Integrating Real-Time User Input with Gesture Classification

Pranav K Nair
College of Engineering
Northeastern University
Boston, MA, USA
nair.pranav@northeastern.edu

Sai Prasasth Koundinya Gandrakota
College of Engineering
Northeastern University
Boston, MA, USA
gandrakota.s@northeastern.edu

Pratik Chatterjee
College of Engineering
Northeastern University
Boston, MA, USA
chatterjee.pr@northeastern.edu

***Abstract*—The significance of gesture recognition systems in enhancing user interactivity and accessibility within virtual environments has grown substantially. This study evaluates the performance of two deep learning models, MobileNetV2 and EfficientNetV2, in interpreting hand gestures for real-time inputs in 2D games. Both models undergo training and validation using the HaGRID dataset, which encompasses diverse gestures captured under various conditions. After determining the most effective model, it is integrated into user inputs for a car racing game and lunar lander simulation. By converting physical gestures into digital commands, the system aims to improve the intuitiveness of human-computer interaction and provide a smooth user experience. The implementation of this system in these games not only showcases the potential of gesture-based interfaces but also suggests broader applications in fields requiring accessible and intuitive user interfaces. Future iterations will concentrate on refining the chosen model and enhancing responsiveness to further enhance the interactive experience.**

I. Introduction

In recent years, the advance of virtual environments and interactive technologies has created a demand for intuitive and interactive user interfaces. Traditional input methods, such as keyboards and controllers, while effective, can be limiting in complexity and accessibility. Gesture recognition systems offer a promising alternative by enabling users to interact with digital environments through natural hand movements. Gesture classification, a key component of gesture recognition systems, involves the categorization of gestures into distinct classes based on their visual characteristics. This process requires robust algorithms capable of identifying subtle differences between gestures and generalizing across diverse inputs [1].

Central to the development of gesture classification systems are convolutional neural networks (CNNs), a class of deep learning models specifically designed to process visual data. CNNs excel at learning hierarchical representations of images, making them well-suited for tasks such as image classification and object detection. Through layers of convolutions and pooling operations, CNNs can effectively extract features from raw image data [2]. Transfer Learning, a benefit of using CNNs allows the use of pre-trained models for further fine-tuning for specific purposes. Two major pre-trained models used for this purpose are MobileNetV2 and EfficientNetV2.

MobileNetV2 is a type of convolutional neural network architecture that is designed for efficient and lightweight mobile applications [3]. It is well suited for tasks like image classification and object detection on resource-constrained devices such as smartphones and embedded systems. It achieves this by using lightweight depth-wise convolutions and a special kind of residual block called an inverted residual block [4]. EfficientNetV2 is an advanced convolutional neural network architecture that is aimed at maximizing model efficiency while maintaining high accuracy across a wide variety of tasks, including image classification and object detection [5].

In this project, we aim to utilize and optimize these two models using transfer learning to classify hand gestures accurately in real-time. Subsequently, the most effective model is integrated into user inputs for a car racing game and lunar lander simulation, demonstrating the potential of gesture-based interfaces to enhance user interaction and accessibility. The integration of gesture recognition systems into gaming environments hints at broader implications across industries requiring accessible and intuitive user interfaces. As advancements in deep learning continue to refine these systems, future versions will focus on further enhancing responsiveness and refining the chosen model to provide an even more immersive and interactive user experience.

II. Implementation

The project consists of two main components, the setup of the gesture classification model and the game environment followed by the implementation where real-time input is provided to the model and the output is observed in the game.

A. Setup

The necessary data for training the machine learning model is gathered and prepared. This involves cleaning and preprocessing the data to make it suitable for the model by resizing the image as required and then normalizing the pixel intensity values. Once the data is ready, it is split into training, testing, and validation sets to evaluate the performance of different models. Both MobileNetV2 and EfficientNetV2 models are trained using the training data with simultaneous testing on the validation data at each step, and their performance is assessed on the test data.

