Digital Media Final Project

GestureControl: Exploring Immersive Gameplay through OpenCV-Enabled Gesture-Based Controls in a Game

Programmer's Report

Nikhil Subramanian

Registration Number: K2218203

Kingston University

Submitted in Jan 2024

Introduction

Within the dynamic landscape of contemporary game development, the pursuit of heightened player interaction has led to the exploration of innovative input methods. Conventional interfaces, reliant on keyboards, mice, and controllers, have established longstanding norms. However, this final project seeks to transcend these conventional paradigms, harnessing the capabilities of OpenCV's gesture recognition within the Unity game development engine.

The primary objective is to cultivate an immersive gaming experience, leveraging human gestures as a natural and fluid form of interaction. Recognizing the limitations of traditional input devices in delivering truly engaging experiences, the project endeavours to explore and implement gesture-based controls as a means of enhancing player engagement.

The integration of OpenCV, a powerful computer vision library, within the Unity game development environment introduces a novel dimension to player interaction. By capturing and interpreting a diverse range of hand gestures, the project aims to provide users with an intuitive means of navigating virtual environments, executing in-game actions, and interacting with virtual elements.

Motivated by the evolving landscape of user experience and the potential offered by gesture-based controls, the project embarks on the meticulous development of robust gesture recognition algorithms. These algorithms are envisioned to seamlessly translate human gestures into meaningful in-game interactions, fostering a heightened level of immersion.

As the project unfolds, the following sections will delve into the intricacies of the methodology, encompassing the development of gesture recognition algorithms, the integration of OpenCV with Unity, and the creation of a compelling game that responds dynamically to user gestures. The endeavour is not merely a technical exploration but a strategic foray into the realms of user experience and gameplay design.

Through the synthesis of computer vision and game development, this final project aspires to contribute to the evolving discourse on immersive gameplay. The subsequent sections will provide a detailed account of the methodology, challenges encountered, solutions devised, and the ultimate outcomes achieved. In essence, the project seeks to redefine the boundaries of player interaction, ushering in a new era of gesture-enabled gameplay within the Unity environment.

Background

In the contemporary landscape of digital entertainment, the quest for heightened immersion and natural interaction within gaming experiences has led to the exploration of innovative input mechanisms. Traditional interfaces, reliant on keyboards, mice, and controllers, have been foundational in shaping the gaming experience. However, the demand for more intuitive and immersive alternatives has spurred the investigation of novel input paradigms.

Gesture-based controls emerge as a promising avenue in this pursuit, offering users the ability to interact with virtual environments through natural hand movements. This departure from conventional input methods aligns with a broader industry trend of enhancing user experiences by bridging the divide between the physical and digital realms. The integration of human gestures as a means of controlling gameplay introduces an element of intuitiveness that resonates across diverse user demographics.

The catalyst for this project lies in the recognition of the inherent limitations of traditional input methods, particularly in delivering a truly immersive gaming experience. Acknowledging the potential of gesture-based controls, the project seeks to leverage the capabilities of OpenCV, a robust computer vision library, to introduce a more natural and responsive form of interaction within the Unity game development engine.

The inclusion of OpenCV is pivotal to the project's success, given its established functionalities in image processing, object detection, and gesture recognition. By harnessing the capabilities of OpenCV, the project aims to implement robust gesture recognition algorithms capable of accurately interpreting a spectrum of hand movements.

Recent research literature, exemplified by the work of Le et al. [5], Kumar et al. [4], and Rautaray et al. [1], attests to the feasibility and potential benefits of incorporating hand gestures into virtual environments. These studies not only underscore the enhancement of gaming experiences but also highlight the promise of addressing accessibility challenges for users with physical disabilities [5].

Moreover, the intersection of computer vision and game development has witnessed increased exploration, leading to innovative applications in virtual reality, rehabilitation, and human-computer interaction [1][2][3]. The present project draws inspiration from these advancements, aspiring to contribute to the growing body of knowledge at the convergence of computer vision and immersive gameplay.

As the project progresses, it will navigate the intricacies of integrating OpenCV with Unity, designing, and implementing gesture recognition algorithms, and crafting a gesture-controlled game. Subsequent sections will illuminate the methodologies employed, challenges encountered, and the ultimate outcomes achieved in the pursuit of redefining player interaction through gesture-based controls within the Unity game development environment.

Literature Review:

Hand Gesture Recognition System for Games (Nhat Vu Le et al IEEE 2022)[5]-

The author talks about how easily a gesture control can be incorporated into modern games as, most laptop computers and many desktops come equipped with a webcam, so naturally, that would be the starting point. Users would be able to perform various hand gestures, with each being mapped to a set of button combinations on a virtual gamepad. As gesture detection would have to be done in real-time, fast Computer Vision libraries such as OpenCV are needed to process images.

This establishes the basis for author’s the research, a program that can be deployed on any computer with a webcam to instantly create a gamepad out of thin air. The final program features an intuitive user interface with customizable game profiles that can be saved to or loaded from storage. The program captures webcam input sixty times per second, performing multiple levels of processing on each image. Using techniques such as thresholding, gaussian blurring, and grayscale conversion, an ideal image is fed to OpenCV’s contour detection algorithm. By calculating the angles between contours, the number of fingers held up can be determined. When a gesture is detected, the program communicates with a kernel-mode driver to send controller inputs directly to games.

