[1]**Krithika Vaidyanathan**

[2] **Subramani Chinnamuthu**

[3]**Partheeban Pon**

[4]**Balaji Ravindaran**

[5]**Nitish Sundarraj**

[6]**Sasidhar Vuppala**

Real-Time Telerobotic Manipulator Control Using Hand Gestures

Abstract: - Control of Objects from a remote distance became an important factor for mankind about decades ago. Since then there has been a tremendous amount of improvements in the field of remote control. Once the "Internet of Things(IoT)" had been introduced, remote control technology across various applications has also opted for this method of data transmission from device to device over the internet. Through this work, a new way of controlling a robotic arm that can mimic/repeat human hand gestures and operate in a remote location through the use of IoT technology is proposed. The robotic arm design involves generative designing for weight reduction over a five end-effector model for cost and weight reduction purposes, also for the cause of hand gesture recognition a Neural Network has been trained and implemented behind an application that has been completely programmed in python. As compared to traditional methods of controlling a robotic arm the proposed method provides better cost efficiency, data transmission, data collection, and accessibility due to the abundant availability of computers, and adaptability towards the growing technology.

Keywords: Remote control; Hand gesture recognition; Internet of Things; Robotic arm control; generative designing, robotics, neural network

I. Introduction

Remote control of robots through hand gestures has garnered significant attention in recent years due to its potential applications in various fields [1-3]. This research aims to develop a novel method for remotely controlling robots using simple hand gestures, focusing on a five end-effector right-hand robotic arm as a demonstration model. This study integrates elements such as neural networks [4], deep learning (DL) methods [5,6], Python programming [7], Internet of Things (IoT) [8-10], machine learning (ML) [11], graphical user interface (GUI) [12], computer vision (CV), and Fusion 360 (integrated 3D modeling, CAD, CAM, CAE software). All the code developed for this research are stored in the GitHub repository [13].

The goals of this research encompass both the user end and the receiver end. At the user end, the goal is to collect a substantial amount of data being different hand positions and use it for transfer learning on a Neural Network (NN). This NN is trained to recognize hand positions and integrated into a Python program with a GUI, helping live predictions using computer vision techniques [14]. The predictions are then transmitted over the internet via Firebase Real-Time Database, a cloud-hosted NoSQL DBMS by Google. At the receiver end, the goal is to retrieve the transmitted data and employ it to control the robotic arm remotely. Additionally, weight reduction of the robotic arm model is pursued through generative designing using Fusion 360.

This research draws upon a diverse range of prior studies and resources. IoT-based robotic arm designs [15], gesture-controlled robotic arms [16], project-based learning examples [17], and image processing techniques [18] contribute to the foundation of this work. Additionally, machine learning algorithms for big data analytics [19], and IoT applications in robotic arm control [20] inform the method. The use of Python programming [21], deep learning methodologies [22], and Fusion 360 for generative design [23] are essential components of this research. These references, along with others cited in later sections, collectively support the development and execution of the proposed method for remote robotic arm control.

[1*,4,5,6] Department of Mechatronics Engineering, SRM institute of Science and Technology, Kattankulathur, Chennai 603203, India
Corresponding Authors: Krithika Vaidyanathan ; krithikv@srmist.edu.in
[2]Department of Electrical and Electronics Engineering, SRM institute of Science and Technology, Kattankulathur, Chennai 603203, India
[3] Department of Computer Science and Engineering, Stella Mary's college of Engineering, India

II. Research Methodology

The proposed research work undergoes the following chronological order:

1. Training Neural Network
2. Dedicated Software and Remote Data Transmission
3. Design of Robotic Arm
4. Data Retrieval and Robotic Arm Control

Initially collection of numerous data is required and pre-processing need s to be done. Then the neural network should be trained and tested with the collected train and test dataset. After completion of proper training testing of the neural network model, the proposed model is applied for live predictions. The GUI application will be created with the designed neural network model in the background. Then the predictions need to be uploaded to the internet. After successful completion of uploading the predictions, it needs to be retrieved from the internet in a remote location. The data retrieval can be successful completed by building a prototype, i.e a robotic arm with proper hand gesture control needs to be designed. The robotic arm can be 3D printed. The weight reduction on solid parts of the robotic arm needs to be considered for effective outcome. Finally proper control system needs to be designed for the designed and developed prototype.

III. Design of dedicated Software Modell for the Proposed system

Initially, data is collected and the training process for a Convolutional Neural Network (CNN) tailored specifically for hand gesture recognition.

A. Data Collection and Pre-processing

The dataset used for training includes 30,000 grayscale images, each measuring 224 x 224 pixels, and being 20 distinct hand gesture classes. Pre-processing steps were applied to enhance feature extraction. These steps included converting RGB images to grayscale, applying Gaussian blur to reduce noise, flipping images for orientation differentiation, denoising, sharpening, and precise cropping to focus solely on the hand region. Figure 1 illustrates this process with great accuracy.

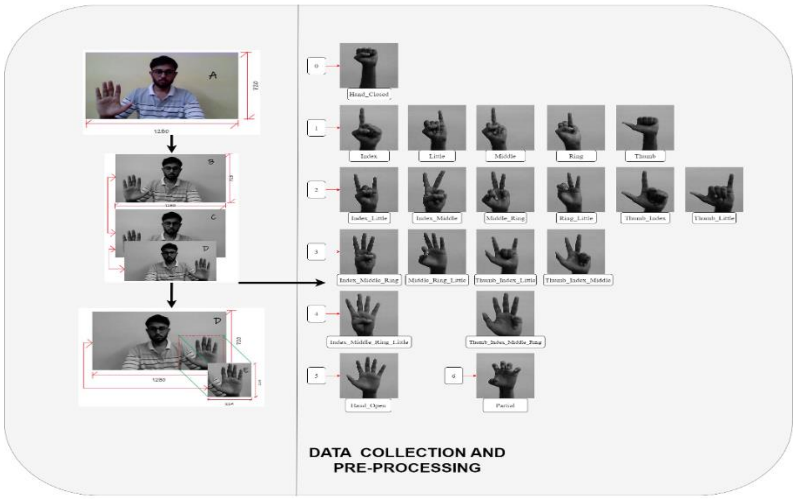


Figure 1. Collected Data and Classes

To predict all the hand gesture positions, there were 20 classes created and each class represents one hand gesture position. The classes with their corresponding class numbers are depicted in Table 1.

