check the shape of our data which is (5, 2000, 96, 96, 3), where 5 represents the number of input classes (Figure 6), 2000 represents the number of images in each dataset, 96x96 represent the height and width of each image (Pranoto et al., 2018), and 3 represents the RGB (Red Green Blue) streams (also known as a colour channel). The ratings have been designed in a matrix format, i.e. [a][b], which means [a] class and [b] picture number. The increase in computing speed is contributing to all that is happening.

Then we use a variable called 'X' which is a list of all the images. 'X' and 'Y' are two different list variables where 'X' contains the images and 'Y' their class indices. In order to store all of the images for all of the classes, we set up a loop that runs over all of the classes' data and images. This whole array is created, with 'X' having a shape of 10,000, 96, 96, 3, where 10,000 is the number of images, 96*96 is the dimension, 3 is the colour channel, whereas the shape of 'Y' is 10,000. So, if for example, we try to view the 3455th image in 'Y', it is the 1455th image of the second class, i.e., 'Y' = '1'. We normalize 'X' on dividing by 255, aiding in being the input for the neural network (Raju et al., 2020).

Figure 6 Resized five classes used in processing.



Notes: The figure contains all the five classes, which are resized and added to a black screen with a pixel size of 96 * 96 in an RBG channel.

4 Data Splitting

Concerning the training and testing models, the optimum scenario is for the test model size to be 20% of the whole dataset size and the training model size to be 80% of the total dataset size (Luo et al., 2018). To ensure that the pictures are randomised, we have shuffled the whole dataset with iterations so as to learn all the dataset classes in a generalized order. We may choose any arbitrary state, thus it will generate randomness, which we set to 2 and the shuffle is set to true (Figure 1). If it is set to false then, images are not shuffled and, in the epoch, it will train images according to paper only.

5 Training Stage

Neural networks have a variety of architectural techniques. The different architectures have differing levels of complexity, varying numbers of layers, and varying numbers of neurons in each layer. In this part, we utilise the most efficient design also explaining two architectures.

5.1 Artificial Neural Network

One of the most popular machine learning approaches is ANN, which is capable of finding correlations. ANN is based on the neural system of the cerebral cortex and in that sense, it is meant to mimic the functioning of the human brain. Most of the ANN's methods utilise weights and an activation function. ANN uses these weights because they're an information resource. Each layer of ANN has three levels: input, hidden, and output. Inputs are taken in by the input layer, hidden layers process the inputs, and the output layer outputs the result. To be more specific, each layer is concerned with learning weights. However, ANN faces several problems such as insignificant human interference, challenging computational scalability, and very poor learning speed.

ANN provides the output based on picture characteristics such as colour pixels, rather than featuring the whole of the image. The ANN model cannot precisely anticipate objects if they are far away and showcases no such object to be predicted, even though it has the same features. For example, in a face detection model, to detect a face for the model its training gets based on colour pixels and not the features, thus if the picture is far away ANN has a hard time predicting and returns no object being present in the image (Chauhan et al., 2018), even though it has some trainable features. An increase in the size of the picture leads to an exponential rise in the number of trainable parameters. Images without location information are lost. In image processing, spatial characteristics pertain to the positioning of the pixels in a picture. In order to cope with sequential data, ANN is unable to gather information sequentially in the input data. In this situation, the ANN algorithm fails. A convolutional neural network model was developed in order to solve this challenge, which included the use of a neural network model, along with convolution.

5.2 Convolutional Neural Network

CNN is one of the most known deep learning models. The ability of Convolutional Neural Networks (ConvNets) to isolate certain patterns in the data is incredible. This has helped them become quite proficient at identifying patterns, and then they utilise patterns to categorise pictures. Using this technique identifies the hand, use them to classify and identify hand movements. CNN is a neural network that learns by itself to become more efficient. In the design of CNNs, there are three types of layers: a subsampling layer, a max-pooling layer, and a fully related layer (Alzubaidi et al., 2021). Stacked regularly throughout CNN is the small number of levels mentioned above. This CNN sequential model has many levels of building blocks.

