REVOLUTIONIZING INTERACTION: VIRTUAL MOUSE AND SPACE BAR CONTROL THROUGH HAND GESTURE DETECTION

MUBASHWIRA ZAYAN,
FARHANA AKTER OESHI,
ARIFUL ISLAM SARKER
Department of Electrical and Computer Engineering, North South University

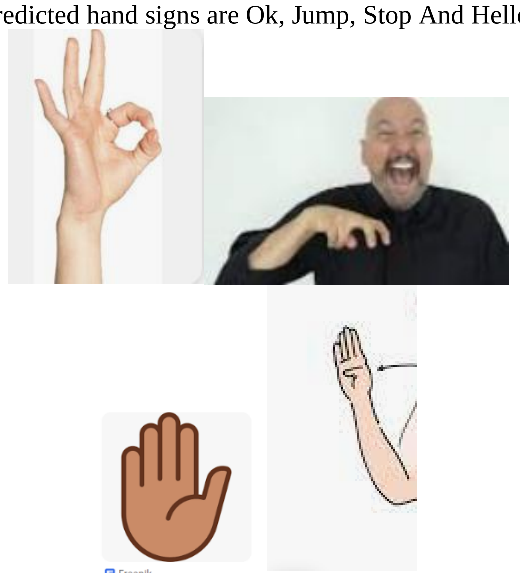
I. ABSTRACT

In today's digital age, the fusion of technology and human expression has paved the way for innovative and engaging interactive experiences. "Gesture-Dino" is a cutting-edge game that leverages hand signs as a means of control, creating a unique and immersive gaming environment. Players use a virtual mouse to interact with the game, while hand signs serve as intuitive gestures to navigate and manipulate the virtual world.
Our project decodes four kinds of hand gestures from the American Sign Language with near-to-perfect accuracy. Moreover, the virtual mouse acts as a guideline to drive the game. As a result, our game blends traditional mouse-based input with the expressive power of hand signs, allowing players to make the lonely dinosaur 'jump' across several obstacles by making specific gestures. By using a webcam or motion-sensing technology, the game detects and interprets the hand signs, providing a seamless and dynamic user experience.
"Gesture-Dino" introduces a novel approach to gaming, encouraging players to explore their creativity and coordination in a fun and interactive manner. This abstract concept aims to promote not only entertainment but also a new way of interacting with technology through the physicality of hand gestures, offering an exciting and immersive gaming experience.

Keywords— sign language detection, ASL, virtual mouse, keyboard, gamejump, hand signs

II. INTRODUCTION

In the modern era of technology, interactive gaming experiences have evolved significantly, offering immersive and engaging ways to interact with virtual worlds. "Gesture-Dino" is an innovative game that incorporates hand signs alongside traditional mouse control to create a unique and captivating gaming environment. This introduction outlines the problem addressed by the game, discusses the existing approaches in the field, identifies limitations, and presents a brief overview of how these limitations are mitigated. Along with these, we'll also provide some features which assisted in making our game unique in some circumstances. Our predicted hand signs are Ok, Jump, Stop And Hello:



A. Problem Statement

The problem at the core of this interactive gaming concept is the need to enhance the user experience by introducing a novel and intuitive control mechanism. Traditional mouse-based gaming interfaces can sometimes lack the tactile and immersive qualities that players desire. Conversely, purely gesture-based gaming can be challenging to control precisely, leading to frustration for many players. Therefore, this new approach of gaming is extremely user-friendly so it's safe to say that toddlers or even elderly people can enjoy it.

B. Existing Solutions

Various attempts have been made to address this problem. Some games incorporate gesture recognition technologies that rely solely on hand movements, but these often lack the accuracy and responsiveness required for precise control. Others provide touch or motion-sensing controls, which have their limitations and may not offer the same tactile feedback as a traditional mouse.

