Hand Gesture Recognition For MP3 Player Using Convolution Neural Network

Sushma Reena P

Synopsis

This report presents a study on static hand gesture recognition for controlling an audio player using Convolutional Neural Networks (CNNs). The study aimed to develop a system capable of accurately identifying a set of predefined hand gestures from images captured by a webcam. The study involved collecting a dataset of hand gesture images made by the user and performing basic operations such as play, pause, volume up, volume down, next, and previous. and training a CNN model on this dataset. The model was then tested on a separate set of hand gesture images to evaluate its performance. The results of the study showed that the CNN model achieved high accuracy in recognizing hand gestures, demonstrating the effectiveness of CNNs in this task. The study contributes to the development of hand gesture recognition systems, which have applications in various fields such as human-computer interaction and sign language recognition.

Introduction

Traditional methods of interacting with computers, such as using a keyboard and mouse or a touch screen, can be difficult to do or manage and take a lot of time and effort and limit the ability of humans to express themselves naturally. Hand gestures are a natural and intuitive means of communication that humans use in their everyday lives. They are a non-verbal way of conveying information and can be used to express emotions, intentions, and commands. Hand gestures are also more expressive and useful in many ways than traditional methods of input, such as typing on a keyboard. Therefore, researchers and engineers began exploring ways to leverage hand gestures for human-computer interaction. This led to the development of hand gesture recognition technology, which enables computers to understand and interpret hand gestures made by humans. Hand gesture recognition technology has the potential to revolutionize the way humans interact with computers and devices, providing a more natural and intuitive means of communication. It can also enable new applications and use cases, such as virtual reality, gaming, and robotics.

One of the applications of hand gesture recognition technology is controlling an MP3 player. With this technology, users can control music playback, volume, and other features of an MP3 player by simply making hand gestures. This technology can provide a more convenient and hands-free way of controlling an MP3 player, especially when the user is engaged in other activities, such as exercising, driving, or working. The use of hand gestures can also help to reduce distractions and increase safety in certain situations, such as driving. This report focuses on the development of a static hand gesture recognition system that can be used to control an MP3 player. The system utilizes a webcam with the help of OpenCV to capture hand gesture images and save them as a dataset, which is then trained by PyTorch using computer vision algorithms (i.e., CNN) and tested to recognize the intended hand gesture. The recognized gesture is then mapped to a specific command such as play, pause, previous, next, volume up and volume down which are used to control the MP3 player. The report further discusses the design and development of hand gesture recognition and presents the results of testing and evaluation and concludes by highlighting the potential benefits and limitations of the hand gesture recognition technology in controlling an MP3 player.

Literature Review

Hand gesture recognition for controlling an MP3 player has been a topic of interest for researchers for over two decades. The history of this research area can be traced back to the early 2000s when researchers started exploring the potential of using hand gestures as a natural and intuitive interface for interacting with digital devices. In the early years, the focus was mainly on developing hand gesture recognition systems for controlling computer systems, but with the advent of portable music players such as the iPod, researchers started exploring the use of hand gestures for controlling music players.

One of the earliest works on hand gesture recognition for controlling an MP3 player was done by W. Gao and H. Ai in 2006, where they proposed a system that used a single camera to recognize hand gestures for controlling a music player. The system used a rule-based approach for recognizing gestures, where a set of rules was defined to map the hand gestures to corresponding actions. The system relied on a set of predefined rules to map the hand gestures to corresponding actions. For example, if the user made a "swipe right" gesture, the system would recognize it and perform the corresponding action, such as skipping to the next track. The system achieved an accuracy of 85%.

In 2008, A. Dipietro and L. Vezzaro proposed a system that used a colour camera to recognize hand gestures for controlling an MP3 player. The system used a combination of image processing techniques such as skin colour segmentation, edge detection, and template matching to recognize hand gestures. The system first segmented the hand region using skin colour segmentation and then applied edge detection to extract the contour of the hand. Finally, the system used template matching to recognize the gesture based on the extracted contour. The system achieved an accuracy of 91.5%.