- **Convolutional Layer:** The first layer is a convolutional layer, comprising several parameters that do computational work. A filter that can be trained is applied to the input picture, convolved, and utilised as a featured image on every structure known feature map. After that, a bias is added to the result, and the final feature map is named (Figure 1).

  These are the parameters to be used:

  1. **Filters:** It's designed to make images seem more natural by identifying patterns in the picture and is set to 32.
  2. **Kernel size:** Kernel is the matrix that moves over the data and performs computation, the kernel size is specified which is (3,3)

3. **Strides:** The stride is the number of pixels shifted over the image. The number of pixels shifted is "1" for both the axes, i.e, x & y, as specified and set to (1,1).

4. **Padding:** Padding has two different types: "valid" and "same". Only the legitimate portion of the padding is kept in “valid” padding; while pictures are padded with zeros in “same” padding to fit. To make sure the output layer is of that size as the input neurons, the padding here is set to “same”.

5. **Activation:** Three activation functions, ReLU, Sigmoid, and Tanh, stimulate the neurons (Phat and Trinh, 2010). The activation function employed throughout this model is the Rectified Linear Unit (ReLU). ReLU is R(x) = Max (0, x) wherein x is just the inputs to a neuron. Sparsity inactivation, improved gradient transmission, and efficient computing are some of the benefits of using ReLU. In contrast to the other two ReLU functions, this one adds nonlinearity into the model, therefore performing better.

- **Max-Pooling Layer:** Even when the input pictures are extremely large, the pooling layer can reduce the number of parameters, thereby minimising the pooling layer’s complexity. It is configured to choose the biggest feature vector from the feature vector. The pool size here is (2,2). At the end levels of the model, the pool size is lowered to avoid reducing the spatial dimensions too fast (Lin et al., 2020).

- **Dropout layer:** One of the most essential layers to prevent overfitting, the dropout method is used to ensure that each layer only sees the feature once and learns a new collection of neurons. The percentage of the total that typically falls between 0 and 1 is 55%, and we have tested several values ranging from 0.1 to 0.4 (Nitish, 2014). When we obtained the generalised model, the most optimal fraction for dropping the percentage of neurons in a particular layer is 20% (Table 1).

Table 1 Comparison of Average scores and F1 Scores on 5 classes used in dataset over various parameters like Anti Segmentation, Epochs, Image Size and Dropout Percentage.

Scenario		*Average Score*		*F1 Score*				
		Training	*Testing*	*Class 1*	*Class 2*	*Class 3*	*Class 4*	*Class 5*
Anti-Segmentation	*Black*	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
	White	*0.98*	*0.98*	*1*	*0.98*	*0.98*	*0.98*	*0.98*
Image size	*28 x 28*	*0.9804*	*0.9804*	*1*	*0.98*	*0.98*	*0.99*	*0.98*
	32 x 32	*0.98*	*0.98*	*1*	*0.98*	*0.98*	*0.98*	*0.98*
	64 x 64	*0.99*	*0.99*	*1*	*1*	*0.99*	*0.98*	*0.98*
	96 x 96	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
	128 x 128	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
	320 x 320	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
Epochs	*5*	*0.99*	*0.94*	*0.98*	*1*	*1*	*0.96*	*0.99*
	10	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
	15	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
	20	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
	25	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
Dropout percentage	*0.1*	*0.99*	*0.98*	*1*	*0.99*	*0.99*	*0.99*	*0.99*
	0.2	*1*	*1*	*1*	*1*	*0.99*	*0.99*	*0.99*
	0.45	*1*	*1*	*1*	*0.98*	*0.98*	*0.99*	*0.98*