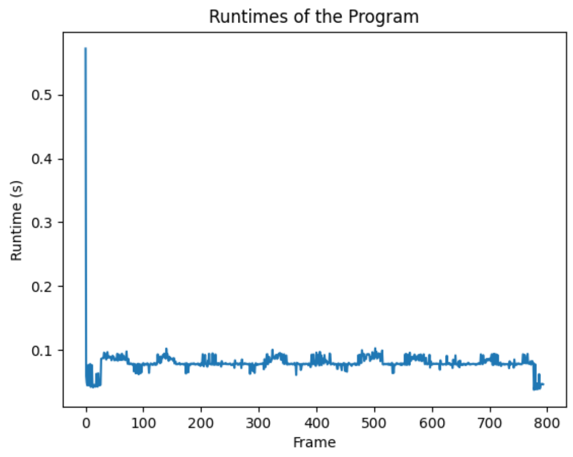
C. Limitations

Since our game is mostly gesture-based, accuracy can be a hassle. It can sometimes take a while for the virtual mouse to move up or down.

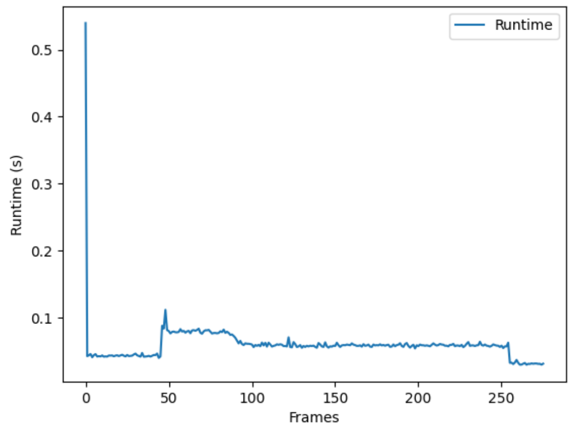
Other accuracy-related problems with our game included the dinosaur's constant hopping in defiance of our wishes which made it difficult to control.
Moreover, learning and memorizing a wide range of hand gestures can be challenging for players, hindering accessibility.

Time limitation Graph:

For Gamejump Runtime:



For Torch Runtime:



D. Solving Limitations

For preventing the constant jumping, we included a timer to limit the gaps between the jumps. This made the dinosaur jump in a more seamless way. So, by combining the best of both worlds – mouse control and hand signs – "Gesture-Dino" aims to offer an inclusive and enjoyable gaming experience, overcoming the limitations of previous solutions and revolutionizing the way players interact with virtual environments.

E. Unqiue Contributions

- Our project manipulates the direction of the mouse as well as spacebar completely with the assistance of hand gestures.
- Multiple hand gestures can be detected at the same time.
- Users don't require a GPU to play the game which is a core requirement for other related projects.
- It's a strategy that's easy to apply because we've built a complex project using basic techniques. Other projects involve CNN, tensor, pytorch, etc. but we've managed to use mediapipe, neural network, pyautogui, OpenCV, randomforest classfier, and numpy.

III. RELATED WORK

After researching ten papers, we narrowed it down to the following. Certain games use hand-only gesture detection technology, and some feature motion-sensing or touch-sensing controls.
Many related works include virtual mouse or keyboard usage but not to play games. Others use sign language detection only to assist people with speech disabilities. Numerous hand sign detection research papers are there which focuses on the fingerspelling method and that can be pretty time-consuming. Furthermore, in most cases, the technology stack to build the projects may seem overwhelming.