Table 1. Class Names and its associated Class Numbers

Class Number	Class Name	Class Number	Class Name
1	Hand_Closed	11	Thumb_Index
2	Index	12	Thumb_Little
3	Middle	13	Index_Middle_Ring
4	Ring	14	Middle_Ring_Little
5	Little	15	Thumb_Index_Little
6	Thumb	16	Thumb_Index_Middle
7	Index_Little	17	Index_Middle_Ring_Little
8	Index_Middle	18	Thumb_Index_Middle_Ring
9	Middle_Ring	19	Hand_Open
10	Ring_Little	20	Partial

As discussed earlier each class depicted above is populated with 1,500 data points(images) resulting in a total amount of 30,000 images forming the whole train data set, and every image in the data set went through the processing techniques. Figure 1 represents one data point from each class distributed over the whole data set. The nomenclature used in this work represents a data point through the words “Index, Little, Middle, Ring, Thumb, Hand_Closed, Hand_Open, Partial”. The index, little, middle, ring, and thumb correlate to one finger in our hand and represent its position as open; this can be viewed in Fig. 1 under category 1, a combination of this with a “_” is used to indicate different hand positions. Then the hand open(corresponds to category 5) and hand closed(corresponds to category 0) correlate that either all the fingers are open or closed, and partial(corresponds to category 6) is a particular case in which all the fingers are partially closed. Also, throughout this work, a generalized indication ie. from 0 to 6 is used as categories to differentiate the number of fingers in the open position, where 0 means no fingers are open to 5 means all fingers are open with 6 being a particular case.

B. Training

Training used Google's Teachable Machine platform, employing transfer learning techniques with the MobileNetV2 architecture. Due to computational constraints, the dataset was divided into two batches for training. Each batch underwent separate training sessions to refine the model's ability to classify hand gestures accurately.

C. Testing and Evaluation

Manual testing involved predicting hand gesture classes for 400 pre-processed images. Predictions were organized into lists, showing the likelihood of belonging to specific classes. Evaluation of the model’s performance is undertaken by analyzing the produced confusion matrix, additionally accuracy and loss of the model over the course of learning is also used to understand its performance development.

The Fig. 2 depicts the step involved in training the Neural Network with the acquired data while the Fig. 3, represents the confusion matrix and Figure. 4 and 5 represents the Neural network and its performance over the learning.

A confusion matrix is taken over 226 samples from each class to determine the performance of this classification algorithm. Since MobileNetV2 is already pre-trained with a large data set it shows maximum performance, other important predictive analytics such as accuracy, specificity, and precision are further calculated with the True Positives(TP), False Positives(FP), True Negatives(TN), False Negatives(FN) that are obtained from the confusion matrix.