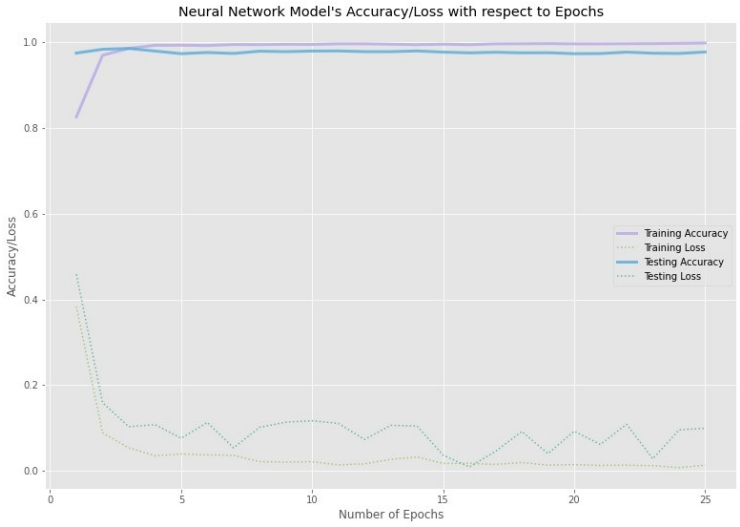
To increase accuracy, we utilize two convolutional layers that influence the size of data. The flatten output layer follows and the previous layers' outputs are combined into a single layer before being used in the output. We also include a hidden layer and a dropout layer, both of which aid in producing a more generic model. Another layer, the Output layer, receives the parameters provided and converts them to units. We use the ReLU activation layer only for the hidden layers, and for the output layer, we are using the SoftMax function because it gives the probability for classifying the image in different classes.

Once the model is built, we have to compile it with the optimization settings, loss function, and measurements. Adam optimiser has been used here to increase the efficiency of CNN visuals. Sparse categorical cross entropy determines the difference between the actual and anticipated ones. After determining the weights, the weights are raised and reduced as needed. The metrics are set to the accuracy metrics.

When we build the model, we set the epoch count to 10, since it enables faster and more accurate training. If you set it to a lower epoch, it reduces the amount of crunching. Computation depends on the epoch where if we set it less it decreases the crunching (Arif et al., 2018). We tried increasing the epochs to 15 and 25, while the model was not being generalized due to less data it performed with less accuracy. The total time is taken for each epoch and to train the model was just 55 seconds, the results were due to higher computations and faster GPU. The result for the testing model accuracy we achieved was 100%. With the classification report obtaining f1 scores, which generates a single score which accounts for both precision and recall, for two classes at 100% and the remaining three classes at 99%, being exceptional. Checking the real-time accuracy at 30 fps, we add a prediction function that displays the image's gesture and its class.

Our Neural network model, as opposed to the traditional ANN, was built using CNN as a primary model because of the smaller number of units in the network, resulting in fewer

Figure 7 Accuracy/Loss vs Number of Epochs over Training Accuracy, Training Loss, Testing Accuracy and Testing Loss.



Notes: In this figure, took the model and fed data to the neural network architecture where it does forward and backward propagation for the epoch. We can see how the model reduces its losses and gains accuracy over the testing and training set.

parameters to train and less risk of overfitting. To obtain a better prediction in image data, this characteristic is very critical. Images operate in tandem with CNN to provide the input data. Feature maps are created by using filters on imagery results. As is the case with CNN, rather than processing data in a forward-facing manner, maps created by CNN relate to the same data numerous times.

Converting 3-channeled RGB images to 1-dimensional pixel coded vectors makes image classification tasks more challenging (Vani et al., 2019; Alzubaidi et al., 2021; Lin et al., 2020). This causes the number of trainable parameters to exponentially grow, while simultaneously increasing the capacity to train trainable parameters. The result is that the capacity for training trainable parameters greatly improves, while simultaneously boosting the number of trainable parameters. More generally, CNNs tend to do better in classifying issues.

6 Linker Stage

AI is used in a variety of approaches in defence technologies, including hand, face, and voice recognition. Using OpenCV, Haar Cascade Classifiers, Matplotlib, CNN, Google's Mediapipe, and other ways that aid to acquire the desired results through patterns, our issue statement works with hand recognition to help improve technology, equipment, and electronics (Suresh and Niranjanamurthy, 2021; Alzubaidi et al., 2021; Leibe et al., 2008). As a result, we bind those results in order for them to be useful in real-life scenarios. To make the best use of our recognition technique, we use a library called gTTs to connect recognised gestures to audio describing the gesture and their action to take in response about the gesture, which is played in their ears, making it easier for soldiers on the battlefield to communicate over long distances using only gestures.