As video games are among the most popular forms of entertainment in the modern world, the author argues that many gamers with physical disabilities are impeded by traditional controllers. The author also suggests that this gesture system could be a solution for gamers faces by such issues.

The author's design to enabling user customization allows for any Xinput compatible game to be controlled. This allowed for the implementation of not just racing games and driving simulators, but also first-person shoots and side-scrolling platformers. While devices exist on the market to serve various physical limitations, these can prove to be cost-prohibitive for many gamers.

Developers must implement necessary API calls into their games to interpret controller inputs from the user. Both DirectInput and Xinput were designed to read the states of physically connected game controllers, whereas the gesture recognition system must send virtual inputs to a game.

Writing the driver from scratch also presented several issues, with the sheer timeframe needed to write one lying beyond the scope of the project. After researching alternatives, the program DS4Windows was discovered. This program remapped button inputs from a physical Sony DualShock 4 controller to XInput commands, enabling compatibility for all games that supported Xinput. As the majority of PC games on the market with controller support utilize Xinput, DS4Windows served as a first basis for what the gesture recognition program would be capable of. As a result, the author has created a method to use Human Computer Interface principles to use Hand Gestures to interact with multiple games.

Game Controlling using Hand Gestures (GVS Manoj Kumar et al. IEEE 2023)[4]-

The main goal of this paper is to demonstrate how to play a computer game using human gestures. The system's secondary goal is to develop a system that allows a player to play a game without using a physical controller. This system seeks to create a gesture recognition application. The system's attached camera or webcam can be used to identify human hand gestures. The operations on the Game will be performed with the game's default gaming controls based on the system's study of detecting human Hand Gestures. A collection of instructions is included in this system for identifying human hand movements. The gestures should be performed using the palms of the hands.

To recognize gestures, the gestures recognition module can be utilized. Hand gesture recognition systems that are dependable are currently being researched and developed. Hand gestures are the most natural and intuitive Human Computer Interaction approach that can be used as a feasible alternative for the computer keyboard. Hand gesture recognition is employed in sign language identification in addition to Human Computer Interaction, making hand gesture recognition even more important.

There is a variety of different strategies for vehicle tracking, monitoring, and warning systems have been put forth in this paper.

Game-based human computer interaction using gesture recognition for rehabilitation (Chiang Wei Tan et al. IEEE 2013)[3] –

This paper introduces a human computer interaction system by using hand gesture recognition to apply into game-based rehabilitation application. The basic concept of this project is to assist patient to conduct rehabilitation with hand gesture recognition, meanwhile therapist can follow recovery progress of patient through results.

Approach proposed in this paper are hand region segmentation, hand palm partition and finger counting. Experiment evaluation as shown has successfully demonstrate the effectiveness and robustness in performing the mouse function and feasible to be implemented into rehabilitation games.

Interaction with virtual game through hand gesture recognition (Siddharth S. Rautaray et al. IEEE 2012)[1]-

Hand gesture recognition systems for virtual reality applications provides the users an enhanced interaction experience as it integrates the virtual and the real-world object. Comprehensive user acceptability has been considered to exhibit the accuracy, usefulness, and ease of use to the proposed and implemented hand gesture recognition system. Hand gesture recognition systems for virtual reality applications provides the users an enhanced interaction experience as it integrates the virtual and the real-world object. Growth in virtual environments based upon computer systems and development of user interfaces influence the changes in the Human-Computer Interaction.

Gesture recognition-based interaction interface, endow with more realistic and immersive interaction compared to the traditional devices. The system enables a physically realistic mode of interaction to the virtual environment. The application of virtual game controlling through hand gestures proposed and implemented in the present paper provides a suitable efficient and user-friendly interface between human and computer for interaction with virtual game using hand gestures.

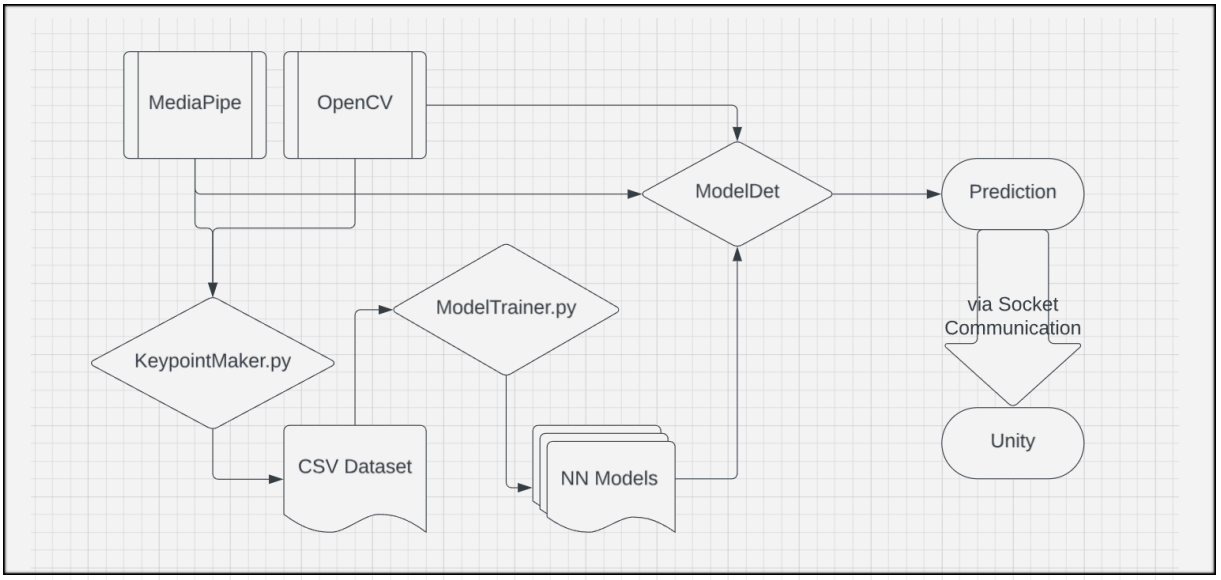
Real Time Hand Gesture Recognition for Human Computer Interaction (Rishabh Agarwal et al. IEEE 2016)[2]-

