DE-42 (EE) KARIM HUSSAIN, FARZAM TAIYAL ZULFIQAR, MUHAMMAD HAROON, MEER MUZAMMEL KHAN

REAL TIME HAND GESTURE CONTROLLED HOLOGRAPHIC DISPLAY

COLLEGE OF

ELECTRICAL AND MECHANICAL ENGINEERING NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY RAWALPINDI

2024

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

DE-42 EE

PROJECT REPORT

REAL TIME HAND GESTURE CONTROLLED HOLOGRAPHIC DISPLAY

Submitted to the Department of Electrical Engineering

in partial fulfilment of the requirements

for the degree of

Bachelor of Engineering

in

Electrical

2024

Project Supervisor:

Dr. QASIM UMAR KHAN

Submitted By:

KARIM HUSSAIN

FARZAM TAIYAL ZULFIQAR

MEER MUZAMMEL KHAN

MUHAMMAD HAROON

ACKNOWLEDGEMENTS

We are thankful to **Allah Almighty**, the Most Merciful, for the courage, and the guidance He has given us, which have made it possible for us to complete this project.

We would like to express our gratitude to our project supervisor “Dr. Qasim Umar Khan” and all the faculty for their guidance and support throughout this journey. We are grateful for their expertise, constructive feedback and encouragement that has guided us to refine our approaches and ideas throughout the course of the project. We are also grateful to our fellow colleagues for their consistent encouragement has been a potential motivation to accomplish this task. Lastly, we would also like to thank our families for their support and belief that made this journey an enjoyable experience.

ABSTRACT

This project aims to develop an advanced human-computer-interacting system designed to seamlessly control holographic 3D visuals using real-time dynamic hand gestures. The primary goal is to enhance the learning experience in the educational sector by effectively displaying and controlling complex 3D concepts such as models of human anatomy and various engineering problems. This innovative approach leads to improved academic performances and enhances the engagement and participation of students. The core of the system's functionality lies in detecting hand gestures and translating them into real-time computer cursor control, thereby enabling interactive control of holographic displays.

To achieve accurate hand gesture detection, two distinct approaches are employed: a vision-based approach and a sensor-based approach. In the vision-based approach, a camera continuously captures hand movements. This visual data is then used to detect and locate different gestures of the hand by employing a learning-based gesture-detecting algorithm. Conversely, the sensor-based approach involves the design of hardware (a sensor-based motion tracking glove) to detect the position of fingers. An ANN model, capable of detecting the same hand gestures as vision-based approach using sensor's data, is developed. The primary purpose of utilising these two different approaches is to enhance gesture recognition accuracy. Therefore, the outputs from both methods are fused together using stacked ensemble learning, with an additional ANN serving as a meta-model to further improve accuracy. These detected gestures are then translated into mouse commands. Furthermore, to facilitate user interaction, a desktop application is designed to initialise and manage the gesture-controlled mouse system. Additionally, another Windows application is developed to upload and control various 3D models displayed on a holographic interface.

SUSTAINABLE DEVELOPMENT GOALS

This project targets two Sustainable Development Goals (SGDs).

SDG 4:

This project supports the achievement of Sustainable Development Goal (SDG) No. 4, which focuses on ensuring access to quality education. This project aims to create a Human-Computer Interaction (HCI) system that uses hand gestures instead of typical input devices such as keyboards and mice. Furthermore, this project uses a holographic device to display 3D models that are interactable with hand gestures, creating an accessible and user-friendly interface that can be very beneficial, especially in the education sector. Teachers and students can use their hands to interact with the educational content displayed in 3D space, making teaching, and learning more engaging and immersive. This project also accommodates individuals suffering from disabilities who may find it difficult to use input devices, ensuring lifelong learning opportunities for all.

SDG 9:

This initiative is in complete accordance with Sustainable Development Goal (SDG) No. 9, which focuses on industry, innovation, and infrastructure. A hand gesture-based Human-Computer Interaction (HCI) system is a cutting-edge technology that can be applied in diverse areas such as healthcare, commercial advertising, automotive, and consumer electronics to enhance industrial infrastructure. This advanced system reduces the dependency on traditional input devices for HCI, making interaction with digital devices more seamless and immersive than ever before. This fosters a culture of innovation, as most modern industries, especially the consumer electronics industry, will be eager to use such innovative approaches in their products and services. Furthermore, by promoting the use of modern technology, this project supports industrial sustainability and opens the door for the development of technologically advanced systems that can meet the needs of an evolving society.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

Greek Letters

Ω resistance

Acronyms

HCI Human Computer Interaction

GUI Graphical User Interface

MEMS Micro Electronic Mechanical Systems

HOG Histogram of Gradients

SIFT Scale Invariant Feature Transform

HWT Haar Wavelet Transform

LBP Local Binary Pattern

PCA Principal Component Analysis

SVM Support Vector Machine

CNN Convolutional Neural Network

ANN Artificial Neural Network

RNN Recurrent Neural Network

k-NN k-Nearest Neighbours

IMU Inertial Measurement Unit

YOLO You Only Look Once

FLOPs Floating Point Operations

IDE Integrated Development Environment

CSV Comma-Separated Values

mAP50 Mean Average Precision 50

mAP50-95 Mean Average Precision 50-95

UI User Interface

Chapter 1 – INTRODUCTION

Human-Computer Interaction (HCI) is the way we communicate with computers. In our modernised world, we are using digital gadgets like smartphones, laptops, TVs, tablets, gaming consoles, and Virtual Reality (VR) headsets, and without these, our lives are incomplete. Human-Computer Interaction (HCI) serves as a crucial connection between users and electronic devices, playing a significant part in our daily interactions with computers and other digital devices. Moreover, with further advancements in computer technologies, HCI is becoming more and more important, and researchers are putting in more effort to enhance HCI, creating immaculate interfaces that enhance user experiences and productivity.

The evolution of HCI goes back to the 1940s and 1950s. In those years, computers were big, complex machines mainly used only for performing complex calculations by scientists and engineers, and interaction was through punch cards, switches, and dials. In the 1960s and 1970s, command-line interfaces were introduced. This interface allowed the user to use command lines to interact with the computer. Although this did improve HCI to some extent, it is still very complex for laymen's users. In the 1980s, the first Graphical User Interface (GUI) was introduced, which revolutionised HCI forever. The GUI provided the users with icons and folders to interact with. A pointer device like the mouse was introduced, which allowed the user to control a pointer on the GUI and interact in a very user-friendly environment. In the 2000s, touchscreen interfaces were introduced in mobile devices and laptops, thus eliminating the need for input devices like the keyboard and mouse. These touch screens use touch gestures like tapping, swiping, pinching, etc., making HCI simpler than ever. Currently, there is a significant focus on utilizing voice commands and hand gestures to interface with computers and other digital devices.

This project focuses on the use of hand gestures for HCI. Different hand gestures are used to control the display screen of the computer. HCI through hand gestures offers a lot of advantages when compared to traditional HCI using input devices like a keyboard and mouse. For example, this kind of HCI provides enhanced teaching and learning experiences in the education sector. Teachers and students can use hand gestures to interact with educational content, making

learning a very engaging and fun experience. Similarly, in the medical sector, surgeons can maintain sterility during surgery by manipulating medical images just by using their hand gestures and not touching anything. Hand gesture-controlled HCI is also useful for people who suffer from mobility disabilities and find it difficult to use input devices.

Over the past several years, research into HCI through hand gestures has seen significant advantages. [1] proposed a virtual mouse that can be moved in the air by using a 5-DOF IMU (Inertial Measurement Unit) and a Kalman filter to fuse the gyroscope and accelerometer data. [2] surveyed mouse implementation through hand gestures and discussed techniques like the Hidden Markov machine learning model (HMM), the convex hull algorithm based on hand contour detection, and MEMS accelerometer-based gesture recognition. [3] implemented a hand gesture-based mouse system using camera and image processing techniques like Otsu's adaptive thresholding and morphological operations. More recently, [4] developed a virtual mouse system based on fingertip detection using coloured caps on fingertips and hand gesture recognition. The system uses image processing techniques and a convex hull algorithm for gesture recognition. Various combinations of coloured caps and hand gestures are then translated to perform various mouse functions. [5] used the camera and MediaPipe libraries for hand pose estimation and the Pyautogui Python library to implement cursor control.

This project implements dynamic hand gesture recognition using two approaches. The first approach employs a vision-based strategy that utilizes a deep learning algorithm to identify several hand categories and key points. The second method employs hand flex sensors positioned on the glove and an Artificial Neural Network (ANN) to identify identical hand gestures. An approach of ensemble machine learning is used to combine the decisions made by the two algorithms, with the goal of improving the accuracy of dynamic gesture identification. The recognised gestures are then translated to perform different mouse functions like cursor movement, scrolling, right-click, double-click, and left-click. A Windows-based application is developed that is used to initialize and startup the system. Lastly, another Windows-based application is developed that is used to display different 3D models on a holographic display. The hand gesture-controlled mouse is then used to interact with these 3D models.

This project will contribute to the research work on hand gesture-based HCI as it provides a natural, immersive, and practical way of communicating with a computer that will enhance the user experience and productivity.

Chapter 2 – LITERATURE REVIEW

2.1. Related Work

Hand gesture recognition has been a significant subject of research in the field of human-computer interaction during the last few decades. In the literature, there have been two main ways of recognising hand gestures, vision-based technique, and sensor glove-based technique. The related work has been arranged in the following order:

2.1.1. Hand Gesture Recognition Using Vision-Based Technique

Hand gestures can be further classified into two categories: static and dynamic. Image processing is being used for pre-processing in countless research papers before applying machine learning models such as HMM, SVM to detect gestures. [6] implemented two image processing techniques namely, (HOG) and (SIFT) algorithms in MATLAB to generate feature vectors of images. The features extracted were passed to (SVM) to classify gestures. [7] introduced a novel technique for accurately detecting hand gestures. Their procedure includes the image segmentation through canny edge detector and employing Oriented Fast and Rotated Brief (ORB) for feature extraction, generating vocabulary codewords using k-means clustering Bag of Words model, and finally passing the feature vector of codewords to various classifiers. [8] introduced a technique for feature fusion to recognize the gestures robustly by fusing features from HWT and LBP and by doing some modifications to the original AlexNet architecture by adding new layers that can learn new patterns and fused features. [9] designed dynamic hand gestures trajectory and remodelling model based on Hidden Markov Model (HMM) algorithm by first detecting hand using Adaboost algorithm and then tracking the hand using a contour-based hand tracker. Invariant curve moments represent global feature whereas local features are represented by orientation to calculate the trajectory of hand gesture.

[10] introduced a system for recognising hand gestures in a continuous manner. The initial step involves tracking and extracting the movement of the hand from the image. Subsequently, the Fourier descriptor (FD) algorithm is utilised to extract spatial characteristics, while motion analysis is employed to identify temporal characteristics. The spatial and temporal characteristics are integrated and utilised as a feature vector for real-time dynamic hand gesture identification in Hidden Markov Models (HMMs).

[11] developed a Multi Class Support Vector Machine and utilised the Biorthogonal Wavelet Transform approach to build the input feature vector for classifying hand movements. [12] created a live American Sign Language Recognition system by employing (PCA) and SVM. The hand motion is separated from the image by extracting the ROI using the HSV color space. The ROI is then processed using PCA to extract 8 features, which are subsequently used for real-time gesture detection by the SVM algorithm. [13] developed a novel feature extraction approach by integrating the K curvature and convex hull algorithms. The newly developed "K convex hull" extractor demonstrates high precision in detecting fingertips. The gestures were categorised by inputting the features into an Artificial Neural Network (ANN).

Numerous researches are underway develop the modern hand gesture recognition system employing deep learning techniques since the rise of deep learning models after AlexNet [14] won the 2012 ImageNet Large Scale Visual Recognition Challenge. ImageNet _[15] is a large-scale image database. Hand gesture classification systems based on preprocessing and feature extraction suffer with poor accuracy and poor robustness particularly in complex backgrounds and varying lighting conditions _[16]. Additionally, some useful information in the image may get discarded when performing feature extraction _[17]. Use of (CNN) to detect visual hand gestures was explored by [18]. The CNN architecture was trained on images of ten static hand gestures with complex background and different hand sizes. Without data augmentation and skin segmentation, the CNN model achieved 82.5% accuracy on NUS II dataset. The 90.5% accuracy was achieved by introducing the dropout in the model and skin segmentation further improved it to 93.5%. The accuracy of gesture recognition further elevated to 96.5% after using data augmentation techniques.

The authors of [19] suggested a convolutional neural network (CNN) consisting of four convolutional layers, each followed by a max pooling layer. The network also includes a fully connected layer and a softmax function at the end. A novel dataset of Irish Sign Language (ISL) was created by gathering footage of individuals demonstrating ISL hand gestures. The frames were retrieved from the videos and converted to grayscale. The background was then eliminated from each frame using a pixel value threshold.

Afterwards, two gesture recognition techniques were trained and tested on the dataset. One approach utilized (PCA) for the extraction of features and k-nearest neighbour for classification purpose while second approach implemented a CNN model for gesture recognition. The k-NN and PCA approach recorded 95% accuracy while CNN-based approach recorded 99% accuracy on the ISL dataset. _[20] designed a robotic arm, controlled by gestures of hand to collect different objects. . To recognise hand gestures, they proposed a CNN architecture (DAG-CNN). For training, a dataset containing 10 different hand gestures was constructed and the model recorded an accuracy of 84.5%. The authors of _[21] introduced the hand gesture recognition approach by using the fully connected layers, FC6 and FC7, of pretrained AlexNet to extract deep features. The deep features were reduced by using (PCA) based dimension reduction technique. (SVM) was used as classifier to classify different static hand gestures. An American Sign Language (ASL) dataset was crafted using five subjects on 36 different hand gestures. The dataset contained variations in background and illumination conditions. The proposed system achieved 87.83% accuracy on 36 hand gestures.

A simple CNN architecture was used to detect Indian Sign Language gestures by the authors of _[22]. The dataset contained upto 35000 images of about 100 distinct types of Indian Sign Language gestures. The proposed CNN model recorded 98.80% validation accuracy on the dataset while using Adam as optimizer and 98.56% validation accuracy while using Stochastic Gradient Descent (SGD) as optimizer. A 2D-CNN and feature fusion based dynamic hand gesture classification system has been proposed by authors of _[23]. The authors proposed a new clustering algorithm for frames extraction and improved the existing HS optical flow algorithm by incorporating fractional order method. The original frames were passed to Spatial Feature Extractor Network to extract spatial features. Similarly, the optical flow frames were passed through a Temporal Feature Extractor Network for temporal feature extraction. The spatial and optical features were fused and passed to a pooling layer followed by two fully connected layers to recognise the dynamic hand gesture. To verify their approach, the authors trained and tested their system on the Hand Gesture Datasets of Northwestern University and Cambridge. The model reached 97.6% and 98.6% accuracy on the Hand Gesture datasets of Northwestern University and Cambridge respectively.

Dynamic hand gestures are recorded in a sequence of frames instead of a single frame. Dynamic hand gesture recognition is difficult and inefficient using traditional two-dimensional convolutional neural networks unless the 2D-CNN is provided with both spatial features and temporal features. A new kind of deep learning model which can learn both spatial and temporal features was presented by authors of [24]. The new proposed model, C3D, is a 3-dimensional convolutional neural network (3D ConvNets). The authors found out the optimum temporal kernel depth for 3D-CNNs and concluded that C3D has the capability to learn spatial as well as temporal features concurrently. C3D outperformed other modern methods on ASLAN dataset by achieving 78.3% accuracy. In [25], a three-dimensional convolutional neural network has been proposed by combining a High Resolution Network and Low Resolution Network for driver hand gestures recognition. The final prediction is given by element-wise multiplication of class probabilities given by the two networks. To reduce the chances of overfitting and effectively train the model, spatiotemporal data augmentation was done on the dataset. The authors claimed that their proposed system achieved 77.5% accuracy on VIVA dataset.

In [26], authors presented a 3DCNN-LSTM-based dynamic hand gesture classification system that recognises hand gestures by using a sequence of frames from two-dimensional RGB videos. The model acquired spatiotemporal characteristics by integrating a 3D-CNN and a Long-Short term Memory (LSTM). The initial architecture underwent training on a comprehensive dataset and subsequently underwent fine-tuning on a specialised smaller dataset using the transfer learning technique. This smaller dataset was specifically created to evaluate the model's capability. The authors documented a heightened level of precision, reaching 93.95%, when evaluating the bespoke dynamic hand motion dataset. The authors of [27] have presented a method for dynamic gesture identification that combines depth and RGB modalities as input to a deep learning model. They also incorporate a finite state machine controller to enable context-aware decision-making. The proposed model utilises a combination of a 3D-CNN and LSTM to acquire spatiotemporal information through two concurrent input streams. One stream processes RGB data, while the other stream processes depth input.

Ultimately, the features acquired from both streams are merged and transmitted to a fully connected layer to generate the final prediction. In order to enhance the precision of real-time predictions, the authors integrated a finite state machine controller into the system. This controller effectively creates a context-aware model by manipulating the softmax decision probability through the adjustment of weights in the final layer. The suggested technique attained a precision of 97.8% on eight distinct gestures derived from a specialised video collection. The authors asserted that the employment of FSM resulted in an enhancement of the real-time recognition rate from 89% to 91%.

Simultaneous dynamic hand gestures detection and classification in real-word systems is a difficult job due to the large variation in the way people perform features and a considerable delay between gesture performing and classification. Ideally, the system should be able to classify the hand gesture before the gesture has ended to provided instantaneous response to the user. To deal with this, authors of [28] came up with an online Recurrent 3D-CNN model, capable of detecting and classifying dynamic gestures of hand at the same time on a multi-modal input. To predict hand gestures from in progress hand gestures, the authors trained the model using connectionist temporal classification. To evaluate their model, a new video dataset is introduced by the authors which was captured with the help of colour, depth, and stereo-IR sensors. The Recurrent 3D-CNN architecture clocked an accuracy of 83.8% on the custom dataset. The authors of [29] proposed a CNN-RNN based architecture to classify dynamic gestures of hand. A new Indian Sign Language dataset containing 33 gestures corresponding to the months, days, and weeks of the year. 12 months of the year has been generated. The images in the dataset were pre-processed by converting them from RGB to HSV colour space and to make the images noise free, operations of dilation and erosion were performed on images segmented. The suggested architecture was trained on a smaller version of the original dataset comprising of 12 hand gesture classes representing 12 months of year. The authors concluded that CNN-LSTM model achieved 99.14%.

The authors of [30] utilized the hand keypoints and bounding box information of the hand to classify static hand gestures. A top-down approach based human pose estimation is used as a feature extractor to give hand pose keypoints and hand bounding

box information. These features are normalized and processed before passed to a classifier. For classification, the authors introduced a new Two-pipeline model. One pipeline implemented a CNN, mobileNetv2, while the other pipeline used dense layers for classification of gestures of hand. The outputs of both pipelines were concatenated and passed to layer with softmax for classification of gestures of hand. The authors reported their approach achieved 91% accuracy on HANDS dataset, 95% accuracy on SHAPE dataset and 94% correct prediction rate on OUHANDS dataset. Another technique using 3D hand pose estimation to recognise dynamic hand gestures has been presented by [31]. Firstly, Faster R-CNN with BA's is used for the detection and extraction of the hand from the input RGB and depth image. Then, 2D Hand Pose of the hand is estimated using OpenPose. The hand pose coordinated of RGB image were mapped on depth image to generate 3D Hand pose of the hand. Finally, the authors presented a new framework by combining 3D-CNN and ConvLSTM. The authors introduced a data fusion technique based on weighted sum of inputs to generate the input feature vector for the deep learning model. Data fusion was done by pixel-level fusion of RGB, depth, and 3D hand pose images by multiplying pixels with a weight and adding the pixel values. The proposed architecture was trained on a custom dynamic hand gesture dataset constructed for this model. The authors reported 92.4% accuracy for the outlined approach.

Nearly all the dynamic gestures of hand are recognized by the systems that are based on deep learning models with high parameters count. Deploying such systems in practical world for real-time recognition of gestures is not feasible. To overcome this, authors of [32] came up with a simple deep learning model that combined CNN and Transformer for accurate detection of dynamic hand gestures. The proposed architecture fused Convolutional Neural Network with a Transformer model using spatio-temporal attention mechanism. The authors evaluated the model on two publicly accessible datasets, one is Jester [33] and other one is NVGesture [28] datasets, and one custom built dataset. The MetaFormer model achieved 96.72% accuracy on Jester dataset, 92.16% accuracy on NVGesture dataset, and a 90.71% accurate predictions rate on custom generated dataset. In [34], a transformer based dynamic hand gesture recognition multi-modal architecture has been introduced. The model utilized ResNet-18 [35] as a feature extractor followed by a transformer encoder based on self-attention

mechanism for temporal features analysis for dynamic hand gestures classification. The researchers demonstrated that employing solely depth maps and the derived surface normals as input can yield impressive outcomes. The suggested model was trained and tested on two datasets, namely Briareo and NVGesture datasets. The model recorded 82.4% accuracy on NVGesture dataset on single input. The accuracy jumped to 87.6% when several modalities were employed as input. On Briareo dataset, the proposed architecture achieved 87.3% when using depth and normal as input while a very slight improvement, 87.6% accuracy, was depicted when colour, depth, IR, and normal were used as input.

2.1.2. Hand Gesture Recognition Using Sensor-Based Technique

Similar to the vision-based approach to gesture recognition, the other most common technique for recognizing hand gestures is using various physical sensors. Researchers have used different sensors for this purpose, such as flex sensors, accelerometers, gyroscopes, electromyography, photoplethysmography, etc. Some of them use these sensors alone, while others use a combination of sensors for better results. As time progresses, we've witnessed significant advancements in wearable sensor-based hand gesture recognition systems. This is due to a growing interest in human-computer interaction, which has heightened the need for these systems in critical applications such as sign language interpretation, virtual reality (VR) and AR, gaming and entertainment, education, and training, among others. Initially, researchers used these sensors to create models that could only identify static objects. Nevertheless, progress in this domain has resulted in the creation of models that can accurately recognize a wide range of dynamic hand movements.

Researchers started leveraging machine learning for gesture detection as a result of its advancement. However, prior to this evolution, this task was performed by comparing real-time data from sensors with previously stored data, then predicting the gesture that most closely matched the stored data for the relevant gesture [36]. Most gesture-sensing systems use flex sensors, motion sensors, and electromyogram (EMG) sensors. However, [37] suggests a new way to recognize different hand gestures by decoding

the information from photoplethysmography (PPG) sensors. This is based on the idea that tendons and muscle movements change the shape of the arteries, and then a learning-based algorithm learns these changes to recognize the gestures. Similarly, [38] uses the concept of capacitive sensing for gesture recognition. The capacitive sensors on the board capture the types of gestures, and then a machine learning algorithm recognizes the specific static gesture.

Flex sensors serve as potential sensors used to distinguish various hand gestures. These sensors are variable resistors, which change their resistance values if we bend them. Therefore, we place these flexible sensors over the fingers, allowing them to detect bending by altering their resistance values. Most of the papers used these sensors alone or in combination with other sensors for gesture recognition. [39] designed a flex sensor-based wireless operating glove to extract information about position and velocity of fingers. They recognized the six common hand gestures using a combination of SVM and a neural network. A similar concept is used in [40]. They only used artificial neural networks for their case, achieving an accuracy of 88.32%. Similarly, [41] developed a system for dynamic gesture recognition using a similar glove construction. The system picks up on different hand movements by using a gated recurrent neural network (GRU) and maximum posteriori (MAP) estimation to watch the change between two different gestures. [42] has developed a sensor-based glove that measures the angles of the finger joints. With this information, they developed a learning-based approach to detecting different dynamic systems of the hand. Their algorithm tracks the gesture's start-to-end data for classification. Moreover, [43] claims that they introduced a novel glove with flex sensors for finger motions and motion sensors over the arm to detect the motion of the arm as well. Using a combination of these two sensors, they developed a mechanism for the recognition of American sign language (ASL) and Chinese sign language (CSL).

Moreover, electromyography (EMG) serves as another potential sensor for gesture recognition. Skeletal muscles produce electrical activity, which electromyography (EMG) sensors measure. They sense the electrical signals produced by muscle

contractions, which can be processed to detect specific movements or gestures. The position of the electrodes is crucial in this case; even a small deviation can impact the results. Nevertheless, past research has attempted to use these sensors alone or in combination with other motion sensors. [44] put different EMG sensors on the arm and collected data based on how the arm and fingers moved. They then used the extreme gradient boosting (XGBOOST) algorithm to recognize the signs for American Sign Language (ASL). Artificial intelligence provides you with a very diverse path for every problem. Using the same setup [45], you can detect hand gestures using a fully connected artificial neural network. [46], [47] achieved this task by combining data from IMU and EMG sensors, but these methods are not considered user-friendly as they necessitate the placement of numerous sensors on the user's naked arms for proper operation.