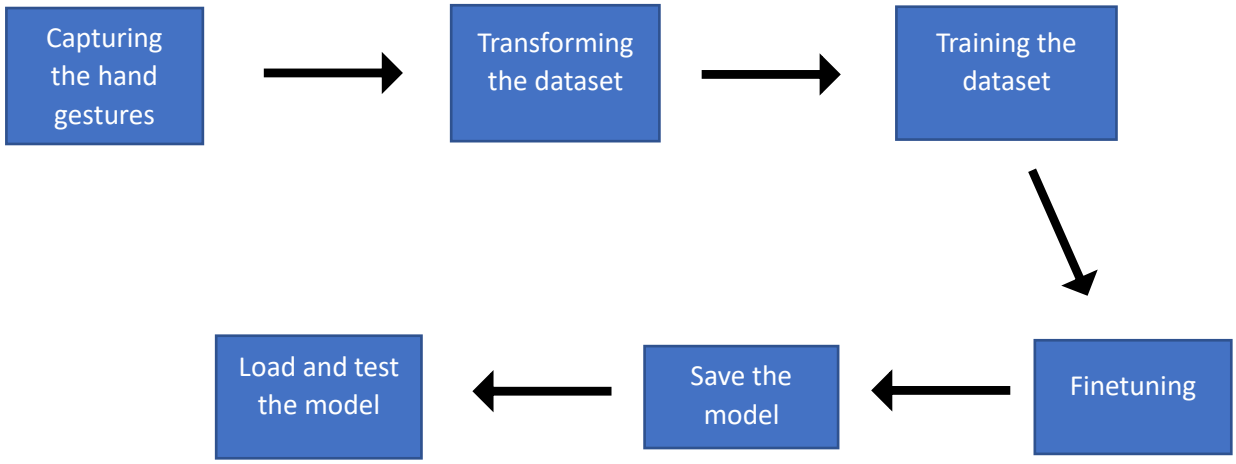
The above two systems did not utilize any machine learning algorithms. The accuracy of these rule-based and image-processing techniques depends on the ability of the system to accurately detect and analyse the relevant physical characteristics of the hand, as well as the complexity and variability of the gestures being recognized. Simple gestures with distinct physical characteristics, such as "swipe left" or "swipe right," may be recognized with high accuracy using these techniques. However, more complex gestures with subtle variations in movement or orientation may be more difficult to recognize accurately. It's worth noting that while these techniques can achieve a certain level of accuracy, they may not be as flexible or adaptable as machine learning algorithms, which can learn from large datasets of examples and generalize to new examples. As a result, many recent hand gesture recognition systems have incorporated machine learning algorithms to achieve higher accuracy and flexibility.

In 2012, J. Choi et al. proposed a system that used a Kinect sensor for recognizing hand gestures to control a music player. The system utilized the depth information provided by the Kinect sensor to recognize the gestures accurately. The system employed a machine learning algorithm called the Hidden Markov Model (HMM) to recognize the gestures. The system first trained the HMM using a dataset of hand gesture sequences and then used the trained model to recognize the gestures in real time. The system achieved an accuracy of 95%.

In recent years, with the advancement in machine learning techniques and the availability of large datasets, researchers have started exploring the use of deep learning algorithms for hand gesture recognition. For example, in 2019, E. M. Hemalatha et al. proposed a real-time hand gesture recognition system that used the convolutional neural network (CNN) algorithm to recognize hand gestures. The system first captured the hand gesture using a camera and then pre-processed the image to remove noise and normalize the lighting. The pre-processed image was then passed through the

CNN, which classified the gesture into one of the predefined classes. YOLO (You Only Look Once) object detection algorithm to detect the hand and data augmentation techniques to improve the performance of the CNN model and Transfer learning, where a pre-trained CNN model was used to extract the features and then trained on a smaller dataset. The system achieved an accuracy of 98.5%.