IV. METHODOLOGY

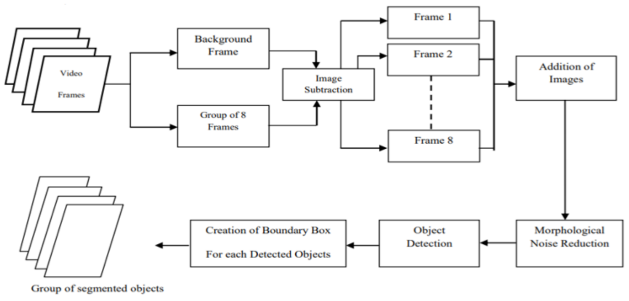
A. Data Collection

We use the OpenCV (Open Source Computer Vision) library to capture images from the default camera (usually a webcam) and save them into separate directories, organized by classes. We created 4 classes or categories to collect data for and per class we collect 250 images.
First opens a default camera index 0 or if we choose another camera its index is 1. After opening the camera it displays a message on the frame, prompting the user to press 'Q' to start data collection. After 'Q' key is pressed, it breaks out of this loop and proceeds to collect data for the classes within the data directory named '0', '1', '2', '3'. It captures frames from the camera, displays them, and saves them to the corresponding class directory as image files with names like

```
Data----
 |
 |--0---
    |
    |-------1.jpg
    |--------2.jpg
    |............
    |---------250.jpg
```

```
|
|---1—
     |
     |-----1.jpg
     |------2.jpg
     |…………..
     |------250.jpg
………………..
```

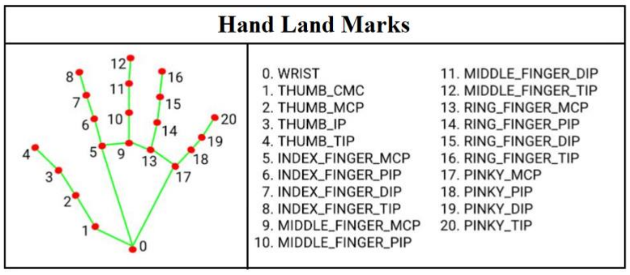
……………….. etc. It continues to do this until it has collected 250 images for the current class. After collecting data for all classes, it releases the camera resource the camera resource and OpenCV windows.



B. *DataSet Generation:*

For each image in a subdirectory, it does the following:

- Read the image using OpenCV and convert BGR to RGB format (MediaPipe works with RGB images).
- Process the image with the MediaPipe Hands model.
- Check if there are multiple hand landmarks detected in the image.
- If hand landmarks are detected, it extracts the x and y coordinates of each landmark and stores them in x_ and y_ lists.
- It then calculates the relative coordinates by subtracting the minimum x and y values from all landmark coordinates and stores them in the data_aux list. This step normalizes the landmark coordinates to a common reference point.
- The data_aux list, which contains the normalized hand landmark coordinates for a single image, is appended to the data list.
- The corresponding label (the subdirectory name) is appended to the labels list.
- After processing all images in all subdirectories, it opens a file named 'data.pickle' in write binary mode ('wb').
- It serializes a dictionary containing the data and labels lists using pickle.dump().
- Finally, it closes the file.



Data preprocessing for NN:

- An image transformation pipeline is used to resize captured images to 64*64 pixels and convert them to tensors.
- A folder is used to load image data from the data folder and apply the specified transformations.

V. MODEL SLECTION AND TRAINING

A. *RandomForest Classifier*

Here our model is Random forest classifier object called ‘model’ and the classifier is trained.

B. *Neural Network*

We uses a customed Pytorch neural network model named ‘CustomNN’ with two fully connected layers to classify the gestures.

C. *Training on Random Forest classifier model*

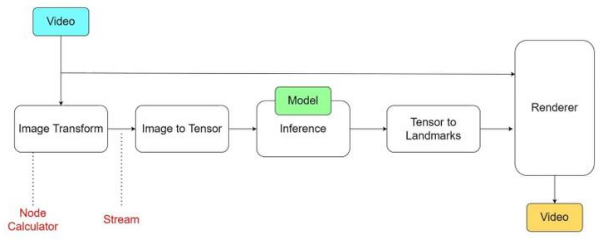
- Data pickle file has a dictionary containing two array: ‘data’ (Hand landmark data) and ‘labels’(class labels) and we have Split the dataset into training and test sets.
- 80% of the data is used for training and 20% of the data is used for testing and shuffles the data before splitting
- Mediapipe Hands model is initialized to detect hand landmarks on each frame
- The RandomForest Classifier model trained on the training data.
- The trained model makes predictions on the testing data

D. *Training on Neural Network model*

- Data pickle file has a dictionary containing two arrays: ‘data’ (Hand landmark data) and ‘labels’ (class labels) and we have Split the dataset into training and test sets.