gTTs stands for Google Text to Speech, and it's used in Python to convert text to speech. It supports a variety of languages, including English, French, German, Tamil, and Hindi, with more languages being added as time goes on. We tried implementing different libraries, such as BytesIO from IO, but those caused issues with the module frequencies. However, because of this assistance, gTTs stood out and proved to be ideal for our needs. This is time-controlled, meaning it plays for a set amount of time before changing when the next gesture is prompted. It supports both server-based and client-based/embedded recognition and synthesis, which is the motto of our problem statement, which could change dynamically as per the user's settings. This library is extensively used in our application and gives the commander a choice to not just see the gesture displayed and action to be taken on the headgear, but also have the gesture described through the earpiece soldier wears to communicate.

In our problem statement, we took 5 classes, each denoting a gesture. Palm, Fist, Thumbs-Up, Gun, Call each of them have a unique description and a unique situation to be used at (Figure 6). We added a description frame using OpenCV to be displayed, which gave us a brief description of each particular class. We add a definition feature frame as well, which tells a short description about the gesture, to explain the gesture more and let the soldier know the task to be performed on the screen of the headgear as well as through the earpiece, in just a few milliseconds which makes it easier for the soldier to adapt without any wastage of time and attention staying on the battleground. For example, in a situation a commander with their team is present on a battleground, against an opposing military troop, moving slowly, without any notice. All of them in a line following each

other's footsteps with "be on point" positioned soldier leading them, if the commander needs to give the foremost soldier a command where only the gestures are viable and the commander is not visible to the foremost soldier, then this gesture recognition method is utilized with the commander portraying their gesture, that is for example to "hold the fire", that gets recognised in one-shot and real-time, also gets transmitted to the embedded recognition system present in their headsets, which could be passed on to the specific soldier or the whole troop as well. The gesture is transmitted to the leading soldier with a brief description to continue his movement holding the fire, and also a definition of the gesture played in the earpiece making sure the eye stays on the ground with the alertness created having the message received (Figure 1).

With the CNN model implementing high accuracy, it will make it easier for soldiers and troops to not worry about the distances making sure the messages are transmitted within no time. Our model novels in utilising a unique approach to transmit the messages in long-distance, by being implemented in headsets which could transmit the message encrypted from each nearby soldier's headset creating a chain and reaching to the discrete commander. The result is a chain, which connects every soldier in a given battle location (regardless of their proximity to one another).

These sound modules are timed for every action assigned to the gesture and created and passed through parameters using loop.gtts. It makes sure the gestures which get detected every second repeatedly, aren't falsely sent to the soldiers disturbing the routine and confusing them for the action to be performed. All these raw testing's have been done on separate windows showing a description of gesture, the definition of the gesture, sound modules and the gesture recognised using OpenCV. The real-time accuracy of the CNN Model has been exceptional predicting with a minimum of 99% accuracy for the model.

7 Experimental Results

This study reviews Deep Learning classification research results which included the use of a relu activation for the CNN model. The dataset containing images were mapped utilizing the MediaPipe and Haar Cascade, obtained from OpenCV and Matplot and also switched from BGR to RGB. The collection of 12,000 pre-processed photos includes 5 distinct hand movements which were vividly used in military operations to communicate. By using live hand tracking with dynamic zooming in or out, whenever the hand moved front or back, our model understood and all the images were resized to set dimensions, which was then recorded and stored, we were able to arrive at an optimal outcome. This all assisted in enhancing the ability to accurately identify the hand. This specific round of ideation focused only on the images displayed utilising mediapipe and the previously black-framed picture being scaled and stored.

Using Mediapipe's state-of-the-art feature extraction system, we don't have to design a convolutional neural network from the beginning, reducing the time it takes to acquire and use features. All CNNs were trained using a large number of layers, with numerous parameters: Filters = 32, Kernel size = (3,3), Strides = (1,1), Padding = 'same' and Activation = ReLU. Using the most minimal computing resources, this suggested approach takes less time to train the model than existing state-of-the-art methods. This approach assisted in recognising the hand, thanks to the addition of flicker and fuzzy pictures of the hand. We took 5 classes that delivered the messages to the soldier in 5 specialized modules which were the hand gesture's name, its definition, task attached to the gesture which is called description, also the description said over through audio in real-time and the at last adding