Most of the human computer interaction interfaces that are designed today require explicit instructions from the user in the form of keyboard taps or mouse clicks. As the complexity of these devices increase, the sheer amount of such instructions can easily disrupt, distract, and overwhelm users. Human Computer Interaction with a personal computer today is not just limited to keyboard and mouse interaction. In this paper, a novel method is proposed to recognize hand gestures in real time with high accuracy and precision. The author also notes that the developed system should be both flexible and expandable which maximize efficiency, accuracy, and intuitiveness.

The proposed method in this paper, gives commendable results in both experimental and real word hand scenarios.

Method

Flow-chart to explain the working:



Integration with OpenCV and MediaPipe:

The initial phase involves the seamless integration of OpenCV, a critical component facilitating image processing and communication within the development pipeline. Using the OpenCV packages from the python packages, A frame of video capture is taken using the webcam and processed to make the next step easier.

MediaPipe takes center stage, seamlessly integrated into the development pipeline through Python. The exploration of MediaPipe's features and its role in hand pose detection and skeleton tracking lays the foundation for subsequent stages. With that said, the processed image is put through MediaPipe to detect and plot hand landmarks. These landmarks are recorded to form a dataset of various hand gestures and saved as Comma Separated Value (CSV) files. This is also used when detecting the hand gestures in while integrated with the game. Using the MediaPipe Documentation[9] and YouTube Tutorial[10], scripts for my use case were created.

This data is of size 21,2 (21 landmarks of x-y coordinates) with a single integer appended to specify the class that hand gesture belongs to. These 21 co-ordinates are relative to each other, and it is recorded with the wrist as the base. Each finger has 4 coordinates. Making a total of 21 points per hand. As the use case doesn't demand both hands, only 21 points are recorded. But an option to use both hands in this venture also exist if the requirement to use it is found.

For this use case roughly 2249 co-ordinates were used to train the model at the time of writing this report. Since any number of gesture data can be added at any time just by running this script, this number will always increase. This recording of hand data can be done by simply running the logging script and pressing the corresponding class number to log those landmarks into a CSV file. In this project, 0 is base, 1 is Duck, 2 is Jump, 3 is Right and 4 is Left. By pressing this button while running the logging script, the current landmarks will be saved under that class and when training the Neural Network model, these 21 data will be used as input and the class will be the classification labels.

TensorFlow for AI Model Training:

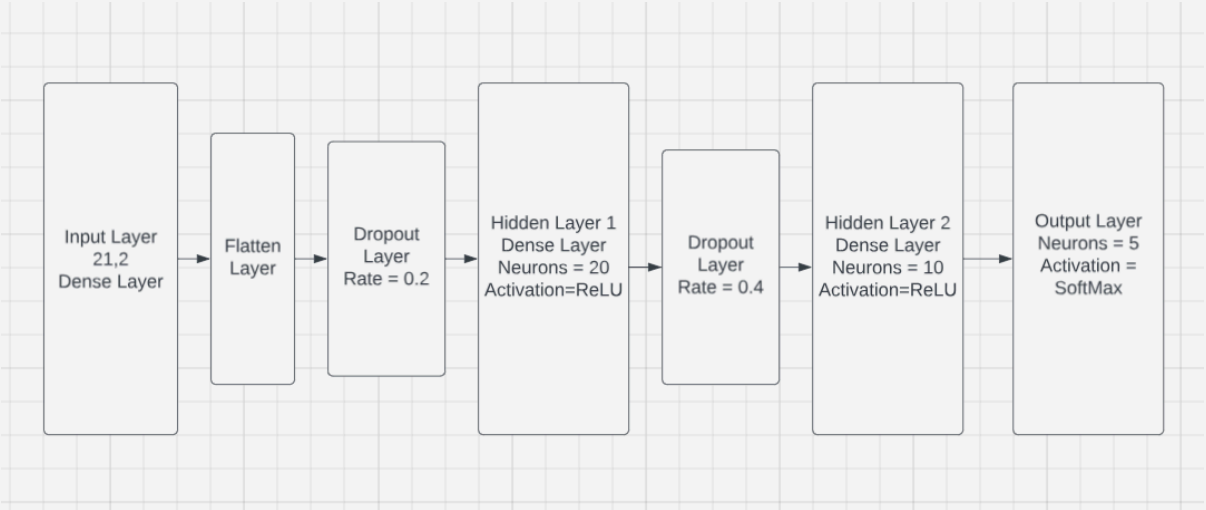
TensorFlow, a cornerstone of the methodology, plays a pivotal role in training two distinct neural network models—Feedforward and Long Short-Term Memory.

The data from csv file is used to train these models. The input provided will be in the shape of -1,21,2. -1 is to specify that the number of landmarks points, as the csv file has many points. So, in general it can be said that the input size is no of points=-1, no of landmarks points=21, no of coordinates=2 (x-y coordinates). The train-test split done is 75-25 split.