- 80% of the data is used for training and 20% of the data is used for testing and shuffles the data before splitting
- A data loader is used to create data loaders for both the training and testing sets. Each loader loads data in batches of 64 samples.
- An instance of the model is created with input size 64 and number of classes 64.
- The model is trained in a loop for a specified number of epochs 10. For each epoch, it loops through the training data in batches:
    - Zeroes the gradients.
    - Performs a forward pass and computes the loss.
    - Computes gradients and performs backpropagation.
    - Updates the model parameters using the optimizer.

- Neural Network model trained on the training data.
- The trained model makes predictions on the testing data



VI. TESTING AND EVALUATION

A. *Model Evaluation*

Random Forest Classifier:
Calculates the model's accuracy by comparing the predicted labels with actual labels in the testing set using the accuracy_score function and the accuracy is printed as a percentage.
Neural network model:
Iterates through the test data in batches. And performs a forward pass to get predictions. Then Computes the accuracy by comparing predicted labels with true labels. The accuracy is printed as a percentage.

B. *Real-time Evaluation:*

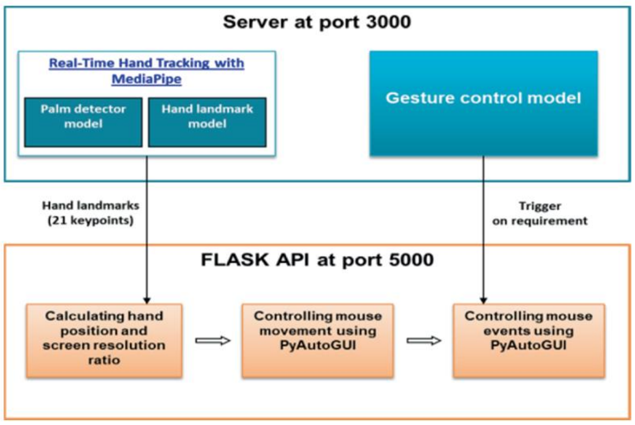
Random Forest Classifier Model:
Captures a frame from a continuous loop of video frames from webcam and retrieves the height and width of the frame and then converts the frame from BGR to RGB format. For the detected hand, the model extracts the x,y coordinates of each landmark and stores and normalizes them. The pre-trained model predicts the gesture based on the normalized hand landmark data and calculates the prediction confidence and the predicted label index using a minimum detection confidence threshold (euclidean distance). After prediction

- if the recognized gesture is ‘Back’ it simulates the keyboard press of the spacebar using pyautogui.
- If the gesture is ‘Option’, it simulates a right-click action pyautogui.
- If the gesture is ‘Open’, it simulates a double-click action pyautogui.
- If the gesture is ‘Free’, it moves the mouse cursor to the position of the hand, and moves the cursor quickly to follow the hand’s movement action pyautogui.

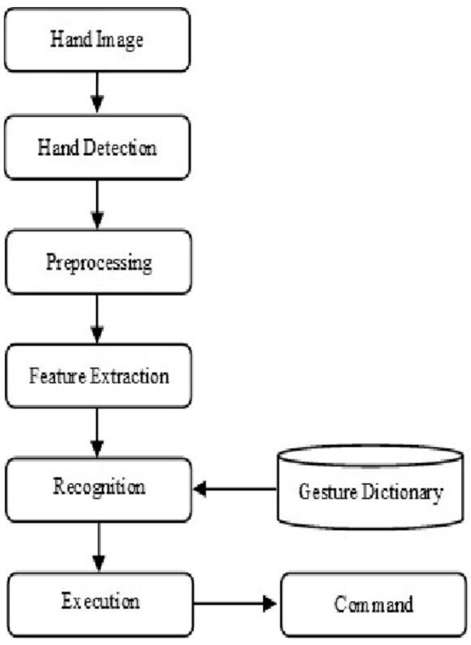
Draws a rectangle around the hand with color that depends on the prediction confidence and displays the recognized gesture and its confidence level in the top left corner of the frame.
Neural Network Model:
After opening the default camera, a continous loop to caputure and process vedio frames. For each frame, it does the following:

- Captures a frame from the webcam and checks for any errors in frame retrieval.
- Retrieves the height (H) and width (W) of the frame.
- Converts the frame from BGR to RGB format.