The results are analyzed to identify the strengths, weaknesses, and areas for improvement of each model. Based on

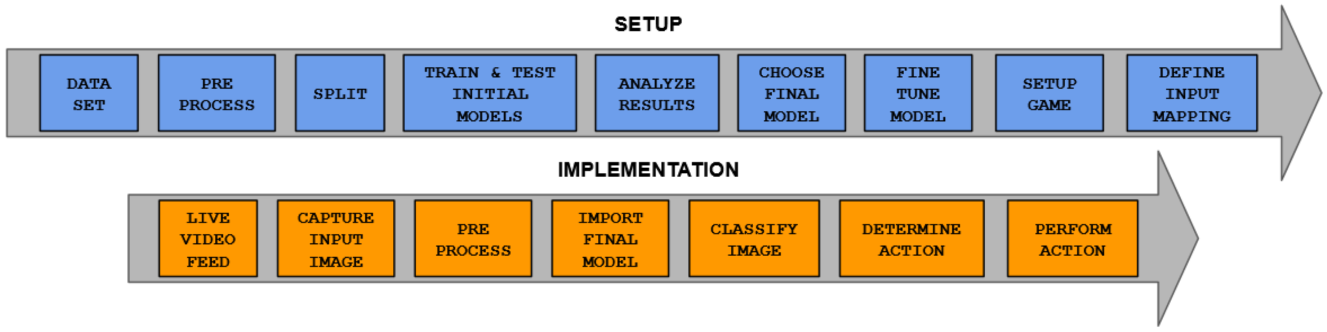

Fig. 1. Workflow for setup and implementation of project

this analysis, the best-performing model is selected to move forward with. The selected model is fine-tuned to further optimize its performance. With the final model trained and optimized, the game environment and infrastructure are set up to accommodate the model. The mapping of the gesture classes to the input pool of the game is defined.

B. Demo

A live video feed for real-time input to the model is set up by using a camera or video capturing device connected to the system where the model is running. This could be a webcam, a smartphone camera, or any other video source capable of capturing real-time footage. Input images are captured and pre-processed. The final trained and optimized model is imported and used to classify the input image or data. Based on the classification output from the model, a decision is made regarding the action to be performed in the game or application. This decision could be determined by a set of rules, a policy defined by the application, or by directly mapping the model's output to specific actions. Finally, the determined action is executed within the game or application.

C. Tools

The entire setup and implementation of the project is done in Python, utilizing various libraries for different tasks as shown below:

- Pandas, Numpy, PIL, OpenCV - Importing and pre-processing images.
- Tensorflow, Keras and Scikit-learn - Splitting data set and developing CNN models.
- Matplotlib, Seaborn and Plotly - Visualizing and analyzing results.
- Gymnasium and Box2D - Setting up game environment and mapping gesture classes to inputs.

III. Experimentation

A. Dataset

For the purpose of training the models, we have used the HaGRID (HAnd Gesture Recognition Image Dataset). The original dataset contains 18 + 1 classes with 554800 images and is 723GB in size. The "+ 1" class is meant to be an extra "no gesture" class to see if there is a second free hand in the frame [6]. However, these images are in FHD resolution and may require more computational resources than what is available to us and thus we have decided to choose the smaller image set.

This smaller image set contains about 30000 images at a 380p resolution and is less than 1GB in size [7]. We have chosen to take 6 classes of gestures to train our models on. These 6 classes are "like", "dislike", "peace", "peace inverted", "ok", and "stop". Fig 2. depicts the different gestures to be recognized by the model. Fig 3. shows the label distribution of the dataset.

