WIRELESS APPLIACTION CONTROL USING HAND GESTURES.

Abstract— **Hand gestures provide a natural way for humans to interact with computers to perform a variety of different applications. The human-computer interaction became an importance in bringing the idea that the link between a user and a computer should look more and more like one between two human beings. Thus, with increasing improvement in technology, response time and ease of operations are the concerns. Here is where human-computer interaction comes into show. This interaction is open and challenges the used devices such as the keyboard and mouse for input. Gestures are natural and are frequently used in day-to-day communications. Therefore, communicating using gestures with computers creates a whole new standard of interaction. In this project, with the help of computer vision and deep learning techniques, user hand movements (gestures) are used to control the media player, PPT slides, minimizing and closing the applications etc. The proposed web application enables the user to use their local device camera to identify their gesture and execute the control over the media player and similar applications (without any additional hardware). It increases effectiveness and makes communication easy in allowing the user control his/her laptop/desktop from a particular distance. We are thinking that in future people will move to wireless technologies and gestures are part of that so, we have made our choice to do this project.**

Keywords— human-computer interaction, hand gestures recognition, computer vision, Deep learning, Machine learning, Python.

I. INTRODUCTION

A. Gestures

The ability to transmit meaning, emotions, or instructions through a variety of physical expressions and movements is made possible through gestures, which are non-verbal modes of communication. They are crucial to human connection because they let people to exchange information without just using spoken or written words. The use of gestures in technology has grown significantly in importance with the advancement of processing power and sensor technologies.

Gesture recognition uses cameras, sensors, and algorithms to recognize and understand human gestures and convert them into commands for controlling digital devices. Smartphones, gaming consoles, virtual reality systems, and smart home appliances are just a few of the areas where this technology has found integration.

Hand gestures provide a simple and straightforward way to communicate with wireless apps and devices when used for wireless application control. With the help of pre-set gesture-to-control mappings, users may carry out operations like managing media players and minimizing and shutting programs without actually touching the device. With the help of this program, you may remotely operate computer gadgets in a more practical and relaxing manner while also increasing productivity.

B. *Computer vision, Deep learning, Machine learning uses in Gestures*

- The identification, interpretation, and use of gestures across a variety of applications depend heavily on the underlying technologies of computer vision, deep learning, and machine learning.

1) **Computer vision:** In order to replicate human visual perception, the discipline of computer vision involves the processing and interpretation of visual input from the outside environment. Computer vision techniques are used to record and decipher the motions and patterns of human gestures in the context of gestures.

   i. **Hand Detection and Tracking:** Computer vision algorithms are employed to identify and track hands in video feeds or images, determining their positions and movements.

   ii. **Feature Extraction:** These techniques extract pertinent features from images or video frames, aiding in the differentiation of various gestures.

   iii. **Image Processing:** Methods such as edge detection, contour analysis, and filtering are utilized to preprocess visual data, enhancing the accuracy of gesture recognition.

   iv. **Motion Analysis:** Computer vision facilitates the analysis of motion patterns, speeds, and trajectories of hand movements, contributing to gesture recognition.

   v. **3D Depth Sensing:** Advanced computer vision systems, like depth cameras and structured light devices, can capture three-dimensional information, enriching the gesture data.

2) **Machine Learning:** Machine learning involves training models to discern patterns from data and make predictions or decisions, forming a crucial component in gesture recognition.

   i. **Gesture Recognition Models:** Machine learning models, including Support Vector Machines (SVMs), Random Forests, or Decision Trees, can be trained on labeled gesture data to identify specific gestures.

ii. **Feature Classification:** Machine learning algorithms can classify extracted features from images or videos into distinct gesture categories.

iii. **Real-Time Processing**: These models can be optimized for real-time operation on devices, enabling instant gesture recognition.

iv. **Data Augmentation:** Techniques such as data augmentation can be employed to expand the training dataset artificially, resulting in improved model generalization.

3) **Deep Learning:** Deep learning, a subset of machine learning, leverages neural networks with multiple layers to excel in complex pattern recognition tasks, revolutionizing gesture recognition.

i. **Convolutional Neural Networks (CNNs):** CNNs are particularly suited for image-based gesture recognition tasks, as they autonomously learn relevant features from input images.

ii. **Recurrent Neural Networks (RNNs):** RNNs can capture temporal dependencies in gesture sequences, making them suitable for continuous gesture recognition.

iii.

iv. **Gesture Generation:** Deep learning models can generate realistic synthetic gestures, expanding training datasets and aiding in data augmentation.

v. **Gesture Pose Estimation:** Deep learning can estimate the skeletal pose of the hand or body from images, enhancing fine-grained gesture analysis.

- These technologies find applications in various fields, such as virtual reality, gaming, human-computer interaction, healthcare, automotive interfaces, and more. As they continue to advance, gesture-based interaction is likely to become even more refined, intuitive, and widespread. For this project, we will be utilizing computer vision Python libraries like Open-CV for motion detection and a deep learning technique - Convolutional Neural Networks (CNN).

II. Related work

1. Adithya V and Rajesh R [1] suggests leveraging CNN and deep parallel architectures to recognize automated hand posture. By utilizing CNN's hierarchical design, which automates feature extraction by examining the high-level abstractions in pictures, it hopes to bypass the laborious feature extraction stage. The suggested model achieves high accuracy while cutting down on calculation time [1].

2. Chen-Chiung Hsieh and Dung-Hua Liou and David Lee[2] presents a summary of the hand gesture recognition methods that have been suggested, the types of gestures that have been taken into consideration, the detection techniques that have been employed, the assessment findings, and the larger context of computer vision applications in the field of human-computer interaction. It also emphasizes the necessity for more research and advancement in the field of multi-camera techniques for more complex gesture identification. Some systems feature restrictions that require the user to sit in front of the camera within a certain range. Here, we attempt to loosen these restrictions. We first provide a face-based adaptive skin color model for segmenting the hand region. Second, both the static and dynamic hand motions used are easy to understand. The directional dynamic hand movements in MHI representations are finally classified using quick and easy Haar-like properties. [2].

3. Recognizing the hand gesture pictures (also known as frames) based on the shapes and orientations of the hands that are retrieved from the input video stream and captured in stable lighting and plain backdrop circumstances. Additionally, it may employ various gestural instructions to operate multimedia programs running on a computer, such as Windows Media Player, VLC Player, etc., using motions that are detected by vision. [3].

4. A.Haria, N.Asokkumar, S.Jyothi [6] gives information on the many approaches that may be used and put into practice to recognize hand gestures. Understanding the benefits and drawbacks of the various strategies is also aided by this. The camera module and the detection module are the two primary sections of the literature review. The many cameras and markers that can be utilized are identified by the camera module. The pre-processing of picture and feature extraction is handled by the detection module. They were able to develop a powerful gesture recognition system without using any markers, which made it easier to use and less expensive. We sought to provide gestures for nearly all areas of HCI in this gesture recognition system, including system functionality, application launching, and window opening some popular websites[6].