It uses the MediaPipe Hands model to detect hand landmarks in the frame and extract the x and y coordinates of each landmark. AndCalculates the bounding box coordinates (x1, y1, x2, y2) around the detected hand landmarks. It uses the pre-trained PyTorch model to predict the gesture based on the normalized hand landmark data. The predicted label and prediction confidence are obtained. Conditional statements for different gestures are provided but commented out. Depending on the recognized gesture, we can uncomment the corresponding actions (e.g., double-click, right-click, space key press, cursor movement). Draws a rectangle around the hand with a black color. Displays the recognized gesture and its confidence on the video feed. The color for displaying the recognized gesture text is determined based on the confidence level. AndShows the confidence level in the top left corner of the frame.



Server at port 3000
Real-Time Hand Tracking with MediaPipe
Palm detector model
Hand landmark model
Gesture control model
Hand landmarks (21 keypoints)
Trigger on requirement
FLASK API at port 5000
Calculating hand position and screen resolution ratio
Controlling mouse movement using PyAutoGUI
Controlling mouse events using PyAutoGUI

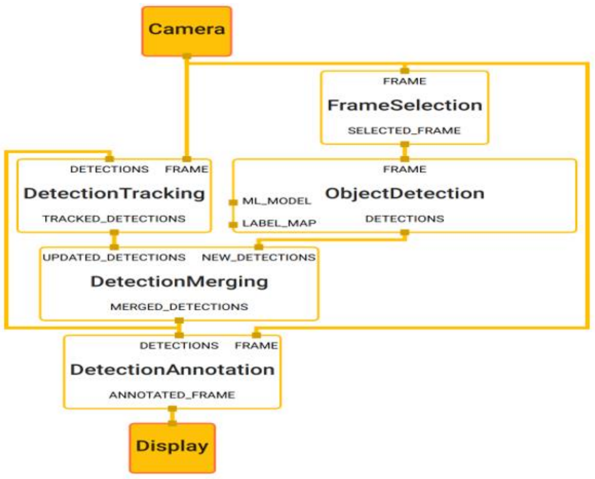


VII. FEATURE LIST

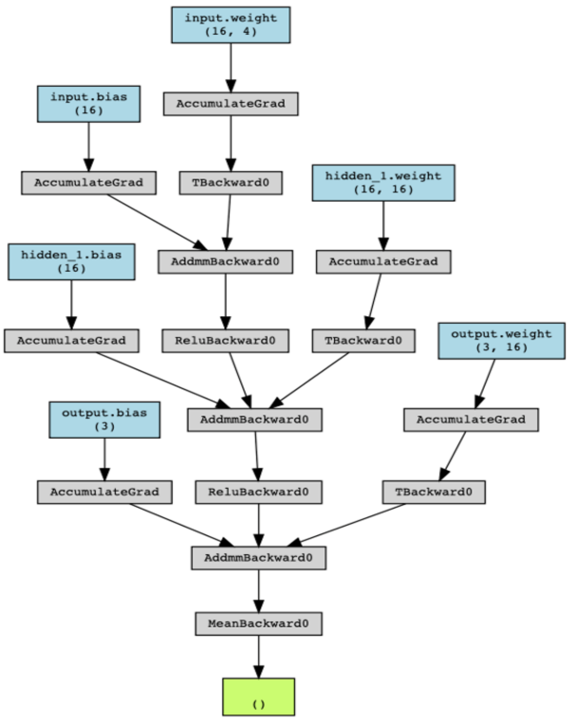
Mediapipe: MediaPipe is an open-source framework developed by Google that provides a wide range of tools and components for building real-time multimedia processing applications. It is particularly popular for its capabilities in computer vision and machine learning tasks, including but not limited to:

1. Hand tracking and gesture recognition.
2. Face detection and recognition.
3. Pose estimation for human body tracking.
4. Object detection and tracking.
5. Audio and speech processing.
6. Augmented reality (AR) effects and filters.
7. Multi-modal data fusion, combining information from various sensors.