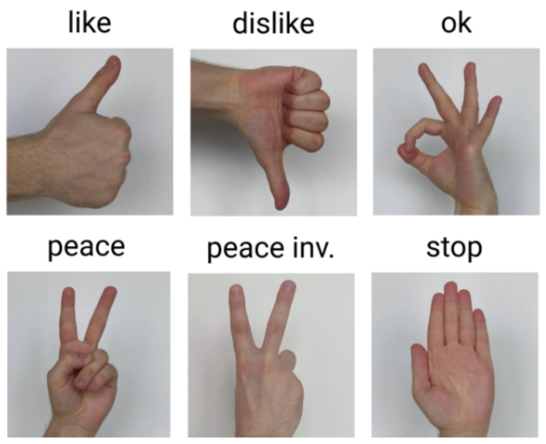

Fig. 2. 6 Classes of Gestures

B. MobileNetV2

The architecture used in the first model is a combination of the MobilbeNetV2 as the base model along with a GlobalAveragePooling2D layer followed by 2 Dense layers. The final dense layer is meant to output an array of size 6 whose values are the probabilities for each of the 6 classes. The shape of an image that is being inputted to the model is (224, 224, 3), where the "3" signifies the R, G, and B values. The model uses an Adam Optimizer with a learning rate of 0.001, Sparse Categorical Cross Entropy for the loss, and Sparse Categorical Accuracy for the metric. The model is trained for 20 epochs with a batch size of 128. A Reduce on Plateau

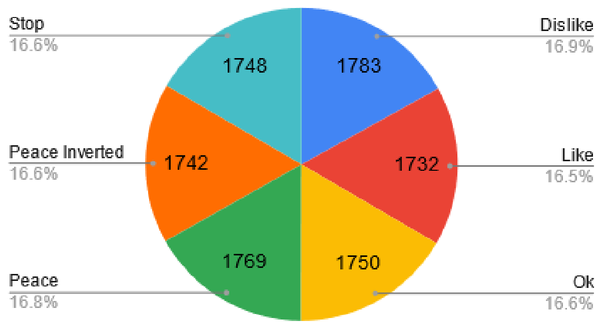

Fig. 3. Label Distribution of Dataset

scheduler is used so that if the validation accuracy starts plateauing between 3 epochs, the learning rate is reduced. This helps in preventing overfitting of the model. The model specifications mentioned here are the same when developing the EfficientNetV2 architecture. This is done so as to be able to compare the performance of the 2 models.

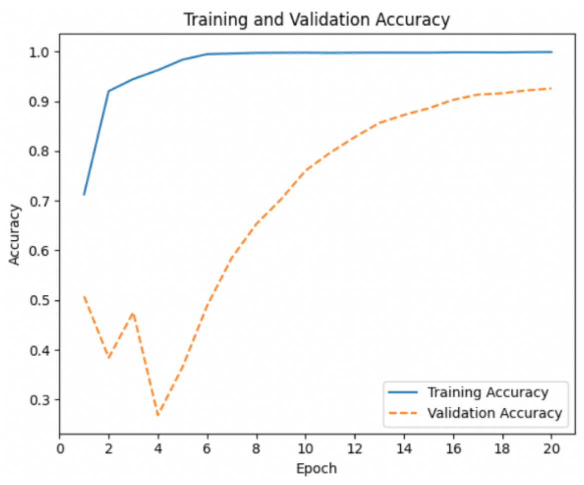

Fig. 4. MobileNetV2 Resulting Accuracy

Fig 4. and 5. show the resulting training and validation accuracies and losses of the model. It can be seen in Fig 4. that between the 4th and 6th epoch, the training accuracy starts plateauing and it is around this same time that the validation accuracy starts rapidly increasing. Another observation is that in Fig 5. while the training loss is quite low from the beginning, it does decrease and start plateauing somewhere between the 4th and 6th epoch. Simultaneously, the validation loss ends up having a sharp decrease.

C. *EfficientNetV2*

The architecture used for the second model is the same as the MobileNetV2 model except the base model would be the EfficientNetV2S. We use the ”S” version here as it is the small version of EfficientNetV2 and would be less computationally taxing.