5. Vision-Based Hand Gesture by Pavlo Mylonas, Antonis, A. Argyros [8] This detailed review of several vision-based hand gesture recognition techniques and their applications is provided by the survey study. It discusses methods, data sets, and difficulties in the area. [8].

6. Hakim NL et. al., [9] explain the application of the 3DCNN model to the identification of dynamic hand gestures. RGB and depth camera data were combined to provide a superior input for deep learning. The demand for spatio-temporal characteristics increases as the recognition must function for dynamic motions. The 3DCNN model and the Long Short-Term Memory (LSTM) model were combined to address this. Finite State Machine (FSM) simplified the model's operation and increased accuracy by focusing the model's search for gesture classification into a condensed one [9].

7. Muhammad Idrees , Ashfaq Ahmad, Muhammad Arif Butt , and Hafiz Muhammad Danish[10] was able to map specific gestures for one action on the slides, such as the next slide, previous slide, zoom in and out, opening a highlighter or pen, and playing or pausing videos. This research was focused on eliminating one such distraction by letting the presenter manage slides

solely by gesturing in front of the camera. The machine learning library used in this project is Py Torch, which is written in the Python programming language. For the project, they used the 148,092 motions from the 20BN-jester Dataset V1 dataset. The outcomes were outstanding, which can make hand gestures a moral way to convey[10].

III. METHODOLOGY

Information may be extracted from the input (image/video) using a variety of ways. Differential image is one of these methods. The technology we suggested and created for recognizing hand gestures using vision had many steps. Additionally displayed in Fig. 1 is the functioning flowchart for the gesture recognition system.

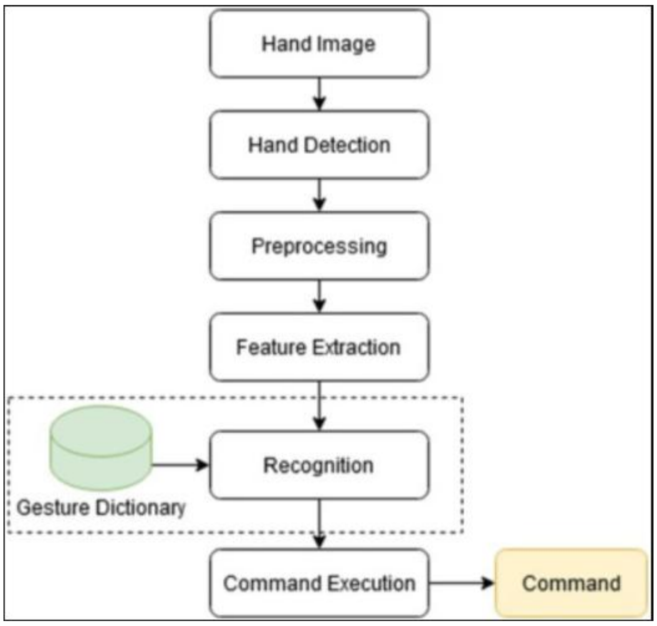


Fig1. Flow chart of hand gestures recognition.

The methodology that we used for our project consists of different phases.

1. **Hand Image:** The hand symbols that will govern your project's applications are depicted in this phase's hand pictures We utilized a collection of hand gesture photos that had been divided into many groups to represent various gestures. For the purpose of evaluating the model, the dataset was split into training and testing sets. In this research, we solely used color photos from the dataset. The folder names with 200 photographs each are "01_one," "02_two," "03_three," "04_four," "05_five," "little & index fingers," "thumb & index fingers," "thumb & little fingers," "thumb, index & little fingers," and "none." Totaling 4000 photos, there are 2000 images in the training data and 2000 images in the testing data. We may think of the dataset we used as a midrange dataset for deep learning methods.
2. **Hand Detection:** These phase detect the symbols of hands with the help of webcam.
3. **Preprocessing:** Preprocessing is an important stage in many computer vision and machine learning applications, especially those that include hand detection or gesture recognition. The preparation procedures we need to carry out might change based on the particular needs of your project and the caliber of your data. Here are a few typical preprocessing procedures.
   i. **Data Collection and Labeling:** Collect a diverse and representative dataset of hand images or videos that covers various hand poses, lighting conditions, and backgrounds. Label the data by annotating hand regions or gestures in each frame or image. Proper labeling is essential for supervised learning.
   ii. **Data Augmentation:** Augment your dataset by applying random transformations to the images or frames. Common augmentations include rotation, scaling, translation, and flipping. Data augmentation helps improve model robustness and generalization.
   iii. **Normalization:** Normalize pixel values in images or frames to a standard range (e.g., [0, 1] or [-1, 1]). Normalization helps improve model convergence during training.
   iv. **Data Splitting:** Split your dataset into training, validation, and testing subsets. This allows you to train your model on one subset, tune hyper parameters on another, and evaluate performance on a separate subset.
   v. **Data Serialization:** Serialize your preprocessed data into a suitable format for efficient training. Common formats include TF-Records or HDF5.
   vi. **Data Loader:** Implement a data loader or generator that loads batches of preprocessed data during training. This helps optimize memory usage.
   vii. **Data Normalization:** Apply normalization techniques specific to your project. For gesture recognition, you may need to calculate features like hand key points or use other methods to represent hand positions.
4. **Feature Extraction:** Feature extraction is a critical step in many computer vision and machine learning projects, including those involving hand detection and gesture recognition. Feature extraction involves transforming raw data (such as images or video frames) into a more compact and representative representation that can be used for model training and analysis.
5. **Recognition:** It is used for Recognizing the symbols and control the applications of the project.
6. Input dataset as the symbols to control hand gestures applications. The symbols are used for controlling hand gestures applications are shown in fig 2.
7. The dataset we taken can be consider as a medium dataset for deep learning techniques.