If we investigate motion sensors, the most commonly used devices are the gyroscope and accelerometer. These are 3-dimensional motion sensors. Gyroscope sensors provide information on angular velocity on each of the three axes (X-axis, Y-axis, and Z-axis), whereas accelerometers try to provide information regarding linear motion in the form of acceleration in each direction. As a result, these sensors could be placed on the hand or arm in combination for gesture recognition. [48] used the data from both sensors and fed it into their developed gesture spotting algorithm. This algorithm detects the start and end points of the gestures, and then recognizes them by comparing them with their gesture database. With the advancement of recurrent neural networks, continuous gesture spotting has improved significantly. So, leveraging these concepts, [49] presented a general continuous hand gesture algorithm based on long-short-term memory (LSTM). It requires data from a gyroscope and an accelerometer for training. Similarly, [50] developed an algorithm for sensory data called PairNet.

2.1.3. Fusion in Machine Learning

The majority of modern machine learning-based systems employ more than one machine learning model to achieve a certain task. A single classifier is inefficient when a wide variety of data is present, the types of features are dissimilar, and generalization

is required. To improve the prediction accuracy of the system in such a situation, multiple classifiers are preferred over a single classifier [51]. In the literature, there are multiple techniques for fusing multiple machine learning models to achieve better accuracy. Fusion is a technique whereby more relevant data is generated by combining data or information from multiple sources [52]. Fusion techniques are broadly divided into three categories: early fusion (data fusion), joint fusion (feature fusion), and late fusion (decision fusion). Out of these three, joint and late fusion involve multiple machine learning models, while in early fusion, there is only one classifier, but the input is combined from several sources.

Early fusion, or data fusion, refers to combining data coming from heterogenous sources. Both decision fusion and feature fusion can be classified as further types of data fusion. Input coming from multiple sources is referred to as multi-modal input, and the fusion of multiple modalities of data has been extensively used in deep learning models. The paper [53] discusses different types of modalities available, like text, video, image, and other signals for multi-modal architectures. The classical technique to fuse data in the past was based on probabilistic fusions such as Bayesian fusion, rough-set based fusion, etc. [54]. To combine data coming from various sources, the most widely adopted procedure is to extract features using machine learning technique and use them as input for the actual classifier. The most popular machine learning techniques present in the literature to perform data fusion are k-NN, Principal Component Analysis, Bayesian Classifier, and clustering. [55] provides a comprehensive review of the use of machine learning techniques in data fusion. The authors of [56] fused data coming from sensors, in numerical form, and data from medical records, in the form of text, using the information gain method to improve the accuracy of healthcare monitoring systems. A new data fusion technique, Attributed Heterogenous Network Fusion, to combine multi-related data from different sources by learning inter-relations and minimizing loss by avoiding insufficient relations, has been proposed in [57].

In any classification problem, features play the most crucial role as they provide useful information. One of the reasons for feature fusion is that in the practical world, rich features as well as noisy and irrelevant features are present. For accurate prediction, noisy and irrelevant features should be filtered as they confuse classifiers. To remove noisy features, feature fusion is done to reduce the size of features so that only relevant features are passed on. The most common type of feature fusion is early fusion, which is similar to data fusion. In early feature fusion, features are fused from a set of features extracted from data obtained from several sources. The methods used for early fusion included averaging, concatenation, and weighted linear combination. [58] developed a multi-modal depression recognition system by employing feature-level fusion on data collected from various sensors. Non-linear and linear feature extraction from the data collected from every source is followed by feature fusion utilising linear combination method.

The authors of [59] increased the classification accuracy of the system by employing multiple models and fusing features extracted by multiple classifiers in the network. [60] improved the housing property value assessment system by fusing features extracted by two deep neural networks, each model working on two different data sources. The ability of the model to automatically learn relevant features based on the enhancement deep feature fusion technique has been proposed in [61]. The authors built an auto-encoder for automatic feature learning and used locality preserving projection (LPP) method for the fusion of deep features. Attention-based architectures like Transformer also perform feature fusion due to their capability of focusing on useful features and suppressing irrelevant features. Graph-based feature fusion has been utilised by Graph Attention Networks (GATs) and Graph Convolutional Networks (GCNs) by collecting features from surrounding nodes in a graph [62].

Decision fusion is a very popular technique used in many machine learning systems to raise the accuracy whereby combining predictions of multiple models. In the literature, multiple techniques for decision fusion can be found, like averaging, weighted averaging, majority voting, stacking, etc. [63] enhanced the accuracy of the tree species

recognition system by employing multiple SVMs, each operating on a separate feature set, and fusing their decisions based on Murphy's average method. [64] increased the prediction accuracy of structural damage detection systems by utilising several 1-D Convolutional Neural Networks and fusing their results based on majority voting. The fuzzy-based decision fusion technique has been utilised by [65] to increase the accuracy of heart disease classification. Their system included two Support Vector Machines (SVMs), each receiving a different set of features from multiple sources, and the final prediction was given by a fuzzy-based system. Decision fusion using stacking ensemble learning is widely used in multiple machine learning and deep learning-based systems. [66] utilised the concept of ensemble learning to elevate the accuracy of Maize Chlorophyll Content estimation system by using three classifiers to make predictions. The predictions given by the classifiers were stacked and given to a meta-model to give the final prediction. In their system, they used a Multiple Linear Regression model as a meta-model. Other techniques used in ensemble learning include bagging and boosting.

2.2. Limitations in Prior Work

The different types of architectures based on deep learning and machine learning models discussed in the related work section of the vision-based approach are vulnerable to certain limitations. Hand gesture detection traditionally relies exclusively on image processing techniques. These techniques exhibit subpar performance in complex backgrounds, diverse lighting conditions, various distances between the hand and the camera, and occasionally owing to differences in skin color and clothing worn. The adoption of a machine learning classifier for hand gesture detection effectively resolved many of the challenges encountered by image processing techniques. Machine learning classifiers do not perform well on large datasets with a wide variety of data and high interdependencies of features because these algorithms do not possess the ability to learn complex relations between features. Therefore, this approach also fails in situations where generalization is required.

Deep learning models offer a solution to the problem of generalization. Deep learning architectures like CNNs extract features from input data and learn the interconnections between features and variations in the data due to their backpropagation technique to

update filter weights. This makes them suitable for situations where machine learning algorithms fail, but the added cost comes in the form of an increase in computational complexity. Although there are quite a few CNNs available that provide a good speed and accuracy trade-off, Similarly, 3D-CNN and CNN-RNN architectures suffer from high computational complexity and require large amounts of data for training, making them largely unsuitable for real-world applications. Moreover, attention-based architectures like Transformer were recently introduced in computer vision tasks, so not much information is available about their training details, dataset requirements, different models, limitations, etc.

Given the extensive research in the area of sensors-based gesture recognition, most suggested methods rely on physical sensors, which can be challenging to integrate in practical settings. Placing EMG sensors all over the arm is not a user-friendly approach. Additionally, many of the approaches use a combination of multiple learning-based algorithms, which requires a lot of computation power.

2.3. Proposed Approach

This project involves detecting a mixture of static and dynamic hand gestures to control the computer cursor. For computer cursor control, classification of gestures as well as localization of the hand are required. The suggested deep learning model for the vision-based approach is called YOLO (You Only Look Once), and it can complete both tasks in real-time. The cutting-edge object detector YOLO offers an excellent compromise between speed and accuracy. It was first introduced in [67]. Since then, YOLO has continued to evolve, with different versions coming after the initial release of YOLOv1. After YOLOv1, several versions of YOLO came in the form of YOLO9000 [68] or commonly known as YOLOv2, YOLOv3 [69], YOLOv4 [70], YOLOv5 [71], Scaled YOLOv4 [72], YOLOR [73], YOLOX [74], YOLOv6 [75], YOLOv7 [76], DAMO-YOLO [77], and YOLOv8 [78]. In this project, YOLOv8 is utilised for real-time gesture detection and localization as well as pose estimation of hand keypoints.

The aim was to fuse sensor-based gesture recognition with a vision-based approach. The traditional approaches discussed in related work section of sensor, restricts us to use such an approach which had larger execution time. We had to choose an approach to little inference time. Overall, we had to choose the path to implement the system that should meet these three objectives:

- More user-friendly.
- Less computations.
- More Robust.

We decided to design gloves based on flex sensors to detect our fingers' movements and use a fully connected artificial neural network for algorithm development after reviewing all the literature and comparing it to our objectives. This was because flex sensors are extremely flexible, light, and easy to place over the glove. Furthermore, they are capable of identifying even the smallest finger movements. To further improve the system's accuracy, the predictions given by both models, vision-based approach model, and sensor-based approach model are fused using another ANN model. After fusing the results of YOLOv8 and the sensor model, an algorithm is developed that can predict dynamic hand gestures and execute mouse commands corresponding to the gesture detected. The complete flow diagram of the proposed approach is given in Fig. 1.

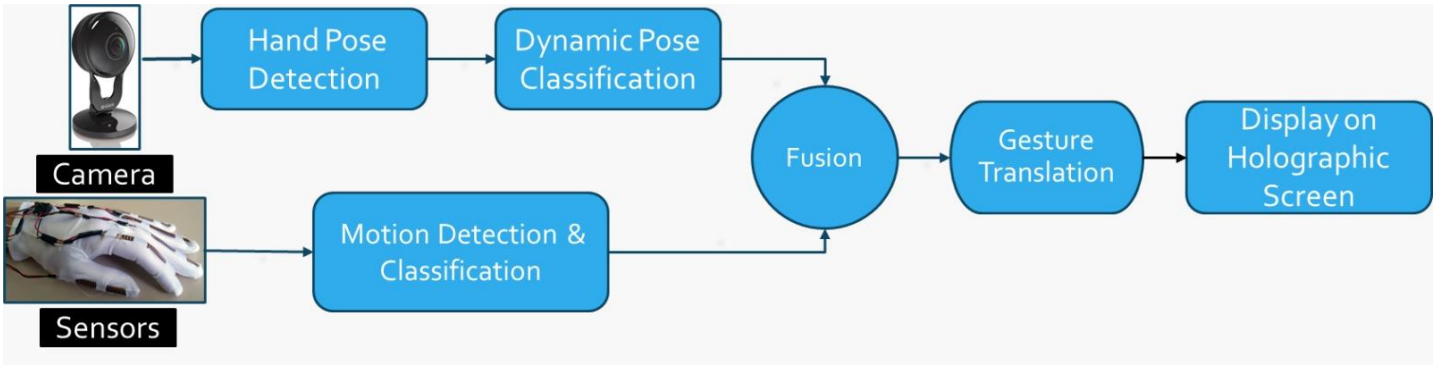


Figure 1. System flow diagram of proposed approach

Chapter 3 – METHODOLOGY

3.1. Vision-Based Approach Methodology

3.1.1. Model Architecture

YOLOv8 comes in different sizes, each having more depth than the previous ones. Additionally, YOLOv8 can perform other tasks besides object detection, such as classification, pose estimation, segmentation, and tracking. For vision-based approach, the model chosen is YOLOv8s with pose estimation. YOLOv8's architecture is shown in Fig. 2. Additional information related to YOLOv8s is given below:

- Model Parameters = 11.6 million
- Number of FLOPs = 30.2 billion

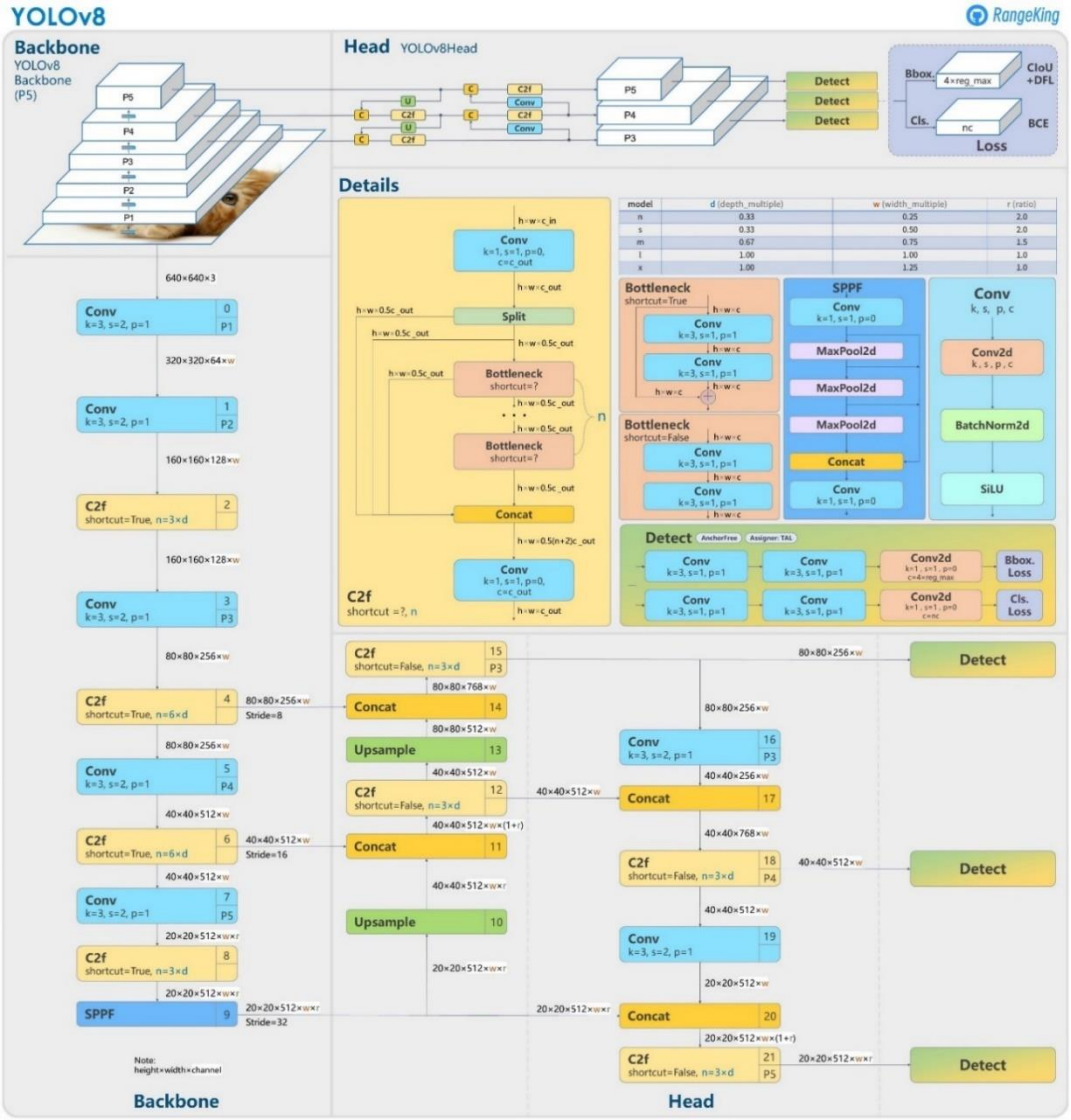


Figure 2. Architecture of YOLOv8

The architecture is split into three main parts: the backbone, neck, and head.

Backbone:

The backbone's roll in YOLOv8 is to extract features from input. It progressively reduces the spatial dimensions of the input visual data while capturing increasingly abstract and robust features.

- **Convolutional Layers (Conv)**:
    - **Role**: Perform the primary operation of extracting features through convolution, a mathematical operation which combines an input image with filter to produce feature maps.
    - **Details**: In the Fig. 2, Conv layers are represented with their kernel size (k), number of output channels (c), padding (p), and stride (s). For instance, a Conv layer with k=3, p=1, and s=2 indicates a 3x3 filter with stride 2 and padding 1, which reduces the spatial dimension by half.
- **C2f (Cross-Stage Partial Networks)**:
    - **Role**: Enhances feature reuse and propagation with fewer parameters.
    - **Details**: This module, denoted with shortcut=True, implements skip connections that allow direct pathways for gradients during backpropagation. It's particularly useful in preventing vanishing gradients and enabling deep networks to train effectively.
- **SPPF (Spatial Pyramid Pooling - Fast)**:
    - **Role**: Aggregates features at multiple scales in a fast and efficient manner.
    - **Details**: The SPPF layer aggregates features at multiple levels and combines them, enabling the network to handle objects of varying sizes effectively. This multi-scale pooling operation is efficient and aids in capturing global context without increasing computation significantly.
- **Downsampling Operations**:
    - **Role**: Reduce the spatial dimensions to distil abstract features.
    - **Details**: Strided convolutions with stride=2 are utilised to downscale the feature maps, while retaining essential features. This hierarchical reduction is crucial for handling large input images efficiently.

- **Multi-Scale Feature Maps (P1 to P5)**:
    - **Role**: Provide feature maps at different resolutions.
    - **Details**: P1 to P5 represent different scales of feature maps extracted from the backbone. P1 is the finest scale (highest resolution), and P5 is the coarsest (lowest resolution). This approach enables YOLOv8 to locate objects of varying sizes effectively.

Neck:

The purpose of the neck in YOLOv8 is to enhance and combine characteristics from various scales generated by the backbone. It bridges the gap between feature extraction and final detection.

- **Upsample Layers**:
    - **Role**: Increase the spatial resolution of feature maps.
    - **Details**: Upsampling is typically done using nearest-neighbour interpolation or transposed convolutions. This process allows the network to regain spatial information that may have been lost during downsampling, crucial for precise localization tasks.
- **Concat Operations**:
    - **Role**: Merge features from different stages or scales.
    - **Details**: By concatenating feature maps from different scales, the network combines both low-level and high-level information, enriching the feature representation and making it robust for detecting objects at various scales and complexities.
- **C2f Layers**:
    - **Role**: Enhance the feature representation and propagation.
    - **Details**: Similar to their use in the backbone, C2f layers in the neck continue to leverage residual connections and efficient feature fusion, ensuring that the refined feature maps retain rich contextual information.

Head:

The head of YOLOv8 is where the final detection predictions are made. It processes the refined multi-scale features from the neck and outputs the bounding boxes, object class scores, and other relevant information.

- **Detection Layers**:
  - **Role**: Generate predictions at multiple scales.
  - **Details**: YOLOv8 uses detection layers at different feature map scales (P3, P4, P5) to handle small, medium, and large objects. This multi-scale approach is integral to the model's capability to localise objects of different sizes.
- **Convolutional and C2f Layers**:
  - **Role**: Further process the feature maps before making predictions.
  - **Details**: These layers refine the features through additional convolutions and feature fusion, ensuring that the features are well-suited for precise and accurate predictions.
- **Output Layers**:
  - **Bounding Box (BBox)**: Predicts the coordinates of the bounding box around detected objects.
  - **Classification (Cls)**: Predicts the class label for the detected objects.
  - **IoU (Intersection over Union)**: Calculates the intersection of actual and anticipated boxes, which serve as an indicator of the reliability of the predictions.
- **Loss Function**:
  - **Role**: Guides the training process by quantifying the difference between predictions and ground truth.
  - **Details**: YOLOv8 typically uses a combination of losses, such as Binary Cross-Entropy (BCE) for classification, CIoU (Complete IoU) for bounding box regression, and other components to ensure robust and precise model training.

The architecture utilizes the SiLU (Sigmoid Linear Unit) activation function. In general, YOLO (You Only Look Once) models typically use multiple loss components, including:

- **Objectness Loss:** This component specifically examines if the bounding box prediction includes an object or not. This is computed using binary cross-entropy loss function. The function is given as:

$$\frac{1}{N}\sum_{i=1}^{N} -\left(y_i * \log(p_i) + (1-y_i) * \log(1-p_i)\right)$$

  Where p_i shows the probability of the detected class, and (1-pi) shows the probability of class undetected.

- **Localization Loss:** This component quantifies the precision of the anticipated bounding box coordinates. This calculation is performed using the Mean Squared Error Loss function. The mathematical expression of this function is given below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

  y_i shows the actual output value and $\hat{y}_i$ shows the predicted output value.

- **Class Confidence Loss:** This component evaluates the confidence scores for the predicted class labels. It is computed using the binary cross-entropy loss function.
- **Keypoints Loss:** This loss component calculates the key point loss and key point object loss for a given batch. Mean squared error loss function is used to calculate the keypoints loss.

3.1.2. Model Training

The model is trained using HaGRID (Hand Gesture Recognition Image Dataset) [79] . The hand gestures chosen to train the model are depicted in Fig. 3.

Figure 3. Gestures used for training

The labels assigned to every class during training are given below:

palm: 0

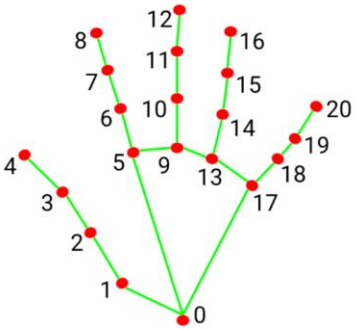
fist: 1

one: 2

two up: 3

three: 4

The training dataset contained 21615 images, while the validation dataset contained 1749 images. Following the completion of training, the model underwent testing using a separate validation dataset. For keypoint annotation, the images were annotated based on the hand landmarks given by MediaPipe hand Pose. The hand keypoints given by MediaPipe are illustrated in Fig. 4.



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

Figure 4. Hand key landmarks given by MediaPipe

3.1.2.1. Hyperparameters

Hyperparameters are configurations for training a neural network. They are not part of the network itself, but they control how the network is trained. The user specifies the

values, and any alterations to these variables have a direct impact on the results of the network. Several crucial hyperparameters include:

- **Optimizer:**

  An optimizer is an essential element of a neural network's training process. It continuously changes the biases and weights of the model in order to minimize the loss function. Among optimizers, the stochastic gradient descent function makes the weights converge quickly due to its ability to process large data sets. It estimates the gradient using a random sample of data points, which is why it is highly computationally efficient. Its formula is given as:

$$\theta = \theta - \eta \nabla J\left(\theta; x^{(i)}; y^{(i)}\right)$$

  Where, ∇J is given as:

$$\nabla J\left(\theta; x^{(i)}; y^{(i)}\right) = \nabla L\left(\theta; x^{(i)}; y^{(i)}\right)$$

  ∇L represents gradient of the loss function discussed in the section 3.1.1.

Other Training Hyperparameters includes:

- Batch Size = 16
- Number of Epochs = 100
- Initial Learning Rate = 0.001

3.2. Sensor-Based Approach Methodology

In the order to detect the mentioned gestures using sensor-based approach, firstly a sensor-based glove was designed from the scratched in the order to sense the required gestures of the hand and then by using the data from that designed glove, a learning-based approach is developed in the order to recognize different gestures and tested in the real-time.

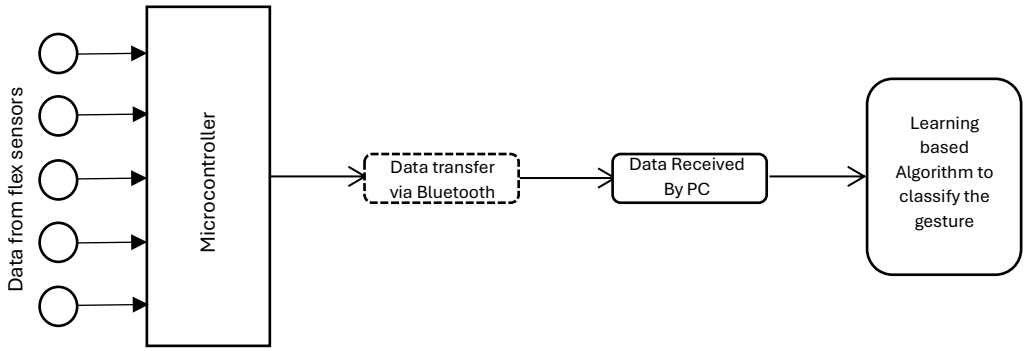


Figure 5. System flow diagram of sensor-based approach

Fig. 5 illustrates the implementation of sensor-based approach. The microcontroller reads data from the flex sensors via an analogue-to-digital converter, then sends the collective data via Bluetooth to the connected PC. In real time, a learning-based algorithm reads the data and classifies the performed gesture.