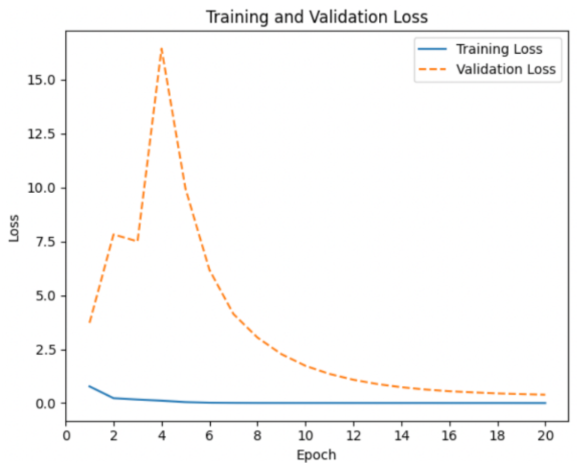

Fig. 5. MobileNetV2 Resulting Loss

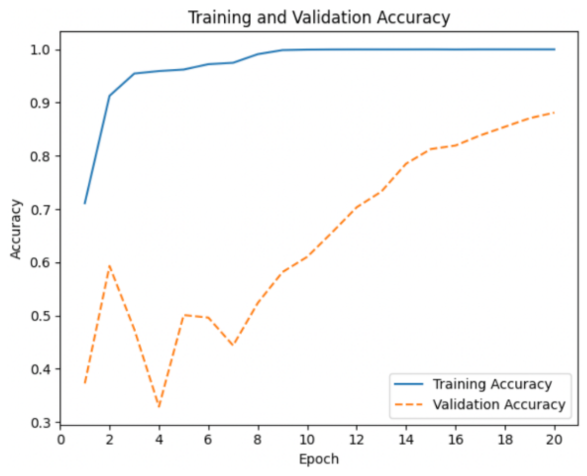

Fig. 6. EfficientNetV2 Resulting Accuracy

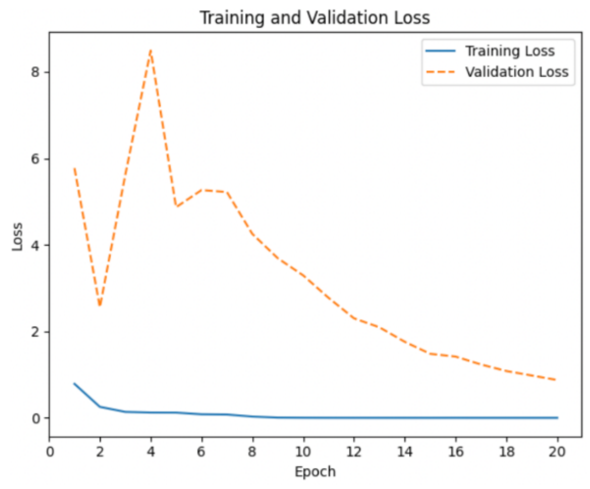

Fig. 7. EfficientNetV2 Resulting Loss

Fig 6. and 7. show the resulting training and validation accuracies and losses of the model. It can be seen in Fig 6. that between the 6th and 8th epoch, the training accuracy starts plateauing and it is around this same time that the validation accuracy starts rapidly increasing. Another observation is that in Fig 7. while the training loss is quite low from the beginning, it does decrease and start plateauing somewhere between the 6th and 8th epoch. Simultaneously, the validation loss ends up having a sharp decrease.

D. Input Mapping

The second part of this project revolves around using the output of our model as input for the games we have chosen. We chose to use the Car Racing and Luna Lander Box2d environments from the Gymnasium library as the games we implement our model on.

In the Car Racing environment, there are 3 main parts of the action space, i.e. the steering (ranging from -1 to 1), the acceleration (ranging from 0 to 1), and the braking (ranging from 0 to 1). Table I shows the mapping of the output of the model to the input/action of the environment.

TABLE I
CAR RACING GAME

Gesture	Action	Action Input
Dislike	Hard Left	[-1, 0.5, 0]
Like	Soft Left	[-0.5, 0.5, 0]
Ok	Forward	[0, 1, 0]
Peace	Hard Right	[1, 0.5, 0]
Peace Inverted	Soft Right	[0.5, 0.5, 0]
Stop	Brake	[0, 0, 1]

In the Luna Lander environment, there are 4 actions available in the actions space. These actions and their mapping are detailed in Table II.