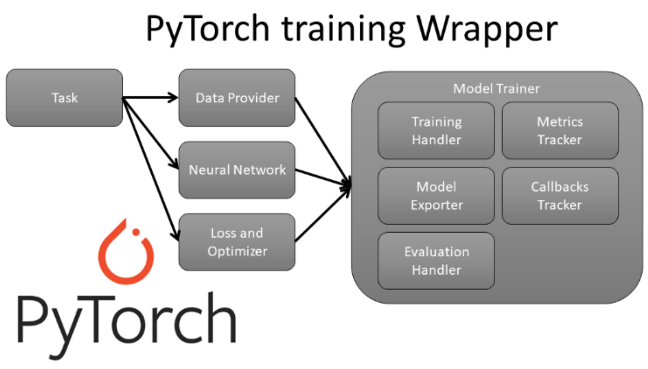
MediaPipe is designed to be versatile, efficient, and user-friendly, making it a valuable resource for developers and researchers working on projects that involve real-time video and audio analysis. It provides pre-trained models and a customizable pipeline, allowing developers to adapt it to their specific use cases.



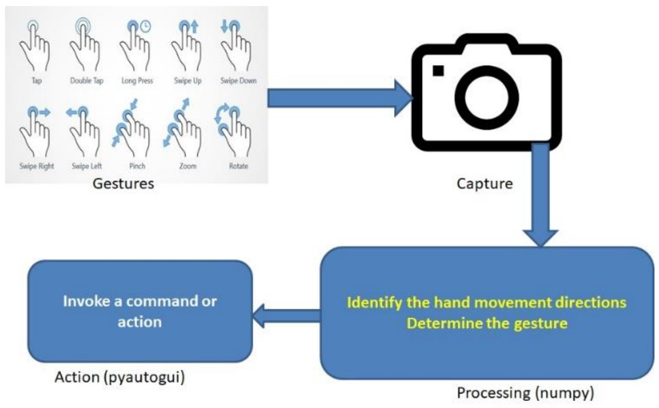
Neural Network (NN): A neural network is a machine learning model inspired by the structure and functioning of the human brain. It consists of interconnected nodes, or artificial neurons, organized into layers. Each neuron receives input data, processes it, and passes the result to the next layer. Neural networks are used for a wide range of tasks, including pattern recognition, image and speech recognition, natural language processing, and more. They learn by adjusting the strengths of connections (weights) between neurons during training to make accurate predictions or classifications. Deep neural networks, specifically deep learning, have gained popularity in recent years for their ability to handle complex, high-dimensional data and have been instrumental in advancing various applications.



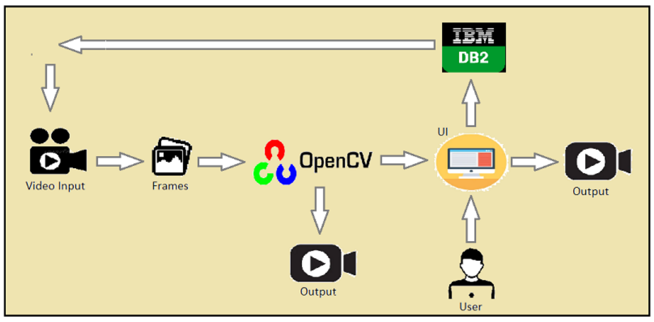
Torch: Torch, or PyTorch, is a popular open-source deep learning framework primarily developed by Facebook's AI Research lab (FAIR). It provides a flexible and dynamic approach to building and training neural networks. Key features of PyTorch include dynamic computation graphs, which make it easier for researchers to experiment and debug, and a strong focus on deep learning, with support for GPU acceleration and automatic differentiation. PyTorch is widely used in the machine learning and deep learning communities and is known for its ease of use and strong community support. It has been used to develop state-of-the-art models for various applications, including computer vision, natural language processing, and reinforcement learning.