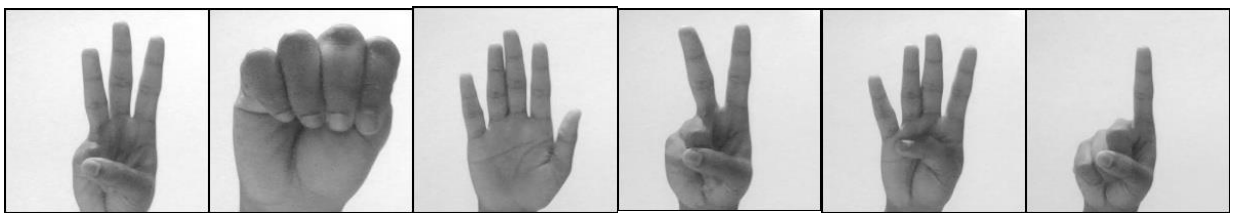
Procedure



Flowchart :1

Data Collection:

In this project collection of data refers to the process of collecting the hand gesture images and save under the corresponding gesture named folder. We collected the images by capturing the gesture with the help of a webcam by converting the frame to gray before capturing the image using OpenCV. OpenCV (Open-Source Computer Vision Library) is used to convert the frame into grey, capture, read and save the images and destroy the windows being used. The dataset consists of six hand gesture sets which consist of 100 images each for the training dataset and 50 images each for the test dataset. The six hand gestures represent six commands used to control the MP3 player. Following are the hand gestures used,



a) previous b) play c) pause d) next e) volume down f) volume up

Fig:1

Data Pre-processing:

We have pre-processed the dataset by transforming and normalizing the data of both training and testing using PyTorch. PyTorch is a fully featured framework for building deep learning models, which is a type of machine learning that's commonly used in applications like image recognition and language processing. Transformation is an important step in hand gesture recognition to pre-process the input data and make it suitable for the neural network to learn. The 'Compose' function from the 'transforms' module is used to chain the transformations together for each set of data. The transformations applied in the training data are resizing the image to (224, 224), converting the image to grayscale, randomly flipping the images horizontally, converting the image to tensor, and normalizing it with a mean and standard deviation of 0.5. For the testing data, resizing to (224, 224), converting the image to grayscale, converting the image to tensor, and normalizing it with a mean and standard deviation of 0.5 is performed.

Neural networks require a fixed input size, so we may need to resize the input image to a fixed size before feeding it to the network. This can help to reduce the computational complexity and make the training more efficient. Normalization helps to reduce the differences in brightness, contrast, and colour between the input images. This makes the network more robust to changes in lighting conditions and improves its ability to generalize to new data. Grayscaling the images can significantly reduce the amount of memory and computation required by the neural network during training and inference. This is often done in image processing and computer vision applications as it can simplify the image representation, reduce the dimensionality of the data, and speed up processing. Additionally, grayscale images can preserve important features such as edges and textures that can be useful for object recognition tasks. Augmenting the data can help to increase the size of the training dataset and improve the network's ability to generalize to new data. After loading the data and applying the transformation, the data transformed as the following,

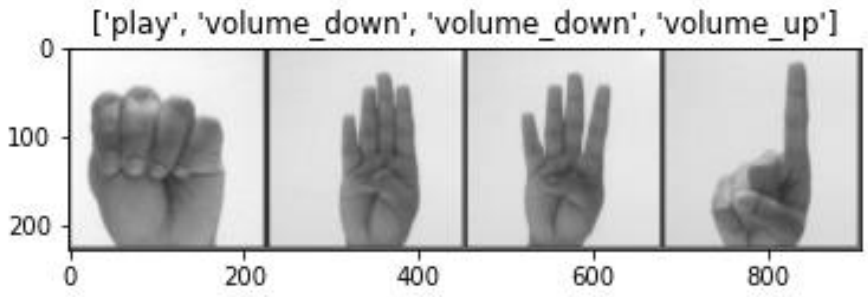


Fig:2

Training the data:

We have used **Convolutional Neural Network (CNN)** algorithm to train our model in this project. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data. There are four layers in CNN. The first layer is the convolutional layer which puts the input images through a set of convolutional filters, each of which activates certain features from the images. The output of the convolutional layer is then passed through an activation function such as ReLU (Rectified Linear Unit) to introduce non-

linearity. Rectified linear unit (ReLU) allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as activation because only the activated features are carried forward into the next layer. The output of the activation function is then passed through a pooling layer, which simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn. This process of convolution, activation, and pooling is repeated multiple times to extract more and more complex features from the input image. The final output is typically passed through one or more fully connected layers, which give the outputs a vector of K dimensions (where K is the number of classes able to be predicted) and contains the probabilities for each class of an image being classified. The final layer of the CNN architecture uses a classification layer to provide the final classification output.

A pre-trained **ResNet-18 model** with the default weights provided by **PyTorch** is used as the base architecture of the CNN which is a variant of the original ResNet (Residual Network) architecture and designed to address the problem of vanishing gradients in very deep neural networks. The ResNet-18 architecture consists of 18 layers, including 16 convolutional layers and 2 fully connected layers. It uses skip connections to enable the gradient to flow directly through the network without encountering the vanishing gradient problem. These skip connections are added between every two convolutional layers, allowing the input to bypass some layers and flow directly to the next layer. This architecture has proven to be very effective in image classification tasks, and it has achieved state-of-the-art performance on many benchmark datasets, such as ImageNet, which has millions of images and thousands of classes. In addition to image classification, ResNet-18 can also be used for other computer vision tasks, such as object detection and segmentation. The ResNet-18 architecture has become a widely used architecture for many computer vision applications due to its simplicity, effectiveness, and efficiency.

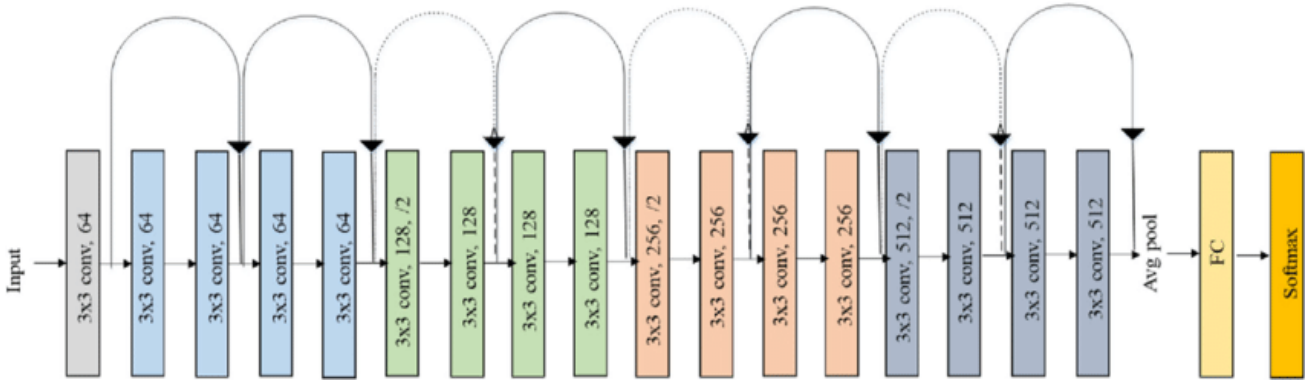


Fig:3

In this project **Transfer Learning** is used, which is the reuse of a pre-trained model on a new problem. It's currently very popular in deep learning because it can train deep neural networks with comparatively little data. It has two ways to customize a pre-trained model, which are feature extraction and fine-tuning. Here fine-tuning is used. Transfer learning is used by loading a pre-trained ResNet-18 model from the PyTorch 'models' library and achieving state-of-the-art performance. The pre-trained ResNet-18 model is then modified by replacing the fully connected layer with a new one consisting of two linear layers and a dropout layer in between, which is randomly initialized and has the appropriate number of output units for the classification task at hand (in this case, 6 classes). By