3.2.1. Selection of Hardware Components

Below is a list of the components utilized in the design of the glove for detecting finger flexion:

- 5” flex sensors, 04
- 2.2” flex sensor, 01
- 3.7-volt rechargeable Li-ion battery, 01
- Cell Holder, 01
- Schottky diode: 01
- Right-handed gardening glove, 01
- T-blocks: 03

- Wires
- Resistors 100kΩ: 05

The sensing-glove includes the placement of the above-mentioned components on the glove for its proper functioning. The reasoning behind each component's selection and its working is given below.

3.2.1.1. Glove

Figure 6. Glove used for Hardware design

The glove we used is a gardening glove. We use only the right-hand side of the gardening glove. On the glove, we had to place different components, including sensors, a battery, a microcontroller, etc. We chose this glove due to its material, which allows for easy component placement, and its colour, which is crucial as the camera will be the main component of this system. This colour is very helpful for the proper functioning of the vision-based approach.

3.2.1.2. Flex Sensor

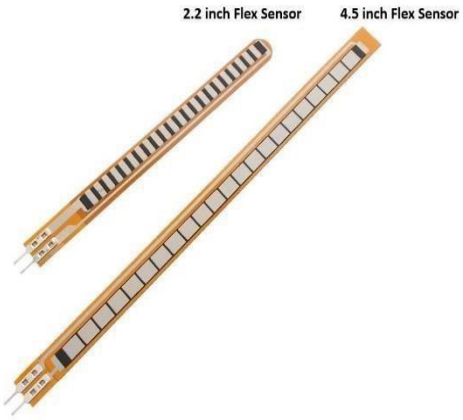


Figure 7. Sensors used for sensing the finger movements

In order to detect the movements of the fingers, a set of five distinct flex sensors were employed, with each sensor dedicated to a specific finger. Typically, these flex sensors are offered in two distinct sizes. The initial dimension of the flex sensor is 4.5 inches, while the subsequent dimension is 2.2 inches. A total of five flex sensors were used, consisting of four 4.5-inch sensors and one 2.2-inch sensor. The smaller 2.2-inch flex sensor placed over the little finger to observe it movements whereas for all other fingers, 4.5-inch flex sensor were used for glove design.

These flex sensors are nothing but a variable resistor. Their resistance values change with the degree of bending. When we let these sensors stand, they offer less resistance. However, as the flex sensors bend, their resistance increases. The flex sensor's idea is that when we bend our fingers, it (placed over the gloves) will also bend, increasing resistance. We will use this information to detect different hand gestures.

3.2.1.3. Microcontroller

Figure 8. Selected Microcontroller (ESP-32)

A microcontroller was selected to receive the signals from the sensors and send them collectively to the computer via Bluetooth. The microcontroller used for this purpose was an Espressif module (ESP-32) with 38 pins. The ESP's preference was that it comes with a built-in advanced Bluetooth module; this feature reduced the need for a separate Bluetooth module. Furthermore, it was economical and required less power to operate (3.3 volts to 3.7 volts) than any other microcontroller.

3.2.1.4. Battery

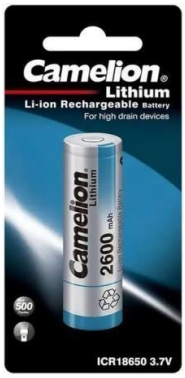


Figure 9. Battery used to power the components

To receive data from the five sensors and to power the ESP, a 3.7-volt lithium-ion Camelion battery and its holder were used to establish the connections.

These rechargeable batteries do not come with a fixed 3.7 volts. Upon full charge, the voltage of these rechargeable batteries consistently exceeded the nominal voltage, reaching 3.9 to 3.8 volts. However, this ESP 32 operates within a voltage range of 3.3 to 3.6 volts. The ESP could easily burn if powered directly from the battery.

To mitigate this issue, a Schottky diode was used to drop the voltage before reaching the ESP. This diode has a forward voltage drop of up to 4.5 mV. The voltage from the battery passed through a Schottky diode. This diode dropped the incoming voltage to less than 3.6 volts. Therefore, this diode was connected in series with the ESP.

Moreover, the same battery was used to power up each flex sensor, connecting them all in parallel.

Note that the Schottky diode was only required for the connection with the ESP; it was not required for the connection with sensors.

Caution:

Avoid using local product because they offer lower current and ultimately it will end with a drop in voltage in no time.

3.2.1.5. T-blocks, Resistors, and Wires

Five flex sensors and a microcontroller required a connection from one battery. There was a need for a T-block for reliable connections. One of the T-blocks received the power wire from the battery holder. Different wires from this T-block were used to provide connections to sensors and the controller. Similarly, for common ground, two other T blocks were used for the connection of the ground wires of the whole system.

In addition, wires were used to establish different connections. Additionally, 100 kΩ resistors were used in series with each flex sensor to drop the voltage across the sensors.

3.2.2. Glove Design

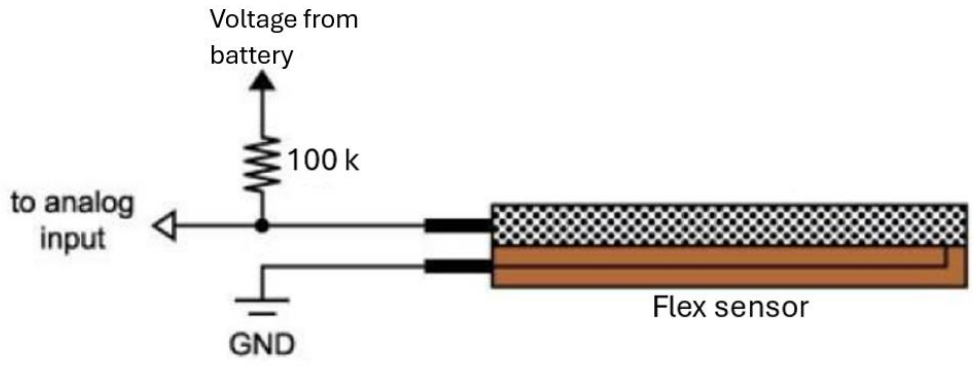


Figure 10. Connections for Flex sensor

Fig. 10 explains the connection necessary for the working of the flex sensors used in the glove design. Initially, a 3.8–3.9-volt battery was wired to the 100k resistor and then connected in series with the flex sensor. The aim was to read the voltage across the flex sensor, which appears in the above figure. The left-pointing arrow shows the connection to the analogue input of the ESP, which digitally reads the voltage across the sensor. Note that the ESP-32 features a 12-bit ADC (analogue-to-digital converter) with a nominal voltage of 3.3 volts. In other words, if we tried to input a voltage greater than 3.3 volts to the analogue pin of the microcontroller, it read the value as 4095 (the maximum). Therefore, a resistor (100 kΩ) was required for the connection. It dropped the majority of voltage in the resistor and passed 1–1.5V to sensors, which could be readable by the microcontroller. When this sensor is bent, the resistance increases, which increases the voltage across the sensor, and the microcontroller's ADC reads the changing voltage.

The above theory is applicable to all flex sensors. Different analogue pins on the microcontroller were used to read the voltage across each sensor.

Figure 11. Final design of sensing-glove

Fig. 11 is the final designed product, showing all the components embedded in it with proper connections. The installation of the components on the glove starts with the placement of the ESP-32 (on the glove, it is upside down) by stitching it with the glove using thread. A battery holder was placed directly over the ESP and sewn to the glove; this enabled these components to be properly fitted to the glove. This battery holder acted as a connection between the battery and other components. Additionally, three T-blocks were placed on the glove using glue. The left T-block served as a power source (it was connected with the power wire from the battery). Both right-sided blocks were used for ground purposes.

Using the glue, flex sensors were attached precisely to each finger of the glove. These flex sensors have two pins, and one of the pins from each was connected to a common ground. The other pin of the flex sensor was connected to a resistor (100 kΩ), which in turn was connected to a battery. The microcontroller's analogue pin received the voltage across each of the sensors via a wire connection. Through this pin, the ESP reads the voltage across each sensor.

Additionally, the wires used for the connections were also attached to the glove using glue. Furthermore, Fig. 12 shows the complete circuit diagram and demonstrates the above written theory clearly and shows how all the components were placed in the glove and their connections with each other.

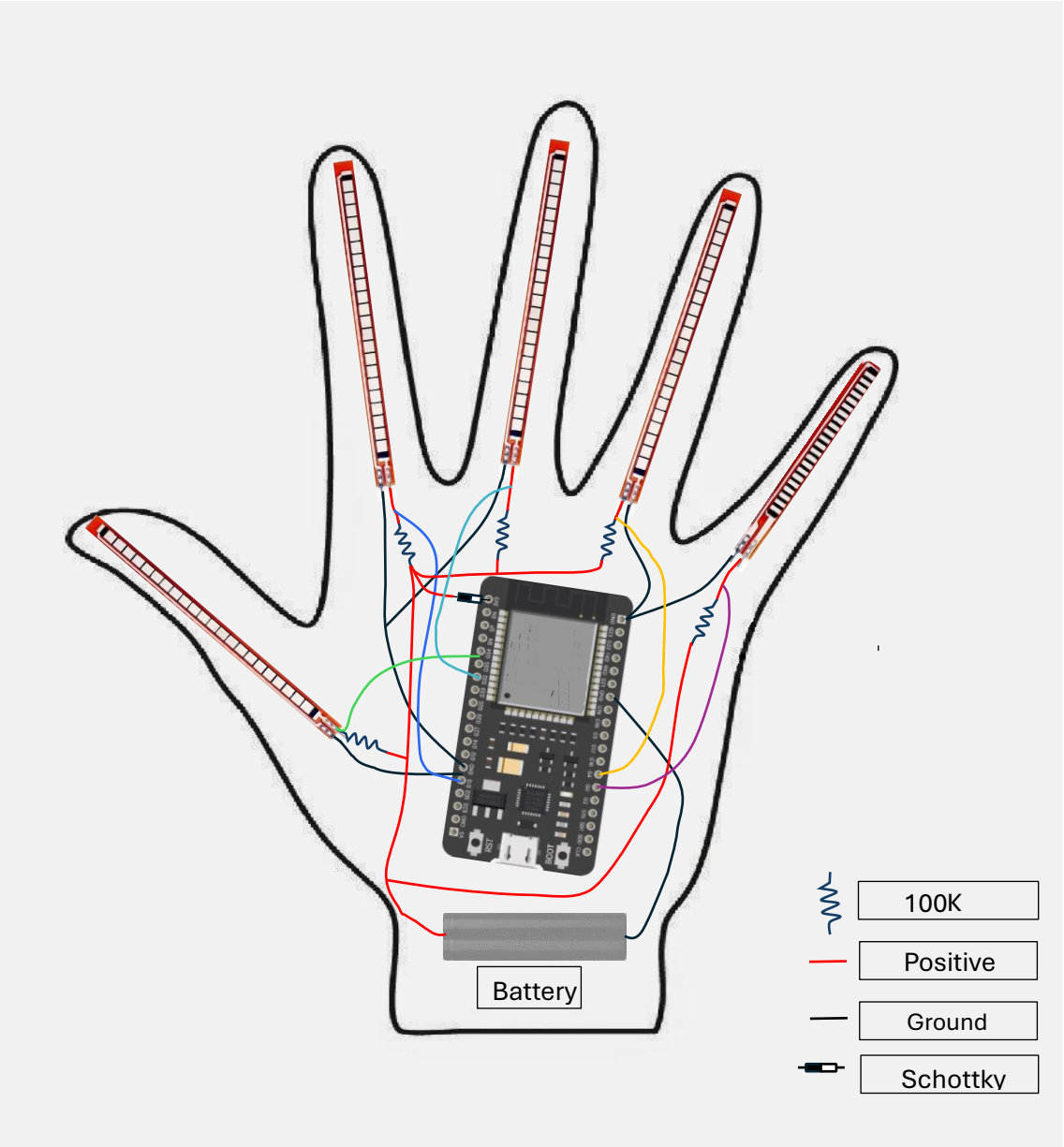


Figure 12. Diagram showing the circuitry of sensor-based glove

3.2.3. Software Implementation

3.2.3.1. Working of ESP-32

The microcontroller reads the data across each flex sensor, according to the Fig. 12. The algorithm is written in Arduino IDE software in the C language to run on the microcontroller. The algorithm is such that it runs in real time in a loop. In the loop, it reads the voltage across each of the sensors and converts it into a digital number in the range of 0–4096. After reading the data from each sensor, it groups together all five

sensors' data in a string, separating each data point using a comma. After the while loop ends, the string data looks like this:

For Example,

343,434,288,854,435

The above is a single example. The first data point on the left side represents information about the forefinger bending, followed by a comma, and then the next data point is for the middle finger, ring finger, and little finger. The final data point represents the thumb's bending information. With the degree of the bend of each finger, the value associated with each finger increases. Therefore, by activating the Bluetooth option of the ESP-32, the entire data will be sent. After every 10 milliseconds, the loop runs and sends the values via Bluetooth in the same format.

3.2.3.2. Bluetooth Connection

After uploading the above algorithm and powering up the ESP and glove sensors, it started working. The next step involves the Bluetooth connection between the device and the associated laptop. After activating the PC's Bluetooth, we paired it with the device named "ESP32." In the code, we have written this name. Once we paired with the ESP, we needed to ascertain its connected port. To do so, we went to the option "More Devices and Printer Settings." The connected device's name can be found here. A double-click on it allowed us to see the connected port of this connection.

3.2.3.3. Data Collection

After enabling to read the data wirelessly on PC, the next step was data collection for each of the gestures necessary to develop the neural network. Python language was used in Visual Studio Code to write the algorithm that reads the data in real-time and stores it. In real time, all the necessary connections were established, wore the sensor-based glove, and performed a single gesture. The microcontroller read the values of that specific gesture and sent them via Bluetooth to the computer. The computer received the data, separated the values of each finger, and converted them into an integer value.

We began storing the data for each finger in a CSV file. During the process of data generation for each gesture, we made the data diverse by moving the fingers while making that gesture. As a result, we stored 1500 data sets for each gesture. Fig. 13 shows how the data stored in CSV file looked like.

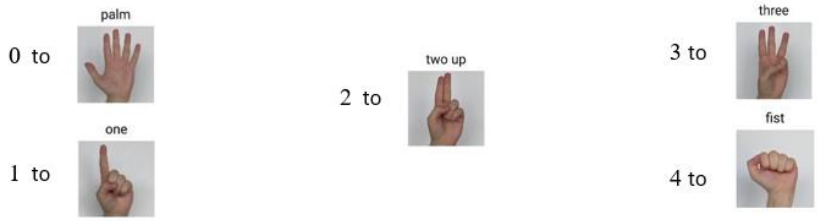
A	B	C	D	E
Index1	Middle	Ring	Little	Thumb
2862	2394	2293	3218	3216
2860	2414	2298	3213	3216
2885	2414	2326	3216	3211
2864	2368	2305	3215	3213
2885	2395	2306	3200	3217
2866	2369	2299	3213	3293
2864	2380	2288	3189	3263
2864	2382	2288	3179	3230
2867	2367	2298	3226	3227
2877	2382	2303	3198	3229
2869	2369	2294	3195	3243
2871	2384	2291	3201	3228
2864	2371	2293	3190	3217
2869	2382	2305	3204	3223
2864	2382	2271	3195	3221

Figure 13. Illustration of data collected for single gesture using sensor glove

3.2.3.4. Features Selection

To go for the selection of the features for the neural network, we combined the data stored for each gesture into a single CSV file. First five features were five values from each finger. To include non-linearity and enhance the variety of gestures, we incorporated the square of each finger value as a feature. In this way, we ended up with a data with ten features.

To label the data, each integer number is associated to each gesture:



The header of the data in CSV file is shown in Fig.14.

Forefinger	Middle Finger	Ring Finger	Little Finger	Thumb	Forefinger Square	Middle Finger Square	Ring Finger Square	Little Finger Square	Thumb Square	Label
427	910	859	1955	697	182329	828100	737881	3822025	485809	1
426	896	901	1950	692	181476	802816	811801	3802500	478864	1
424	896	883	1955	705	179776	802816	779689	3822025	497025	1
427	889	881	1957	703	182329	790321	776161	3829849	494209	1
429	894	880	1967	705	184041	799236	774400	3869089	497025	1
430	893	884	1979	705	184900	797449	781456	3916441	497025	1
427	892	882	1984	707	182329	795664	777924	3936256	499849	1
429	892	880	1990	708	184041	795664	774400	3960100	501264	1
426	893	865	1989	709	181476	797449	748225	3956121	502681	1
431	891	866	1985	706	185761	793881	749956	3940225	498436	1
431	887	880	1984	707	185761	786769	774400	3936256	499849	1
430	893	857	1985	708	184900	797449	734449	3940225	501264	1
430	891	880	1979	706	184900	793881	774400	3916441	498436	1
427	896	859	1973	706	182329	802816	737881	3892729	498436	1
429	905	881	1973	709	184041	819025	776161	3892729	502681	1
424	901	891	1968	706	179776	811801	793881	3873024	498436	1
422	903	879	1968	704	178084	815409	772641	3873024	495616	1
422	907	907	1968	705	178084	822649	822649	3873024	497025	1
419	906	880	1967	703	175561	820836	774400	3869089	494209	1
422	906	867	1968	705	178084	820836	751689	3873024	497025	1
425	906	875	1968	705	180625	820836	765625	3873024	497025	1

Figure 14. Illustration of data used for training the model used for sensor-based approach

3.2.3.5. Model Selection

To train the data to recognize the particular gestures, Artificial Neural Network (ANN) is chosen. The architecture is shown in Fig. 15.

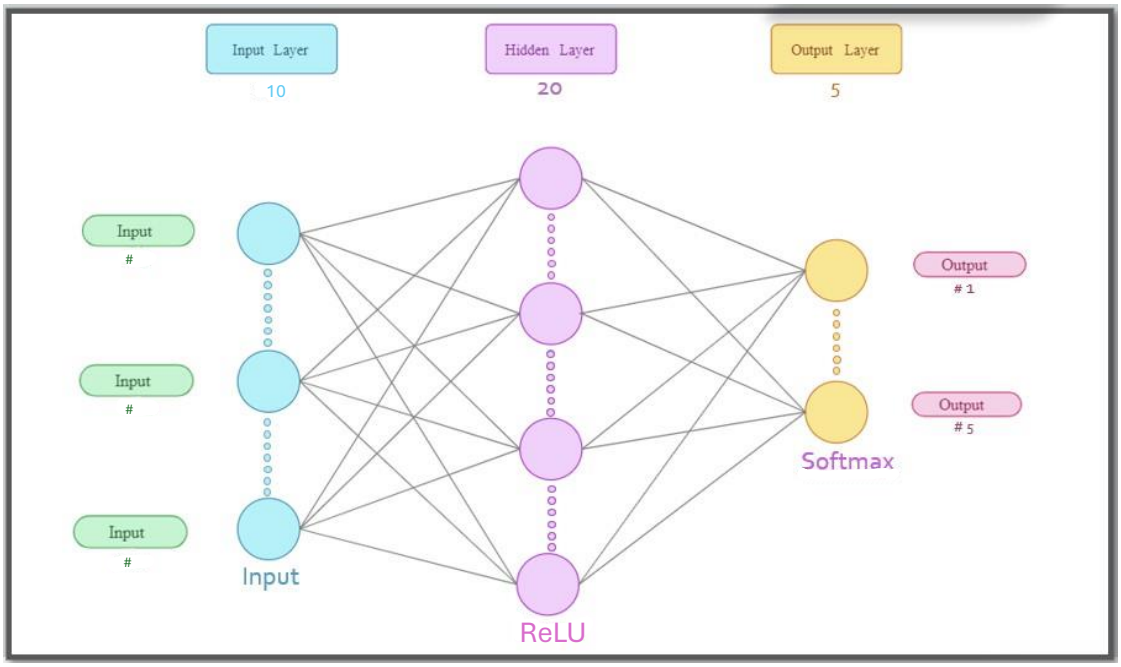
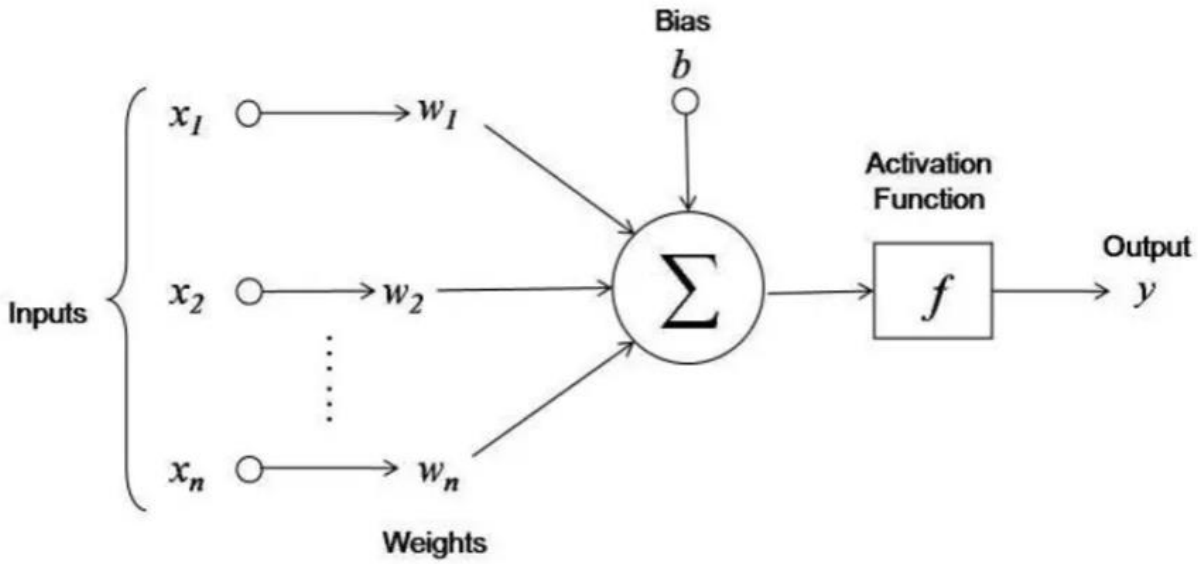


Figure 15. Architecture of the model used in sensor-based approach

One hidden layer is included in the model with 20 neurons. Each neuron in the hidden layer receives the data with 10 features, scales each input with a respective weight, and adds all the resulting products and a bias term. An activation function named ReLu receives this operation's result and allows it if it exceeds zero; otherwise, its output remains zero. Fig. 16 shows the operation of each neuron. This applies to all the 20 neurons of hidden layer.



$n = 20$

Figure 16. Mathematical operation of each neuron in the hidden layer

The final output layer consists of 5 neurons, with each neuron receiving the output values from each activation unit in the hidden layer. The Softmax activation function has been implemented in this layer. The goal of utilizing this activation function in the last layer is to get values that represent probabilities. In this layer, each neuron's output represents the probability of each gesture. This implies that every neuron's output is associated with a specific gesture. The neuron with the highest probabilistic output indicates that a specific gesture was executed. This relation is shown in Fig. 17.

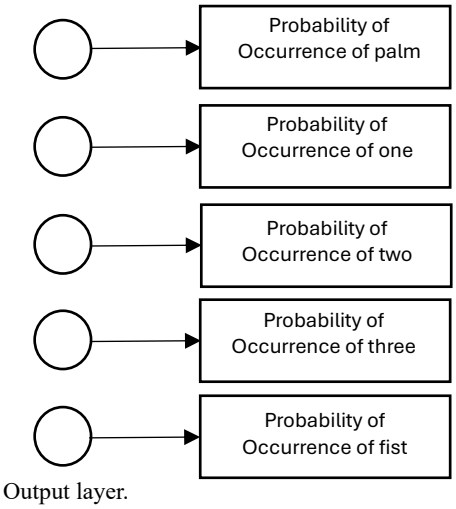


Figure 17. Association of the outputs of the last layer of sensor model with gestures

3.2.3.6. Hyperparameters

We had to select numerous hyperparameters based on which we had to train the model to get the accurate values of weights and biases.

- **Loss Function:**
  We chose a sparse cross-entropy function to calculate the loss function during training. The metric quantifies the discrepancy, or discrepancy, between the real probability distribution and the anticipated probability distribution of the output category. This sparse cross-entropy assumes that the output label is an integer value that directly represents the class indices. This loss function operates by first converting the true labels into one-hot encoded vectors, then applying the conventional cross-entropy loss function. Loss function is given as:

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

Here,

m indicates total number of examples

yi is the true label vector.

ŷi indicates the predicted label vector.

- **Optimizer:**

  Optimizer helps neural networks learn. It helps to find the best weights and biases for good results by adjusting model parameters and reducing the error. Parameters are adjusted according to the data and the loss function.