The setting chosen for both these models are as so, Adam Optimizer is used which is an efficient stochastic gradient descent optimizer. It adapts the learning rate for each parameter, allowing faster convergence. As Loss function, sparse categorical cross-entropy is used as it is more appropriate when dealing with integer class labels. This is a measure of probability error for multi-class single-label problems. Metrics of comparison is set to be Accuracy to help with comparison between models.

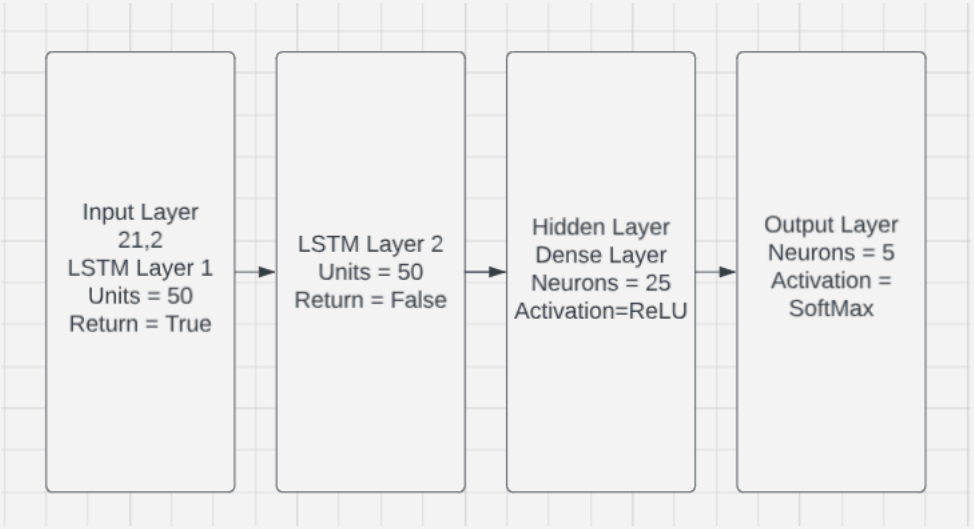
Batch size used is 128 for faster training iteraction while full utilization of GPU parallelization capabilities. Epochs is set at 1000 with callbacks like ModelCheckpoint and EarlyStopping to get a perfect accuracy for the model. Given next are the architecture for both models used.

Feedforward Neural Network Model



Compile information:	**Training Information:**
Optimizer = Adam Loss Function = Sparse Categorical Crossentropy Metrics = Accuracy	Epochs = 1000 Batch Size = 128 Callbacks: ModelCheckpoint and EarlyStopping

Long Short Term Memory Model



Compile information:	**Training Information:**
Optimizer = Adam Loss Function = Sparse Categorical Crossentropy Metrics = Accuracy	Epochs = 1000 Batch Size = 128 Callbacks: ModelCheckpoint and EarlyStopping

These are two models created and used. Its evaluation metrics are given in the next section.

Model Evaluation and Selection:

Using the CSV data from the recorded landmarks is used to train both these models and depending on the performance metrics, the best on is chosen for the game.

The trained models undergo rigorous evaluation, scrutinizing aspects such as accuracy, responsiveness, and suitability for real-time gesture prediction.

The culmination of this phase results in the strategic selection of the model—be it Feedforward or LSTM—that exhibits the highest accuracy for seamless integration into the Unity game environment.

This model is saved as HDF5 file and used in the final python code that will be sending the prediction into Unity.

The evaluation of deep feedforward network has given these scores, (in macro average) the precision is 89%, recall is 83%, f1 score is 85%, and accuracy is 89%. The evaluation of LSTM model has given these scores, (in micro average) the precision is 94%, recall is 94%, f1 score is 94%, and accuracy is 96%. By comparing these scores, it can be said that the LSMT model demonstrates superior performance. The accuracy itself is a sure shot proof of the model's superiority over feedforward. The difference of 9% in difference in the recall score shows LSTM's superiority in correctly predicting positives cases without any missing gestures. Thus, LSTM was used in the final script as the main model for the hand gesture.

The performance metrics of both the models is attached at the outcomes section of the report.

Socket Communication Optimization:

Socket communication, the linchpin between Python and Unity, undergoes meticulous optimization to enhance efficiency and reliability. The focus lies on reducing latency and ensuring data integrity, guaranteeing a seamless flow of information for precise gesture prediction and optimal gameplay responsiveness.

As a prototype, currently this latency is set to facilitate non-erroneous socket communication between Unity and python.

A separate python file that runs OpenCV and MediaPipe detects hand gesture and using the NN model a prediction is run and the gesture is detected.

The socket is a TCP server socket which listens for Unity as a client on initialization. No data will be sent until unity is connected as client; this is done to have reliable synchronous data transfer in real time interactivity. The prediction is packed as a 4-byte structure and sent over the connection. Once Unity receives the data and it sends over an acknowledgement back to this script. This way prediction can be sent over while preventing data loss.

Unity Game Integration and Testing:

With Hand gesture detection working, the focus is shifted to the game. An Infinite runner[8] with basic controls like jump, duck, left and right. A base hand pose is added to keep a position where the player is idle hence the player will just run infinitely. The models used in the game was obtained from Unity Asset Store and Mixamo for animations. Initially the game was created to run on WASD keys to

fix basic movements of the game. A Script was also created to read and parse data from socket. But later, this WASD movement was switched to use only the data from socket to control the player. The game generates level infinity every 2 seconds. The world is created using free assets from unity asset store. The level chunks are pre-created and loaded. There have not been any issues of multiple instances of levels being loaded. But if the game was to be publicly released, a destroy script must be added. Since this is just a proof of concept, it has been omitted.