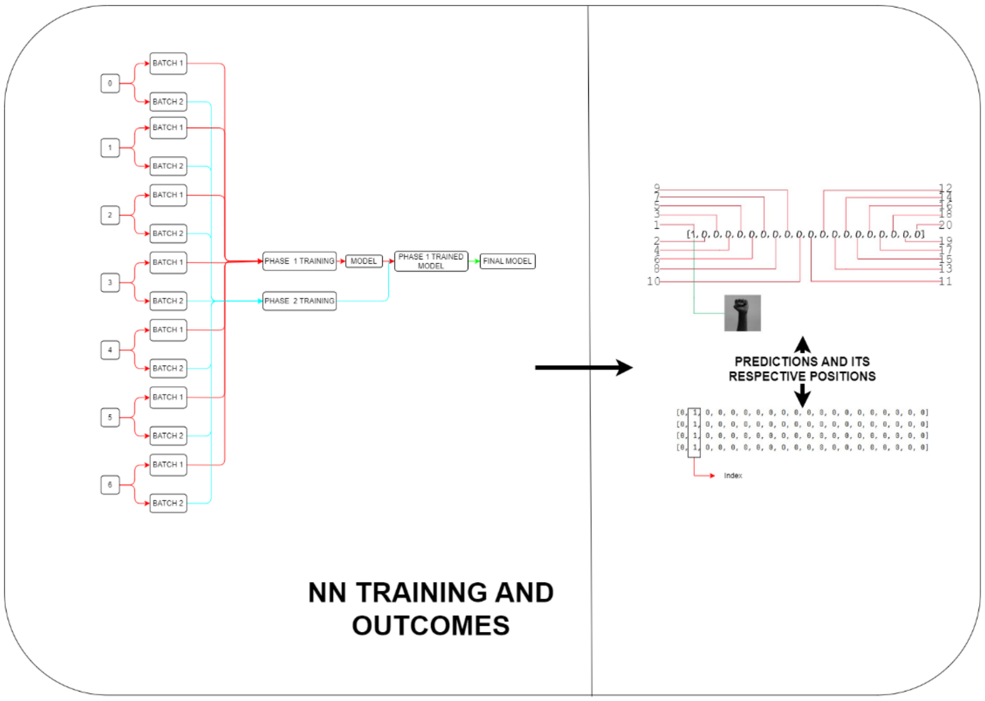
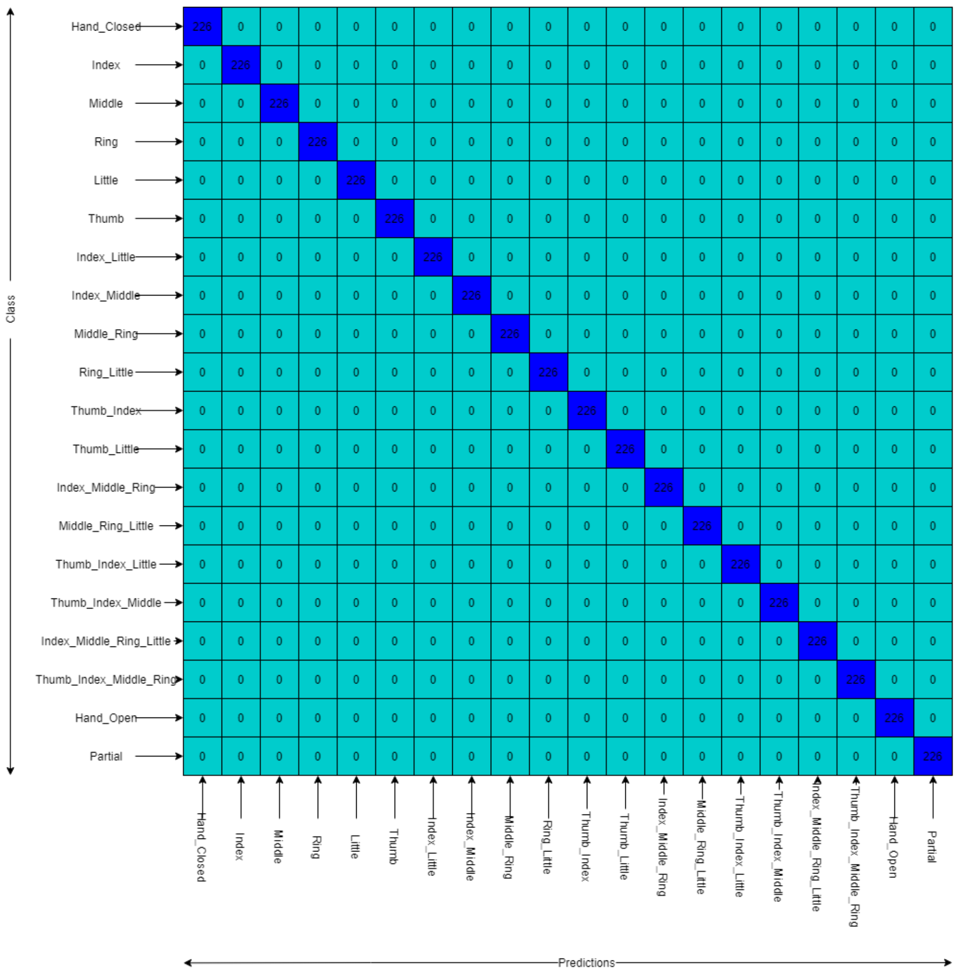


Figure. 2. Neural Network Training Process and Outcomes

One epoch is when the whole data set is passed forward and backward through the NN once, increasing the number of epochs gradually increases the accuracy of the model yet leading to Overfitting, thus an optimal number of epochs ie. 50 is used, and the accuracy per epoch is represented in Fig. 4, from 0 being minimum to 1 being maximum accuracy.

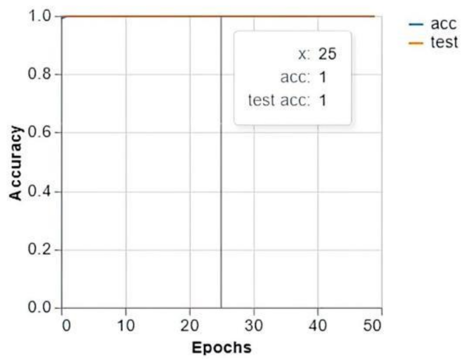
Loss in NN refers to the “Prediction Error”, and the method used to calculate the loss is called the “Loss Function”, there are various optimization techniques used to reduce loss, here every time one epoch is performed the loss reduces by a significant amount 12, this is represented in Fig. 5, as initiated by the first epoch the loss dropped from 0.025 to near 0 which is negligible, hence further epochs also tends to reduce the loss while increasing the performance by a very small amount.

The classes Hand_Closed and Index are considered, observing the values present in the confusion matrix, the TP and TN for these classes are 226 and 226 meanwhile, the FP and FN are found to be 0 and 0, hence calculating the accuracy using the below formula, the accuracy obtained is 1.0 which is considered as the best accuracy.


Class
Predictions

	Hand_Closed	Index	Middle	Ring	Little	Thumb	Index_Little	Index_Middle	Middle_Ring	Ring_Little	Thumb_Index	Thumb_Little	Index_Middle_Ring	Middle_Ring_Little	Thumb_Index_Little	Thumb_Index_Middle	Index_Middle_Ring_Little	Thumb_Index_Middle_Ring	Hand_Open	Partial
Hand_Closed	226	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index	0	226	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Middle	0	0	226	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ring	0	0	0	226	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Little	0	0	0	0	226	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Thumb	0	0	0	0	0	226	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index_Little	0	0	0	0	0	0	226	0	0	0	0	0	0	0	0	0	0	0	0	0
Index_Middle	0	0	0	0	0	0	0	226	0	0	0	0	0	0	0	0	0	0	0	0
Middle_Ring	0	0	0	0	0	0	0	0	226	0	0	0	0	0	0	0	0	0	0	0
Ring_Little	0	0	0	0	0	0	0	0	0	226	0	0	0	0	0	0	0	0	0	0
Thumb_Index	0	0	0	0	0	0	0	0	0	0	226	0	0	0	0	0	0	0	0	0
Thumb_Little	0	0	0	0	0	0	0	0	0	0	0	226	0	0	0	0	0	0	0	0
Index_Middle_Ring	0	0	0	0	0	0	0	0	0	0	0	0	226	0	0	0	0	0	0	0
Middle_Ring_Little	0	0	0	0	0	0	0	0	0	0	0	0	0	226	0	0	0	0	0	0
Thumb_Index_Little	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226	0	0	0	0	0
Thumb_Index_Middle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226	0	0	0	0
Index_Middle_Ring_Little	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226	0	0	0
Thumb_Index_Middle_Ring	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226	0	0
Hand_Open	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226	0
Partial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226