  For the training of our data, we used Adam Optimizer. This optimizer is a combination of other optimizers called momentum-based gradient descent and root mean square propagation. It updates the parameters using the following formula:

$$W = W - \alpha * \frac{V_{dW}}{\sqrt{S_{dW} + \epsilon}}$$

$$B = B - \alpha * \frac{V_{dB}}{\sqrt{S_{dB} + \epsilon}}$$

  where, V_{dW}, V_{dB}, S_{dW}, and S_{dB} is given as:

$$S_{dW} = \beta_2 SdW_{prev} + (1 - \beta_2)dW^2 \qquad S_{dB} = \beta_2 S_{dB_{prev}} + (1 - \beta_2)dB^2$$

$$V_{dW} = \beta_1 V_{dW_{prev}} + (1 - \beta_1)dW \qquad V_{dB} = \beta_1 V_{dB_{prev}} + (1 - \beta_1)dB$$

3.3. Fusion of Vision and Sensor Based Approaches

The vision-based approach includes the YOLOv8s model, and the sensor-based approach includes an artificial neural network. Fusion of both approaches is implemented based on decision-level fusion of both models by employing a meta-model. The meta-model, the fusion model, is an artificial neural network.

3.3.1. Model Architecture

To develop an ANN model, it is necessary to have some crucial points to consider. These important points include high accuracy and minimum inference time. When designing an ANN, the important part is to set the right depth of the model. The number of hidden layers sets the depth of an ANN model. Similarly, the computational complexity also depends on these hidden layers. For this project, the ANN model was designed in such a way that the number of hidden units decreased as the depth increased. In this way, computational complexity can be minimised as the input progresses

through ANN layers. The architecture of the Fusion model based on artificial neural network is depicted in Fig. 18.

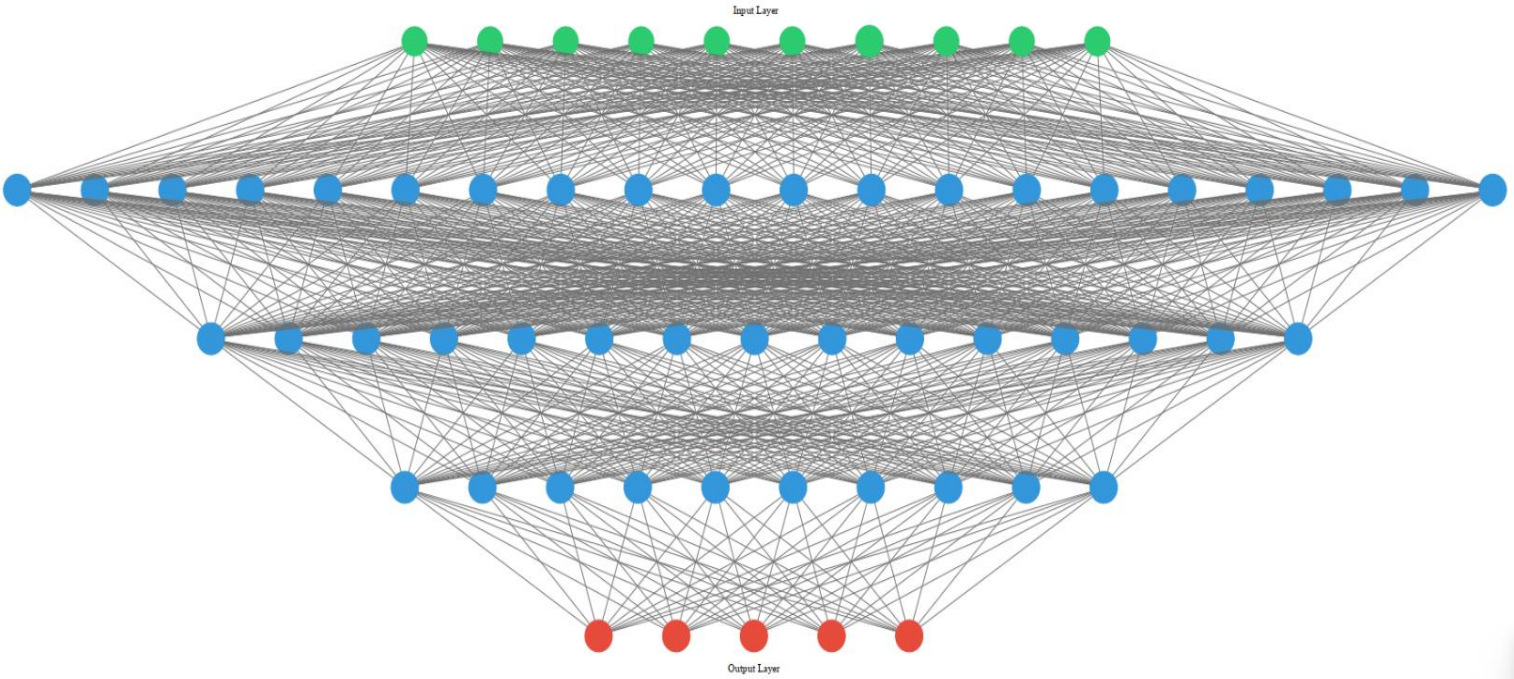


Figure 18. Architecture of model used for fusion

The architecture of the fusion model is explained below.

Input layer: 10 neurons in the input layer corresponding to 10 features in the dataset.

Hidden Layers:

- **First Hidden Layer:** 20 neurons
- **Second Hidden Layer:** 15 neurons
- **Third Hidden Layer:** 10 neurons

Output Layer: 5 neurons (there are 5 classes for gestures)

In order to introduce non-linearity into the model, the hidden layers utilize an activation function known as ReLU (Rectified Linear Unit), whereas the output layer does not employ any activation function. All the layers in the model apply linear transformations

to the incoming data. The mathematical expression of a linear transformation is given below:

$$y=xA^T+b$$

x = input feature vector

y = output feature set

A = weights of every layer

b = bias

Total parameters = 750

3.3.2. Model Training and Hyperparameters

To train this fusion model, a custom dataset was prepared. The dataset contained the output probabilities of YOLOv8s, vision-based approach model, and the output probabilities of ANN, sensor-based approach model. To generate the probabilities given by the camera, the same HaGRID dataset was used. In this case, the images of the classes used to generate the dataset for the fusion model were not used to train the vision-based approach model.

Similarly, the data containing the probabilities given by the sensor model was generated. For this, a sensor glove was worn, and all five hand gestures were performed separately for a set period of time. The data sent by ESP-32 was passed to the sensor-based approach model, and the output probabilities were stored in a CSV file. This process was repeated for all five gestures, and the sensor output probabilities were stored in separate csv files for all five gestures.

The final step was to merge all the data stored in the CSV files into one CSV file. One important thing to ensure is that the first five columns should contain the probabilities given by the vision-based approach model, and the next five columns should contain the probabilities given by the sensor model for a specific label or hand gesture class.

The labels assigned to all five classes in the dataset are shown below:

palm: 0

one: 1

two up: 2

three: 3

fist: 4

	A	B	C	D	E	F	G	H	I	J	K	L
		class1_model1	class2_model1	class3_model1	class4_model1	class5_model1	class1_model2	class2_model2	class3_model2	class4_model2	class5_model2	label
0		0.889	0.02775	0.02775	0.02775	0.02775	0.999	0	0	0	0.001	0
1		0.912	0.022	0.022	0.022	0.022	0.998	0	0	0	0.001	0
2		0.903	0.02425	0.02425	0.02425	0.02425	0.998	0	0	0	0.002	0
3		0.915	0.02125	0.02125	0.02125	0.02125	0.997	0	0	0	0.002	0
4		0.885	0.02875	0.02875	0.02875	0.02875	0.998	0	0	0	0.002	0
5		0.907	0.02325	0.02325	0.02325	0.02325	0.998	0	0	0	0.002	0
6		0.915	0.02125	0.02125	0.02125	0.02125	0.998	0	0	0	0.002	0
7		0.912	0.022	0.022	0.022	0.022	0.999	0	0	0	0.001	0
8		0.921	0.01975	0.01975	0.01975	0.01975	0.998	0	0	0	0.002	0
9		0.926	0.0185	0.0185	0.0185	0.0185	0.998	0	0	0	0.001	0
10		0.897	0.02575	0.02575	0.02575	0.02575	0.998	0	0	0	0.001	0
11		0.916	0.021	0.021	0.021	0.021	0.998	0	0	0	0.001	0
12		0.917	0.02075	0.02075	0.02075	0.02075	0.998	0	0	0	0.002	0
13		0.908	0.023	0.023	0.023	0.023	0.998	0	0	0	0.001	0
14		0.884	0.029	0.029	0.029	0.029	0.998	0	0	0	0.002	0
15		0.911	0.02225	0.02225	0.02225	0.02225	0.997	0	0	0	0.002	0
16		0.908	0.023	0.023	0.023	0.023	0.998	0	0	0	0.001	0
17		0.901	0.02475	0.02475	0.02475	0.02475	0.998	0	0	0	0.001	0
18		0.917	0.02075	0.02075	0.02075	0.02075	0.998	0	0	0	0.002	0
19		0.897	0.02575	0.02575	0.02575	0.02575	0.998	0	0	0	0.001	0
20		0.939	0.01525	0.01525	0.01525	0.01525	0.999	0	0	0	0.001	0

Figure 19. Illustration of data collected for training of fusion model

The fusion model is trained using the cross-entropy loss function.

Other Training Hyperparameters Settings:

- Number of Epochs = 150
- Batch Size = 1028
- Initial Learning Rate = 0.001
- Optimizer = Adam

3.4. Algorithm for Executing Mouse Commands

After training the camera model, YOLOv8s, sensor model, and fusion model, the next step is to make an algorithm that can predict dynamic hand gestures and execute mouse

commands corresponding to the gesture detected. A total of six mouse commands needs to be executed using a mixture of dynamic and static hand gestures. The mouse commands to be executed are listed below:

1. Double Click
2. Left Click
3. Cursor Movement
4. Right Click
5. Scroll Down
6. Scroll Up

The types of hand gestures associated with every mouse command are shown below:

Cursor Movement: static hand gesture; hand gesture class **one**.

Left Click: Dynamic Hand Gesture; Gesture starts with Hand Gesture class **one** and ends with Hand Gesture class **fist**.

Right Click: Dynamic Hand Gesture; Gesture starts with Hand Gesture Class **two up** and ends with Hand Gesture Class **fist**.

Double Click: No specific hand gesture is associated. Perform left-click gestures twice in a short period of time.

Scroll Up: Static Hand Gesture; Hand Gesture class **fist**.

Scroll Down: Static Hand Gesture; Hand Gesture class **fist**.

The above algorithm is illustrated in the flowchart shown in Fig. 20.

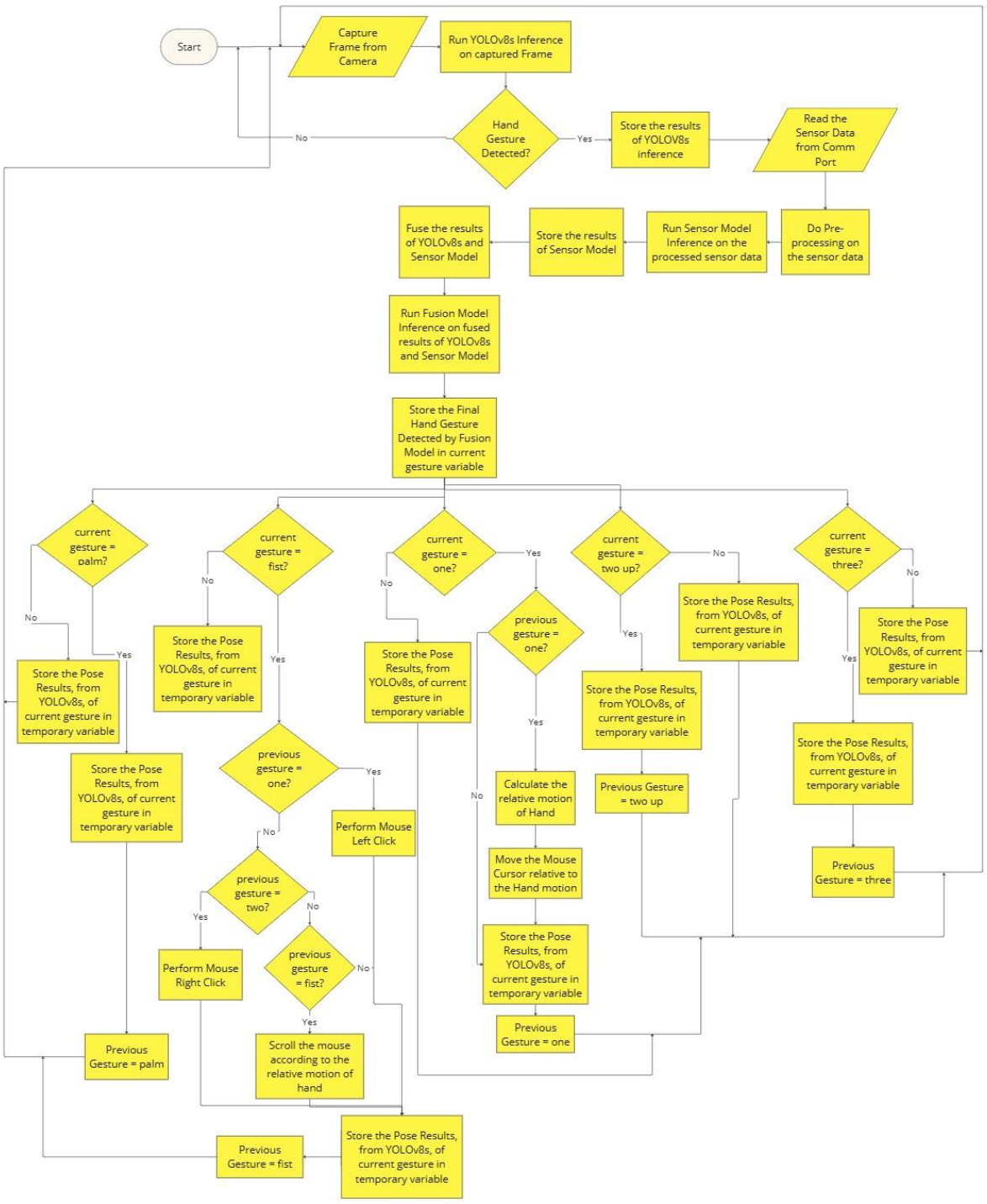


Figure 20. Flowchart of algorithm developed for gesture translation into mouse functionalities

The algorithm starts by reading a frame from the camera input video stream. The frame read is passed to the camera model for prediction. If the camera model predicts a hand gesture class in the image, then the algorithm moves to the sensor model. It reads the

data from the communication port, processes it, performs normalisation, and passes it to the sensor model for the sensor model's prediction. After receiving sensor model probabilities, the algorithm combines both camera and sensor models probabilities for a final prediction using the fusion model. After obtaining the fusion model prediction, it monitors the classes predicted by the fusion model for the purpose of recognizing dynamic hand gestures and executing mouse commands. Now, how this algorithm executes mouse commands is shown separately for every mouse command.

3.4.1. Translating Gestures into Mouse Functionalities

3.4.1.1. Executing Cursor Movement

Cursor movement depends on static hand gesture class **one.** When class **one** is initially detected by the fusion model, read the pose key points of the hand given by the YOLOv8s model and store them in the initial coordinates variable. Now, move to the next frame, and if class **one** is detected again, store the pose key points of the hand in the final coordinates variable. Now, determine the relative movement of the reference key point and adjust the cursor according to the relative motion of the hand key point. The hand key point set as reference is the WRIST key point given by Mediapipe hand landmarks. The threshold for minimum relative motion detected is set to 5.

3.4.1.2. Executing Left Click

Left click depends on a dynamic hand gesture. The dynamic hand gesture starts with class **one** and ends with class **fist**. This dynamic hand gesture is detected by first detecting class **one** and then detecting class **fist** in the very next frame. The hand's motion should have a high velocity relative to the overall execution duration of the algorithm. Finally, one last check is that when the gesture ends, that is, on class fist, the index fingertip key point should be close to the thumb_IP key point. Left click is performed if the threshold set for accurate detection is crossed by the Euclidean distance calculated. The threshold is set to 20.

3.4.1.3. Executing Right Click

Right click depends on a dynamic hand gesture. The dynamic hand gesture starts with class **two up** and ends with class **fist**. This dynamic hand gesture is detected by first

detecting class **two up** and then class **fist** in the very next frame. The hand's motion should be rapid in comparison to the overall execution time of the algorithm. Finally, one last check is that when the gesture ends, that is, on class fist, the middle fingerdip key point should be close to the thumb tip key point. The right click is triggered when the Euclidean distance is below a certain threshold for precise detection. The threshold is set to 20.

3.4.1.4. Executing Double Click

There is no specific hand gesture associated with the double-click command. The cause for the execution of a double click is the rapid execution of two left clicks in close succession. To conduct a double click, simply repeat the actions associated with the left click twice in rapid succession.

3.4.1.5. Executing Scroll Up

Scroll Up depends on static hand gesture class **fist** . When initially class fist is detected by the fusion model, read the pose key points of the hand given by the YOLOv8s model and store them in the initial coordinates variable. Proceed to the subsequent frame. If the hand is once again recognized as being in a fist position, save the positional key points of the hand in the final coordinates variable. Now, determine the relative movement between the reference key point and scroll upwards if vertical hand movement is identified in the image based on the relative motion of the hand key point.The hand key point set as reference is the WRIST key point given by Mediapipe hand landmarks. The threshold for minimum relative motion detected is set to 5.

3.4.1.6. Executing Scroll Down

Scroll down depends on static hand gesture class **fist**. When the fist class is initially detected by the fusion model, read the pose key points of the hand given by the YOLOv8s model and store them in the initial coordinates variable. Proceed to the subsequent frame. If the hand is once again recognized as being in a fist position, save the positional key points of the hand in the final coordinates variable. Now, determine the relative movement between the reference key point and scroll upwards if the hand's motion is identified as moving vertically downwards in the image, based on the relative

motion of the hand key point. The hand key point set as reference is the WRIST key point given by Mediapipe hand landmarks. The threshold for minimum relative motion detected is set to 5.

Chapter 4 – RESULTS

4.1. Evaluation Criteria of Vision-Based Approach Model

The YOLOv8s model, which is a vision-based technique, functions as an object detector. Its capability is assessed by different metrics such as F-1 score, mAP50, mAP50-95, precision, and recall. The measurements are displayed relative to the confidence threshold. Precision gives a measure of true positives that were actually correct. Recall quantifies the ratio of true positive predictions by the model on the entire collection of positive instances in the data. Combining both, or taking the harmonic mean, results in an F-1 score. The F-1 score gives the complete assessment of the model. The mean average precision 50 describes the performance of the model at a threshold of 0.5. This is also known as the model's capability to detect easy detections. While the mean average precision 50-95 measures the model's performance at multiple difficulty levels of detection. Moreover, several training and validation losses are also monitored.

4.1.1. Training Results

Results are shown separately for pose estimation and object detection.

4.1.1.1. Results of Object Detection

The Precision-Confidence curve of YOLOv8s for bounding box is shown in Fig. 21.

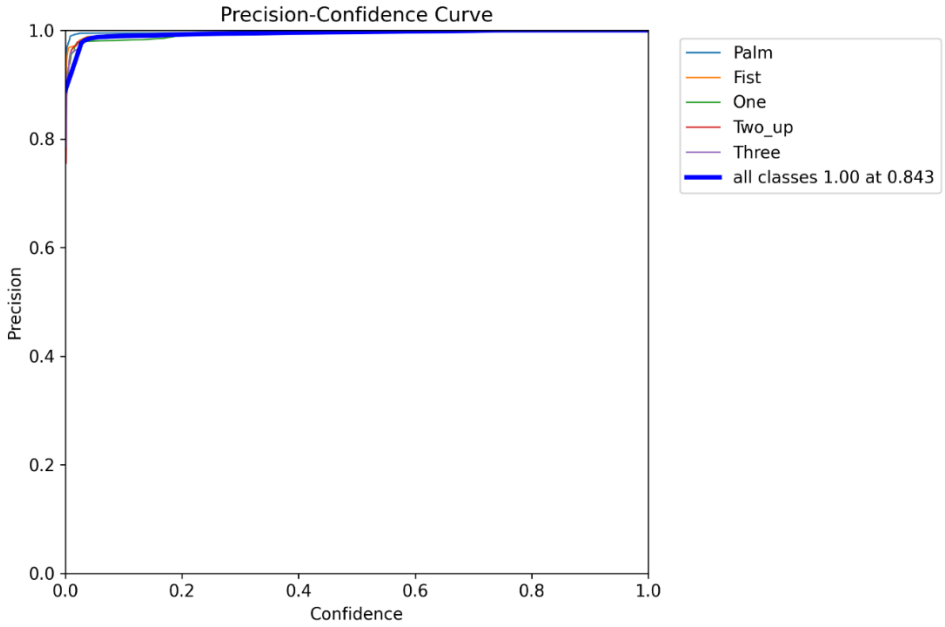


Figure 21. Precision-Confidence curve of YOLOv8 object detection

The Precision-Confidence Curve is crucial for optimizing the performance of YOLOv8 object detection model. By plotting precision against varying confidence thresholds for different classes, this curve allows the selection of an optimal confidence threshold that maximizes precision while minimizing false positives. Determining the ideal confidence threshold enables you to achieve a harmonious equilibrium between precise detections (with a high level of accuracy) and catching a maximum number of pertinent items (with a high level of recall).

Analysis:

Each coloured line in the Fig. 21 represents a specific class (Palm, Fist, One, Two_up, Three), showing how precision changes with varying confidence thresholds. The precision for each class fluctuates as the confidence threshold changes, indicating how certain the model is in its predictions for different classes. The blue line represents the combined precision performance for all classes, giving a summary of the model's overall accuracy. The mean precision for all categories is 1.00 when the confidence threshold is set at 0.843, which is the high level of confidence in its classification predictions. This is especially crucial in this context, as incorrect good results can lead to substantial repercussions.

The Recall-Confidence curve of the model for bounding box is shown in Fig. 22.

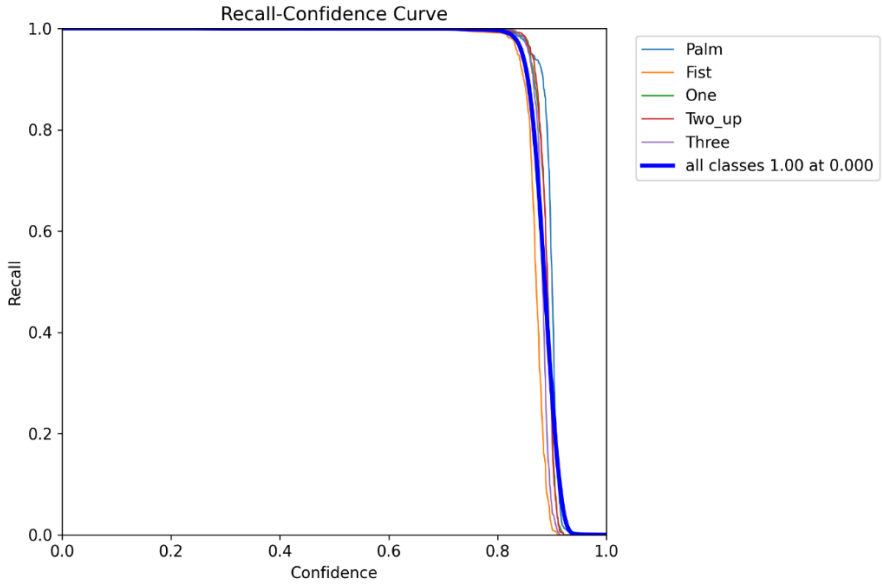


Figure 22. Recall-Confidence curve of YOLOv8 object detection

The curve shows the relationship between confidence levels and recall for different classes. It indirectly shows the trade-off with precision because as the threshold

decreases (allowing lower confidence predictions), recall typically increases, but precision may decrease due to more false positives. A model that maintains high recall across a wide range of confidence thresholds is generally more robust. The curve can help identify an optimal confidence threshold where the model achieves a desirable balance of high recall and acceptable precision. This is particularly useful for tuning the model for specific applications.