Gesture-Based Gameplay Refinement in Unity:

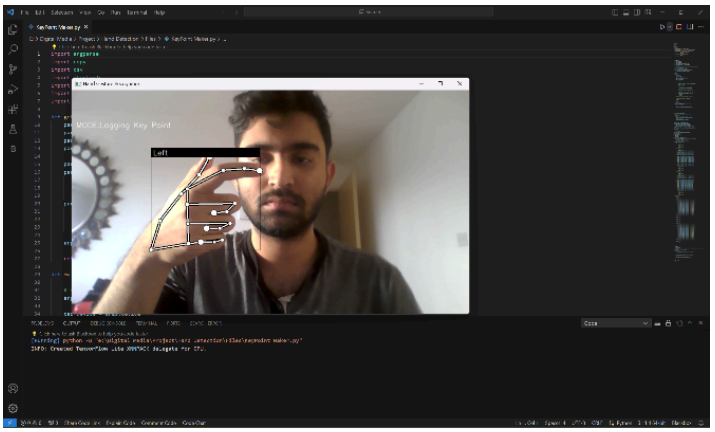
In the final stage, the parsed data from the socket is connected with the player movement script and calls the functions depending on the prediction given by the OpenCV python code.

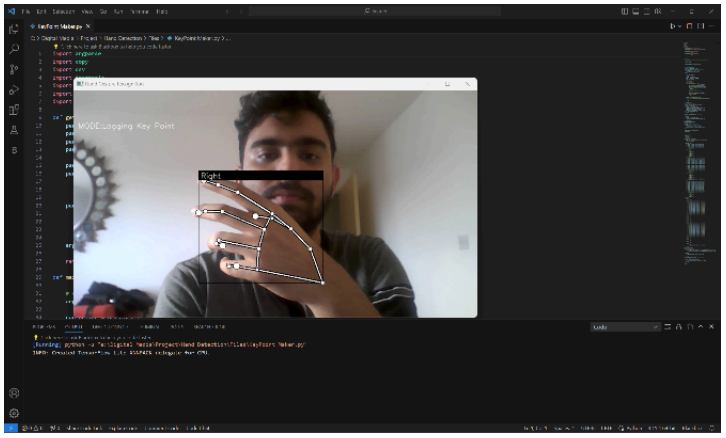
There is only one script that is responsible for parsing data. Upon receiving the data, it is sent to another script which by using switch case calls the function on the player movement script like Jump(), Left(), Right() or Duck(). Just for Debugging purpose, it is logged in the terminal when that specific function is called. By calling these functions, the player will move accordingly hence completing connecting the hand gesture to player movement perfectly. Other scripts on this that were created were scripts for running python file but that prevented the camera to open separately. This made it hard for playing the game as the hand gesture being predicted was not visible. This file is placed as depreciated and the Camera script is run separately. But as the communication between unity and python takes place through socket, both can run asynchronously making the working on them easier and reliable.

Some game settings were tweaked to make the prediction and the game feel smoother. Collision script has been turned off as using hand gesture to play a game is considerable hard. Collision made the game to be on the hard side, hence now it functions as an obstacle course.

Outcomes

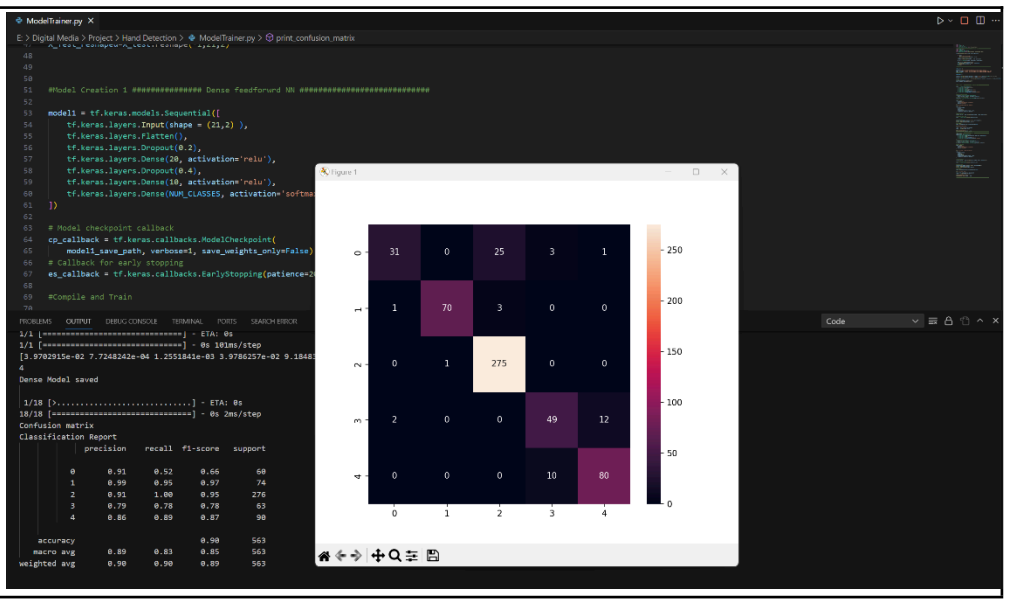
Images from the Hand marker logging code