Figure. 3. Confusion Matrix



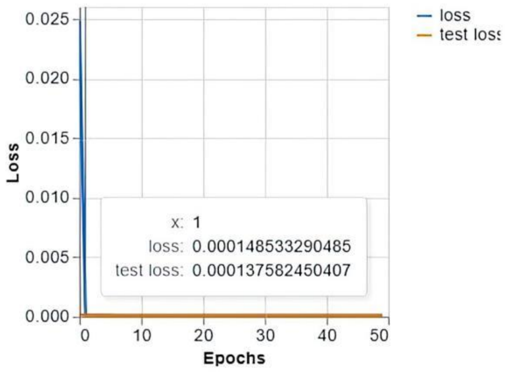


Figure. 4. Accuracy per epoch **Figure. 5. Loss per epoch**

D. Dedicated Software and Remote Data Transmission

This section delineates the functionalities of dedicated software designed for real-time hand gesture recognition, including the process of remote data transmission.

1. *Real-time Predictions*

A Python-based program, empowered by the OpenCV library, orchestrates the capture of live video footage at a smooth rate of 30 frames per second (fps). This footage undergoes meticulous processing, with each frame subjected to the same pre-processing methods outlined earlier. The resulting frames, akin to data points in the dataset, are aggregated and presented as a coherent video stream for user visualization.

2. *Data Transfer*

Following the real-time prediction phase, the next step involves uploading these predictions to an online Database Management System (DBMS). Leveraging Google's Firebase DB real-time database, the predictions are seamlessly transmitted to a remote location. Firebase DB eases data storage and synchronization between publishers and subscribers, ensuring real-time access to the transmitted data.

To optimize data transfer efficiency, predictions are uploaded with a slight delay of 1 second. This delay accounts for upload constraints between Firebase DB and the local machine, as well as the response time required for downstream processes. Each prediction is uploaded in string format, optimizing storage efficiency, with the size of each prediction estimated at 90 Bytes or approximately 8.6e^ (-05) Megabytes.

3. *Graphical User Interface (GUI)*

To enhance user experience and accessibility, a Graphical User Interface (GUI) encapsulates the functionalities of the real-time prediction system. Fig. 6 illustrates the graphical representation of the GUI, showcasing its intuitive design and user-friendly interface.

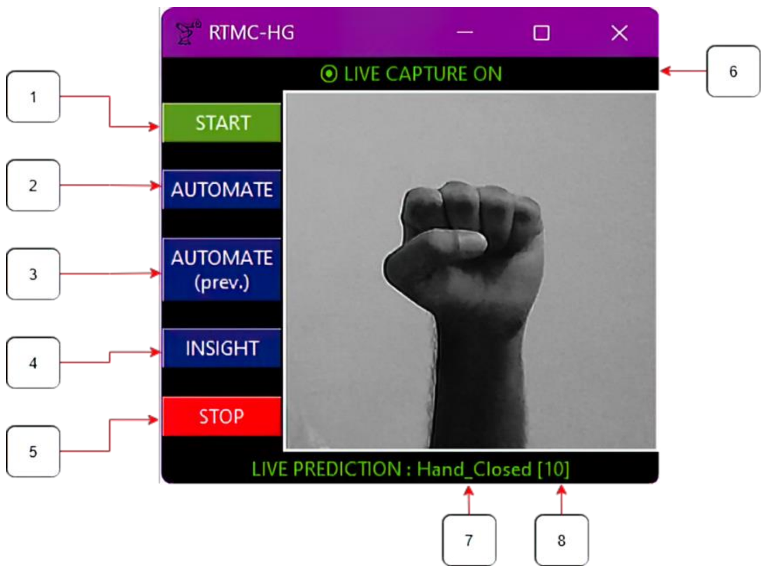


Figure. 6. Dedicated Software for real time prediction

The GUI comprises five intuitive buttons:
START (1): Initiates the live video capture, processing, prediction, and upload process, providing users with real-time visualization of the video stream.
AUTOMATE (2): Replicates the prediction upload process at a predefined interval, leveraging internally stored predictions generated during the live session.
AUTOMATE (prev.) (3): Like AUTOMATE, but utilizes predictions stored in a log.csv file for upload, automating the process based on previously generated predictions.
INSIGHT (4): Generates a cluster map to visualize the intensity of each class in the predictions stored in the log.csv file, aiding in identifying outliers and false predictions as shown in Fig. 7.

STOP (5): Halts all running processes, including the application itself, and optionally creates a log.csv file from internally stored predictions when used in conjunction with the START button.