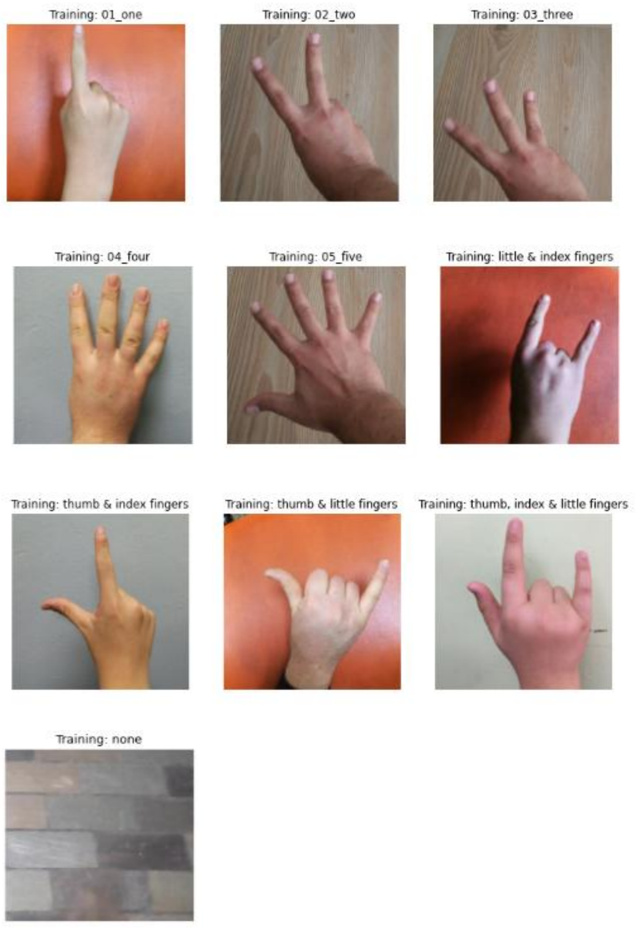

Training: 01_one
Training: 02_two
Training: 03_three
Training: 04_four
Training: 05_five
Training: little & index fingers
Training: thumb & index fingers
Training: thumb & little fingers
Training: thumb, index & little fingers
Training: none

Fig2. Images symbols for controlling hand gestures applications.

A. CNN model and VGG16 CNN model

1. Convolutional Neural Network(CNN) :

A particular deep learning technique called a convolutional neural network (CNN) is ideally suited for jobs that require the recognition and processing of images. Convolutional, pooling, and fully linked layers are just a few of the layers found in CNNs. The crucial part of a CNN is its convolutional layers, where filters are used to extract characteristics like edges, textures, and forms from input pictures. The results of the convolutional layers are subsequently transmitted to pooling layers, which are in charge of condensing the feature maps and lowering the spatial dimensions while maintaining essential data. After that, one or more fully linked layers that categorize the picture or generate predictions come next. Utilizing a sizable dataset of labeled pictures, a CNN is trained to detect patterns and attributes related to certain objects or classes. After being trained, CNNs may identify fresh pictures by classifying them or extracting features for use in a variety of tasks, including object identification and image segmentation. Numerous image recognition tasks, such as object categorization, object detection, and picture segmentation, have been successfully completed by CNNs with state-of-the-art performance. Their usefulness extends to computer vision, image processing, and related domains, with applications in anything from security systems to medical imaging to self-driving automobiles.

- **Convolutional Neural Network Design:**

A hierarchical attribute learning system is created by gradually stacking numerous layers to create a convolutional neural network. In CNNs, activation layers often come after convolutional layers to create hidden layers. The visual cortex of the human brain served as the model for this design. Each feature map in a CNN's convolution layer is coupled to a local patch in the layer before it through a filter bank. All of the feature maps in a layer utilize the same filter bank, however various filter banks are used in separate layers. With the help of this design, the network can recognize distinct local patterns in photos even when they are dispersed throughout the image. To stabilize the convolved outputs, the local weighted sums generated during filtering operations are passed through a non-linear function, often ReLU (Rectified Linear Unit). To aggregate characteristics from the convolutional layer that are semantically related, pooling techniques are incorporated into the CNN framework. In a typical CNN design, there are two to three convolutional layers with non-linear activations and pooling layers, followed by further convolutional layers with more pooling and activation. Usually, the network's last layer is a completely linked layer that handles categorization.

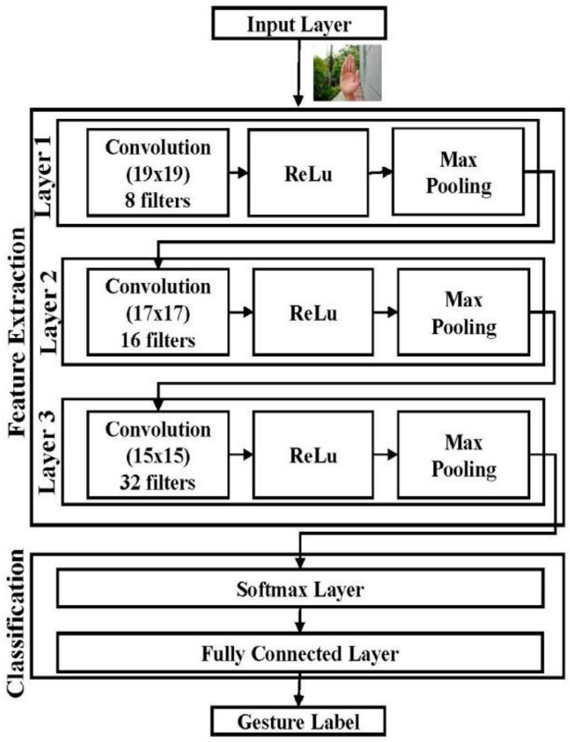


Fig3. CNN model

- **Different Types of CNN Models:**
  1. LeNet.
  2. AlexNet.
  3. ResNet.

4. GoogleNet.
5. MobileNet.
6. VGG.

- **Working of CNN model in this project**

Importing necessary libraries, such as Tensor Flow, Keras, NumPy, Pandas, Matplotlib, Seaborn, and OS, is required for code commencement. It specifies the locations of the training and testing data folders and uses Matplotlib's imshow function to display a selection of randomly chosen pictures from the training dataset that represent different classes. For model training, variables such as batch size, picture dimensions, the number of classes, and epochs are specified. By adding transformations like rotation, shifting, shearing, zooming, and flipping to the training dataset, the Keras Image Data Generator is used to improve the generalization skills of the model. During model training, this generator makes it easier for data to flow from the training and testing directories. Utilizing the Sequential API of Keras, the CNN model is built. Convolutional and MaxPooling layers are interspersed with flattened and fully linked (Dense) levels before the final layer. Dropout regularization is used to stop overfitting from happening. The categorical cross-entropy loss function and Adam optimizer are used to create the model. The fit function is used to train the model, recording the training history along with accuracy and loss values for both the training and validation sets. After training, the model is assessed using testing data, and its accuracy is shown. The accuracy curves for training and validation are displayed using Matplotlib. To calculate accuracy metrics, the model's predictions on the test data are obtained and contrasted with the actual class labels. Using the Scikit-Learn classification_report function, a classification report is produced that includes the precision, recall, F1-score, and support for each class. Using Seaborn's heatmap, a confusion matrix is created to show how the model's predictions compare to the actual labels. The code describes the whole procedure for developing, training, and assessing a CNN model for image classification, including data preparation, model design, training loop, performance assessment, and performance visualization. This provides insights into the model's accuracy and its effectiveness in classifying images within various categories.

2. VGG 16 CNN Model

The Visual Geometry Group at the University of Oxford created the Convolutional Neural Network (CNN) architecture known as VGG-16, which stands for Visual Geometry Group 16. It is praised for its simple yet efficient method for handling picture categorization challenges.