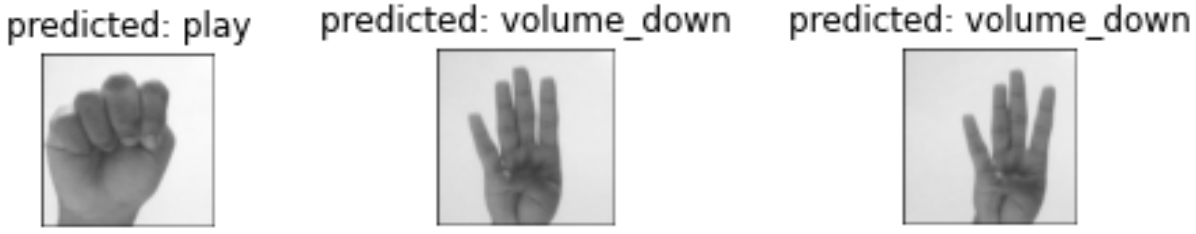
using a pre-trained model, the network can take advantage of the learned features and weights from the ImageNet dataset, which can help to improve the accuracy and speed up the training process on the new dataset. During training, only the weights of the newly added layer are updated, while the weights of the pre-trained layers are frozen, as they already contain useful features for image recognition. This approach is known as **fine-tuning**, where a pre-trained model is further trained on a new dataset to adapt its learned features to a new task.

Fine-tuning is used to Unfreeze a few of the top layers of a frozen model base and jointly train both the newly-added classifier layers and the last layers of the base model. This allows us to "fine-tune" the higher-order feature representations in the base model in order to make them more relevant for the specific task. Fine-tuning a pre-trained model is a common and effective way to perform transfer learning in deep learning, as it can save time and resources and improve the performance of the model on the new dataset.

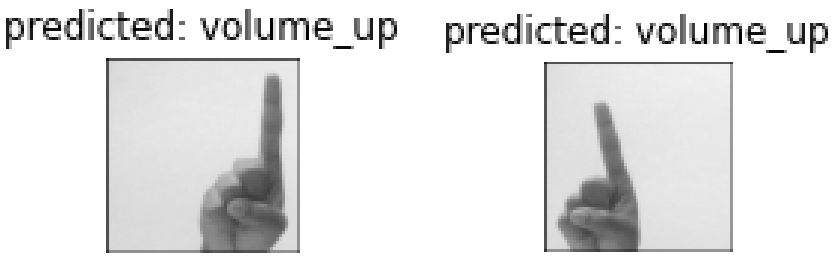
In a classification task, the model is required to assign a class label to each input image. The **cross-entropy loss** measures the difference between the predicted class probabilities and the true class probabilities. It is commonly used as the loss function for classification tasks. For each input image, the cross-entropy loss is calculated by taking the negative logarithm of the predicted probability for the true class label. The loss is then averaged over all the input images in a batch. The goal of the training process is to minimize this loss by adjusting the weights of the neural network.

The optimization algorithm used to train the model uses the stochastic gradient descent (SGD) algorithm to update the model parameters during training. **SGD** is a widely used optimization algorithm in deep learning and is often used as a baseline optimization algorithm. During training, it computes the gradients of the loss with respect to the parameters of the model and updates the parameters in the direction of the negative gradient. The learning rate (lr) determines the step size of the update, while the momentum parameter controls how much of the previous gradient direction should be incorporated into the current update. The algorithm uses 0.001 as the learning rate and 0.9 as the momentum.