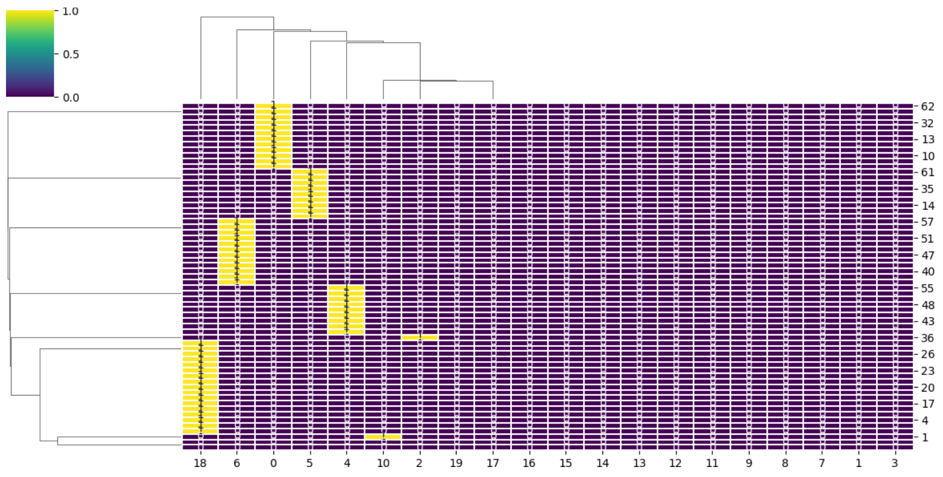


Figure. 7. Cluster map of the predictions

Additionally, the GUI incorporates visual indicators (6-8) to provide real-time feedback on the status of various processes, including camera status, prediction upload status, and row number for prediction logging.

IV. Design of Robotic Arm

Designing and manipulating objects is a complex task that requires a high level of dexterity and precision. For humanoid robots, this capability is particularly crucial as they are expected to safely interact with the environment, including humans. To achieve this, significant efforts have been devoted to developing robot hands or mechanisms that can emulate the dexterity and precision of human hands. However, successful design and manipulation control also depend on the availability of suitable contact and force feedback. To address this challenge, a robotic arm can be employed from a reliable source such as "InMoov", which has undergone weight reduction to enhance its responsiveness to predicted hand gestures.

A. Optimization of Robot Weight

Based on the research, it is identified that an opportunity to optimize the design by reducing excess material on solid objects. However, achieving this optimization manually can be quite challenging, especially without an in-depth understanding of material properties, structural rigidity, and shape. To overcome this challenge, the work is turned to Autodesk's Fusion 360 and its innovative tool, "Generative Design". By leveraging the power of AI, machine learning, and cloud computing, Generative Design has enabled us to streamline the weight reduction process while ensuring the structural integrity of the model is preserved.

B. Implementation of Generative Design

Executing generative design within Fusion 360 requires the initialization of several key components over the model. These components are crucial for the process and include "Obstacle Geometry," "Preserve Geometry," "Loads," "Constraints," "Material and Manufacturing method," and optionally, "Starting Shape."

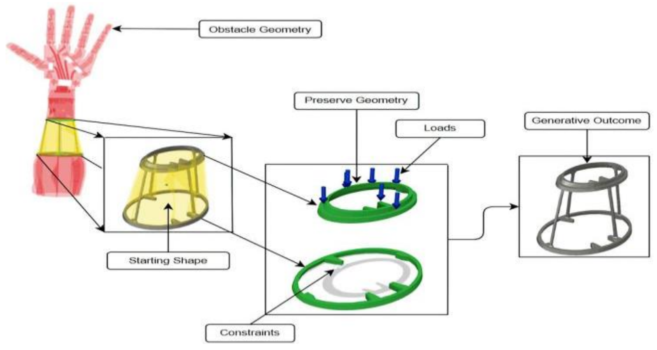


Figure. 8. Input Geometries and Generative Outcome

Obstacle Geometry is highlighted in "Red" and is assigned to objects within the model. These objects are considered real-life constraints, guiding the AI's iterative design process to exclude these constrained models from its modifications. Preserve Geometry, indicated in "Green," instructs the AI to optimize the design process to always include this specific object in the final outcome, ensuring its presence and characteristics are maintained throughout the iterations.

Loads are visually represented by blue arrows, and their direction signifies the direction in which loads are applied to the model. Each arrow is associated with a magnitude, representing the amount of load applied to the object.

Constraints, denoted by a lock symbol, serve to lock specific areas of the object, indicating that the object is anchored or held in place within that particular region. There exists a relationship between loads and constraints, with loads tending to deform the body while constraints work to maintain the object's position. Additionally, the "Starting Shape," marked in "Yellow," is initialized. This serves as a foundational model from which the AI commences its design process. The Starting Shape provides a basis for the AI to begin iterating and refining, ultimately leading to the generation of the final optimized design.

Once all the above constants are initialized the AI starts its iterative design process and provides an n-number of outcomes as shown in Fig. 8. Each outcome as shown in Fig. 9 is different from one another, among those outcomes one or more suitable outcomes can be chosen for the further research purpose.

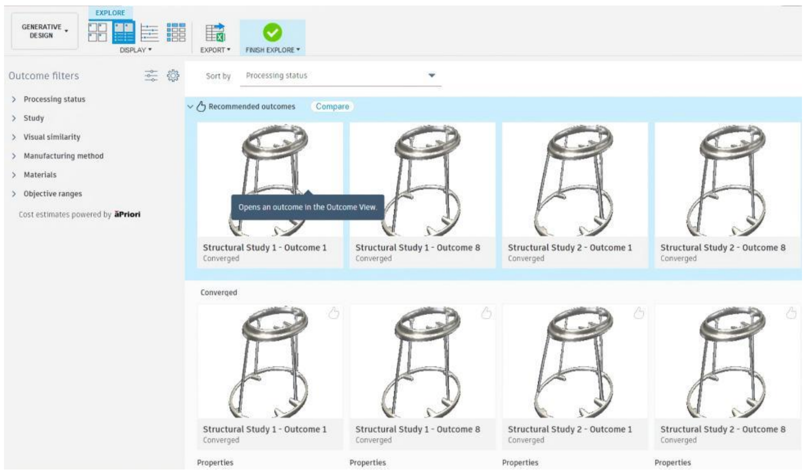


Figure. 9. Multiple generative design outcome

Although Generative outcome has its perks to its finest, the output model is still not that suitable for manufacturing since it has rough edges, uneven geometry, and a very organic design that are hard to manufacture, hence the outcome of Generative design is used as a reference and optimized the existing model in such way that it uses all the structural integrity data present in the outcome and reduces the excess material wherever necessary as shown in Fig. 10.