Analysis:

The curve in the Fig. 22 indicates that the model achieves high recall at lower confidence thresholds, meaning it can identify most of the actual positives when the confidence threshold is low. When confidence threshold rises, recall drops significantly. This indicates that the model becomes increasingly cautious in its outputs, resulting in a reduction in false positives. However, it may also result in the possibility of missing some true positives. The average recall for all classes is 1.00 at a confidence threshold of 0.00. This is not surprising as recall is maximum when threshold is set to zero, because recall quantifies the number of positives the model detects, no matter if they are correct. Although, recall is maximum at 0.0 threshold, it should not be considered as the optimal value as it will lead to fatal consequences where almost every prediction made by the model will be incorrect. To find a proper threshold, select the value just before sharp roll-off starts. From the figure, it can be seen that the recall remains at 1.00 until approximately 0.8 confidence threshold.

The Precision-Recall curve is shown in Fig. 23.

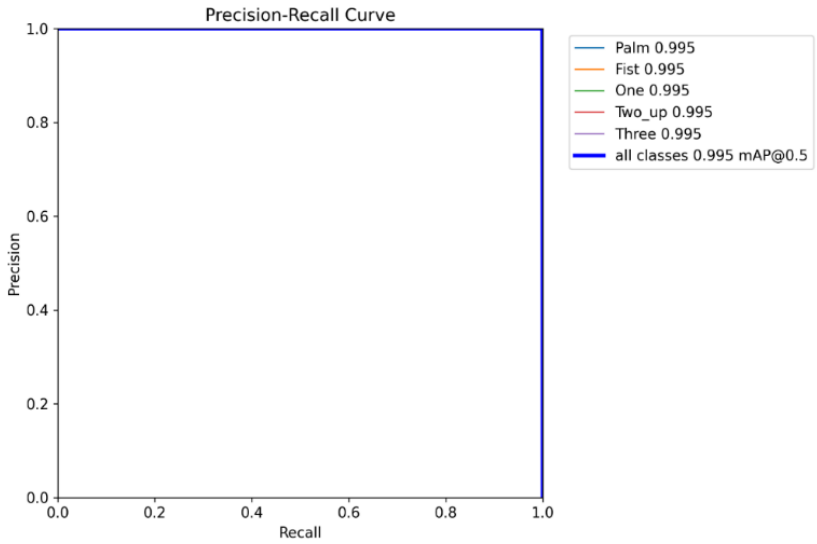


Figure 23. Precision-Recall curve of YOLOv8 object detection

The Precision-Recall curve helps evaluate and improve machine learning models, particularly in situations when there is an imbalance between the classes. This demonstrates the compromise between accuracy and completeness. For instance, increasing recall may lower precision, and vice versa. This curve helps in understanding how changes in one metric affect the other. Through the examination of this curve, individuals can choose the ideal limit for the model's outputs in order to get the desired equilibrium between precision and recall.

Analysis:

The Fig. 23 shows that each class maintains a precision of 0.995, reflecting the model's ability to accurately detect hand gesture classes with minimal false positives. The curves indicate that the model performs consistently well across different classes, maintaining high precision even as recall increases. The model's high mAP@50 score of 0.995 at an IoU threshold of 0.5 depicts that it is well-optimized for object detection tasks. This demonstrates that the model effectively balances accuracy and recall.

F1-Confidence curve of the object detector is shown in Fig. 24.

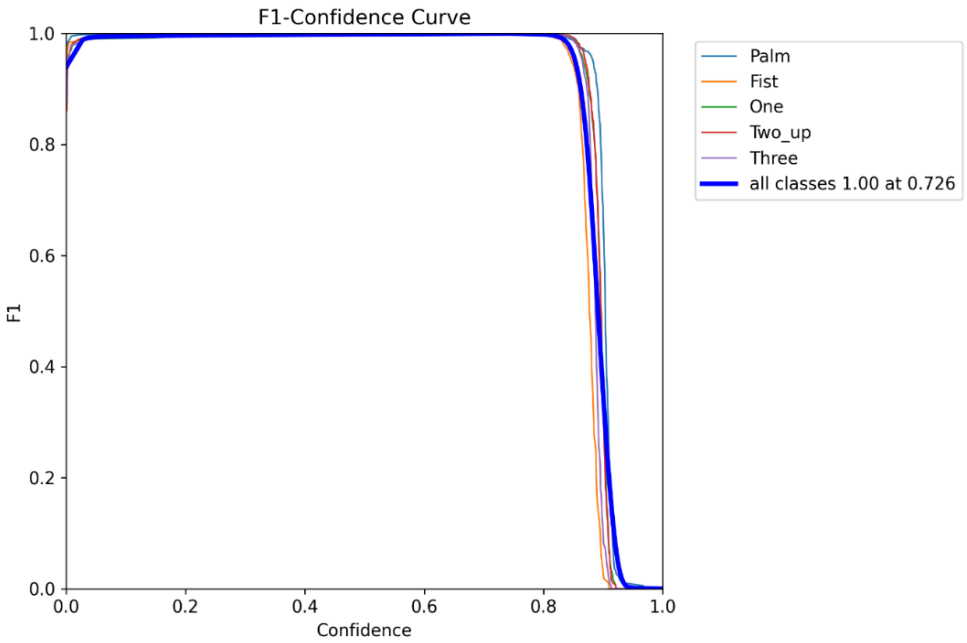


Figure 24. F1-Confidence Curve of YOLOv8 object detection

The F1-Confidence Curve in object detection, particularly in models like YOLOv8, serves as a valuable metric for understanding the trade-off between detection accuracy and confidence thresholding. The F1 score is a quantitative measure that integrates recall and precision and is defined as the harmonic average of recall (R) and precision (P). By examining the curve, one can determine the optimal confidence threshold that maximizes both precision and recall or strikes a balance based on specific application requirements. Lowering the confidence threshold increases the number of detections (higher recall) but may decrease precision. Conversely, raising the threshold reduces false positives (higher precision) but may miss some detections (lower recall). This guides adjustments in model parameters or training strategies to improve both precision and recall across different confidence levels.

Analysis:

The curve in Fig. 24 shows that when the model has a high confidence threshold (close to 1.0), it achieves a near-perfect F1 score of 1.00. This indicates that the model's detections are highly accurate when it is very confident about its predictions. Different hand gestures (classes) are represented by separate curves on the plot. For example, the "Fist" class curve shows a slightly lower F1 score compared to other classes at higher confidence levels. This suggests that detecting the "Fist" gesture may be more challenging for the model. As the confidence threshold increases, F1 score drops significantly. This illustrates that the model gets progressively cautious in its forecasts, resulting in a reduction of false positives. However, there is also a possibility of missing some true positives. The curve shows that the optimal confidence threshold for achieving the highest F1 score across all classes is around 0.726, where the F1 score is 1.00. The average F1 score curve shows that the model maintains a relatively stable performance across different confidence levels. This suggests that the model is consistent in its predictions across various thresholds, albeit with a decrease in accuracy at higher confidence levels.

Mean Average Precision 50 curve for bounding box is depicted in Fig. 25.

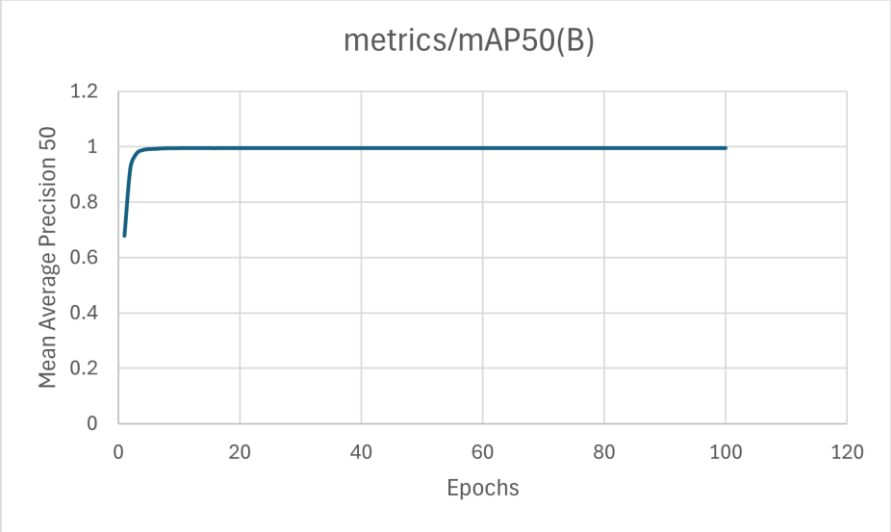


Figure 25. mAP50 curve of YOLOv8 object detection

In object detection, evaluating the effectiveness of a model like YOLOv8 involves several metrics, with Mean Average Precision at 50% Intersection over Union (mAP@50) being a key standard. This metric offers a lucid and perceptive assessment of the model's capacity to precisely identify and pinpoint items in images. mAP@50 measures the average precision of the model's object detections at IoU threshold of 50%. IoU, or Intersection over Union, is a quantitative measure utilized to assess the amount of intersection between the ground truth bounding box and the predicted truth bounding box. mAP is the mean of the APs across all object classes. For mAP@50, it averages the AP values at the IoU threshold of 50%. AP is computed as the area under the precision-recall curve for each object class at the 50% IoU threshold. mAP@50 serves as a feedback metric for tuning hyperparameters (like learning rates, batch sizes) and adjusting model configurations (like network depth, anchor sizes). Improvements in mAP@50 suggest better object detection capabilities.

Analysis:

The graph in Figure 25 illustrates a sharp rise in mAP50 during the early stages, suggesting that the model rapidly acquires the ability to reliably detect and locate hand classes. After the initial increase, the curve plateaus, indicating that the model's performance stabilizes, and further training epochs result in marginal improvements. The curve approaches a value close to 1.0, suggesting that the model achieves near-perfect precision and recall at the IoU threshold of 0.50.

Mean Average Precision 50-95 curve for bounding box is shown in Fig. 26.

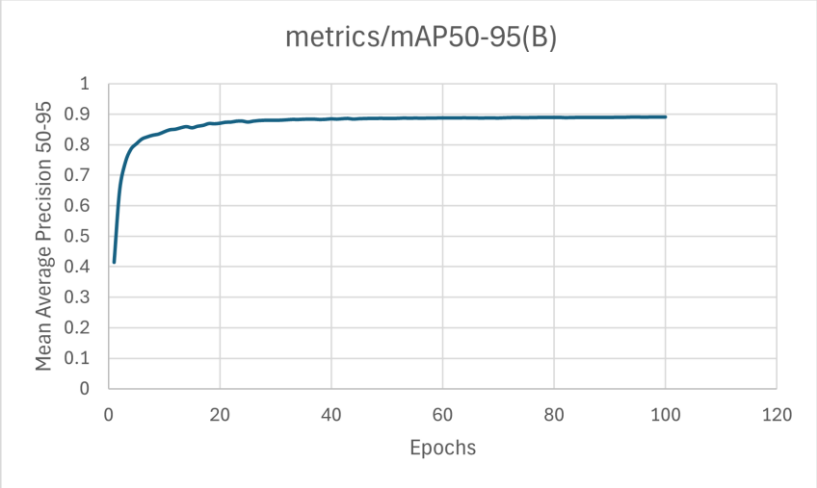


Figure 26. mAP50-95 curve of YOLOv8 object detection

Each mAP score at different IoU thresholds provides insight into how well the model can detect objects with varying levels of overlap with ground truth annotations. Monitoring mAP@50-95 during training helps in assessing the model's progress and guides optimization efforts such as adjusting learning rates, augmenting data, or modifying the model architecture to improve performance at specific IoU thresholds. The utilization of several IoU criteria in mAP50-95 offers a better resilient evaluation of the model's capability in contrast to the utilization of a single IoU threshold. It ensures that the model is not only good at detecting objects but also at accurately localizing them.

Analysis:

The graph in Fig. 26 demonstrates a sharp rise in mAP50-95 during the early stages, suggesting that the model rapidly acquires the ability to reliably detect and locate hand motions. After the initial increase, the curve plateaus, indicating that the model's performance stabilizes, and further training epochs result in marginal improvements. The curve approaches a value close to 0.9, suggesting that the model achieves high recall and precision across a range of IoU thresholds.

The Confusion Matrix of YOLOv8s model is shown in Fig. 27.

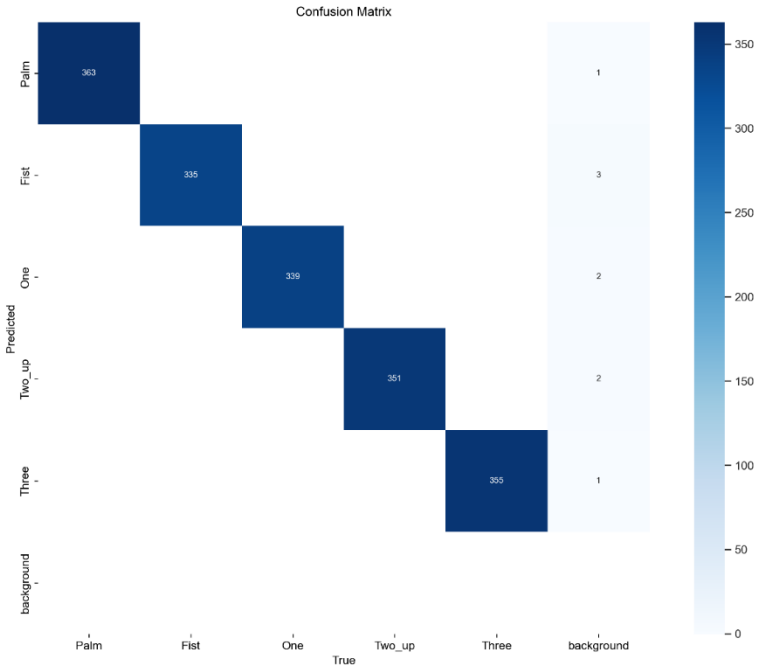


Figure 27. Confusion matrix of YOLOv8 object detection

The confusion matrix is a vital tool in evaluating the capability of object detection models. It offers a comprehensive breakdown of predictions vs ground truth for many classes, enabling a detailed assessment of the model's working. The primary purpose of the confusion matrix is to evaluate how well an object detection model performs in terms of predicting different classes of objects within images. Understanding where the model frequently misclassifies or misses detections (false positives and false negatives) guides improvements in model training, data augmentation, or model architecture.

Analysis:

Overall, the confusion matrix, shown in the Fig. 27, indicates that the model performs relatively well, with a high number of correct classifications along the diagonal. There are some instances of misclassifications, but they are relatively small compared to the number of correct predictions.

$$\text{Test Accuracy} = \frac{363+355+339+351+355}{363+355+339+351+355+1+2+2+2+3+1}$$

Test Accuracy = **99.49%**

The confusion matrix shows that the trained model has a high rate of accurately identifying negative and positive instances, with a negligible number of incorrect positive and negative detections.

4.1.1.2. Results of Pose Estimation

The Precision-Confidence curve for pose estimation is shown in Fig. 28.

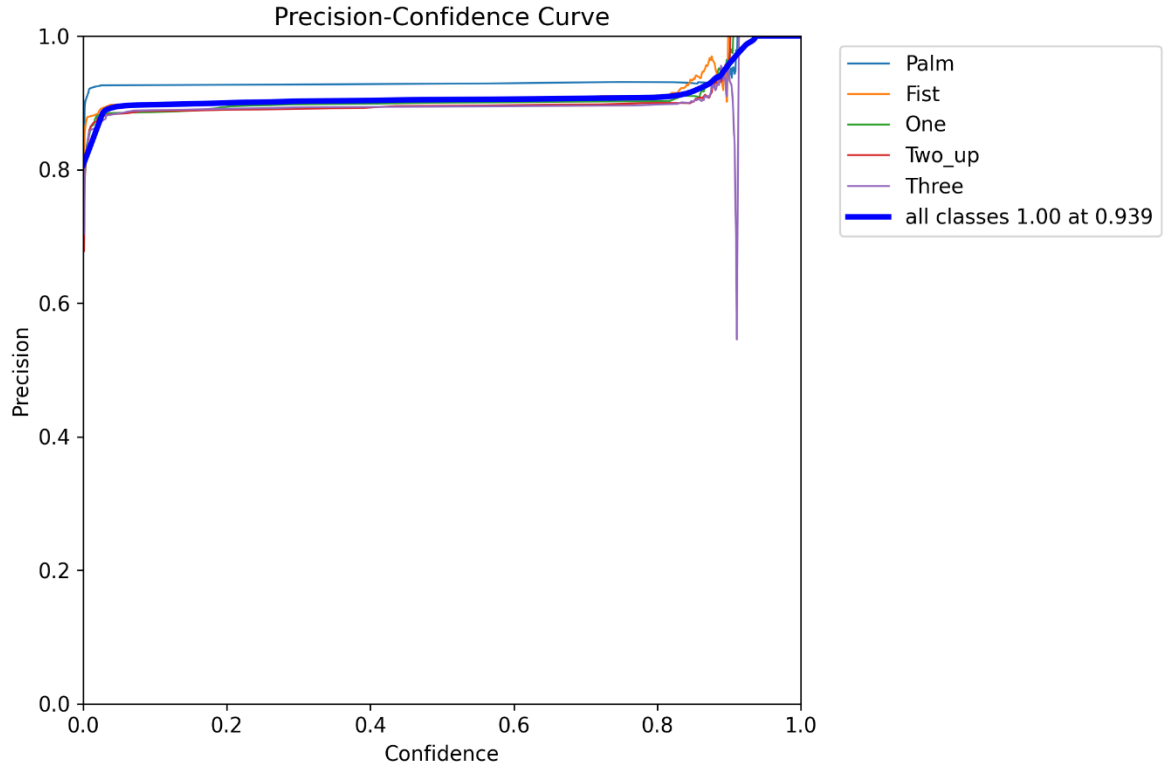


Figure 28. Precision-Confidence curve of YOLOv8 pose estimation

The Precision-Confidence curve serves as an essential tool to understand how the model's precision varies with different confidence thresholds. Pose estimation involves predicting key points (like joints in human body poses) with confidence scores. The precision-confidence curve helps in understanding how precision varies as the threshold for confidence scores is adjusted. High-confidence thresholds typically result in fewer, but more accurate key point predictions. Lowering the threshold may increase the number of detected key points but could include more incorrect predictions. The curve illustrates how well the model maintains precision as the confidence level changes.

Analysis:

The Fig. 28 shows that the model demonstrates high precision across all classes, with precision values close to 1.0 for most confidence levels. This indicates that the model is highly accurate in its predictions, with a low rate of false positives. There is a noticeable sharp increase in precision as the confidence level approaches 1.0. This suggests that at higher confidence thresholds, the model becomes more conservative, making fewer but more accurate predictions. The curve shows that the optimal

confidence threshold for achieving the highest precision across all classes is around 0.939, where the precision is 1.00. The precision for individual classes (Palm, Fist, One, Two_up, Three) follows a similar trend, with high precision values across different confidence levels. This consistency across classes indicates that the model performs well in detecting and classifying each pose based on the 21 key points.

The Recall-Confidence curve for pose estimation is shown in Fig. 29.

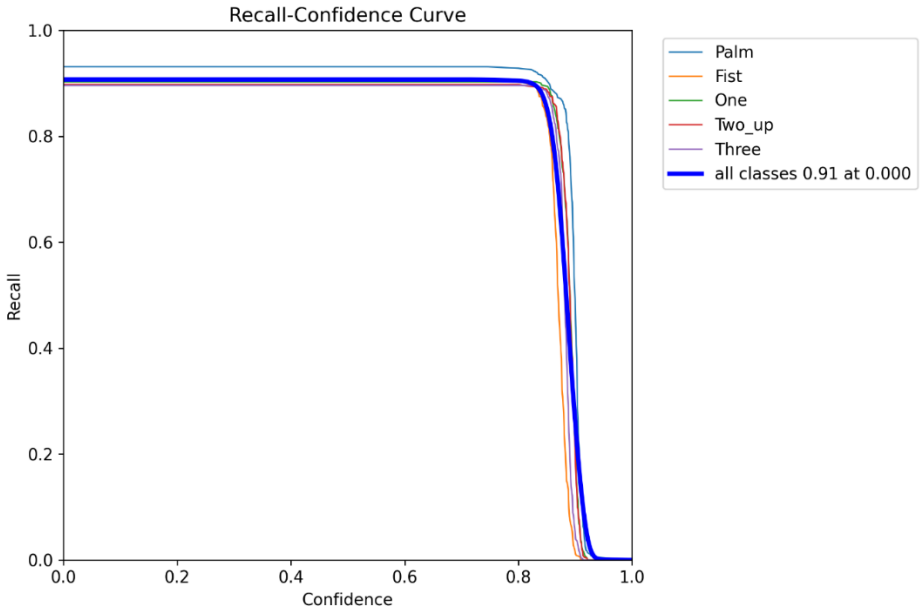


Figure 29. Recall-Confidence curve of YOLOv8 pose estimation

The Recall-Confidence curve is an important tool for evaluating the model's performance to detect and identify key points (such as joints in human body poses) at varying levels of confidence. High recall means that the model can detect a large proportion of actual key points. The curve helps determine the confidence level at which the model maintains a high recall, ensuring that the model captures as many true key points as possible. This is crucial for applications that require comprehensive detection of key points. Although the recall-confidence curve focuses on recall, it helps understand the implicit trade-off with precision. Lowering the confidence threshold generally increases recall but may decrease precision due to more false positives. This insight is crucial for balancing the detection accuracy and the number of detected key points.

Analysis:

The curve in the Fig. 29 shows that the model demonstrates high recall across all classes, with recall values close to 1.0 for most confidence levels. This suggests that the model is highly accurate in detecting real positive cases, while also having a low rate of incorrectly classifying cases as false negatives. There is a noticeable sharp drop-off in recall as the confidence level approaches 1.0. This suggests that at higher confidence thresholds, the model becomes more conservative, making fewer predictions and potentially missing some true positives. The curve shows that the optimal confidence threshold for achieving the highest recall across all classes is around 0.91, where the recall is 1.00. The recall for individual classes (Palm, Fist, One, Two_up, Three) follows a similar trend, with high recall values across different confidence levels. This consistency across classes indicates that the model performs well in detecting each pose based on the 21 key points.

Fig. 30 illustrates the Precision-Recall curve for pose estimation.

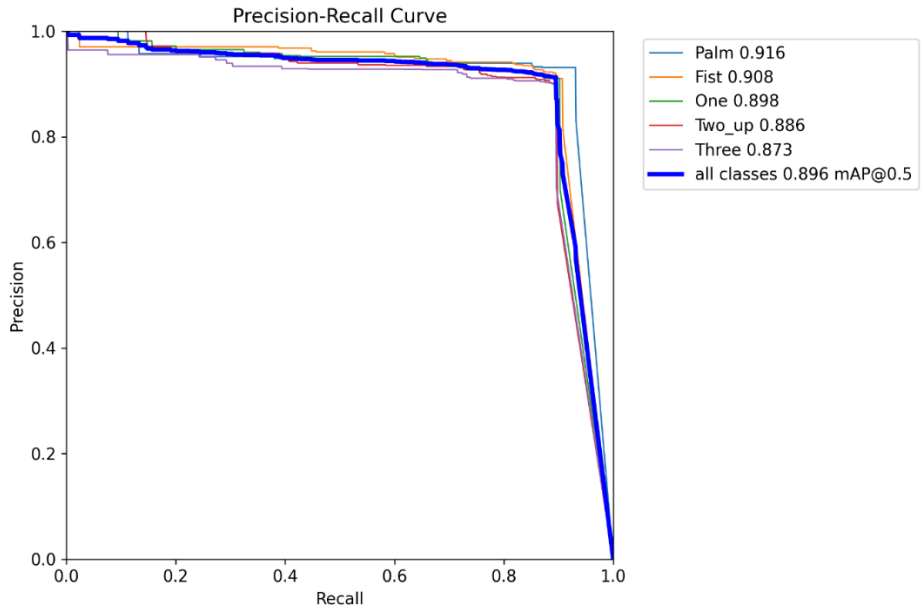


Figure 30. Precision-Recall curve of YOLOv8 pose estimation

The Precision-Recall (PR) curve is a fundamental tool for assessing the performance of pose estimation models like YOLOv8. In pose estimation, precision and recall are often inversely related. In pose estimation, certain key points (e.g., joints obscured by clothing) might be less frequently detected than others. The PR curve is particularly informative in such cases because it emphasizes true positive rate (recall) and the

precision of detections, providing a clearer picture of model performance even when there's a class imbalance.