VGG-16 CNN model architecture overview:

There are 16 layers in the VGG-16, including 13 convolutional layers and 3 fully linked layers. The convolutional layers are divided into five groups, with max-pooling layers after each convolutional layer in each group. The picture categorization is carried out by the last trio of layers that are fully integrated.

1. **Convolutional and Pooling Layers:** In VGG-16, the convolutional layers employ small filters, typically 3x3 in size, to extract diverse features from the input image. As the network deepens, it becomes capable of capturing more abstract and intricate features. Subsequent to each convolutional layer, a max-pooling layer is introduced to reduce the spatial dimensions of the feature maps, thereby reducing computational demands and broadening the receptive field.
2. **Filter Sizes and Depth:** Throughout VGG-16, 3x3 filters are consistently used. Initially, the model maintains a uniform depth of 64 filters for the first two convolutional groups, which then doubles in depth for the subsequent groups (128, 256, 512, and 512). This depth progression equips the model to recognize increasingly complex patterns.

3. **Fully Connected Layers:** Following the convolutional layers, VGG-16 appends three fully connected layers, each comprising 4096 units. These layers amalgamate the high-level features learned by the convolutional layers and perform the ultimate classification.

4. **Activation Function:** Rectified Linear Unit (ReLU) activation functions are applied after each convolutional and fully connected layer. ReLU introduces non-linearity into the network, enabling it to learn intricate mappings between inputs and outputs.

5. **Dropout:** To mitigate overfitting, VGG-16 employs dropout regularization within the fully connected layers. Dropout randomly deactivates a fraction of neurons during training, thereby fostering network robustness and reducing reliance on specific neurons.

6. **Softmax Activation:** The final layer of the network utilizes the softmax activation function to yield class probabilities, rendering VGG-16 suitable for multi-class classification tasks.

7. **Output Layer:** The output layer possesses a number of units equivalent to the classes within the classification problem. The predicted class corresponds to the one with the highest probability within the softmax output.

8. **Image Preprocessing:** Typically, input images are resized to a fixed dimension, such as 224x224, and are normalized by subtracting the mean pixel values.

- VGG-16's principal contribution lies in demonstrating the efficacy of deep convolutional networks in image recognition tasks. Nevertheless, its architectural depth and parameter count make it computationally intensive and prone to overfitting when applied to smaller datasets. Despite these limitations, VGG-16 has laid the foundation for even deeper and more efficient architectures, such as ResNet and Inception, which build upon VGG's principles while introducing additional innovations.

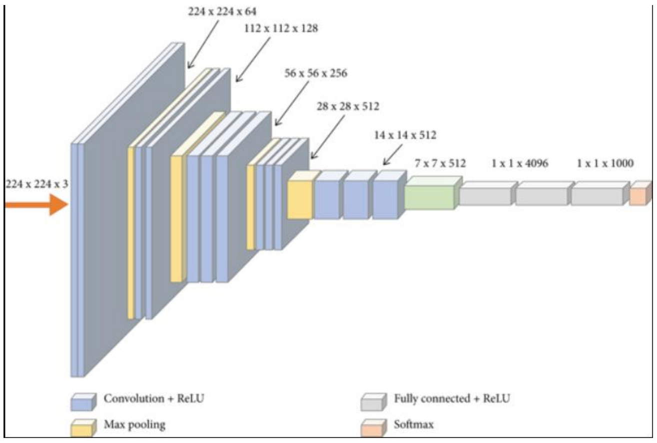


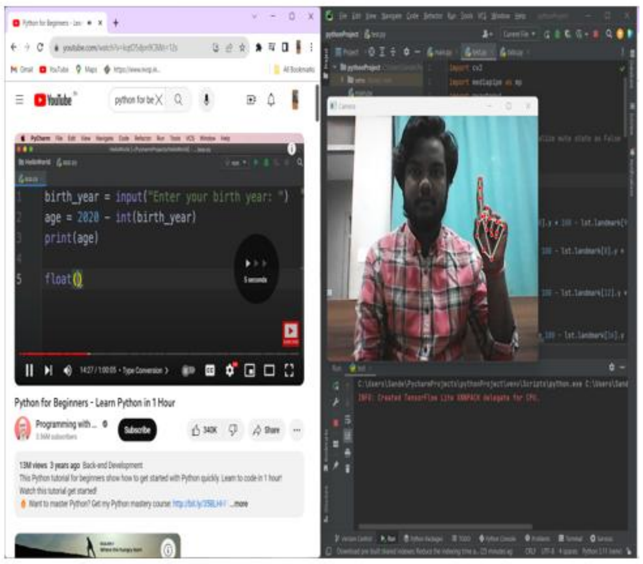
Fig4. VGG 16 CNN Model

- **Working of VGG 16 model in this project**

The script begins by importing the required libraries, including Tensor Flow, Keras, and various utilities for data processing and visualization. It then defines parameters related to the image data and training process, such as image dimensions (image_height and image_width), batch size (batch_size), number of classes (num_classes), and the number of training epochs (epochs). Next, an Image Data Generator object is created for data augmentation. This generator applies a series of image transformations to augment the training dataset, including rotations, shifts, shearing, zooming, and horizontal flipping. This helps increase the model's ability to generalize from limited data. The flow_from_directory method is used to create generators for both the training and testing datasets. These generators read and preprocess images from their respective directories while applying the specified data augmentation techniques. The pre-trained VGG16 model is loaded from Keras' applications module using the VGG16 function. The include_top parameter is set to False to exclude the fully connected layers. The input shape is specified as (image_height, image_width, 3) to match the size and color channels of the images. The script freezes the layers of the pre-trained VGG16 model by setting their trainable attribute to False. This means that only the additional layers added later will be trainable during the fine-tuning process. A new model is constructed using Keras' Sequential API. The pre-trained VGG16 model is added as the base model, followed by a flatten layer, a dense layer with 512 units and ReLU activation, a dropout layer to mitigate over fitting, and a final dense layer with num_classes units and a soft max activation function for classification. The model is compiled using the Adam optimizer with a small learning rate, categorical cross-entropy loss function, and accuracy as the evaluation metric. The model is trained using the fit function, which utilizes the data generators created earlier. Training and validation accuracies are monitored over the specified number of epochs. After training, the model's performance is evaluated on the test dataset, and the test accuracy is printed. A Matplotlib plot is generated to visualize the training and validation accuracy curves over the epochs. The model's predictions on the test data are obtained, and various accuracy metrics are computed, including a classification report with precision, recall, and F1-score for each class. Finally, a confusion matrix is visualized using Seaborn heat map, showing the distribution of predicted vs. actual class labels. In summary, this code demonstrates how to implement transfer learning with the VGG16 model for image classification using TensorFlow and Keras. It showcases the process of data augmentation, model construction, training, evaluation, and performance visualization, providing insights into the model's accuracy and classification capabilities.