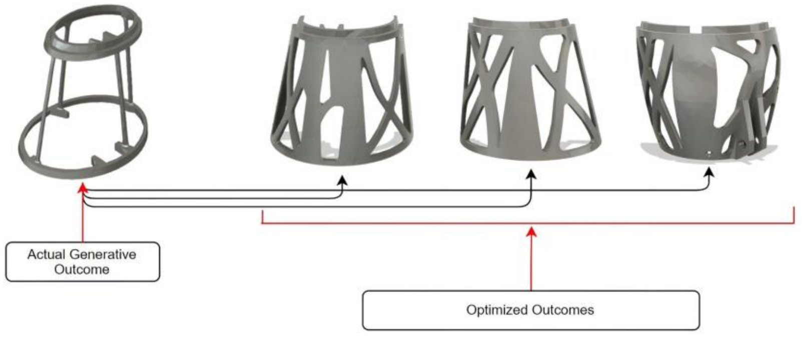


Figure.10. Optimized Outcomes

When creating the final model, it is important to define both the material utilized and the type of manufacturing process employed. In this instance, the "Additive Manufacturing" process is chosen for its capacity to deliver a highly optimized outcome.

V. Design of Data Retrieval and Robotic Arm Control

To control an Arduino Mega 2560 microcontroller, it is necessary to have a communication bridge between the module that transmits the data and the one receiving it. Considering latency aspects, the approach to this problem is to use two computers that send and receive the data over the internet, and then communicate the input data to the microcontroller. The "pyserial," is used to send instructions and data to the Arduino Board. This allows the microcontroller to process commands and efficiently handle incoming predictions with reliability.

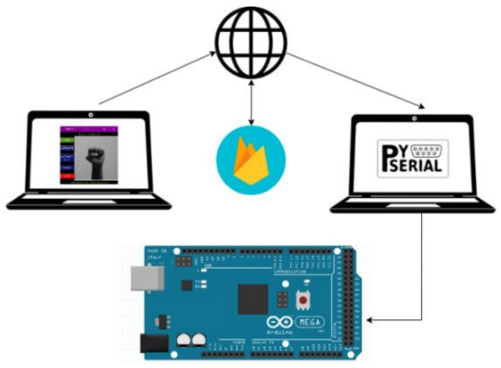


Figure.11. Data Flow

A. *Retrieval of Data*

After transmitting data to the Firebase database, the next step involves receiving the data on the other end and processing it for use by the recipient machine as shown in Fig. 11. However, since the data is transmitted as a string, characters such as "[", "]", and "," are also considered elements of the predictions. To find which class has a "1" in each prediction in the current state, it is necessary to parse the received predictions.

To perfect the received data, unnecessary elements in the string are removed, and a new array is created having only the useful values. This optimized array allows for faster and more efficient data processing, along with improved data organization.

B. Design of Control System

Once the predictions have been pre-processed and are ready to be used, the next step involves building a control system capable of effectively controlling the servo motors of the robotic arm using these predictions. To achieve this, a circuitry that connects all the motors to a microcontroller is set up. The microcontroller is then supplied with distinct functions that correspond to each received hand gesture prediction.

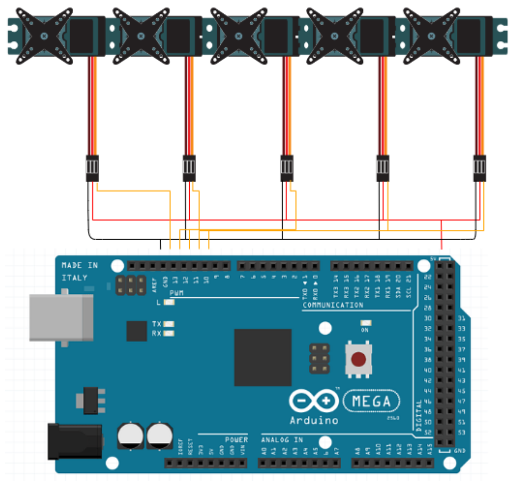


Figure.12. Control System Design of the proposed system

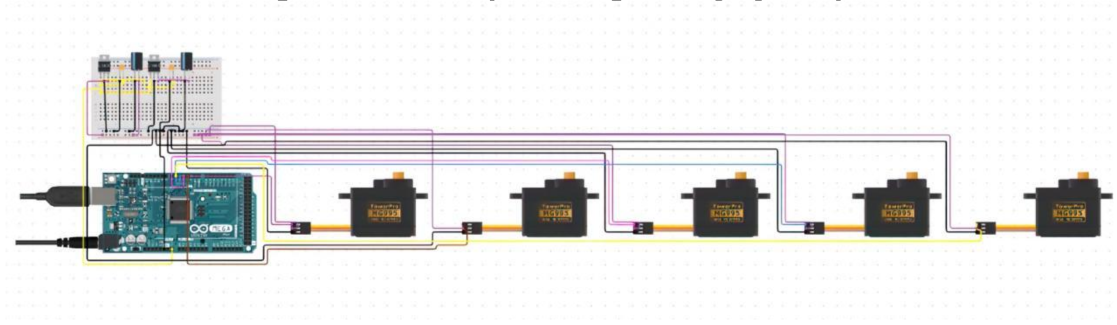


Figure. 13. Final circuitry for Arm control of the proposed system

To control the robotic arm, a circuit diagram, as shown in Fig. 12, is employed. In this setup, all the servo motors are connected appropriately to the circuitry. Each signal input for the motors is then linked to specific PWM ports (i.e., 8, 9, 10, 11, and 12) on the Arduino Board as seen in Fig. 13. This configuration ensures that the microcontroller commands the signals to accurately control the robotic arm's servo motors after data processing.