The model's parameters are updated using the specified optimizer and loss function and the performance is evaluated on both the training and validation sets. The best model parameters are saved, based on the test loss and generate a confusion matrix for the test set, using the trained model to predict the class labels for each image in the test set. The confusion matrix shows the true and predicted class labels for each gesture class, allowing the user to assess the model's performance in recognizing each gesture. Then iterating through each epoch and for each epoch, iterating through the training and testing phases gives the loss and accuracy for both training and testing. By comparing the epochs, we obtained the best epoch which has high test accuracy and low test loss as the best model. After visualising the best model, we got the predicted model as follows,



a) b) c)



d) e)

Fig:4

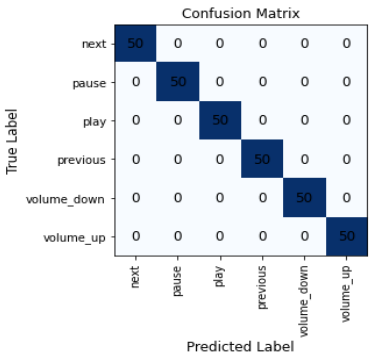
Observation:

While training the model, we have run 5 epochs to find the best model. The following table shows the observation of those epochs,

Epoch	Training Loss	Training Accuracy	Test Loss	Test Accuracy
1	0.8389	0.6917	0.0737	1.0000
2	0.2235	0.9383	0.0728	0.9800
3	0.1082	0.9700	0.0270	0.9900
4	0.1159	0.9650	0.0298	0.9900
5	0.1199	0.9650	0.0032	1.0000

Out of all these epochs, the best epoch is

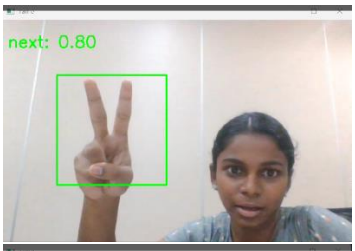
```
Training complete in 6m 56s
Best test Acc: 1.000000
Best test loss: 0.003153
Best confusion matrix:
[[50  0  0  0  0  0]
 [ 0 50  0  0  0  0]
 [ 0  0 50  0  0  0]
 [ 0  0  0 50  0  0]
 [ 0  0  0  0 50  0]
 [ 0  0  0  0  0 50]]
```

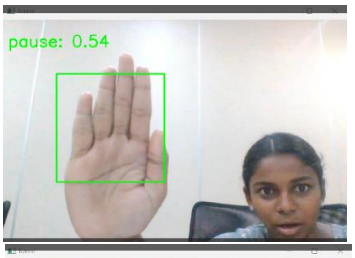


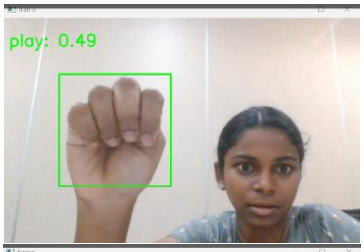
Data validation:

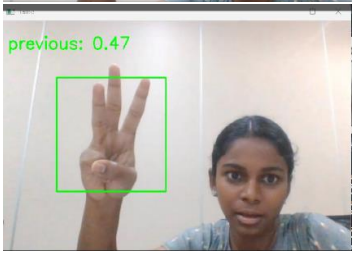
In validation, PyTorch is used to load the obtained trained model. Image transformation is used to pre-process the input image before passing it through the model for prediction. The transformations applied include resizing the image to a fixed size of (224, 224) pixels as required by the ResNet-18 model, converting the image to grayscale with 3 output channels (to match the input channels required by the ResNet-18 model), converting the image to a PyTorch tensor and normalizing the pixel values of the image to have a mean and standard deviation of 0.5 (as required by the ResNet-18 model). The