Pyautogui: PyAutoGUI is a Python library that provides automation capabilities for controlling a computer's mouse and keyboard. It allows users to programmatically simulate mouse movements and clicks, as well as keyboard keypresses, making it useful for automating repetitive tasks, GUI testing, and other automation-related activities. PyAutoGUI is cross-platform and works on Windows, macOS, and Linux. It's a simple and easy-to-use tool for automating user interactions with a graphical user interface (GUI) and can be a valuable resource for tasks like data entry, GUI testing, or automating routine operations on a computer.



OpenCV: OpenCV, or Open-Source Computer Vision Library, is an open-source computer vision and image processing library. It provides a wide range of tools and functions for tasks such as image and video analysis, object detection, facial recognition, machine learning, and more. OpenCV is written in C++ and offers interfaces for various programming languages, including Python, making it accessible to a broad range of developers. It is widely used in computer vision applications, robotics, and machine learning, and it is known for its versatility, efficiency, and extensive community support. OpenCV is a valuable resource for anyone working with image and video data in their projects.



Randomforest classifier: A Random Forest Classifier is a popular machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that combines multiple decision trees to make more accurate predictions.

- **Ensemble Method:** Random Forest is an ensemble method that builds multiple decision trees during training and combines their predictions to improve accuracy and reduce overfitting.
- **Decision Trees**: Each decision tree in the Random Forest is constructed based on a random subset of the data and a random subset of features. This

randomness helps reduce the correlation between trees and increases diversity.

- **Aggregation:** The final prediction of the Random Forest is typically determined by a majority vote (for classification) or an average (for regression) of the individual tree predictions.
- **Robust and Versatile:** Random Forests are robust to noisy data and are less prone to overfitting compared to individual decision trees. They can handle a wide range of data types and are effective for both classification and regression tasks.
- **Feature Importance:** Random Forests can provide insights into feature importance, which helps in understanding the significance of different features in making predictions.
- **Widely Used:** Random Forest is a popular algorithm in various fields, including data science, finance, healthcare, and more, thanks to its effectiveness and ease of use.

Random Forest Classifiers are a valuable tool for improving the accuracy and generalization of machine learning models, making them a go-to choice in many practical applications.

Numpy: NumPy, which stands for "Numerical Python," is a fundamental library in the Python ecosystem for numerical and scientific computing. Here's a short summary of NumPy:

- **Array Processing:** NumPy provides support for multi-dimensional arrays and matrices, making it an essential tool for numerical computations in Python.
- **Efficient Operations:** It offers a wide range of mathematical and logical functions, and these operations are highly optimized for speed and memory efficiency.
- **Broadcasting:** NumPy allows for element-wise operations on arrays of different shapes, a feature known as broadcasting, which simplifies computations and reduces the need for explicit loops.
- **Integration with Python:** NumPy seamlessly integrates with Python and is often used in combination with other libraries, such as SciPy, pandas, and scikit-learn, for various data analysis and scientific computing tasks.
- **Interoperability:** NumPy provides support for integrating C/C++ and Fortran code, enabling the incorporation of low-level, high-performance libraries into Python applications.
- **Open Source:** NumPy is open-source and actively maintained, with a large and supportive community of users and contributors.

NumPy is the foundation for many other scientific computing libraries in Python and is widely used in fields such as data analysis, machine learning, image processing, and scientific research, making it an essential tool for numerical and array-based operations in Python.