Upon receiving a gesture prediction, the microcontroller instantly triggers the PWM signals for every servo motor to perform the desired hand movement, ensuring seamless and precise manipulation of the robotic arm movements.

To ensure that the servo motors perform the right tasks for different hand gestures, they are grouped based on their functions. Each group of servos handles specific movements that collectively achieve the desired actions for the hand gestures. The proposed overall data flow is represented in Fig. 14.

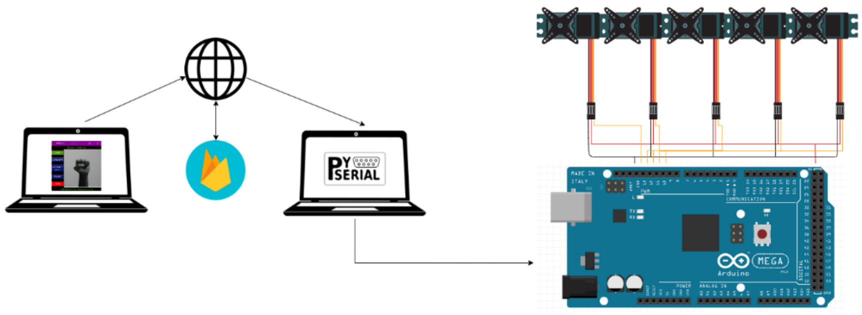


Figure.14. Representation of proposed Overall data flow

For example, let us consider the task of performing the "index finger" operation. In this scenario, the motors connected to the prosthetic fingers, except the index finger servo, should perform a 180-degree rotation to close, while the index finger servo still is at a 0-degree position to keep the index finger open. This grouping and coordination of servo movements allow for precise control of the hand gestures.

This principle applies to all other hand gestures as well. Each gesture has a corresponding group of servos that are programmed to move in a synchronized manner to achieve the intended action. By organizing the servos into functional groups and defining their specific movements for each hand gesture, the robotic arm can accurately replicate a wide range of human hand movements.

VI. Results & Discussion

The final working model of the proposed system is shown below in Fig. 15. This prototype was finally developed incorporating the software and hardware designs as discussed above. The robotic arm are 3D printed. Once, the data is transmitted to the FireBase DB, the next step is to receive the data on the opposite receiving end and to process the data so that the receiving machine can make use of it. Once the received data is parsed the unnecessary elements in the string are removed, and a new array is created with the useful values in it which can be used to process data faster and more efficiently, additionally having a better data organization in it.

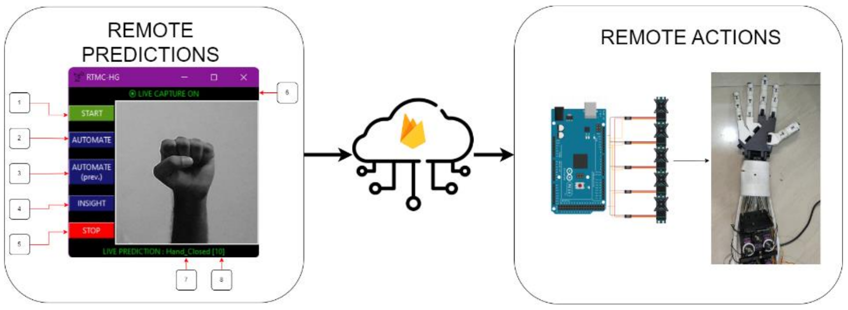


Figure. 15. Final Working Model

To provide commands to the Arduino Mega 2560 microcontroller which can process base C++ programming language with special methods and functions (Arduino Language) while getting the predictions over the internet using Python.

Hence, there is a necessity for communication between these two languages, hence the “pyserial” library is used which encapsulates the access for serial ports in Arduino boards from python which can be essentially used for communication.

The research culminated in the successful development and deployment of a neural network (NN) model capable of real-time hand gesture recognition. Leveraging the power of convolutional neural networks (CNNs) and transfer learning techniques, the model exhibited high accuracy in predicting hand gestures from live video footage. The integration of a graphical user interface (GUI) facilitated seamless interaction and visualization of the prediction process.

Furthermore, the implementation of Firebase DB facilitated the remote transmission of live predictions over the internet. The efficient transfer of prediction data enabled real-time communication between the sender (prediction system) and the receiver (robotic arm control system), the final working model is depicted in Figure 15 for visualization.

VII. Conclusion

This study proposes an innovative process of remote control that uses the latest available technology, providing distinct advantages over traditional robot control methods. The system is easily upgradeable, has low latency with IoT, utilizes MobileNetV2 for enhanced accuracy and robustness, and includes a GUI application for manual correction of predictions and better visualization. This work has elaborated the process of remote control through the latest technology available, also the proposed method shows its advantages when compared to traditional robot control methods that are available. The key attributes of the proposed system are as follows; they can be easily upgradeable meaning better adaptability towards future technologies. Due to the presence of IoT, the latency is greatly reduced, hence providing a better response from the system, also making the system truly wireless. The usage of MobileNetV2 which excels at computer vision problems makes the resultant application more accurate and robust. Due to the GUI application, there is a possibility of a manual correction in the predictions that are being uploaded, and also gives better visualization of the whole process. The future scope of this work is to make the end robot mobile from stationary thus adding some additional functionalities to the application, such as not only uploading pre- dictions of the hand gestures but also uploading other forms of inputs such as keypress in the laptop, etc., Also, there is a need for better optimization of the application which requires additional time and resources to make it more efficient, and reliable. Alternate systems can also be developed with the understanding of this system that functions with the aid of a smartphone containing all the functionality of this proposed system making it easily accessible as well as affordable to all. Future developments aim to enhance mobility, perfect efficiency, and broaden accessibility to users.