Analysis:

From Fig. 30, it can be inferred that the model demonstrates high precision and recall across all classes, with values close to 1.0 for most of the curve. This indicates that the model is both accurate (high precision) and comprehensive (high recall) in its predictions. There is a noticeable sharp drop-off in precision as recall approaches 1.0. This suggests that as the model tries to capture all true positives (high recall), it starts to include more false positives, reducing precision. Palm attains the best performance with the accuracy of 0.916. Fist follows closely with a precision of 0.908. One has a precision of 0.898. Two_up shows a precision of 0.886. Three has the lowest precision among the classes at 0.873. The consistency across classes indicates that the model has high performance in detecting each pose based on the 21 key points. The mAP, which is calculated at the hreshold of 0.5 for all classes, is 0.896. This aggregated metric indicates that the model has high performance overall, balancing both recall and precision across all classes.

Fig. 31 depicts the F1-Confidence curve for pose estimation.

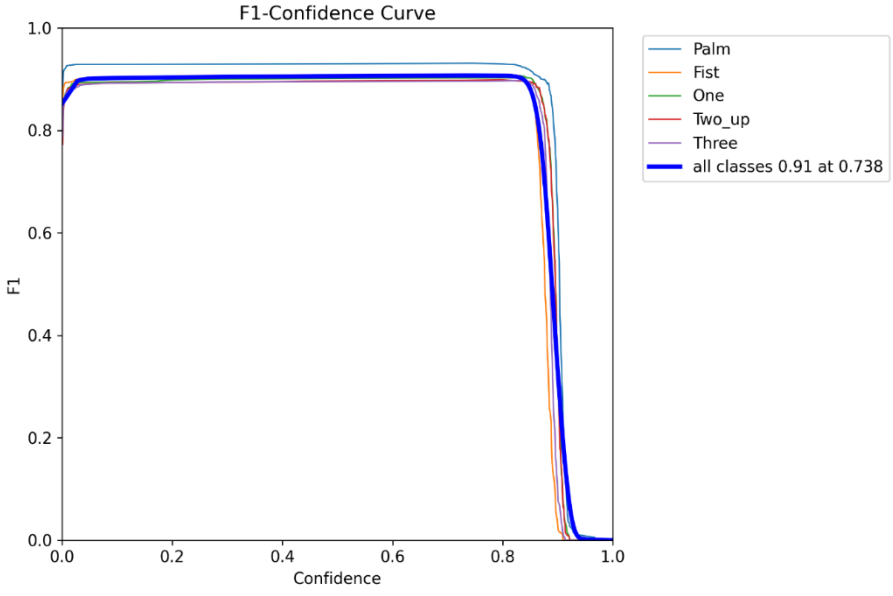


Figure 31. F1-Confidence curve of YOLOv8 pose estimation

In the context of pose estimation with YOLOv8, the F1-Confidence curve is an insightful evaluation tool that helps in understanding the overall model's performance

by combining both recall and precision into a single metric, the F1 score. The graph illustrates the model's responsiveness to the confidence threshold employed for key point predictions, aiding in the comprehension of the confidence levels at which the model attains optimal trade-off between recall and precision. The highest point of the curve of F1-confidence corresponds to the confidence threshold at which the model achieves the best trade-off between precision and recall. The shape of the curve provides insights into how well the model balances precision and recall. A sharp decline on either side of the peak can indicate that the model heavily favours one metric over the other as the threshold changes.

Analysis:

The graph in Fig. 31 demonstrates that the model attains elevated F1 scores when the confidence thresholds are set to lower values. This implies that the model exhibits strong performance in terms of both precision and recall when the confidence threshold is low. As the confidence threshold increases, the F1 score experiences a significant decrease, indicating that the model becomes more cautious in its predictions. This results in a reduction in false positives but also a potential increase in missed real positives. The curve shows that the optimal confidence threshold for achieving the highest F1 score across all classes is around 0.738, where the F1 score is 0.91. Although, the confidence threshold is low compared to when the model measures recall and precision with respect to confidence threshold, it still is more than satisfactory value.
Mean Average Precision (mAP50) for pose estimation is shown Fig. 32.

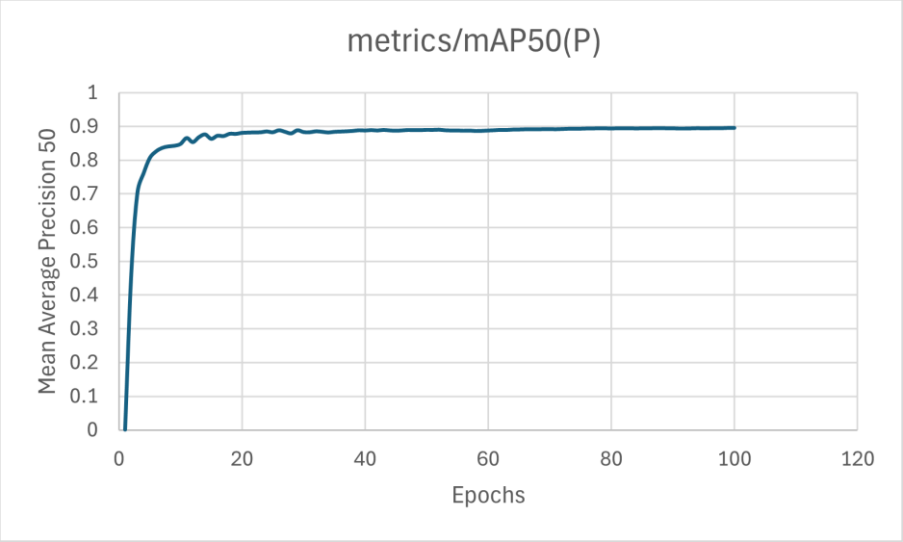


Figure 32. mAP50 curve of YOLOv8 pose estimation

In pose estimation, evaluating the accuracy and robustness of a model like YOLOv8 involves various metrics, with the Mean Average Precision at 50% Intersection over Union (mAP@50) being a prominent one. This metric provides a comprehensive measure of how well the model detects and localizes key points (such as body joints) across the dataset, particularly during the training process. mAP@50 measures the precision of the model's key point predictions when the overlap between the ground truth and predicted key points exceeds 50%. Key points can be evaluated using Intersection over Union (IoU), which measures the proximity of predicted key points to the ground truth key points within a specified area, typically a bounding box. mAP is a metric that calculates the average of the Average Precision values for all important points and classes in a dataset.

Analysis:

The curve in Fig. 32 depicts a steep increase in mAP50 during the initial epochs (approximately the first 10 epochs). This indicates that the model quickly learns to detect and localize poses accurately in the early stages of training. After the initial rapid increase, the curve plateaus, indicating that the model's performance stabilizes. This plateau phase starts around the 10th epoch and continues through the remaining epochs up to 100. The mAP50 value stabilizes at around 0.9, suggesting that the model achieves high precision and recall at the IoU threshold of 0.50. A high mAP50 value (close to 0.9) indicates that the model performs well in detecting and localizing key points, making it suitable for hand pose estimation.

Mean Average Precision (mAP50-95) for pose estimation model is shown in Fig. 33.

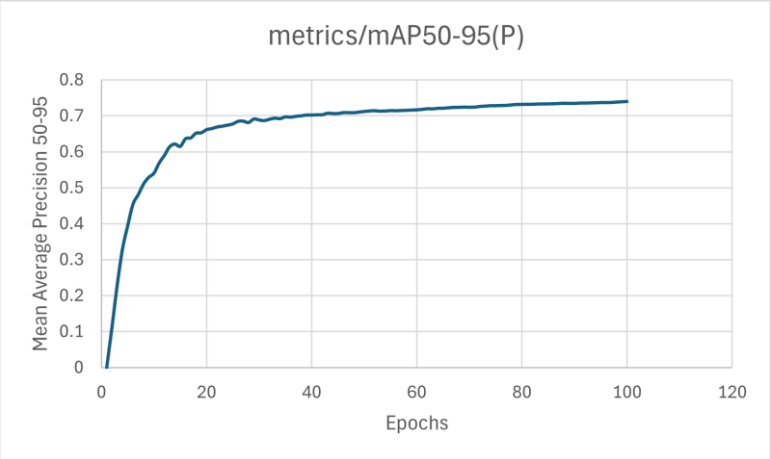


Figure 33. mAP50-95 curve of YOLOv8 pose estimation

The Mean Average Precision (mAP) at IoU thresholds ranging from 50% to 95% is a comprehensive metric that evaluates the model's performance across varying levels of detection stringency. This metric, often referred to as mAP@50-95, is particularly useful in assessing the robustness and precision of key point detection and localization in a nuanced manner. The mAP@50-95 metric calculates the average precision across multiple IoU thresholds from 50% to 95%, typically in steps of 5% (i.e., 50%, 55%, 60%, ..., 95%). This curve shows the model's performance on various difficult hand pose estimation, and it shows that it is detecting all key points of all the classes with a satisfactory 0.75 mAP50-95. This helps in understanding how well the model generalizes to different levels of detection precision.

Analysis:

The curve in Fig. 33 highlights a steep increase in mAP50-95 during the initial epochs (approximately the first 20 epochs). This indicates that the model quickly learns to detect and localize poses accurately in the early stages of training. After the initial rapid increase, the curve continues to rise but at a slower rate, indicating that the model's performance is still improving but at a diminishing rate. This phase continues until around the 80th epoch. The curve begins to plateau after the 80th epoch, suggesting that the model's performance stabilizes, and further training epochs result in marginal improvements. The mAP50-95 value reaches a plateau of approximately 0.75, suggesting that the model attains a favourable equilibrium between precision and recall across various IoU thresholds.

4.2. Evaluation Criteria of Sensor-Based Approach Model

The data was divided into three segments to train the model. 60% of the whole dataset is used for training. During the training process, 20% of the data was set aside as validation data to evaluate the model's performance. Additionally, another 20% of the data was utilized as test data to assess how well the model performs on unknown data. The model underwent 50 epochs of training utilizing the aforementioned architecture and hyperparameters.

4.2.1. Training Results

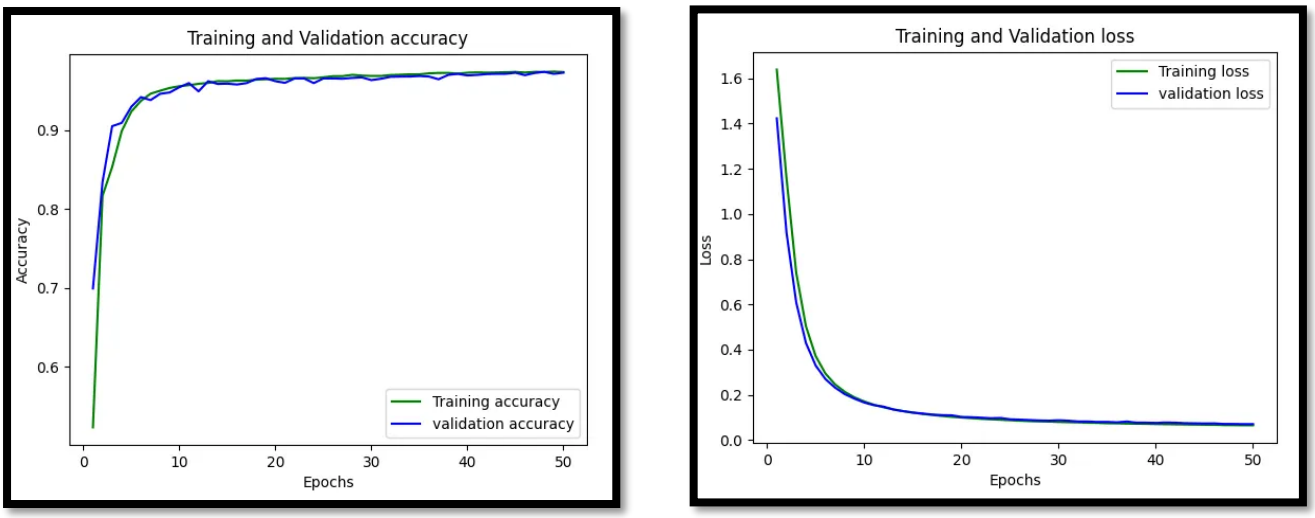


Figure 34. Performance of sensor model on validation data

Both of the above figures explain the model's training results based on the provided data. The right figure shows a reduction in losses with each epoch. At the end of the 50 epochs, both validation and training losses dropped below 0.1. On the other hand, the figure on the right side demonstrates an increase in validation and training accuracy to 98% after 50 epochs.

Finally, the model was tested with unknown data, and once again, its accuracy was close to 98%. These results show the precision of the sensor model.

4.2.2. Real Time Testing

The trained model mentioned earlier was saved in a singular file. The model was imported for real-time testing. Once all the necessary conditions for real-time testing were met, such as wearing the glove and establishing a Bluetooth connection with the PC, the testing was conducted using real-time data. The model produced the expected results. When the gesture changes in real time, the model predicts the corresponding change.

4.3. Evaluation Criteria of Fusion Model

The fusion model is assessed using three assessment criteria: training and validation loss, training and validation accuracy, and test outcomes presented as a confusion matrix.

4.3.1. Training Results

The validation and training losses of the fusion model are shown in Fig. 35.

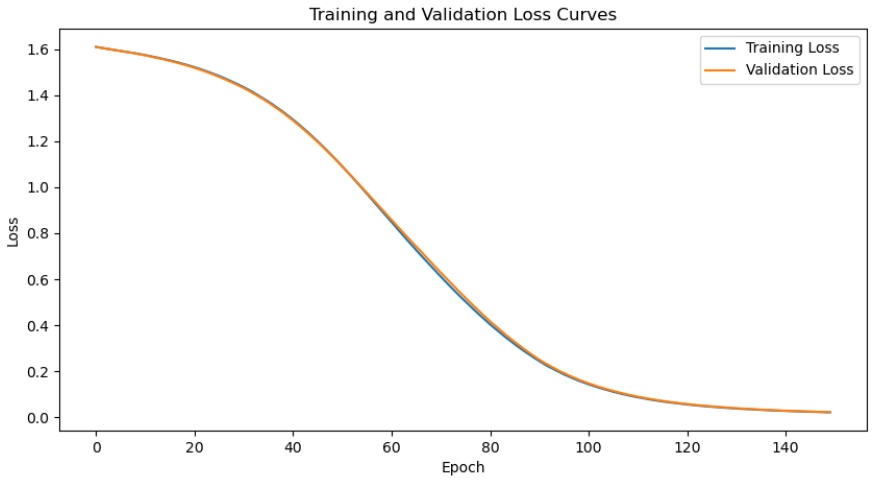


Figure 35. Validation and training losses of fusion model

Monitoring training and validation losses, often visualized as curves during the training of a model, serves several important purposes in machine learning and deep learning tasks. Loss curves track how well the model is learning during training. The objective is to monitor if the loss consistently reduces or levels out, indicating whether the model is approaching an ideal solution. A diminishing training loss shows that the model is acquiring knowledge from the data. The validation loss, computed on a distinct validation set, aids in assessing the model's performance to generalize to unseen and new data. An increasing disparity between the validation and training loss may suggest overfitting, which refers to a case where the model fails to classify unseen data and performs well on the training dataset. Loss curves are useful for finding the optimal stopping point during training in order to prevent overfitting. Loss curves provide insights into the impact of hyperparameters (e.g., learning rate, batch size) on model performance.

Analysis:

The graph in Fig. 35 shows a typical pattern observed in deep learning model training. The model learns to reduce the loss on the training data with each epoch. The blue line represents the training loss. It starts at a relatively high value (around 1.6) and decreases steadily as the number of epochs increases. The orange line represents the validation loss. It also starts at a high value and decreases as the number of epochs increase. It follows a similar trend to the training loss but remains slightly higher. Both the training and validation losses converge to a low value around 0.1, indicating that the model has learned a good representation of the data and is not overfitting.

The validation and training accuracy of the fusion model are shown in Fig. 36.

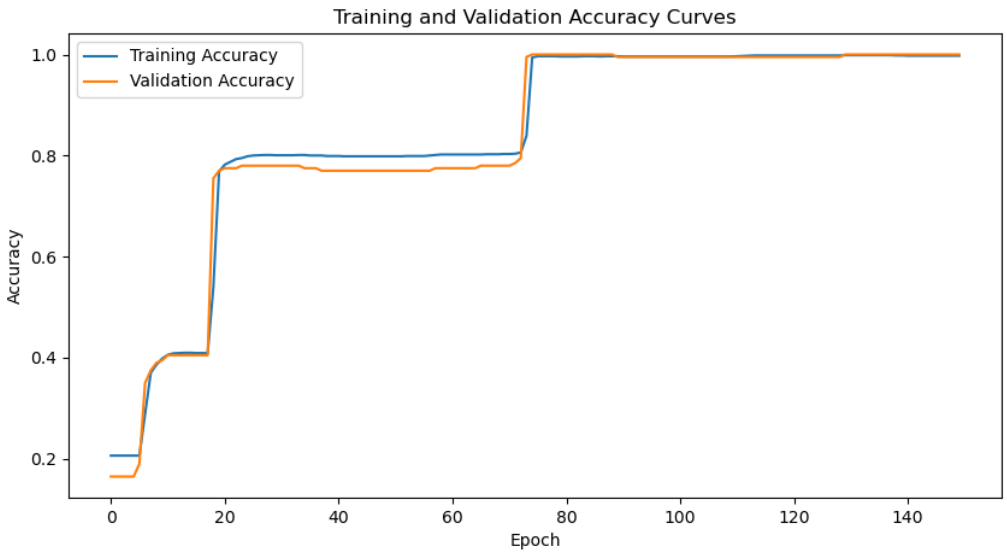


Figure 36. Validation and training accuracy of fusion model

Monitoring validation and training accuracy curves during the training of a model serves several essential purposes in evaluating and optimizing model performance. Accuracy curves track how well the model performs on both the training data and unseen validation data as training progresses. They provide a direct measure of the model's ability to correctly classify or detect objects, reflecting its overall performance. Accuracy curves shows whether the model is improving and learning with time. Improving training accuracy shows that the model is successfully adapting to the training data. Whereas, validation accuracy is a measure of how effectively the model can apply what it has learned to unseen and new data. Monitoring these curves helps ensure that the model is converging to an optimal solution without overfitting (high

training accuracy but low validation accuracy). Accuracy curves provide insights into the impact of hyperparameters (e.g., learning rate, batch size) on model performance.

Analysis:

The Fig. 36 shows that the model has been trained effectively and has high accuracy on both the training and validation datasets, indicating its capacity to comprehend the patterns existing in the data and apply them to fresh data. Training Accuracy is shown with blue line. It starts at a low accuracy, then increases rapidly, and eventually plateaus around 1.0. Validation Accuracy is indicated with an orange line. It follows a similar trend to the training accuracy but generally remains slightly lower.

The Confusion Matrix of the model on test data is shown in Fig. 37.

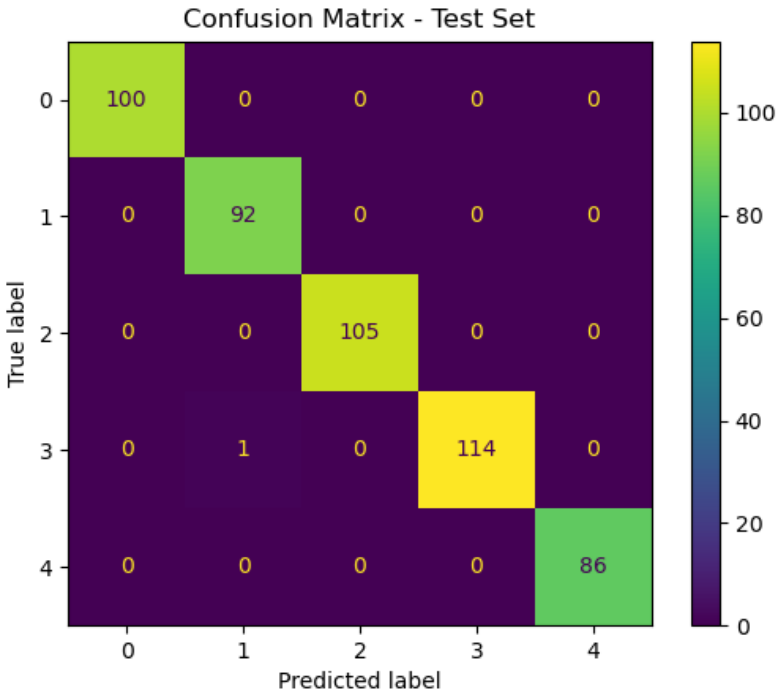


Figure 37. Confusion matrix of fusion model on test data

The confusion matrix is an essential tool for evaluating the performance of machine learning models. It offers a comprehensive analysis of the model's forecasts in relation to the actual ground truth across various groups or categories. The confusion matrix is a tool used to evaluate the effectiveness of a classification or detection model. It provides a summary of predictions in a matrix style.

Analysis:

The confusion matrix in Fig.37 shows that the model performs well on the test dataset. The cells along the diagonal, which indicate appropriate classifications, have the highest frequencies, suggesting that the model successfully predicted the correct class for the majority of the data points. The off-diagonal cells indicate misclassifications, but their counts are relatively low compared to the correct classifications. The model correctly classified 100 data points as class 0, 92 as class 1, 105 as class 2, 114 as class 3, and 86 as class 4. The only noticeable misclassification is one data point from class 3 being incorrectly predicted as class 1.

Here, label 0 refers to the palm gesture, label 1 refers to the one gesture, label 2 refers to the two_up gesture, label 3 refers to the three gesture, and label 4 refers to the fist gesture.

Test Accuracy of Fusion Model = $\frac{100+92+105+114+86}{100+92+105+114+86+1}$

Test Accuracy of Fusion Model = **99.8%**

The graphic indicates that the model performs with a high degree of accuracy on the test set, with very few misclassifications. This indicates that the model has effectively adapted to unfamiliar data.

4.3.2. Real Time Testing

After training fusion model, its weights were saved. The model was imported for real-time testing. Once all the necessary conditions for real-time testing were met, such as connection of the sensing-glove, loading camera model, the testing was conducted using real-time data of the probabilities of two models. The fusion model produced the expected results.

4.4. Results Conclusion

Based on the comprehensive evaluation of all the algorithms developed in this section, the system proved its high accuracy and low latency to be used in real time. In the vision approach, results show robust performance across metrics such as F1 score, mean average precision (mAP), and confusion matrices. Furthermore, figure 34 illustrates the exceptional accuracy and little loss achieved by the model in identifying gestures using

the sensor-based technique. Additionally, the fusion model combining camera and sensor data enhances accuracy further, leading to 99.8% test accuracy. Real-time testing confirms the low latency and reliability of the system for gesture detection and controlling the computer cursor using detected gestures. These results are a testament to the systems' effectiveness in practical applications that require precise and responsive gesture recognition.

4.5. Development of Windows-Based Applications

For this project, two windows-based applications have been developed. The first one is a desktop application to run this project and the second one is a desktop application to display and interact with 3D models.

4.5.1. Development of Desktop Application for User

A desktop application based on Windows has been designed for this project. This application provides a seamless interface for the user to utilise this project in the form of computer cursor control using hand gestures. The user can place the folder that contains the application and its associated files on their computer. Then follow the steps defined in the documentation of the application to run it properly.

This application is developed on the .NET 8.0 platform using the C# language in Microsoft Visual Studio. Both the frontend and backend of the application are developed on the same platform. The application allows the user to select either a camera or a fusion of sensor and camera for hand gesture detection.

The user interface of the application designed for standalone camera is shown in Fig.38.