TABLE II
LUNA LANDER GAME

Gesture	Action	Action Input
Dislike	Fire Left Orientation Engine	1
Ok	Fire Main Engine	2
Peace	Fire Right Orientation Engine	3
Stop	Do Nothing	0

IV. RESULTS

A. Comparision between initial models

In the landscape of neural network architectures, MobileNetV2 and EfficientNetV2 stand out for their unique strengths shown in Fig. 8. MobileNetV2, with a test accuracy of 92.4% and a loss of 0.4936, demonstrates an ability to make accurate predictions efficiently. EfficientNetV2, although slightly behind with an accuracy of 89.17% and a higher loss of 0.8923, excels in managing complex and varied datasets, suggesting a robustness that's valuable for extensive and scalable tasks. Between the two, MobileNetV2 is preferred for its high accuracy and low loss while EfficientNetV2 could

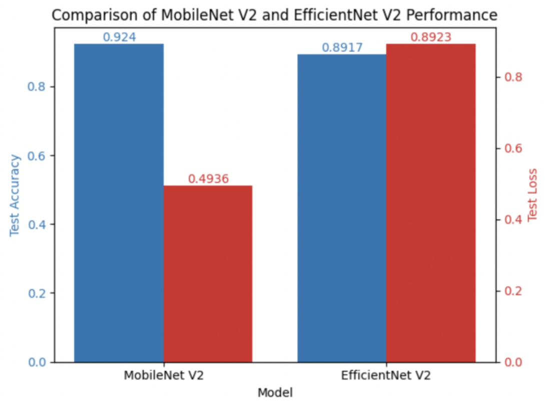

Fig. 8. Test and loss accuracy of MobileNetV2 and EfficientNetV2

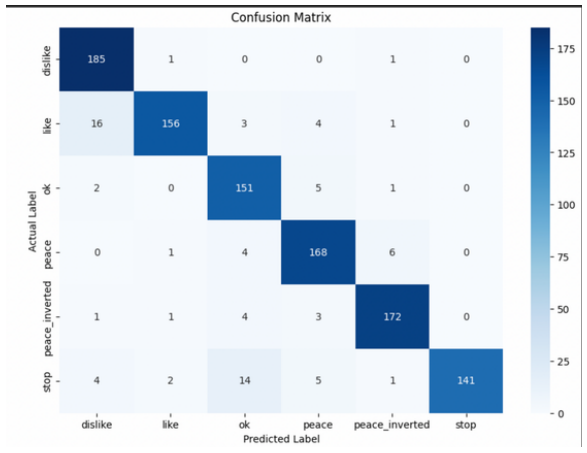

Fig. 9. MobileNetV2 Confusion Matrix

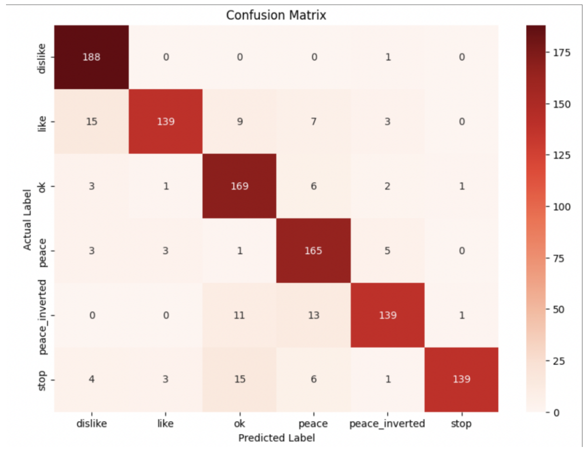

Fig. 10. EfficientNetV2 Confusion Matrix

be the model of choice for its adaptability and scalability in diverse conditions.

The confusion matrix heatmap in Fig. 9 and Fig. 10 provides a detailed view of the model's performance across different classes. The confusion matrix for both models is almost the same with the same insights. Key insights include:

Diagonal Dominance: A well-performing model will have high values along the diagonal (top left to bottom right), indicating correct predictions. This heatmap shows significant diagonal dominance, which suggests that the model accurately classifies a high proportion of the instances.

Misclassifications: Off-diagonal cells indicate misclassifications, meaning if there's a high number in a cell that's not on the diagonal, it signifies that many instances of one class were misclassified as another. In this case, the relatively low numbers in off-diagonal cells indicate that the model makes few misclassification errors.