References

[1]V. Abhilash and P. K. Mani., “IoT based wheeled robotic arm”, International Journal of Engineering & Technology (IJET), vo.7(2.24), pp16-19, April 2018.
[2] A. S. Ahmed, H. A. Marzog, and L. A. Abdul-Rahaim., “Design and implement of robotic arm and control of moving via IoT with Arduino ESP32,” International Journal of Electrical and Computer Engineering (IJECE), vol.11, no.5, pp.3924-3933, October 2021.
[3] B. Ganesh Choudhary, B. V. Chethan Ram “Real time robotic arm control using hand gestures”, in 2014 International Conference on High Performance Computing and Applications (ICHPCA), IEEE, 2014. 10.1109/ICHPCA.2014.7045349.
[4] S. Athmaja, M. Hanumanthappa, and V. Kavitha. “A survey of machine learning algorithms for big data analytics,” in 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). IEEE, March 2017. 10.1109/ICI- IECS.2017.8276028.
[5] Y. Roh, G. Heo, and S. E. Whang., “ A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective,” IEEE Transactions on Knowledge and Data Engineering, vol.33, no.4,pp.`1328-1347, April 2021. 10.1109/TKDE.2019.2946162.
[6] D. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, “Flexible, High Performance Convolutional Neural Networks for Image Classification,” in International Joint Conference on Artificial Intelligence (IJCAI). AAAI, Spain, July 16-22, 2011. 10.5591/978-1-57735-516-8/IJCAI11-210.
[7] E. Matthes., Python Crash Course, 2nd edition, No Starch Press, May 2019.
[8] S. Fu and P. C. Bhavsar., “Robotic Arm Control Based on Internet of Things,” in 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT). IEEE, May 2019. 10.1109/LISAT.2019.8817333.

[9] S. Kumar, P. Tiwari, and M. Zymbler., "Internet of Things is a revolutionary approach for future technology enhancement: a review," Journal of Big Data. Vol.6, December 2019. 10.1186/s40537- 019-0268-2.
[10] S. L. Ullo and G. R. Sinha., "Advances in IoT and Smart Sensors for Remote Sensing and Agriculture Applications", Remote Sensing (MDPI), vol.13, no.13, July 2021. 10.3390/rs13132585.
[11] S. Linge and H. P. Langtangen., "Programming for Computations - Python: A Gentle Introduction to Numerical Simulations with Python", 1st edition, Springer, July 2016.
[12] R. S. Nair, S. Kumar, and N. Soumya., "A study on gesture controlled robotic arms and their various implementations", International Journal of Mechanical Engineering and Technology (IJMET), vol.9, no.3, pp.425-434, March 2018.
[13] Yuvaraj, N., Praghash, K., Logeshwaran, J., Peter, Geno., & Stonier, A. A. (2023). An artificial intelligence based sustainable approaches—Iot systems for smart cities. In B. Bhushan, A. K. Sangaiah, & T.N. Nguyen (Eds.), AI Models for Blockchain-Based Intelligent Networks in IoT Systems: Concepts, Methodologies, Tools, and Applications (pp. 105–120). Springer International Publishing. https://doi.org/10.1007/978-3-031-31952-5_5
[14] M. M. Kopichev, V. V. Putov, A. V. Putov, and K. V. Ignatiev., "Control system for a computer vision-equipped robot.", in 2015 XVIII International Conference on Soft Computing and Measurements (SCM), 2015. 10.1109/SCM.2015.7190421.
[15] B. Sundari and S. Sivaguru., "Design and Implementation of robotic arm using IoT.", SSRG International Journal of Electronics and Communication Engineering (SSRG - IJECE), vol.5, no.8, pp.1-4, 2018.
[16] S. Devine, K. Rafferty, and S. Ferguson., "Real time robotic arm control using hand gestures with multiple end effectors", in 2016 UKACC 11th International Conference on Control (CONTROL). IEEE, pp.1-5, 2016. 10.1109/CONTROL.2016.7737564.
[17] T. P. Cabré, M. T. Cairol, D. F. Calafell, M. T. Ribes, and J. P. Roca., "Project- Based Learning Example: Controlling an Educational Robotic Arm With Computer Vision", IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol.8, no.3, pp.135-142, August 2013. 10.1109/RITA.2013.2273114.
[18] G. James, D. Witten, T. Hastie, and R. Tibshirani., "An Introduction to Statistical Learning with Applications in R", 1st edition, Springer, September 2017.
[19] B. Ratner., "Statistical and Machine-Learning Data Mining; Techniques for Better Predictive Modeling and Analysis of Big Data", 2nd edition, CRC Press, January 2012.
[20] Geno Peter, A. A. Stonier, P. Gupta, D. Gavilanes, M. M. Vergara, and J. Lung sin, "Smart Fault Monitoring and Normalizing of a Power Distribution System Using IoT," Energies, vol. 15, no. 21, p. 8206, Jan. 2022, doi: 10.3390/en15218206.
[21] C. E. Rasmussen and C. K. Williams., "Gaussian Process for Machine Learning", 1st edition, The MIT Press, November 2005.
[22] W. Wang, Y. Li, T. Zou, X. Wang, J. You, and Y. Luo., "A Novel Image Classification Approach via Dense-MobileNet Models", Mobile Information Systems (Hindawi), January 2020.
[23] J. de Jesus Rubio, E. Garcia, C. A. Ibanez, and C. Torres., "Stabilization of the robotic arms", IEEE Latin America Transactions, vol.13, no.8, pp.2567-2573, August 2015. 10.1109/TLA.2015.7332133.