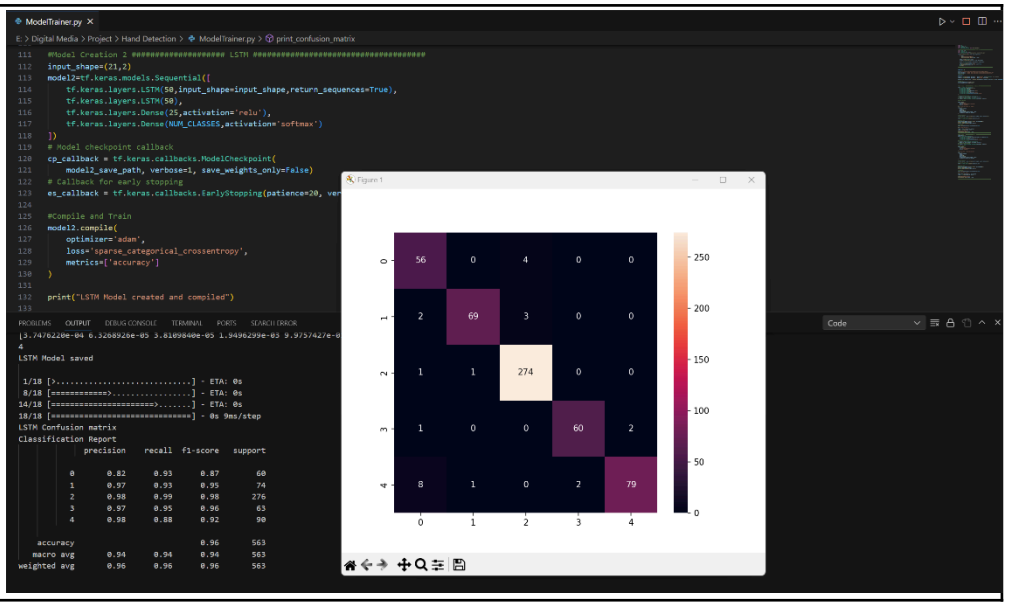


This code is responsible for creating the csv dataset to train the NN models. Below is the dataset collected and its corresponding labels-

Below is the Performance metrics of the two models trained-

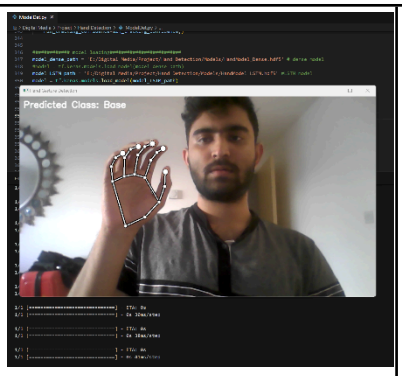
Deep Feedforward Neural Network	

```
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 101ms/step
[3.9702915e-02 7.7248242e-04 1.2551841e-03 3.9786257e-02 9.1848
4
Dense Model saved

 1/18 [>.............................] - ETA: 0s
18/18 [==============================] - 0s 2ms/step
Confusion matrix
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.52      0.66        60
           1       0.99      0.95      0.97        74
           2       0.91      1.00      0.95       276
           3       0.79      0.78      0.78        63
           4       0.86      0.89      0.87        90

    accuracy                           0.90       563
   macro avg       0.89      0.83      0.85       563
weighted avg       0.90      0.90      0.89       563
```

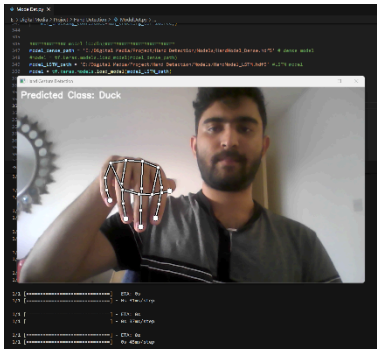
Long Short Term Memory Model	

As it can be inferred the Long Short Term Memory model performs better than the regular Feedforward model. The performance measure here is chosen to be accuracy which is 96% for the Long Short Term Memory but 90% for the feedforward network. Hence in the final model LSTM was chosen to be used with the game.

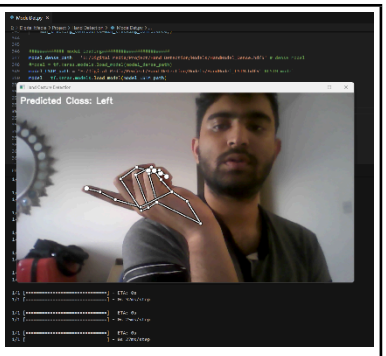
Below is the performance of the model with the real time data from the webcam.