When evaluating the classification reports for MobileNetV2 and EfficientNetV2 (shown in Table III), it becomes apparent that each model has its distinct strengths. MobileNetV2 excels with its high precision in class 5 and substantial recall for class 0, making it particularly adept at minimizing false positives in critical classifications. On the other hand, EfficientNetV2's impressive recall in class 0 suggests it's less likely to miss true positives, which could be crucial in sensitive applications. If we consider the F1 score as a measure of a model's balanced performance, MobileNetV2 shows a slight edge, especially notable in class 4. However, EfficientNetV2's near-perfect precision in class 5 indicates its strong capability to correctly predict this class with almost no false positives. The choice between the two models hinges on the specific requirements of the task, with MobileNetV2 offering broad reliability and EfficientNetV2 providing exceptional precision where it matters most.

B. Final Model

Based on comparisons done between the models, MobileNetV2 was chosen to further improve and utilize as the final model for gesture classification before input mapping. In order to improve the model, data augmentation is performed so that the training data pool size is increasing. Using the images from the data set, various augmented versions are generated by altering scale, size, orientation and other properties of the image, as shown in Fig. 11

Using the base model of MobileNetV2, similar to the initial model a GlobalAveragePooling2D layer followed by 2 Dense layers are used, but this time an additional Dense layer is also added. Additionally, Dropout layers and L2 regularization are applied to the output of the Dense layers to prevent the model from overfitting on the training data. The model is trained for 20 epochs with a batch size of 32.

The training and validation accuracy and loss of the final model follows a much smoother path over the training epochs. The validation accuracy and loss coincide faster with the training accuracy and loss and maintain a steady progression alongside them.

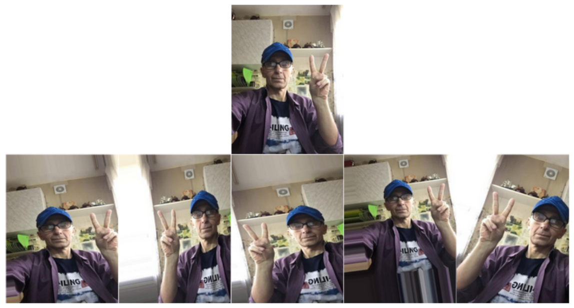

Fig. 11. Augmented Images

After testing the model on the test data set we compared it's performance with that of the intial models. From Table III we can see that highs and lows of the precision, recall and F1 scores of the final model for each class are higher than that of the previous two models. The mean values of these metrics also show that the final model has higher values.

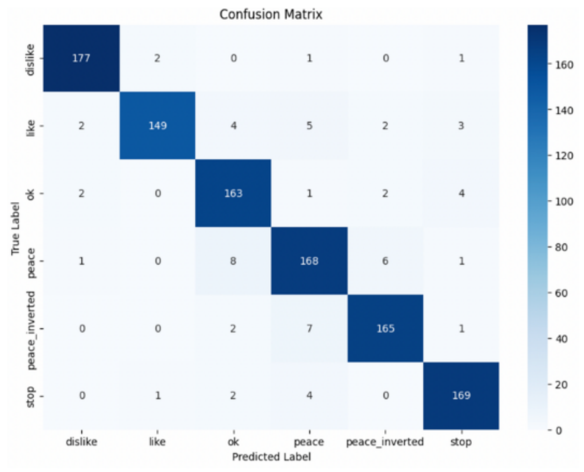

Fig. 12. Final Model Confusion Matrix

Fig. 12 shows that the final model has a much more even performance across classes and With a final test accuracy of 94.1% and a loss of 0.21 (Table IV), the final model can be confirmed to outperform the initial models and that it is as optimized as it can be before moving to the final stage of the setup of the project.

C. Demo

To visualize the final outcome of the project, we created a video game-like system where there is a colorful and interactive menu with music to go with it. The menu allows the user to select the game of their choice. On selection, the game environment opens up and waits for 3 seconds for the user to get ready and finally, the game starts. The link for the project code along with the video of the demo can be found in the reference point [8].