IV. Results and Discussion

- **Implementation of Python codes figures**

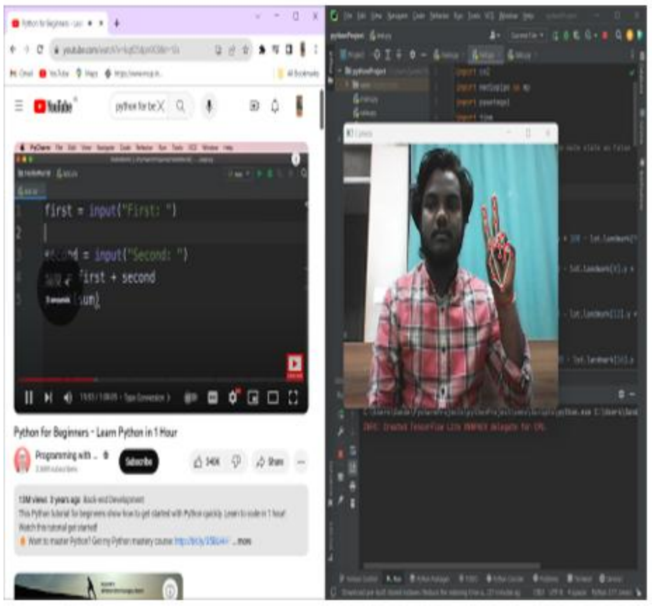


```
birth_year = input("Enter your birth year: ")
age = 2020 - int(birth_year)
print(age)

float()
```

```
C:\Users\lande\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\land
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Fig5. Forwarding gesture



```
first = input("First: ")