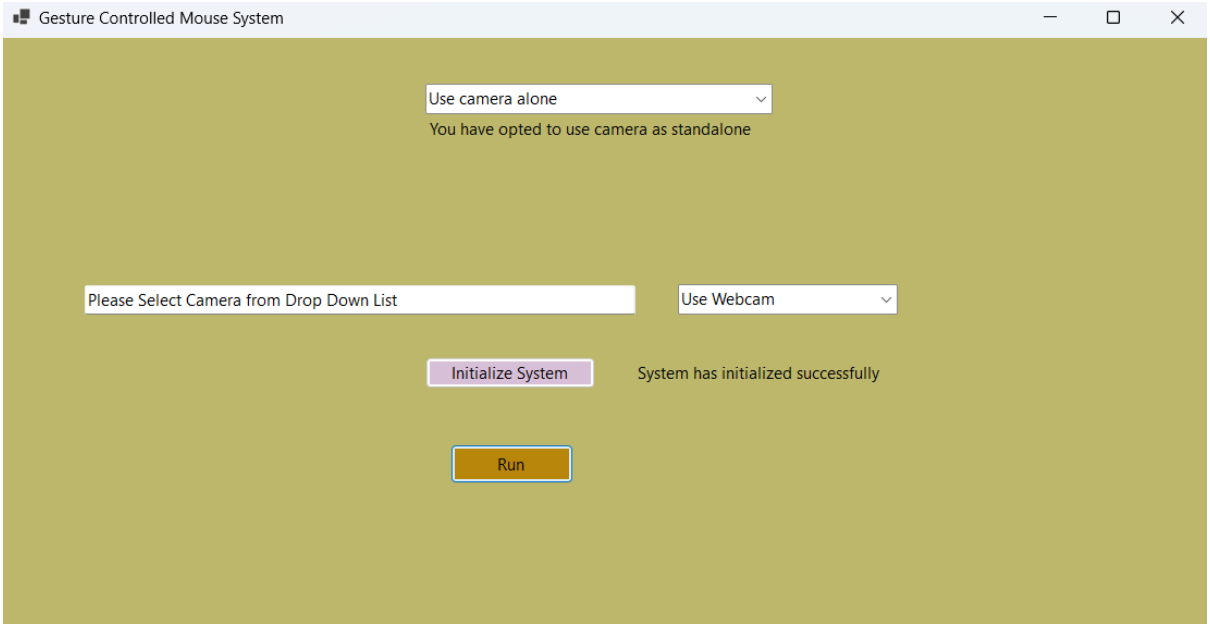


Figure 38. User interface for camera-based gesture recognition

Fig. 39 displays the user interface of the application specifically developed for the integration of camera and sensor.

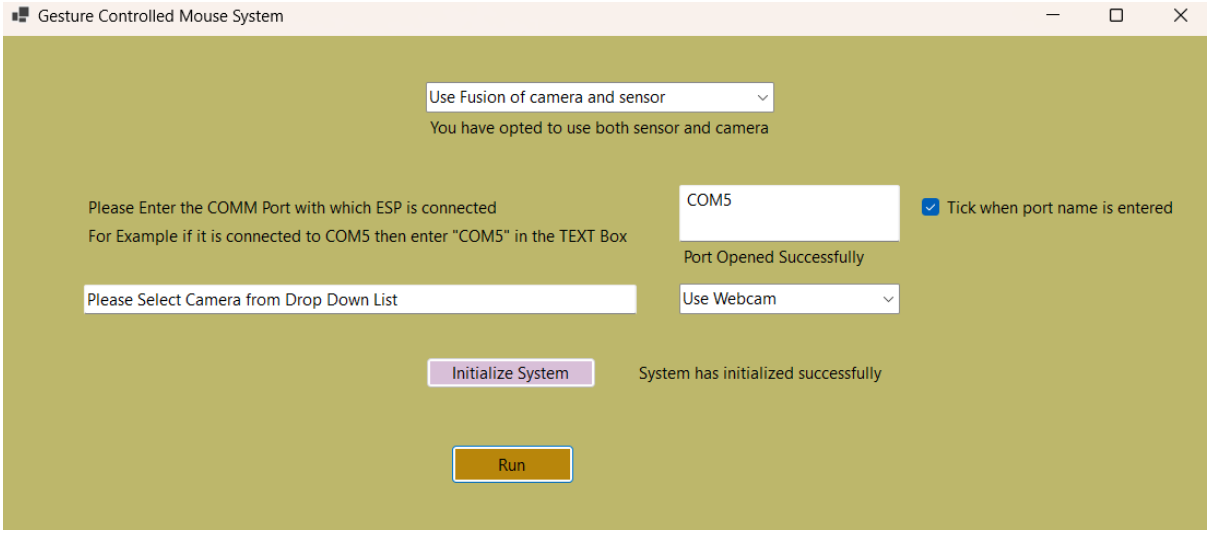


Figure 39. User interface for fusion-based gesture recognition

4.5.2. Development of 3D Models Application

3D-Viewer is a Windows-based application that provides an interface for interactable 3D models. Different 3D models are displayed, and the user can completely rotate the 3D models in all three dimensions and zoom in and zoom out of the 3D models. The tools used include Unity-3D software, 3D models in different formats available online, and VS Code.

This application is developed in the Unity 3D engine. 3D models of the Human Skeleton, the Human Heart, and a Simple Cube are used. The cube model is available in the Unity environment by default, while the skeleton and heart models are available online and are imported into the Unity environment. The 3D models are then scaled and positioned accordingly so that these models appear appropriately in the game view. A user interface is designed with text, buttons, and a drop-down menu. Four UI buttons are designed for the rotation of 3D models. There are three buttons designated for rotation along the z-axis, x-axis and y-axis, respectively. Additionally, there is one button specifically meant to halt the rotation. Two buttons are designed for zooming functionality. One button to zoom into the 3D model, and the other one is to zoom out of the 3D model. Another four buttons are designed to perform pan functionalities to move the 3D models up, down, left, and right. A drop-down menu is added, which is used to select the 3D model. C# script files are attached with 3D models, UI buttons, and the drop-down menu, which are responsible for performing actions like rotation, pan, zooming, and model selection. The following figures represent the final version of the application with various 3D models.

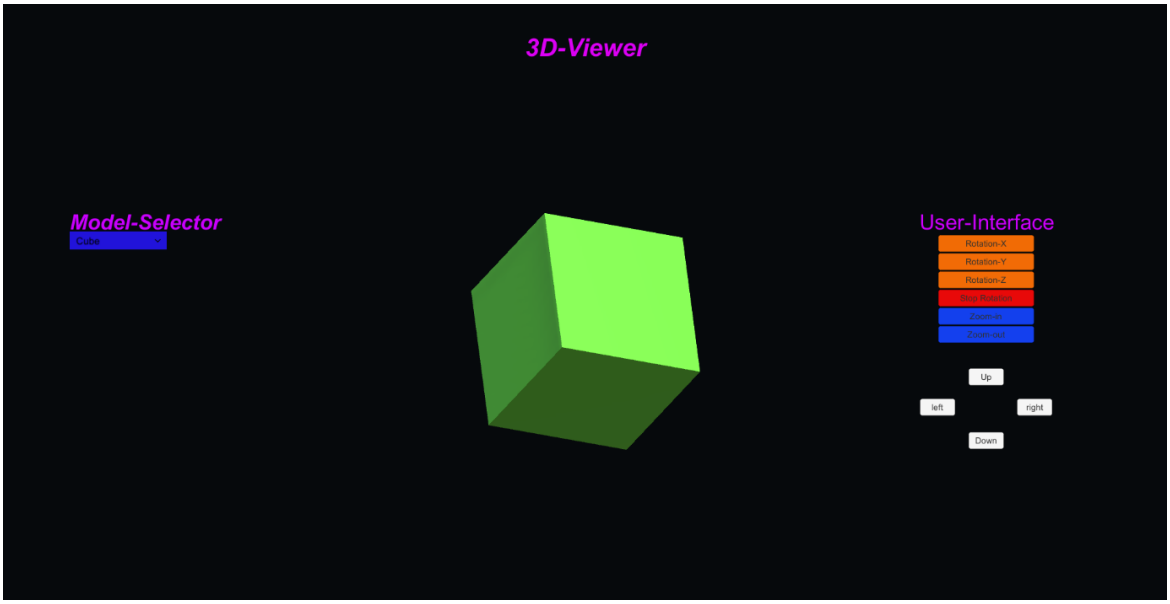


Figure 40. Cube model

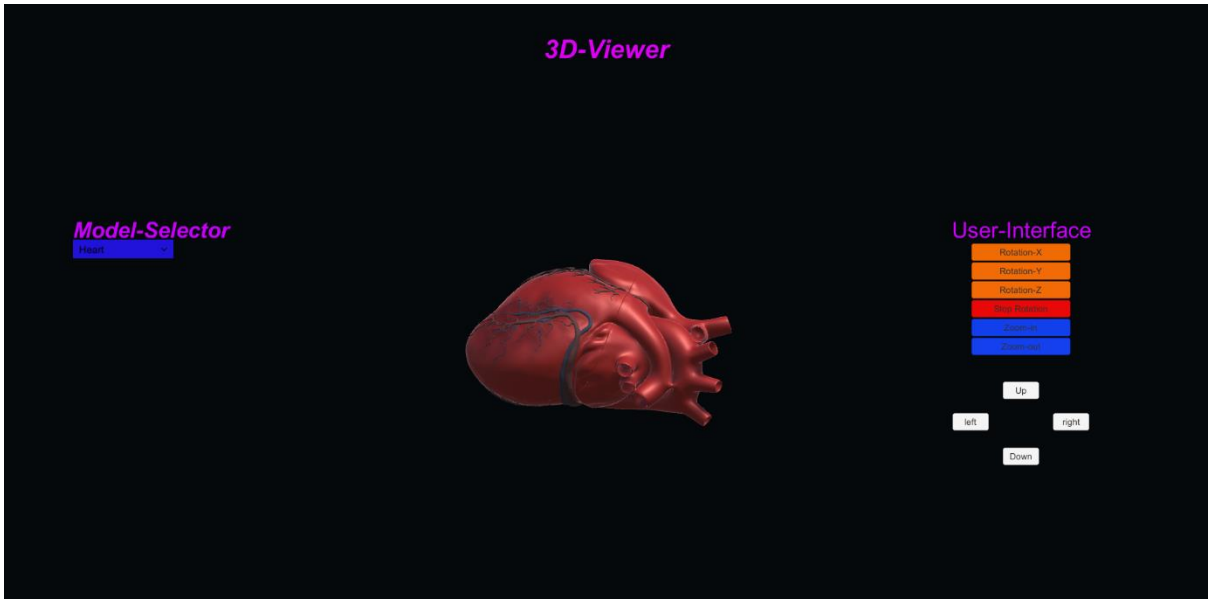


Figure 41. Heart model

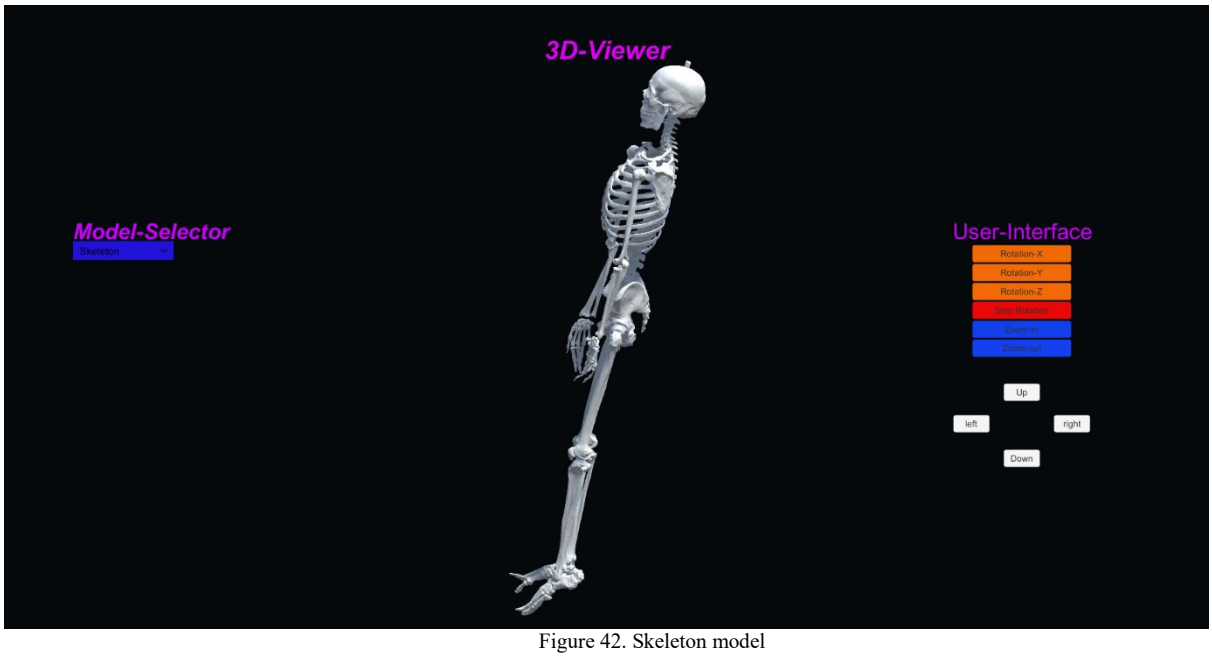


Figure 42. Skeleton model

Chapter 5 – CONCLUSIONS AND FUTURE WORK

5.1. Conclusion

The aim of developing a system to interact with 3D visuals is successfully accomplished. Keeping the view of its high potential in multiple sectors, system is developed with high degree of accuracy using combination of multiple learning-based approaches. The 3D display is interacted with using dynamic hand movements in real time, resulting in a more natural user experience. Therefore, the whole development process involved the detection of hand gestures, translating them into mouse commands, system application development and an application for displaying and controlling 3D models on the holographic display.

Hand gesture detection is accomplished through a combination of two approaches: a vision-based approach and a sensor-based approach. The vision-based approach utilizes either an external camera or a webcam to continuously monitor hand movements. A learning-based gesture detection algorithm is employed to train the model for recognizing specific hand gestures. Based on real-time data input from the camera, the algorithm accurately identifies the performed gestures. Additionally, gestures are annotated in real time to obtain hand pose information in the form of coordinate points, which are essential for controlling the computer cursor. Conversely, the same hand gestures are also detected using a sensor-based approach. The development of this approach began with the design of the hardware. While on the software side, real-time data from the flex sensors is collected for each gesture. Subsequently, the collected data is utilized to construct and instruct an artificial neural network (ANN) model with the ability to identify gestures using the recorded data. The flex sensors, located on the fingers, gather data regarding the flexion of the fingers. This information is transmitted in real-time via Bluetooth from the ESP32 microcontroller to a connected computer. The transmitted data is processed by a fully connected ANN, which computes the probability of each possible gesture occurring. The gesture with the highest likelihood is recognized as the prevailing gesture.

To improve the precision of gesture prediction, the probability vectors from both the vision-based and sensor-based models were merged and utilized as input to an extra ANN model. This model was created and trained utilizing the output data from both

gesture detection methods. Consequently, the output of this model provides the most accurate gesture prediction. This is followed by another algorithm that detects the transition between two consecutive gestures. When a particular set of transitions is detected, it generates a signal to perform an operation, such as a right click, a left click, a double click, etc. In order to move the cursor, the key landmark of the forefinger is used to compute the difference between the previous and present landmark values. The difference was then scaled, and the mouse tracking function was called to move the cursor in accordance with the hand.

Finally, the entire system is integrated into a single window-based application that offers a variety of options for system execution. This Windows-based application allows users to operate the system using either the vision-based approach alone or in combination with the sensor-based approach for enhanced accuracy through data fusion. Lastly, another Windows application is created to show 3D models on the holographic display. This completes the development of all necessary requirements for translating mouse functionalities into dynamic hand gestures. Connecting a hologram to this system enables the user to control the 3-dimensional visuals with hand motions, which would be useful in a variety of fields, particularly education and medical. At this point, the hand gesture-controlled holographic display is complete.

5.2. Future Work

There is a lot of potential for this project in the future. Two main concepts could be incorporated into this project. First, instead of tracking the mouse using a vision-based approach, this could be done using a combination of motion sensors such as a gyroscope and an accelerometer. A well-executed algorithm that uses the information from these sensors could provide more robust tracking of the cursor using hand motions. Additionally, this allows the user to control the cursor even if he is not in the camera's zone. Furthermore, a potential future endeavor would involve transferring the system onto distinct, self-contained hardware. This would grant the user the ability to execute the system on any computer.

REFERENCES

[1] D. I. B. Tarazona, D. A. D. Gómez and S. A. González, "Design and implementation of a computer mouse using a 5-DOF inertial measurement unit and a sensor fusion algorithm," in *2012 Workshop on Engineering Applications*, Bogota, 2012.

[2] R. Suriya and V. Vijayachamundeeswari, "A survey on hand gesture recognition for simple mouse control," in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Chennai, 2014.

[3] S. Song, D. Yan and Y. Xie, "Design of control system based on hand gesture," in *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, Zhuhai, 2018.

[4] V. V. Reddy, T. Dhyanchand, G. V. Krishna and S. Maheshwaram, "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition," in *2020*, Hyderabad, 2020 IEEE-HYDCON.

[5] P. Kalaivaani, R. Kumaresan, A. J. M. Ranjith and S. N. Kumar, "Hand Gestures for Mouse Movements with Hand and Computer Interaction model," in *2023 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, 2023.

[6] A. Jadhav, R. Asnani, R. Crasto, O. Nilange and A. Ponkshe, "Gesture Recognition using Support Vector Machine," *International Journal of Electrical, Electronics and Data Communication,* vol. 3, no. 5, pp. 36-41, 2015.

[7] A. Sharma, A. Mittal, S. Singh and V. Awatramani, "Hand Gesture Recognition using Image Processing and Feature Extraction Techniques," *Procedia Computer Science,* vol. 173, pp. 181-190, 2020.

[8] B. A. &. N. A. Sunanda, "A novel feature fusion technique for robust hand gesture recognition," *Multimedia Tools and Applications,* vol. 83, 2024.

[9] X. Wang, M. Xia, H. Cai, Y. Gao and C. Cattani, "Hidden-Markov-Models-Based Dynamic Hand Gesture Recognition," *Mathematical Problems in Engineering,* pp. 1-11, 2012.

[10] M. HafizurRahman and J. Afrin, "Hand Gesture Recognition using Multiclass Support Vector Machine," *International Journal of Computer Applications,* vol. 74, no. 1, pp. 39-43, 2013.

[11] X. Jiang and W. Ahmad, "Hand Gesture Detection Based Real-Time American Sign Language Letters Recognition using Support Vector Machine," in *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)*, Fukuoka, 2019.

[12] D.-Y. Huang, W.-C. Hu and S.-H. Chang, "Vision-Based Hand Gesture Recognition Using PCA+Gabor Filters and SVM," in *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Kyoto, 2009.

[13] M. M. Islam, S. Siddiqua and J. Afnan, "Real time Hand Gesture Recognition using different algorithms based on American Sign Language," in *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, Dhaka, 2017.

[14] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *In Advances in neural information processing systems,* vol. 1, pp. 1097-1105, 2012.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, 2009.

[16] G. Li, H. Tang, Y. Sun, J. Kong, G. Jiang, D. Jiang, B. Tao, S. Xu and H. Liu, "Hand gesture recognition based on convolution neural network," *Cluster Computing,* vol. 22, pp. 2719-2729, 2017.

[17] K. Xing, Z. Ding, S. Jiang, X. Ma, K. Yang, C. Yang, X. Li and F. Jiang, "Hand Gesture Recognition Based on Deep Learning Method," in *2018 IEEE Third International Conference on Data Science in Cyberspace*, Guangzhou, 2018.

[18] A. Eid and F. Schwenker, "Visual Static Hand Gesture Recognition Using Convolutional Neural Network," *Algorithms,* vol. 16, no. 8, 2023.

[19] J. O. Pinzón-Arenas, R. J. Moreno and R. D. H. Beleño, "Convolutional neural network with a DAG architecture for control of a robotic arm by means of hand gestures," *Contemporary Engineering Sciences,* vol. 11, pp. 547-557, 2018.

[20] J. O. Pinzón Arenas, R. Jiménez Moreno and R. Darío Hernández Beleño, "Convolutional Neural Network with a DAG Architecture for Control of a Robotic Arm by Means of Hand Gestures," *Contemporary Engineering Sciences,* vol. 11, no. 12, pp. 547-557, 2018.

[21] J. Prakash Sahoo, S. Ari and S. Kumar Patra, "Hand Gesture Recognition Using PCA Based Deep CNN Reduced Features and SVM Classifier," in *2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, Rourkela, 2019.

[22] A. Wadhawan and P. Kumar, "Deep learning-based sign language recognition system for static signs," *Neural Computing and Applications,* vol. 32, pp. 7957-7968, 2020.

[23] J. Yu, M. Qin and S. Zhou, "Dynamic gesture recognition based on 2D convolutional neural network and feature fusion," *scientific reports,* vol. 12, 2022.

[24] D. Tran , L. Bourdev , R. Fergus, L. Torresani and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, 2015.

[25] P. Molchanov, S. Gupta, K. Kim and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Boston, 2015.

[26] N. L. Hakim, T. K. Shih, S. P. K. Arachchi, W. Aditya, Y.-C. Chen and C.-Y. Lin, "Dynamic Hand Gesture Recognition Using 3DCNN and LSTM with FSM Context-Aware Model," *sesnors,* vol. 19, no. 24, 2019.

[27] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree and J. Kautz, "Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[28] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree and J. Kautz, "Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[29] V. Sharma, M. Jaiswal, A. Sharma, S. Saini and R. Tomar, "Dynamic Two Hand Gesture Recognition using CNN-LSTM based networks," in *2021 IEEE International Symposium on Smart Electronic Systems (iSES)*, Jaipur, 2021.

[30] T. L. Dang, S. D. Tran, T. H. Nguyen, S. Kim and N. Monet, "An improved hand gesture recognition system using keypoints and hand bounding boxes," *Array,* vol. 16, 2022.

[31] Q. Gao, Y. Chen, Z. Ju and Y. Liang, "Dynamic Hand Gesture Recognition Based on 3D Hand Pose Estimation for Human–Robot Interaction," *IEEE Sensors Journal,* vol. 22, no. 18, pp. 17421-17430, 2021.

[32] Y. Zhang and F. Wang, "HandFormer: A Dynamic Hand Gesture Recognition Method Based on Attention Mechanism," *applied sciences,* vol. 13, no. 7, 2023.

[33] J. Materzynska, G. Berger, I. Bax and R. Memisevic, "The Jester Dataset: A Large-Scale Video Dataset of Human Gestures," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, 2019.

[34] A. D'Eusanio, A. Simoni, S. Pini, G. Borghi, R. Vezzani and R. Cucchiara, "A Transformer-Based Network for Dynamic Hand Gesture Recognition," in *2020 International Conference on 3D Vision (3DV)*, Fukuoka, 2020.

[35] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.

[36] V. Pathak, S. Mongia and G. Chitranshi, "A framework for hand gesture recognition based on fusion of Flex, Contact and accelerometer sensor," in *Third International Conference on Image Information Processing (ICIIP)*, Waknaghat, 2015.

[37] T. Zhao, J. Liu, Y. Wang, H. Liu and Y. Chen, "PPG-based Finger-level Gesture Recognition Leveraging Wearables," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, 2018.

[38] F. Noble, M. Xu and M. Xu, "Static Hand Gesture Recognition Using Capacitive Sensing and Machine Learning," *Sensors,* vol. 3419, no. https://doi.org/10.3390/s23073419, p. 23, 2023.

[39] Y. Ge, B. Li, W. Yan and Y. Zhao, "A real-time gesture prediction system using neural networks and multimodal fusion based on data glove," in *Tenth International Conference on Advanced Computational Intelligence (ICACI)*, Xiamen, 2018.

[40] A. K. Panda, R. Chakravarty and S. Moulik, "Hand Gesture Recognition using Flex Sensor and Machine Learning Algorithms," in *IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, Langkawi Island, 2020.

[41] W.-C. Chuang, W.-J. Hwang, T.-M. Tai, D.-R. Huang and Y.-J. Jhang, "Continuous Finger Gesture Recognition Based on Flex Sensors," *Sensors 2019,19(18), 3986. https://doi.org/10.3390/s19183986.*

[42] M. Lee and J. Bae, "Deep Learning Based Real-Time Recognition of Dynamic Finger Gestures Using a Data Glove," *IEEE Access,* vol. 8, pp. 219923 - 219933, 2020.

[43] G. Yuan, X. Liu, Q. Yan, S. Qiao, Z. Wang and L. Yuan, "Hand Gesture Recognition Using Deep Feature Fusion Network Based on Wearable Sensors," *IEEE Sensors,* vol. 21, no. 1, pp. 539 - 547, 2021.

[44] H. Chen, T. Qin, Y. Zhang and B. Guan, "Recognition of American Sign Language Gestures Based On Electromyogram (EMG) Signal With XGBoost Machine Learning," in *3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, Taiyuan, 2021.

[45] M. Ariyanto, W. Caesarendra, K. A. Mustaqim, M. Irfan, J. A. Pakpahan, J. D. Setiawan and A. R. Winoto, "Finger movement pattern recognition method using artificial neural network based on electromyography (EMG) sensor," in *International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*, Bandung, 2015.

[46] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang and J. Yang, "A Framework for Hand Gesture Recognition Based on Accelerometer and EMG Sensors," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans,* vol. 41, no. 6, pp. 1064 - 1076, 2011.