dynamic encryption to the module. At 30 frames per second, our outcomes were optimum; this was owing to our highly trained model, given the results achieved were not expected at a minimum of 99% for each class accomplishing far above state-of-the-art performance and defining hand motions accurately. Maximum f1 scores of 1.00 are achieved for 2 classes and the rest three having 0.99. With the precision and recall of 1.00 for 3 classes and 0.99 for the other 2 classes (Table 1 and Figure 7). We also had a time-controlled Text to Speech Converter, ie, gTTs, supporting in describing the gesture in real-time with the highest accuracy and minimalistic delay.

8 Conclusion

Our suggested approach shows that MediaPipe may be used as an effective tool to identify complicated hand gestures accurately, with an accuracy of 99 percent across the Military Hand Gestures utilising MediaPipe's technology and analytics. The training timerequired for a model is minimal. As the amount of processing power and adaptability to smart devices cuts down on the costs, the approach is more durable and cost-effective. One thing that is not often discussed, however, is the possibility of training and testing with diverse hand gesture recognition datasets. Doing so demonstrates that this framework can be implemented successfully for any hand sign language dataset, with the highest accuracy possible. Its faster real-time detection allows the model to show its efficiency better than other methods now available. This study still has to go through a number of more iterations before it becomes something people can use, but at least it opens the door to a new and more accessible method to detect hand movements.

In conclusion, our proposed model is well tuned to be utilised for gesture detection as well as real-time picture acquisition, and this model also works with higher frames per second. This research can be advanced in a number of different ways, and the AI and database that were utilised in this study can be optimised for better results. One of the biggest difficulties we may encounter is the additional motions which are included into more gestures, which might be helpful to the issue in real-time.

9 Future Scope

Some distinguishing characteristics may be employed in future studies so that comparable motions may be identified with low failure rates. Attention-based Classification algorithms, in which particular areas of gestures are given greater importance for accurate prediction. The technology in the future would contemplate employing a superior camera and working with a LiDAR sensor or a 3D camera to better comprehend a hand position. In scanning and comprehending the whole categorization we aim to employ the entire hand. We aim to research night mode, in which distance characteristics from the picture taken may be understood so that illumination and the grouped background problems in the gesture detection process are not confined to the system.

However, with much more efficient categories demanding a greater number of movements to detect motions with a minimum error rate, classification methods still need to be enhanced. The most significant item that may help us implement it is the military equipment specifics, which would enhance the operational capability of the proposal. There might be a wide variety of military demands in terms of hazards, quality management,

governmental limits, etc. and certain types of openness have not yet become available. In addition, a certified sign language interpreter isn't yet developed for the datasets on this project. Also, fewer studies on gesture recognition for the defence using AI are researched and more improvements are needed in memory usage utilizing MediaPipe & OpenCV. Artificial Neural Networks provide plenty of opportunities to develop things for the first time in programs to obtain 100% precision with less training data set.

References

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. (2021). 'Review of deep learning: concepts, CNN architectures, challenges, applications, future directions'. *Journal of Big Data*, Vol 8, No 1, pp. 53–53.

Ari, N. and Ustazhanov, M. (2014). 'Matplotlib in python'. *11th International Conference on Electronics, Computer and Computation (ICECCO)*, pages 1–6.

Arif, R. B., Siddique, M. A. B., Khan, M. M. R., and Oishe, M. R. (2018). 'Study and Observation of the Variations of Accuracies for Handwritten Digits Recognition with Various Hidden Layers and Epochs using Convolutional Neural Network'. *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*, pages 112–117.

Cengil, E., Çinar, A., and Güler, Z. (2017). 'A GPU-based convolutional neural network approach for image classification'. *2017 International Artificial Intelligence and Data Processing Symposium*, pages 1–6.

Chandan, G., Jain, A., Jain, H., and Mohana (2018). 'Real Time Object Detection and Tracking Using Deep Learning and OpenCV'. *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 1305–1308.

Chauhan, N., Bhatt, A. K., Dwivedi, R. K., and Belwal, R. (2018). 'Accuracy Testing of Data Classification using Tensor Flow a Python Framework in ANN Designing'. *2018 International Conference on System Modeling & Advancement in Research Trends (SMART)*, pages 44–48.