VIII. CONCLUSION

Interacting with the virtual world gained immense popularity throughout this decade which is why "Gesture-Dino" is an innovate way to play the no-internet dinosaur game purely on the basis of hand gestures. By detecting the American Sign Language system (ASL), our game allows the dinosaur to jump over rocks and cacti. So, the spacebar and mouse interactions which were essentially required to play the game has been single-handedly replaced with sign language. This implies that in addition to functioning as a spacebar, our project uses the ASL to move the mouse up and down. Moreover, multiple hand gestures can be detected instead of one.

As a result, this game is extremely user-friendly and can be played by toddlers or even people who have learning disabilities. Unlike other relevant works, we have also been able to construct our project employing trouble-free methods such as mediapipe, neural network, pyautogui, OpenCV, randomforest classfier, and numpy. Moreover, it doesn't require a GPU to function, which caused issues with other projects. Future works may further translate the entire set of keys of the keyboard using different ASL notations. For instance, one hand gesture may be used to cast spells while others can be used to recharge. The virtual mouse system can also be enhanced for a more smoother gaming experience.

So overall, our project revolutionizes the implication of the American Sign Language System for playing the no-internet Dinosaur game completely handsfree.

ACKNOWLEDGEMENTS

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes me possible to complete the project.
We are really grateful and wish our profound indebtedness to our honorable faculty Mr. Intisar Tahmid Naheen sir for his infinite patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism and valuable advice.
Lastly, we would like to heartiest gratitude to our group members to help use finish our project properly and received lots of suggestions that improved our quality of work.
.

REFERENCES

[1] *Papers with Code - Fingerspelling recognition in the wild with iterative visual attention*. (2019, August 28). https://paperswithcode.com/paper/fingerspelling-recognition-in-the-wild-with

[2] Verma, P. S., Mahanta, S. P., Sevanth, B. N., & B, A. P. S. (2023). Virtual mouse and keyboard for computer interaction by hand gestures using machine learning. *International Journal of Current Science Research and Review*, *06*(08). https://doi.org/10.47191/ijcsrr/v6-i8-13

[3] Badgujar, H. Y., & Thool, R. (2013). MULTISCRIPT MARUTI WITH VIRTUAL KEYBORD. *ResearchGate*. https://www.researchgate.net/publication/236853049_

MULTISCRIPT_MARUTI_WITH_VIRTUAL_KEYBORD

[4] Challa, Y., Prabhakaran, G., & Sameera, N. (2023). AI based mouse controller. *ResearchGate*. https://www.researchgate.net/publication/373332695_AI_Based_Mouse_Controller

[5] Shriram, S., Nagaraj, B., Jain, J., Shankar, S., & Ajay, P. (2021). Deep Learning-Based Real-Time AI virtual mouse system using computer vision to avoid COVID-19 spread. *Journal of Healthcare Engineering*, *2021*, 1–8. https://doi.org/10.1155/2021/8133076

[6] Papers with Code - Virtual Mouse And Assistant: A Technological Revolution Of Artificial Intelligence. (2023, March 11). https://paperswithcode.com/paper/virtual-mouse-and-assistant-a-technological

[7] Rowthu, L. R., Kumar, G. K., Mande, U., & Prahaladh, L. (2023). VIRTUAL KEYBOARD IMPLEMENTATION BASED ON FKU FINGURE RECOGNITION. *ResearchGate*. https://www.researchgate.net/publication/370472018_VIRTUAL_KEYBOARD_IMPLEMENTATION_BASED_ON_FKU_FINGURE_RECOGNITION

[8] Lu, D., & Huang, D. (2018). FMHaSH: Deep Hashing of In-Air-Handwriting for User Identification. *ResearchGate*. https://www.researchgate.net/publication/325709718_FMHash_Deep_Hashing_of_In-Air-Handwriting_for_User_Identification

[9] *Papers with Code - Sign Language Detection*. (2022, September 8). https://paperswithcode.com/paper/sign-language-detection

[10] Obi, Y., Claudio, K. S., Budiman, V. M., Achmad, S., & Kurniawan, A. (2023). Sign language recognition system for communicating to people with disabilities. *Procedia Computer Science*, *216*, 13–20. https://doi.org/10.1016/j.procs.2022.12.106

.