second = input("Second: ")
sum = first + second
sum)
```

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Fig6. Backward gesture

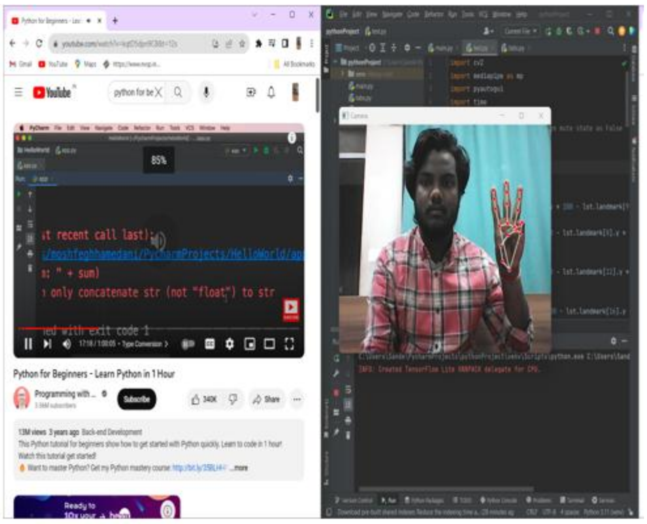


Fig7. Volume Up gesture

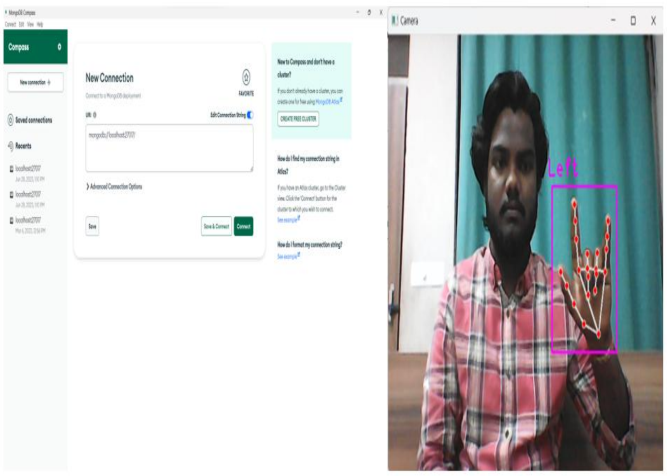


Fig10. Maximize window gesture

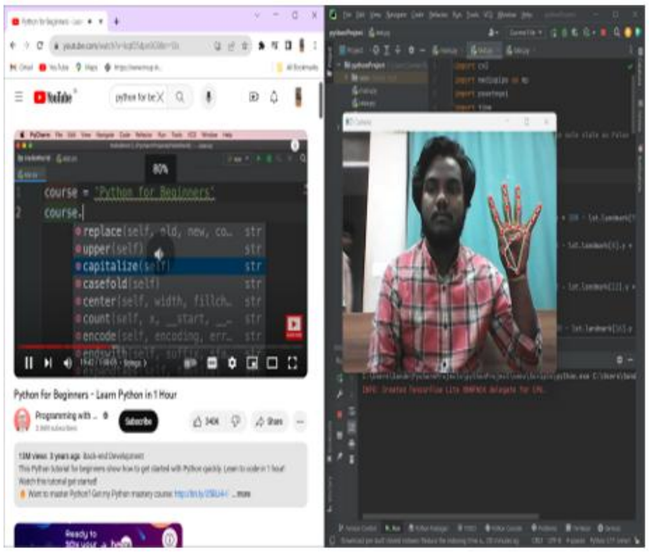


Fig8. Volume Down gesture

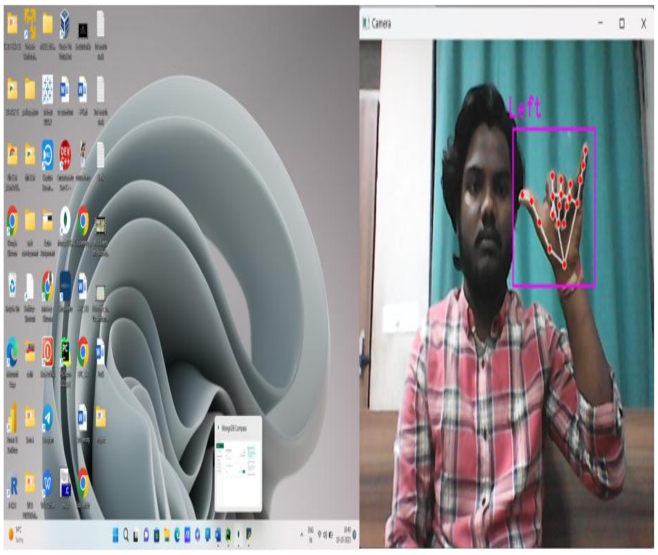


Fig11. Minimize window gesture

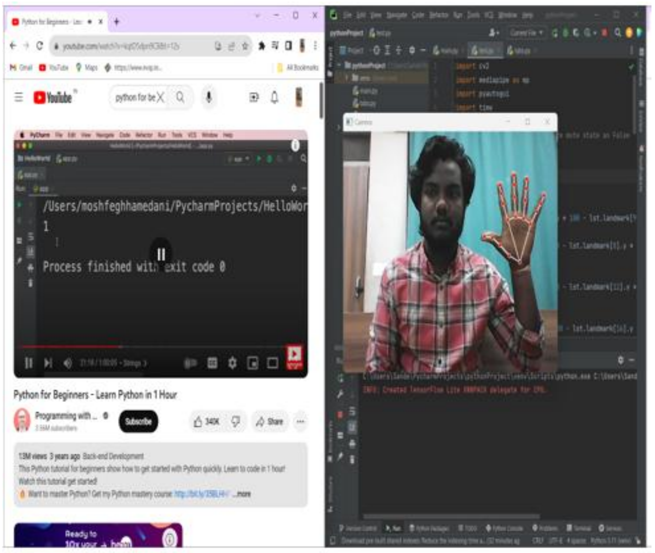


Fig9. Pause/Play gesture

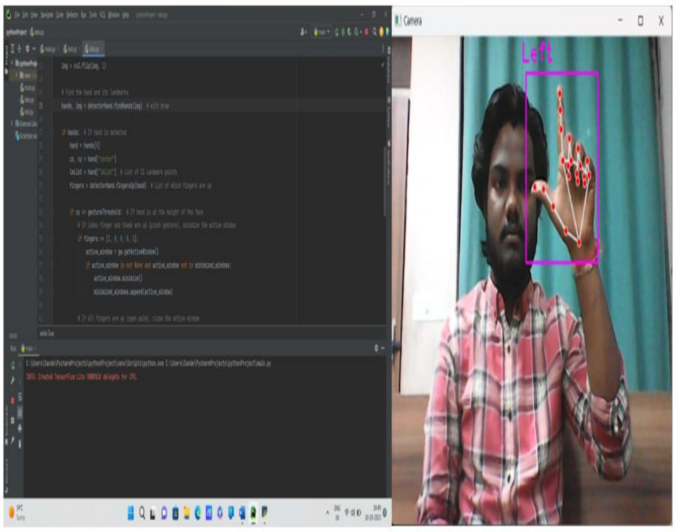


Fig12. Restore window gesture

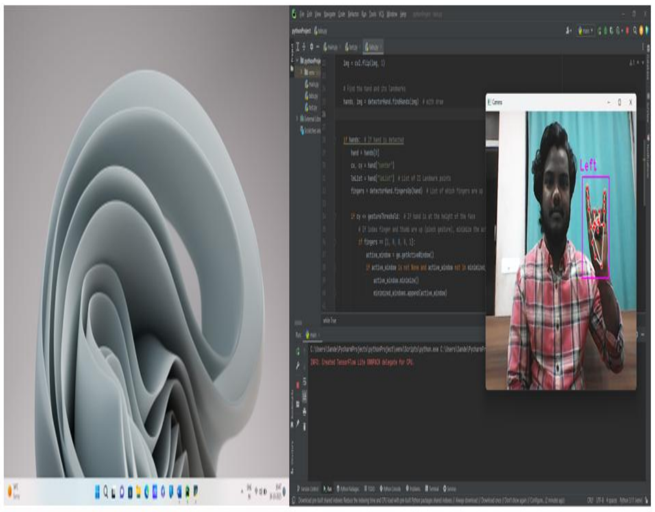


Fig13. Closing window gesture

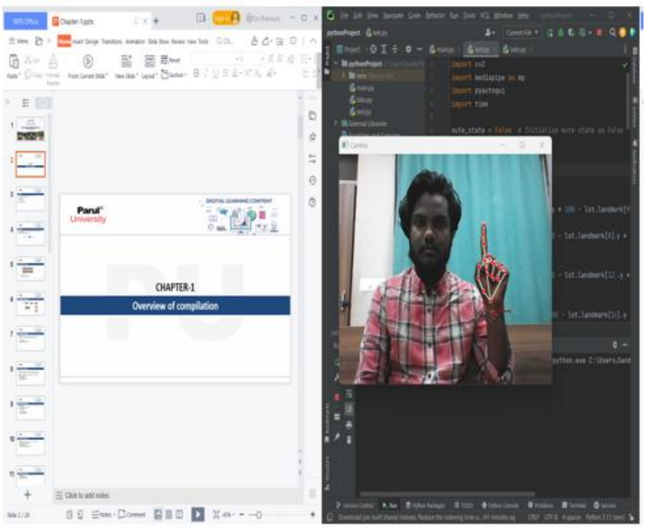


Fig14. PPT Forwarding gesture

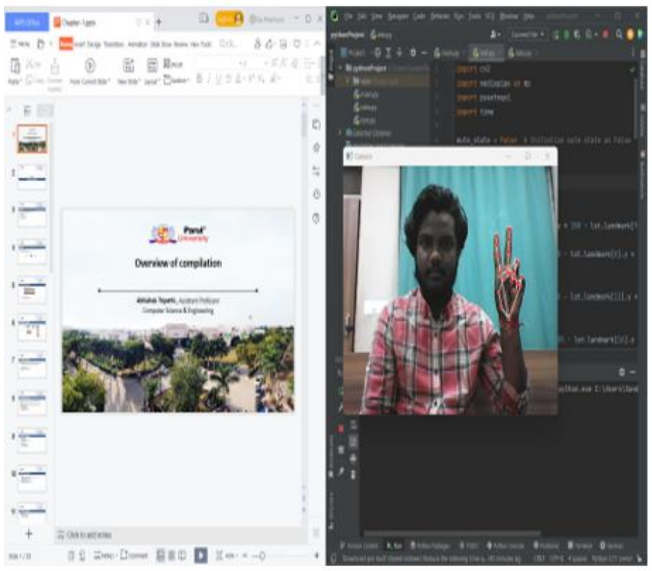


Fig15. PPT Backward gesture

- **CNN Model Performance :**

For the classification challenge, the CNN model had an accuracy rate of 81.85%. Variable performance across classes was highlighted by the categorization report. An F1-score of 1.00 was achieved by the "none" class, which was notable for its flawless precision and recall. Some classes, such "02_two" and "05_five," had accuracy and recall values that were substantially lower, yielding F1 scores of 0.71 and 0.72, respectively. The "little & index fingers" class has a high accuracy but a marginally worse recall, giving it an F1-score of 0.92. A good overall model performance was shown by the macro average F1-score of 0.82 for all classes.

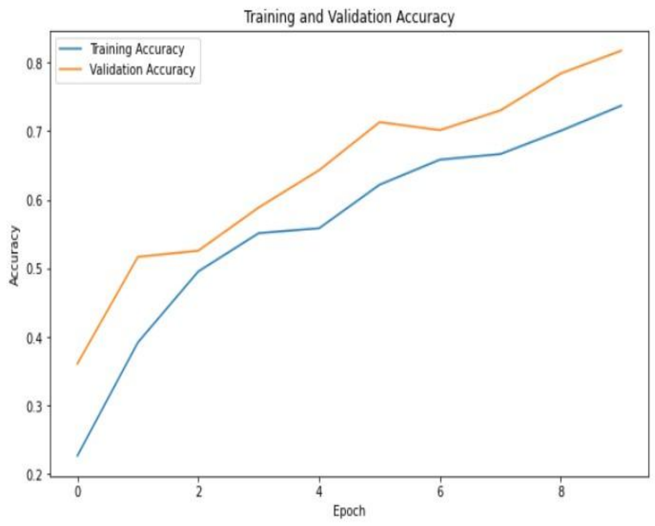


Fig16. CNN Model Accuracy Graph

TABLE-I. CNN MODEL CLASSIFICATION REPORT

	Precision	Recall	F1-score	Support
01_one	0.71	0.91	0.80	200
02_two	0.62	0.84	0.71	200
03_three	0.73	0.55	0.62	200
04_four	0.78	0.86	0.82	200
05_five	0.80	0.66	0.72	200
Little & index fingers	0.97	0.88	0.92	200
None	1.00	1.00	1.00	200
Thumb & index fingers	0.91	0.84	0.88	200
Thumb & little fingers	0.91	0.86	0.89	200
Thumb, index & little fingers	0.90	0.81	0.85	200
Accuracy			0.82	2000
Macro Avg	0.83	0.82	0.82	2000
Weighted Avg	0.83	0.82	0.82	2000

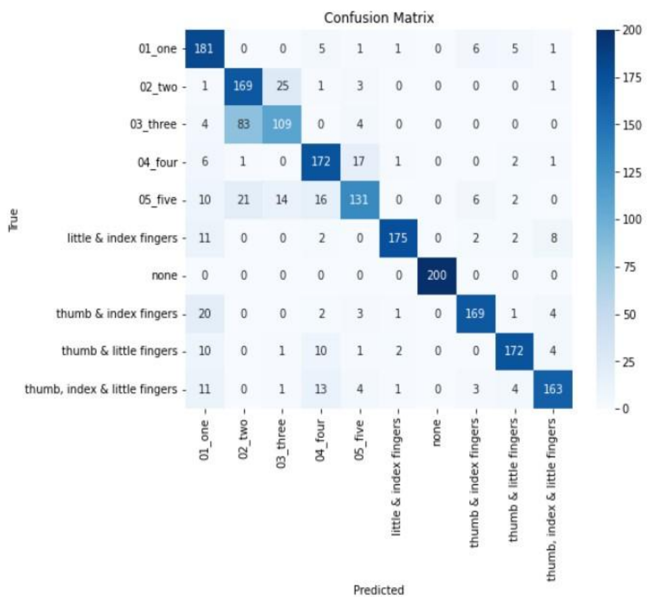


Fig17. CNN Model Confusion Matrix

- **VGG-16 Based Model Performance:**

In comparison, the VGG-16 model excelled in the identical classification job with an accuracy of 91.00%. The VGG-16 model performed well, with the majority of classes exhibiting comparatively good accuracy, recall, and F1-scores. It is noteworthy that the classes labeled "none," "thumb & little fingers," and "thumb, index & little fingers" all obtained high F1-scores of 1.00, 0.95, and 0.94, respectively. The macro average F1-score for all classes was a solid 0.92, despite some performance variances, showing great overall model performance.

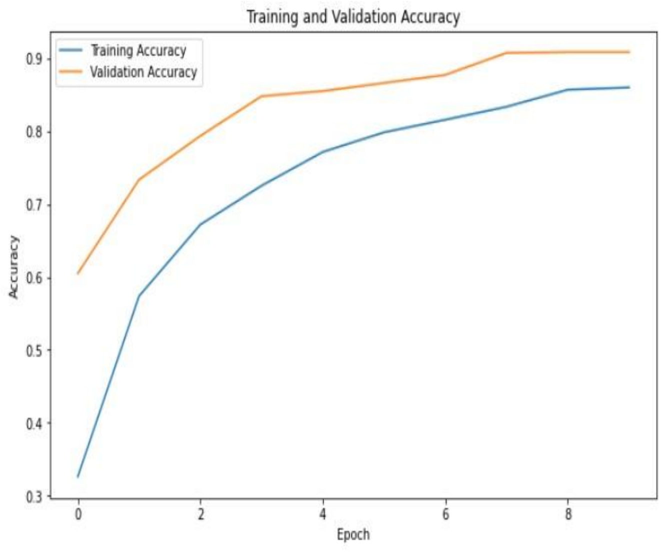


Fig18.VGG-16 Based Model Accuracy Graph

TABLE-II. VGG-16 BASED MODEL CLASSIFICATION REPORT

	Precision	Recall	F1-score	Support
01_one	0.92	0.92	0.92	200
02_two	0.89	0.87	0.88	200
03_three	0.90	0.73	0.81	200
04_four	0.93	0.91	0.92	200
05_five	0.84	0.93	0.88	200
Little & index fingers	0.92	0.95	0.94	200
None	1.00	1.00	1.00	200
Thumb & index fingers	0.95	0.92	0.93	200
Thumb & little fingers	0.94	0.95	0.95	200
Thumb, index & little fingers	0.90	0.99	0.94	200
Accuracy			0.92	2000
Macro Avg	0.92	0.92	0.92	2000
Weighted Avg	0.92	0.92	0.92	2000

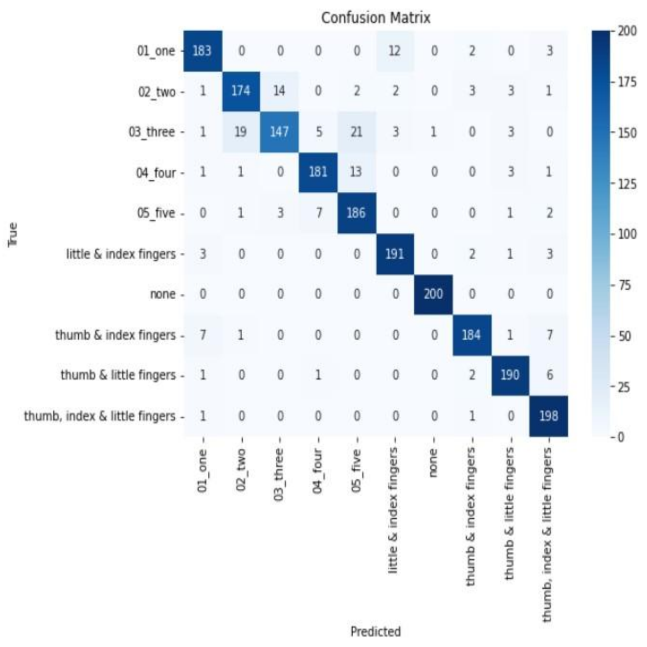