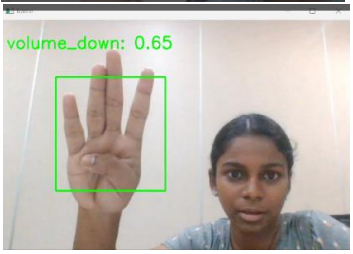
hand gesture is then predicted using webcam by capturing the real-time gestures. The prediction is done using the loaded pre-trained model. The frame will show the predicted gesture name and their probabilities. The probability is computed using softmax for each class and selects the gesture name with the highest probability as the predicted gesture name. The predicted hand gestures are as follows,

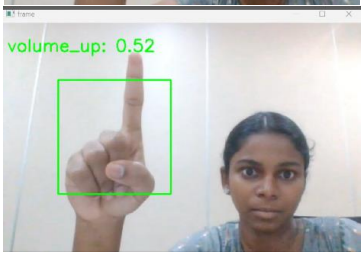












Conclusion

The hand gesture recognition system implemented in this project can accurately recognize six different hand gestures using a ResNet-18 deep learning model trained on a custom dataset of hand gesture images. The system may control an MP3 player based on the recognized gestures, allowing for hands-free control of music playback. The use of transfer learning and data augmentation helped improve the accuracy and efficiency of the model. Overall, this project demonstrates the potential of deep learning-based hand gesture recognition systems for real-world applications such as human-computer interaction and accessibility. However, there is still room for improvement in terms of probabilities, accuracy, speed, and robustness, which could be achieved through further optimization and refinement of the model and system.

Future Scope

There are several areas where the hand gesture recognition project for controlling an MP3 player can be further developed:

Multi-modal input: In addition to using only video input, the project can be expanded to use multiple modalities of input such as audio and sensors. This will help in making the system more robust and able to perform well even in low-light conditions.

Gesture variations: The current implementation recognizes only a few hand gestures. To make the system more versatile, additional gestures can be added to control other features of the MP3 player.

Real-time performance: The current implementation works in real-time, but there is still room for improving the speed and accuracy of the recognition algorithm. This can be achieved by using more advanced machine learning models or by optimizing the current implementation.

Integration with other applications: The project can be integrated with other applications such as virtual assistants, smart homes, and other audio and video applications. This will make the system more versatile and useful.

Scalability: The current implementation is designed to work on a single device. To make it scalable, it can be deployed on cloud infrastructure or on edge devices such as Raspberry Pi or other low-power devices.

Overall, the project has great potential for further development and can be extended to several other applications. The use of hand gestures for controlling devices is an emerging trend, and with advancements in machine learning and computer vision, we can expect to see more advanced and sophisticated gesture recognition systems in the future.

Reference

1. A. Dipietro and L. Vezzaro, "Gesture Recognition for MP3 Players: A Comparative Study," in Proceedings of the 2008 IEEE International Conference on Multimedia and Expo, 2008, pp. 1457-1460.
2. E. M. Hemalatha et al., "Real-time hand gesture recognition for controlling multimedia player using machine learning," Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 2, pp. 731-740, 2019.
3. https://builtin.com/data-science/transfer-learning
4. https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html
5. https://www.nvidia.com/en-us/glossary/data-science/pytorch/
6. https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
7. https://www.tensorflow.org/tutorials/images/transfer_learning
8. J. Choi, K. Lee, and I. Kim, "Hand Gesture Recognition Using a Kinect Sensor," Sensors, vol. 14, no. 4, pp. 7662-7680, 2014.
9. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

10. P.S. Neethu, R. Suguna, Divya Sathish, "An efficient method for human hand gesture detection and recognition using deep learning convolutional neural networks" Springer-Verlag GmbH Germany, part of Springer Nature 2020.
11. Sakshi Sharma, Sukhwinder Singh, "Vision-based hand gesture recognition using deep learning for the interpretation of sign language," ECE Department, Punjab Engineering College (Deemed to be University), Chandigarh, India
12. W. Gao and H. Ai, "A Hand Gesture Recognition System for Controlling Music Players," in Proceedings of the 2006 IEEE International Conference on Multimedia and Expo, 2006, pp. 101-104.