[47] M. Georgi and T. S. Christoph Amma, "Recognizing Hand and Finger Gestures with IMU based Motion and EMG based Muscle Activity Sensing," *BIOSTEC 2015: Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies, Volume 4, January 2015, Pages 99–108, https://doi.org/10.5220/0005276900990108.*

[48] H. P. Gupta, H. S. Chudgar, S. Mukherjee, T. Dutta and K. Sharma, "A Continuous Hand Gestures Recognition Technique for Human-Machine Interaction Using Accelerometer and Gyroscope Sensors," *IEEE Sensors,* vol. 16, no. 16, pp. 6425 - 6432, 2016.

[49] T.-M. Tai, Y.-J. Jhang, Z.-W. Liao, K.-C. Teng and W.-J. Hwang, "Sensor-Based Continuous Hand Gesture Recognition by Long Short-Term Memory," *IEEE Sensors,* vol. 2, no. 3, 2018.

[50] Y.-J. Jhang, Y.-C. Chu, T.-M. Tai, W.-J. Hwang, P.-W. Cheng and C.-K. Lee, "Sensor Based Dynamic Hand Gesture Recognition by PairNet," in *International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Atlanta, 2019.

[51] U. G. Mangai, S. Samanta, S. Das and P. R. Chowdhury, "A Survey of Decision Fusion and Feature Fusion Strategies," *IETE Technical Review,* vol. 27, no. 4, pp. 293-307, 2010.

[52] E. Kanjo, E. M. Younis and N. Sherkat, "Towards unravelling the relationship between on-body, environmental and emotion data using sensor information fusion approach," *Information Fusion,* vol. 40, pp. 18-31, 2018.

[53] J. Summaira, X. Li, A. M. Shoib, S. Li and J. Abdul, "Recent Advances and Trends in Multimodal Deep Learning: A Review," *arXiv preprint arXiv:2015,* 2021.

[54] G. Divya and T. I. Manish, "Machine Learning Techniques and Frameworks for Heterogeneous Data Fusion in Big Data Analytics," in *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, 2020.

[55] T. Meng, X. Jing, Z. Yan and W. Pedrycz, "A survey on machine learning for data fusion," *Information Fusion,* vol. 57, pp. 115-129, 2020.

[56] F. Ali, S. El-Sappagh, S. R. Islam, D. Kwak, A. Ali, M. Imran and K.-S. Kwak, "A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion," *Information Fusion,* vol. 63, pp. 208-220, 2020.

[57] G. Yu, Y. Wang, J. Wang, C. Domeniconi, M. Guo and X. Zhang, "Attributed heterogeneous network fusion via collaborative matrix tri-factorization," *Information Fusion,* vol. 63, pp. 153-165, 2020.

[58] H. Cai , Z. Qu, Z. Li , Y. Zhang, X. Hu and B. Hu, "Feature-level fusion approaches based on multimodal EEG data for depression recognition," *Information Fusion,* vol. 59, pp. 127-138, 2020.

[59] Y. Pan, L. Zhang, X. Wu and M. J. Skibniewski, "Multi-classifier information fusion in risk analysis," *Information Fusion,* vol. 60, pp. 121-136, 2020.

[60] J. Bin, B. Gardiner, E. Li and Z. Liu, "Multi-source urban data fusion for property value assessment: A case study in Philadelphia," *Neurocomputing,* vol. 404, pp. 70-83, 2020.

[61] H. Shao, H. Jiang, F. Wang and H. Zhao, "An enhancement deep feature fusion method for rotating machinery fault diagnosis," *Knowledge-Based Systems,* vol. 119, pp. 200-220, 2017.

[62] N. Mungoli, "Adaptive Feature Fusion: Enhancing Generalization in Deep Learning Models," *arXiv preprint arXiv:2304.03290,* 2023.

[63] B. Hu, Q. Li and G. B. Hall, "A decision-level fusion approach to tree species classification from multi-source remotely sensed data," *ISPRS Open Journal of Photogrammetry and Remote Sensing,* vol. 1, 2021.

[64] S. Teng, G. Chen, Z. Liu, L. Cheng and X. Sun, "Multi-Sensor and Decision-Level Fusion-Based Structural Damage Detection Using a One-Dimensional Convolutional Neural Network," *Sensors,* vol. 21, no. 12, 2021.

[65] M. W. Nadeem, H. G. Goh, M. A. Khan, M. Hussain, M. F. Mushtaq and V. Ponnusamy, "Fusion-Based Machine Learning Architecture for Heart Disease Prediction," *Computers, Materials & Continua,* vol. 67, no. 2, pp. 2481-2496, 2021.

[66] W. Zhai, C. Li, Q. Cheng, F. Ding and Z. Chen, "Exploring Multisource Feature Fusion and Stacking Ensemble Learning for Accurate Estimation of Maize Chlorophyll Content Using Unmanned Aerial Vehicle Remote Sensing," *remote sensing,* vol. 15, no. 13, 2023.

[67] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[68] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, 2017.

[69] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767,* 2018.

[70] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934,* 2020.

[71] G. Jocher, "YOLOv5 by Ultralytics," *https://github.com/ultralytics/yolov5,* 2020.

[72] C.-Y. Wang, A. Bochkovskiy and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling Cross Stage Partial Network," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, 2021.

[73] C.-Y. Wang, I.-H. Yeh and H.-Y. M. Liao, "You Only Learn One Representation: Unified Network for Multiple Tasks," *arXiv preprint arXiv:2105.04206,* 2021.

[74] Z. Ge, S. Liu, F. Wang, Z. Li and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," *arXiv preprint arXiv:2107.08430,* 2021.

[75] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei and X. Wei, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *arXiv preprint arXiv:2209.02976,* 2022.

[76] C.-Y. Wang, A. Bochkovskiy and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696,* 2022.

[77] X. Xu, Y. Jiang, W. Chen, Y. Huang, Y. Zhang and X. Sun, "DAMO-YOLO : A Report on Real-Time Object Detection Design," *arXiv preprint arXiv:2211.15444,* 2022.

[78] G. Jocher, A. Chaurasia and J. Qiu, "Ultralytics YOLOv8," *https://github.com/ultralytics/ultralytics,* 2023.

[79] K. Alexander, K. Karina, N. Alexander, K. Roman and M. Andrei, "HaGRID - HAnd Gesture Recognition Image Dataset," in *2024 Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, 2024.

APPENDICES

Appendix A: Complex Engineering Problem

Depth of Knowledge Required (WP1 >>>WK3 WK4 WK5 WK6):

Project requires fundamental understanding of communication protocols, signal processing, and AI algorithms (WK3). Further it require engineering specialist knowledge of deep learning algorithm to recognize hand gesture (WK4). In addition, knowledge related to engineering design of project involving vision and sensor based approach is required (WK5). Project also require modern software tools for implementation of signal processing and AI algorithms (WK6).

Depth of Analysis Required (WP3): To complete the project strong mathematical analysis is required for finding out objective function and their optimization to recognize hand gesture for controlling holographic display.

Familiarity of Issues (WP4): Involve infrequently encountered issues in fusion of algorithm for hand gesture recognition using vision and sensor based approach.

	WP1						WP 2	WP 3	WP 4	WP 5	WP 6	WP 7
	WK 3	WK 4	WK 5	WK 6	WK 7	WK 8						
PLO1 (WA1)	X											
PLO2 (WA2)		X						X				
PLO3 (WA3)			X									
PLO4 (WA4)									X			
PLO5 (WA5)				X					X			
PLO6 (WA6)												
PLO7 (WA7)												
PLO8 (WA8)												

Appendix B: Sustainable Development Goals

SUSTAINABLE DEVELOPMENT GOALS FOR FYP

FYP TITLE:

Real Time Hand Gesture Controlled Holographic Display

FYP SUPERVISOR: Dr. Qasim Umar Khan

GROUP MEMBERS:

	REGISTERATION NUMBER	NAME
1	348038	MEER MUZAMMEL KHAN
2	344968	FARZAM TAIYAL ZULFIQAR
3	353048	MUHAMMAD HAROON
4	337587	KARIM HUSSAIN

SDGs:

	SDG No.	Justification after consulting
1	4	The project is in accordance with target 4.8, which is to provide an effective learning environment.
2	9	The project is in accordance with the following targets: 9.1, 9.2, and 9.5 to promote innovations, infrastructure and industries.

FYP Advisor Signature: _____

UN website: https://focus2030.org/Understanding-the-Sustainable-Development-Goals#:~:text=The%20Sustainable%20Development%20Goals%20(SDGs,life%20for%20all%2C%20by%20 2030.

Appendix C: Plagiarism Report

REAL TIME HAND GESTURE CONTROLLED HOLOGRAPHIC DISPLAY

Submission date: 01-Jul-2024 02:42AM (UTC-0700)
Submission ID: 2411149295
File name: FYP_report_alpha_3_-karim-sunday.pdf (2.99M)
Word count: 21549
Character count: 125376

KHUSSAIN

ORIGINALITY REPORT

9%	6%	6%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	<1%
2	www.hindawi.com Internet Source	<1%
3	"Data Science and Applications", Springer Science and Business Media LLC, 2024 Publication	<1%
4	Submitted to Liverpool John Moores University Student Paper	<1%
5	Submitted to South Bank University Student Paper	<1%
6	Mingdeng Tang. "A real-time traffic sign detection in intelligent transportation system using YOLOv8-based deep learning approach", Signal, Image and Video Processing, 2024 Publication	<1%
7	dokumen.pub Internet Source	<1%

8	yorkspace.library.yorku.ca Internet Source	<1%
9	Qing Gao, Yongquan Chen, Zhaojie Ju, Yi Liang. "Dynamic Hand Gesture Recognition Based on 3D Hand Pose Estimation for Human-Robot Interaction", IEEE Sensors Journal, 2021 Publication	<1%
10	Submitted to University Politehnica of Bucharest Student Paper	<1%
11	Md Hasan-Ur Rahman, Bichar Dip Shrestha Gurung, Bharat K. Jasthi, Etienne Z. Gnimpieba, Venkataramana Gadhamshetty. "Automated Crack Detection in 2D Hexagonal Boron Nitride Coatings Using Machine Learning", Coatings, 2024 Publication	<1%
12	www.researchgate.net Internet Source	<1%
13	Lecture Notes in Computer Science, 2013. Publication	<1%
14	Submitted to Higher Education Commission Pakistan Student Paper	<1%
15	www.ijimai.org Internet Source	<1%

16	Yi Li, Jun Cheng, Xiaopeng Ji, Wei Feng, Dapeng Tao. "Real-time action recognition by feature-level fusion of depth and inertial sensor", 2017 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2017 Publication	<1 %
17	Sahar Hassani, Ulrike Dackermann, Mohsen Mousavi, Jianchun Li. "A systematic review of data fusion techniques for optimized structural health monitoring", Information Fusion, 2023 Publication	<1 %
18	Tang, Zhicheng. "New Deep Learning Methods for Small to Medium Object Detection in Images", University of Missouri - Columbia, 2024 Publication	<1 %
19	Submitted to University of Northumbria at Newcastle Student Paper	<1 %
20	www.arxiv-vanity.com Internet Source	<1 %
21	www.bag.admin.ch Internet Source	<1 %
22	repository.mines.edu Internet Source	<1 %

23	www.nature.com Internet Source	<1 %
24	ebin.pub Internet Source	<1 %
25	Chinta, Uma. "Pioneering Facial Expression Recognition: Unveiling Multi-Modal Strategies, Confronting Data Imbalance, and Decoding Emotions", University of Colorado Colorado Springs, 2024 Publication	<1 %
26	Submitted to Sheffield Hallam University Student Paper	<1 %
27	scholarworks.iupui.edu Internet Source	<1 %
28	www.projectpro.io Internet Source	<1 %
29	www.techscience.com Internet Source	<1 %
30	"Intelligent Systems Design and Applications", Springer Science and Business Media LLC, 2020 Publication	<1 %
31	dspace.alquds.edu Internet Source	<1 %
32	escholarship.org Internet Source	

		<1%
33	hdl.handle.net Internet Source	<1%
34	www.diva-portal.org Internet Source	<1%
35	"Machine Learning for Cyber Security", Springer Science and Business Media LLC, 2020 Publication	<1%
36	Dingsheng Luo, Yang Ma, Xiangqi Zhang, Xihong Wu. "Multi-Sensor Fusion Based Robot Self-Activity Recognition", 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), 2018 Publication	<1%
37	github.com Internet Source	<1%
38	www.ir.juit.ac.in:8080 Internet Source	<1%
39	www.trademarkelite.com Internet Source	<1%
40	Submitted to British University in Egypt Student Paper	<1%

41	Ishak Pacal, Dervis Karaboga. "A Robust Real-Time Deep Learning Based Automatic Polyp Detection System", Computers in Biology and Medicine, 2021 Publication	<1 %
42	M. Mufti, A. Khanum. "Design and implementation of a secure mobile IP protocol", International Conference on Networking and Communication, 2004. INCC 204., 2004 Publication	<1 %
43	Submitted to Lindenwood University Student Paper	<1 %
44	S.V. Jansi Rani, Iacovos Ioannou, R. Swetha, R.M. Dhivya Lakshmi, Vasos Vassiliou. "A novel automated approach for fish biomass estimation in turbid environments through deep learning, object detection, and regression", Ecological Informatics, 2024 Publication	<1 %
45	scholar.sun.ac.za Internet Source	<1 %
46	Submitted to Australian National University Student Paper	<1 %
47	Submitted to University of Lagos Student Paper	<1 %

48	cds.iisc.ac.in Internet Source	<1 %
49	journals.lww.com Internet Source	<1 %
50	Wan Sieng Yeo, Woei Jye Lau. "Predicting the whiteness index of cotton fabric with a least squares model", Cellulose, 2021 Publication	<1 %
51	injctr.com Internet Source	<1 %
52	login.easychair.org Internet Source	<1 %
53	scholarworks.unist.ac.kr Internet Source	<1 %
54	techscience.com Internet Source	<1 %
55	www.coursehero.com Internet Source	<1 %
56	www.ije.ir Internet Source	<1 %
57	"Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2016)", Springer Science and Business Media LLC, 2018 Publication	<1 %

58	Chun Keat Tan, Kian Ming Lim, Roy Kwang Yang Chang, Chin Poo Lee, Ali Alqahtani. "HGR-ViT: Hand Gesture Recognition with Vision Transformer", Sensors, 2023 Publication	<1 %
59	Lan Luo, Limao Zhang, Qinghua He. "Linking project complexity to project success: a hybrid SEM–FCM method", Engineering, Construction and Architectural Management, 2020 Publication	<1 %
60	Yongfeng Dong, Jielong Liu, Wenjie Yan. "Dynamic Hand Gesture Recognition Based on Signals from Specialized Data Glove and Deep Learning Algorithms", IEEE Transactions on Instrumentation and Measurement, 2021 Publication	<1 %
61	ddd.uab.cat Internet Source	<1 %
62	digibuo.uniovi.es Internet Source	<1 %
63	dspace.bracu.ac.bd Internet Source	<1 %
64	journals.iium.edu.my Internet Source	<1 %
65	www.doria.fi Internet Source	

<1 %

66 "Machine Vision and Augmented Intelligence—Theory and Applications", Springer Science and Business Media LLC, 2021 <1 %
Publication

67 Freitas, Nuno Renato Azevedo. "Automatic Detection of Urinary Tract Abnormalities (ADUTA)", Universidade do Minho (Portugal), 2023 <1 %
Publication

68 Santos, Tomás Moreira. "Real-Time Weapon Detection in Surveillance Footage", Universidade do Porto (Portugal), 2023 <1 %
Publication

69 V. Vijeya Kaveri, V. Meenakshi, R. Meena Devi, A. Kousalya, M. Sujaritha. "Object Tracking Glove", Materials Today: Proceedings, 2022 <1 %
Publication

70 Xinyue Hao, Shaojuan Luo, Meiyun Chen, Chunhua He, Tao Wang, Heng Wu. "Infrared small target detection with super-resolution and YOLO", Optics & Laser Technology, 2024 <1 %
Publication

71 Zhang, Chenxi. "Deep Learning for Automatic Endoscope Assist System, From Disease <1 %

	Detection to Quality Control", University of Massachusetts Lowell, 2024 Publication	
72	elibrary.tucl.edu.np Internet Source	<1 %
73	ijres.iaescore.com Internet Source	<1 %
74	link.springer.com Internet Source	<1 %
75	sensorsportal.com Internet Source	<1 %
76	user.ceng.metu.edu.tr Internet Source	<1 %
77	www.ijitee.org Internet Source	<1 %
78	"Computer Vision – ECCV 2016", Springer Science and Business Media LLC, 2016 Publication	<1 %
79	"Emerging Trends in Electrical, Communications, and Information Technologies", Springer Science and Business Media LLC, 2020 Publication	<1 %
80	0-www-mdpi-com.brum.beds.ac.uk Internet Source	<1 %

81	Dewi Putrie Lestari, Rifki Kosasih. "Comparison of two deep learning methods for detecting fire hotspots", International Journal of Electrical and Computer Engineering (IJECE), 2022 Publication	<1 %
82	Fernandes, Diogo Samuel Gonçalves. "Exploring Metadata in Neural Networks for UAV Maritime Surveillance", Universidade do Porto (Portugal), 2024 Publication	<1 %
83	Ferrão, José Miguel Lopes. "A Multimodal Vision-Based Sensor Fusion Approach for Precise Landing of an UAV", Universidade do Porto (Portugal), 2024 Publication	<1 %
84	Hairong Jiang, , Bradley S. Duerstock, and Juan P. Wachs. "A Machine Vision-Based Gestural Interface for People With Upper Extremity Physical Impairments", IEEE Transactions on Systems Man and Cybernetics Systems, 2014. Publication	<1 %
85	Hongkun Chen, Junyang Chen, Yingjie Xie, Hangfei He, Boyi Zhang, Jingjie Guo, Li Wan, Xiaoyan Chen. "OIDS-45: A large-scale benchmark insect dataset for orchard pest	<1 %

monitoring", Research Square Platform LLC, 2024
Publication

---

86 Hussein, Hagar Mohamed Wahid. "Stress Detection Using New Time-Frequency Decomposition: Progressive Fourier Transform", Hamad Bin Khalifa University (Qatar), 2023 <1%
Publication

---

87 Jogi John, Shrinivas Deshpande. "Intelligent hybrid hand gesture recognition system using deep recurrent neural network with chaos game optimisation", Journal of Experimental & Theoretical Artificial Intelligence, 2023 <1%
Publication

---

88 Kunc, Vladimír. "Inference genové exprese pomocí umělých neuronových sítí", Czech Technical University, 2024 <1%
Publication

---

89 Lijuan Zhang, Gongcheng Ding, Chaoran Li, Dongming Li. "DCF-Yolov8: An Improved Algorithm for Aggregating Low-Level Features to Detect Agricultural Pests and Diseases", Agronomy, 2023 <1%
Publication

---

90 Nilakshee Rajule, Sheetal Pawar, Savita Jadhav, Utkarsh Jambhulkar, Dnyaneshwar Kapse, Onkar Kanthale. "Real-Time Hand <1%

Gesture Recognition with Voice Conversion for Deaf and Dumb", 2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA), 2023
Publication

---

91 Pramod Kumar Vishwakarma, Nitin Jain. "Design and Augmentation of a Deep Learning Based Vehicle Detection Model for Low Light Intensity Conditions", SN Computer Science, 2024 <1%
Publication

---

92 Renda, Henrique Eduardo Espadinha. "Implementation of Deep Learning Models for Information Extraction on Identification Documents", Universidade NOVA de Lisboa (Portugal), 2024 <1%
Publication

---

93 Shuai Teng, Xuedi Chen, Gongfa Chen, Li Cheng. "Structural damage detection based on transfer learning strategy using digital twins of bridges", Mechanical Systems and Signal Processing, 2023 <1%
Publication

---

94 Submitted to University of Sydney <1%
Student Paper

---

95 Yunan Li, Qiguang Miao, Kuan Tian, Yingying Fan, Xin Xu, Rui Li, Jianfeng Song. "Large-scale Gesture Recognition with a Fusion of RGB-D <1%

Data Based on Saliency Theory and C3D model", IEEE Transactions on Circuits and Systems for Video Technology, 2017
Publication

96	academic.oup.com Internet Source	<1 %
97	arxiv.org Internet Source	<1 %
98	dias.library.tuc.gr Internet Source	<1 %
99	export.arxiv.org Internet Source	<1 %
100	ijritcc.org Internet Source	<1 %
101	informatica.si Internet Source	<1 %
102	journals.scholarpublishing.org Internet Source	<1 %
103	khazna.ku.ac.ae Internet Source	<1 %
104	macdorman.com Internet Source	<1 %
105	prod-ruor.uottawa.ca Internet Source	<1 %

106	qmro.qmul.ac.uk Internet Source	<1 %
107	scholarworks.rit.edu Internet Source	<1 %
108	technodocbox.com Internet Source	<1 %
109	uu.diva-portal.org Internet Source	<1 %
110	uwspace.uwaterloo.ca Internet Source	<1 %
111	www.frontiersin.org Internet Source	<1 %
112	www.iaarc.org Internet Source	<1 %
113	www.ijraset.com Internet Source	<1 %
114	www.npmjs.com Internet Source	<1 %
115	www.tnsroindia.org.in Internet Source	<1 %
116	Amir R.Ali, Jessica Magdy, Aya ElNahas, Marina Samir, N. M. Samantha Kamel, Mariam A. Fathy. "Design and Evaluation of a Wearable Glove for Sign Language	<1 %

Translation Using Cross-Correlation and Flex Sensors", 2023 International Telecommunications Conference (ITC-Egypt), 2023
Publication

---

117 Daniel S. Breland, Simen B. Skriubakken, Aveen Dayal, Ajit Jha, Phaneendra K. Yalavarthy, Linga Reddy Cenkeramaddi. "Deep Learning-Based Sign Language Digits Recognition From Thermal Images With Edge Computing System", IEEE Sensors Journal, 2021 <1%
Publication

---

118 Donghyeon Noh, Hojin Yoon, Donghun Lee. "A Decade of Progress in Human Motion Recognition: A Comprehensive Survey from 2010 to 2020", IEEE Access, 2024 <1%
Publication

---

119 Snehal Abhijeet Gaikwad, Dhananjay Upasani, Virendra Shete. "Review and Trends on Hand Gesture Recognition of Sign Language based on Deep Learning Approaches", 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), 2023 <1%
Publication

---

120 Wang, Xizhe. "Deep Learning for Dental Caries Detection With Model Enhancement <1%

and Dental Dataset Expansion", University of Massachusetts Lowell, 2024
Publication

---

121 Yufei Zhang, Bin Liu, Zhiqiang Liu. "Recognizing Hand Gestures With Pressure-Sensor-Based Motion Sensing", IEEE Transactions on Biomedical Circuits and Systems, 2019 <1 %
Publication

---

122 Ahmed Eid, Friedhelm Schwenker. "Visual Static Hand Gesture Recognition Using Convolutional Neural Network", Algorithms, 2023 <1 %
Publication

---

123 Amina Ben Haj Amor, Oussama El Ghoul, Mohamed Jemni. "Sign Language Recognition Using the Electromyographic Signal: A Systematic Literature Review", Sensors, 2023 <1 %
Publication

---

124 Chun Keat Tan, Kian Ming Lim, Chin Poo Lee, Roy Kwang Yang Chang, Ali Alqahtani. "SDViT: Stacking of Distilled Vision Transformers for Hand Gesture Recognition", Applied Sciences, 2023 <1 %
Publication

---

125 Dirisala J Nagendra Kumar, Vidhya K, S. Sagar Imambi, P. V. Pramila, Ashok Kumar, Vijayabhaskar V. "DL-based Rheumatoid <1 %

Arthritis Prediction using Thermal Images", 2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2022
Publication

126	E. Foroughi Asl, S. Ebadollahi, R. Vahidnia, AA. Jalali. "Statistical Database of Human Motion Recognition using Wearable IoT - A Review", IEEE Sensors Journal, 2023 Publication	<1 %
127	Matthew Turk, Alex Pentland. "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, 1991 Publication	<1 %
128	Sepp Hochreiter, Jürgen Schmidhuber. "Long Short-Term Memory", Neural Computation, 1997 Publication	<1 %
129	Tan Kok Liang, Toshiyuki Tanaka, Hidetoshi Nakamura, Akitoshi Ishizaka. "A neural network based computer-aided diagnosis of emphysema using CT lung images", SICE Annual Conference 2007, 2007 Publication	<1 %
130	Tong Meng, Xuyang Jing, Zheng Yan, Witold Pedrycz. "A survey on machine learning for data fusion", Information Fusion, 2020 Publication	<1 %