Fig19. VGG-16 Based Confusion Matrix

TABLE-III. CNN AND VGG-16 BASED MODELS COMPARISON

	CNN Model	VGG16 CNN Model
Accuracy	81.85%	91.00%
Macro Avg Precision	0.83	0.92
Macro Avg Recall	0.82	0.92
Macro Avg F1-Score	0.82	0.92
Weighted Avg Precision	0.83	0.92
Weighted Avg Recall	0.82	0.92
Weighted Avg F1-Score	0.82	0.92

- **Discussion:**

It is clear by comparing the CNN and VGG-16 models using their respective classification reports that the VGG-16 model performs better than the CNN model in terms of accuracy and F1-scores. The accuracy of the VGG-16 model was 91.00%, above that of the CNN model, which was 81.85%. Additionally, it demonstrated a steady and reliable performance with F1-scores ranging from 0.88 to 1.00, producing a solid macro average of 0.92. An outstanding example of the VGG-16 model's accuracy in categorizing the "none" category is its flawless precision and recall. The CNN model, on the other hand, performed admirably with an overall F1-score of 0.92, although it showed higher variation in accuracy and recall between classes. The choice between both models should take into account the particular job requirements. The VGG-16 model may be selected for improved overall accuracy and dependability in class predictions, while additional fine-tuning of the CNN model might be investigated to solve specific class problems, if necessary.

V. Conclusion

This study presents a revolutionary hand gesture recognition-based system for controlling media players, PowerPoint slides, and window management (minimizing, maximizing, shutting, and restoring). The approach uses Open-CV methods to capture pictures, a 2-Dimensional Convolutional Neural Network (2D CNN) for feature extraction and gesture prediction, and PyAuto-GUI to provide keyboard control upon gesture recognition. Real-time testing was done on the suggested model using a custom dataset of 10 different gestures, and the results showed an amazing accuracy rate of 91.00% when utilizing the VGG-16 CNN architecture. This technology offers a user-friendly and economical way to interface with computer systems. Enhancing gesture recognition skills in many contexts, such as those related to artificial intelligence (AI) and the medical industry, may be the focus of future initiatives.

VI. References

[1] V.Aditya, & R.Rajesh (2020). A Deep Convolutional Neural Network Approach for Static Hand Gesture Recognition. Committee of the Third International Conference on Computing and Network Communications (CoCoNet'19), Volume(171), Pages 2353-2361.

[2] Hsieh Chiung, Liou, Dunga-huna, & Lee David (2010). A Real-Time Hand Gesture Recognition System Using Motion History Image. 2nd International Conference on Signal Processing Systems (ICSPS), Volume(II), Page 394-398.

[3] Nagalapuram, G. D., Roopashree, S., Varshashree, D., Dheeraj, D., & Nazareth, D. J. (2021). Controlling Media Player with Hand Gestures Using Convolutional Neural Network. IEEE Mysore Sub Section International Conference (MysuruCon), Volume(1), Pages 1-8.

[4] Idrees, M., Ahmad, A., Butt, M. A., & Danish, H. M. (2021). Controlling PowerPoint Using Hand Gestures in Python. Webology (ISSN: 1735-188X), Volume(18), Page 1372-1388.