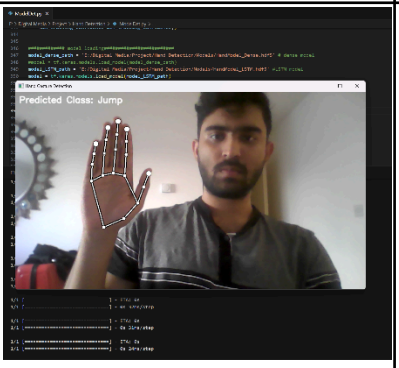
Base



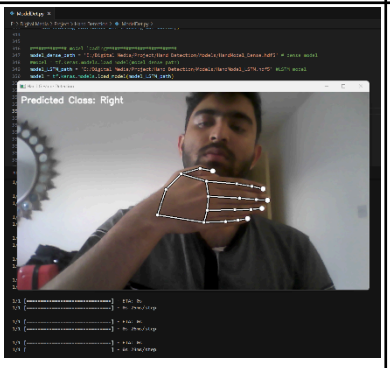
Duck



Left



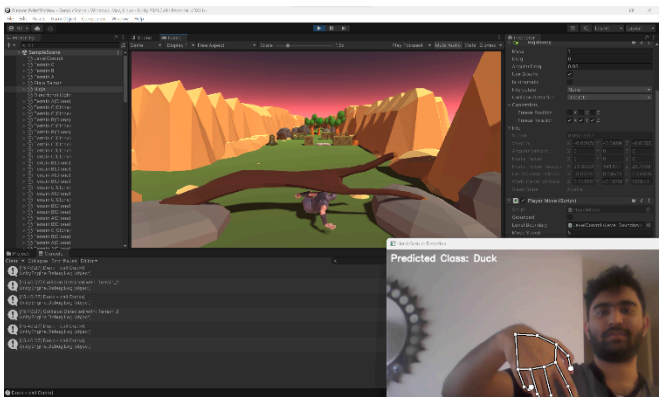
Jump

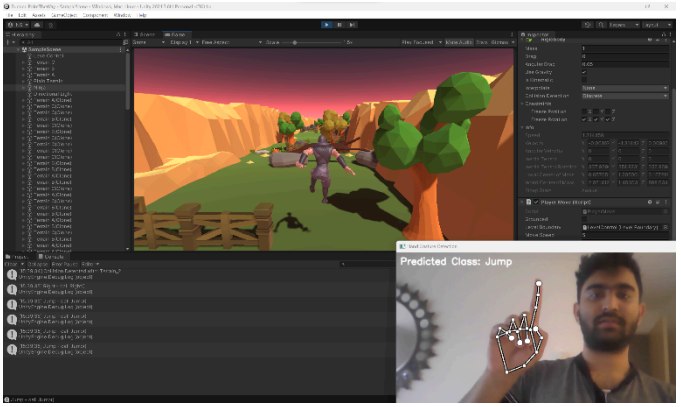


Right

All these detections were done in the LSTM model in the python code before sending it over to Unity via the Socket.

Below are the images of the model in action in the game scene





[Game Demo Link]

Discussions and Conclusion

In Conclusion, a game which uses OpenCV capabilities as input has been created. The model used in this project is super versatile that it can be expanded to make various gesture related games. The responsiveness of the model is high as it can predict at almost sixty frames per second but to facilitate socket communication, this speed has been reduced to prevent erroneous readings by unity.

As the dataset primarily consists of numbers, the LSTM models are better at learning sequential data. This works better for the projects use case as the numbers in the dataset belong to landmarks which determine the hand gesture.

In terms of challenges faced, the socket communication could be improved more to facilitate more speed and fluidity in between prediction and the game. The issue of reading multiple predictions in a second often lead to the crash of the script responsible for retrieving data from the socket. To solve this issue, an acknowledgement from unity is sent back to socket before sending the next prediction.

There were also problems related to how the landmark points were processed. Initially, the landmarks were processed and flattened in python before letting the model predict it. The issue that raised was related to this preprocessing, as the data that trained was not flattened, all the predictions were wrong. Correction for this issue led to re-coding most scripts to have the same datatype and structure.

The reason socket communication was used for its asynchronous processing, decoupled architecture, and flexibility to both python and unity. As a result, Python issues, processing and prediction takes place separately from unity rendering and player controls processing.

In the duration of the project, the massive potential of this line was discovered. Using MediaPipe as a base, any number of gestures could be made into a dataset making it predict many classes. This could mean that if the whole sign language is used to make a dataset, then this could be used to

make AR apps that could help who do not need and know the sign language understand the people who do. It can also be used to a niche anime game for Naruto as 'hand signs' is one of the basic concepts in the story. This system can also be used to design multiple role-playing games and competitive mage style games to increase immersiveness of the game.

With that said, upon keeping some questionnaire in various discord servers regarding having a hand gesture game, a common answer that resonated with everyone said that they do not want a pre-existing game to incorporate this system. They rather have a new game based on gesture over a gesture add-on to an existing game as it will make them lose their touch in that game. Thus, reaching the conclusion that this project can facilitate newer games and newer game mechanics but not old games.

This project was tried by few people after its completion to get their reviews on the topic. The suggestions received showed that games that use gesture has a big learning curve making it harder to play initially. The game currently has the collision management turned off as the players found it hard to cross the first hurdle in their playtime. The players also mentioned that though the game gets interesting once they get the hang of the controls, but as it demands showing their hand constantly to the camera, the hand and arm pain makes the players leave the game quickly but were willing to continue after some rest.

Thus, summarizing everything, this project proves the feasibility of integrating dynamic hand gesture recognition via OpenCV into Unity Gameplay. Though refinements to fluidity and accessibility are possible avenues for future work, the implementation delivers responsive, vision-based interaction. At its core, the endeavour unveils novel potentials at the intersection of computer vision and gaming while contributing pioneering tools facilitate emerging game development frontiers centred on human-computer synergy. As virtual environments continue approximating physical realities, Natural user interfaces will undoubtedly synergize vision technologies and creative design thinking to craft deeper, more intuitive connections between worlds digital and actual.