Kang, S. and Tversky, B. (2016). 'From hands to minds: Gestures promote understanding'. *Cognitive Research: Principles and Implications*, Vol 1, No 1, pp. 1–15.

Leibe, B., Schindler, K., Cornelis, N., and Gool, L. V. (2008). 'Coupled Object Detection and Tracking from Static Cameras and Moving Vehicles'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 30, No 10, pp. 1683–1698.

Lin, J., Ma, L., and Cui, J. (2020). 'A Frequency-Domain Convolutional Neural Network Architecture Based on the Frequency-Domain Randomized Offset Rectified Linear Unit and Frequency-Domain Chunk Max Pooling Method'. *IEEE Access*, Vol 8, pp. 98126–98155.

Lugaresi, C., Tang, J., Nash, H., Mcclanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C. L., Yong, M. G., Lee, J., and Chang, W. T. (2019). *Mediapipe: A framework for building perception pipelines*.

Luo, X., Li, L., Wang, J., He, D., Li, J., and Zhou (2018). 'How Does the Data set Affect CNN-based Image Classification Performance?'. *5th International Conference on Systems and Informatics (ICSAI)*, pages 361–366.

Nitish, S. (2014). 'Dropout: a simple way to prevent neural networks from overfitting'. *Journal of machine learning research*, Vol 15, No 1, pp. 1929–1958.

Phat, V. N. and Trinh, H. (2010). 'Exponential Stabilization of Neural Networks With Various Activation Functions and Mixed Time-Varying Delays'. *IEEE Transactions on Neural Networks*, Vol 21, No 7, pp. 1180–1184.

Pranoto, H., Budiharto, W., Warnars, H. L. H. S., Matsuo, T., and Heryadi, Y. (2018). 'Image Size, Color Depth, Age variant on Convolution Neural Network'. *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, pages 39–45.

Rajnoha, R., Burget, L., and Povoda (2018). 'Image Background Noise Impact on Convolutional Neural Network Training'. *10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–4.

Raju, N. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., and Padma, V. (2020). 'Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification'. *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 729–735.

Ranawat, M., Rajadhyaksha, N., Lakhani, R., and Shankarmani (2021). 'Hand Gesture Recognition Based Virtual Mouse Events'. *2021 2nd International Conference for Emerging Technology (INCET)*, pages 1–4.

Ruslan, R., Ibrahim, M. F., Khairunniza-Bejo, S., Aznan, A. A., Rukunudin, I. H., and Azizan, F. A. (2018). 'Effect of Background Color on Rice Seed Image Segmentation Using Machine Vision'. *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)*, pages 1–4.

Strigl, K., Kofler, S., and Podlipnig (2010). 'Performance and Scalability of GPU-Based Convolutional Neural Networks'. *18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 317–324.

Suresh, H. S. and Niranjanamurthy, M. (2021). 'Image Processing Using OpenCV Technique for Real World Data'. *Intelligent Computing Paradigm and Cutting-edge Technologies. ICICCT 2020. Learning and Analytics in Intelligent Systems*, Vol 21, pp.285-296.

Vani, S., Rao, T. V., and Naidu, C. K. (2019). 'Comparative Analysis on variants of Neural Networks: An Experimental Study'. *5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 429–434.

Wang, S. and Lee, H. (2007). 'A Cascade Framework for a Real-Time Statistical Plate Recognition System'. *IEEE Transactions on Information Forensics and Security*, Vol 2, No 2, pp. 267–282.

Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T. S., and Yan, S. (2010). 'Sparse Representation for Computer Vision and Pattern Recognition'. *Proceedings of the IEEE.*

Yustiawati, R. (2018). 'Analyzing Of Different Features Using Haar Cascade Classifier'. *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pages 129–134.

Zhang, Y., Dai, Z., Zhang, L., Wang, Z., Chen, L., and Zhou, Y. (2020). 'Application of Artificial Intelligence in Military: From Projects View'. *2020 6th International Conference on Big Data and Information Analytics (BigDIA)*, pages 113–116.