[5] Sakthimohan, M., Elizabeth Rani, G., Navaneethakrishnan, M., Revathi, S., Naik, B. T., & Reddy, P. V. (2023). Development of an Automated Hand Gesture Software to Control Volume for Computer. International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS), Volume(1), Pages 382-386.

[6] Haria, A., Subramaniana, A., Asokkumara, N., Poddar, S., & Nayaka, J. S. (2017). Hand Gesture Recognition for Human-Computer Interaction. 7th International Conference on Advances in Computing & Communications, ICACC, Volume(115), Pages 367-374.

[7] Al-Hammadi, M., Muhammad, G., Abdul, W., & Alsulaiman, M. (2020). Hand Gesture Recognition Using 3D-CNN Model. IEEE Consumer Electronics Society, Volume(1), Pages 2162-2248.

[8] Wachs, J. P., Kölsch, M., Stern, H., & Edan, Y. (2011). Vision-based hand-gesture applications. Communications of the ACM, 54(2), 60-71.

[9] Hakim, N. L, Timothy K.Shih, Kasthuri Arachchi.S. P, Aditya Wisnu, Yi-Cheng, Chih-Yang. (2019). Dynamic Hand Gesture Recognition Using 3DCNN and LSTM with FSM Context-Aware Model. Volume(1), Pages 1-19.

[10] Granit Luzhnica, Elizabeth Lex, Viktoria Pammer. A Sliding Window Approach to Natural Hand Gesture Recognition using a Custom Data Glove. In: 3D User Interfaces (3DUI); 2016 IEEE Symposium on ; 2016 Mar 19 ; New York : IEEE; 2016 ; p.81-90.

[11] Raimundo F. Pinto Jr. , Carlos D. B. Borges, Antoˆnio M. A. Almeida , and Ia´lis C. Paula Jr. Static Hand Gesture Recognition Based on Convolutional Neural Networks. Volume 2019, Article ID 4167890, 12 pages.

[12] Ahmad Puad Ismail , Farah Athirah Abd Aziz , Nazirah Mohamat Kasim and Kamarulazhar Daud. Hand gesture recognition on python and opencv. Ahmad Puad Ismail et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1045 012043. 11 pages.

[13] Thippa Reddy Gadekallu,Mamoun Alazab ,Rajesh Kaluri , Praveen Kumar Reddy Maddikunta , Sweta Bhattacharya,Kuruva Lakshmanna,Parimala. Hand gesture classification using a novel CNN-crow search algorithm. Complex \& Intelligent Systems (2021) 7:1855–1868.

[14] Raj Kumar E M, Smitha G Adiga, Md Altaf Raja, T V Anuhya, Dr. C Nandini . Controlling multiple applications with hand gesture using CNN. Publised by International Journal of Creative Research Thoughts (IJCRT) . Volume 10, Issue 4 April 2022 | ISSN: 2320-2882.

[15] Suguna R , Rupavathy N , Asmetha Jeyarani R . A Smart Vision Based Single Handed Gesture Recognition system using deep neural networks. I3CAC 2021, June 07-08 ,11 Pages.

[16] Stella Nadar, Simran Nazareth, Kevin Paulson, NilambriNarkar. Video controlling using hand gestures for disable people. International Journal of Trendy Research in Engineering and Technology. Volume 5 Issue 2 April 2021,ISSN No 2582-0958.

[17] Rajalakshmi J, Kumar P . Gesture Recognition using CNN and RNN. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878 (Online), Volume-9 Issue-2, July 2020.

[18] Kaustubh Jadhav , Abhishek Jaiswal , Abbas Munshi , Mayuresh Yerendekar. Sign language recognition using neural network. Volume 1 Issue 1,I SSN: 2581-4419.

[19] Gaurav Sawardekar, Parthil Thaker, Rishiraj Singh, Vaishali Gaikwad (Mohite). Arduino based Hand Gesture Control of Computer Application. October 29 ,2018 . 6 Pages.

[20] Rutwik Shah, Vinay Deshmukh, Viraj Kulkarni, Shatakshi Mulay, Madhuri Pote. Hand Gesture Control Car. International Journal of Engineering Research \& Technology (IJERT) ,Published by ICSITS - 2020 Conference Proceedings, Volume 8, Issue 05, ISSN: 2278-0181.