Legal, Social, Ethical, and Data Security:

While developing a gesture-based game using OpenCV integration, it is imperative to consider the legal, social, ethical and data security aspects of the project. This ensures that the game complies with relevant regulations, respects user privacy, and promotes responsible usage of technology.

When creating any game, it is crucial to abide with copyright and intellectual property regulations from a legal aspect. Make careful to take precautions to prevent the game from violating any already-existing patents, trademarks, or copyrights. Additionally, if the game uses or collects personal data, compliance with data privacy laws, including the General Data privacy Regulation (GDPR)[6] or other applicable legislation, is essential. This entails getting the proper user consent, processing, and storing personal data securely, and being open and honest about how it is used.

From a social standpoint, it is critical to take into account how inclusive and accessible the gesture-based controls are. Assuring that gesture recognition is not the only means of interaction, the game should be made to suit players with various abilities or physical constraints. To guarantee that a wider range of players can enjoy the game, different input methods or configurable controls should be taken into account. In order to avoid making gestures that can be unpleasant or unsuitable in particular cultural situations, cultural sensitivity should also be taken into account.

The ethical and responsible use of technology is one ethical consideration. The game should not promote negative actions like violence, prejudice, or any other offensive material. It is crucial to

make sure that the game respects the rights and dignity of players, abstaining from any prejudice or privacy infringement. To put user welfare first and advance wholesome societal ideals, the development process should be conducted in a transparent and ethical manner.

A crucial component of the project is data security, especially when collaborating with user data. User data should be effectively safeguarded against misuse, unauthorized access, and security breaches. This entails putting in place secure communication methods, encrypting confidential information, and consistently applying security fixes. To safeguard personal information, user data should, if feasible, be anonymized and aggregated. Additionally, it is in top priority to define clear data retention policies and provide users control over their personal information, including options to delete or modify their data. This is top priority because this project utilizes image of their camera to function, which could develop a user data problem.

A responsible and reliable result is ensured by addressing the legal, social, ethical, and data security issues during the creation of the gesture-based game employing OpenCV integration. The project can produce a pleasurable and secure gaming experience that complies with legal requirements, social norms, and ethical principles by adhering to pertinent rules and regulations, fostering inclusivity and accessibility, respecting ethical standards, and protecting user data.

References:

[1] S. S. Rautaray and A. Agrawal, "Interaction with virtual game through hand gesture recognition," 2011 International Conference on Multimedia, Signal Processing and Communication Technologies, Aligarh, India, 2011, pp. 244-247, doi: 10.1109/MSPCT.2011.6150485.

[2] R. Agrawal and N. Gupta, "Real Time Hand Gesture Recognition for Human Computer Interaction," 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, India, 2016, pp. 470-475, doi: 10.1109/IACC.2016.93.

[3] Chiang Wei Tan, Siew Wen Chin, and Wai Xiang Lim, "Game-based human computer interaction using gesture recognition for rehabilitation," 2013 IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia, 2013, pp. 344-349, doi: 10.1109/ICCSCE.2013.6719987.

[4] G. M. Kumar, V. Manohar, B. Ravi, S. V. S. Prasad, S. Paluvatla and R. Sateesh, "Game Controlling using Hand Gestures," 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), Bhubaneswar, India, 2022, pp. 1-5, doi: 10.1109/ASSIC55218.2022.10088337.

[5] N. V. Le, M. Qarmout, Y. Zhang, H. Zhou, and C. Yang, "Hand Gesture Recognition System for Games," 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Brisbane, Australia, 2021, pp. 1-6, doi: 10.1109/CSDE53843.2021.9718421.

[6] Art. 5 GDPR – principles relating to processing of personal data. (2021, October 22). https://gdpr-info.eu/art-5-gdpr/

[7] Asana. (n.d.). Manage your team's work, projects, & tasks online • asana. Asana. https://asana.com/

[8] Jimmy Vegas, (2021). How to make a 3d endless runner game in unity for pc & mobile - tutorial #01 - introduction [online]. *YouTube*. [Viewed 16 October 2023]. Available from: https://www.youtube.com/watch?v=u5hRtTEhnOA

[9] Gesture recognition task guide | MediaPipe | Google for Developers [online], (no date). *Google for Developers*. [Viewed 21 June 2023]. Available from: https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer#get_started

[10] Ivan Goncharov, (2022). Custom hand gesture recognition with hand landmarks using Google's MediaPipe + OpenCV in python [online]. *YouTube*. [Viewed 26 June 2023]. Available from: https://www.youtube.com/watch?v=a99p_fAr6e4

List of Outsourced materials and assets

[1] Animations - https://www.mixamo.com/#/?page=1&query=&type=Motion%2CMotionPack

[2] Player model - https://www.mixamo.com/#/?page=1&query=ninja&type=Character

[3] MediaPipe Documentation - Gesture recognition task guide | MediaPipe | Google for Developers

[4] Game Assets –

LowPoly environment pack | 3D landscapes | unity asset store [online], (no date). *Unity Asset Store - The Best Assets for Game Making*. [Viewed 28 December 2023]. Available from: https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479

WoodLand - LowPoly environment by unvik_3d | 3D landscapes | unity asset store [online], (no date). *Unity Asset Store - The Best Assets for Game Making*. [Viewed 28 December 2023]. Available from: https://assetstore.unity.com/packages/3d/environments/landscapes/woodland-lowpoly-environment-by-unvik-3d-203187