TABLE III
PERFORMANCE METRICS FOR INITIAL AND FINAL MODELS

	MobileNetV2			**EfficientNetV2**			**Final Model**		
Class	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
0 (Dislike)	0.89	0.99	0.94	0.88	0.99	0.94	0.97	0.98	0.98
1 (Like)	0.97	0.87	0.91	0.95	0.80	0.87	0.98	0.90	0.94
2 (Ok)	0.86	0.95	0.90	0.82	0.93	0.87	0.91	0.95	0.93
3 (Peace)	0.91	0.94	0.92	0.84	0.93	0.88	0.90	0.91	0.91
4 (Peace Inverted)	0.95	0.95	0.95	0.92	0.85	0.88	0.94	0.94	0.94
5 (Stop)	1.00	0.84	0.92	0.99	0.83	0.90	0.94	0.96	0.95
Mean	0.93	0.92	0.92	0.90	0.89	0.89	**0.94**	**0.94**	**0.94**

TABLE IV
ACCURACY COMPARISON

Model	Accuracy	Loss
MobileNetV2	92.4 %	0.49
EfficientNetV2	89.2 %	0.89
Final Model	**94.1 %**	**0.21**

V. CONTRIBUTIONS

Pranav K Nair: Worked on developing the EfficientNetV2 model. Set up the Car Racing and Luna Lander environment and mapped the output of the final model to the inputs of the games. Created a user-friendly UI for the demo so as to improve the user experience.

Sai Prasasth Koundinya Gandrakota: Worked on cleaning and pre-processing images from the dataset for smooth input into CNN models, developing the MobileNetV2 model, and developing and optimizing the final model used for gesture classification.

Pratik Chatterjee: Worked on the data acquisition and conducted data analyses to evaluate model performance across various metrics, identifying key trends and improvement areas. Tasked with creating visualizations that communicated project results and progress, supporting both internal understanding and external reporting.

VI. CONCLUSIONS

This project has significantly advanced the field of human-computer interaction by demonstrating how deep learning models like MobileNetV2 and EfficientNetV2 can be applied to effectively interpret and translate hand gestures into real-time inputs for 2D applications. Achieving a high recognition accuracy of 94.1%, the system not only enhances user accessibility and interaction but also proves its practical applicability in interactive games such as car racing and lunar lander simulations. Key challenges such as computational demands and initial accuracy issues were effectively overcome through strategic optimizations in data pre-processing and model tuning. The success of this project lays a solid foundation for further research, promising even greater advancements in gesture recognition technology to create more intuitive and inclusive user interfaces.

VII. FUTURE WORK

While the current project has successfully demonstrated the application of gesture recognition technology in enhancing user interaction within 2D applications, there are multiple avenues for future development that could further refine and expand the utility of this technology.

Introducing adjustable difficulty levels could cater to a broader user base, from novices to experts, thereby enhancing the adaptability of the system. Integrating additional gestures and diverse datasets can increase the robustness of the model, ensuring it learns from a wider array of gestures and scenarios. Increasing the variability of the mapping system would allow gestures to correspond to a range of controls rather than a single input, providing users with a more nuanced and flexible interaction. Exploring methods to map recognized gestures to games that require complex inputs could enhance gaming immersion and accessibility, particularly for users with varying abilities. Adapting the model for real-world applications, would extend utility beyond controlled environments and into diverse, everyday contexts where gesture-based interactions could be transformative. The development of an interactive dashboard for visualizations can provide users with real-time feedback and deeper insights into the model's performance and their interaction patterns.

REFERENCES

[1] Y. Li, T. Wang, A. khan, L. Li, C. Li, Y. Yang, and L. Liu, "Hand gesture recognition and real-time game control based on a wearable band with 6-axis sensors," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–6.

[2] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[3] M. Akay, Y. Du, C. Sershen, M. Wu, T. Chen, S. Assassi, C. Mohan, and Y. Akay, "Deep learning classification of systemic sclerosis skin using the mobilenetv2 model," *IEEE Open Journal of Engineering in Medicine and Biology*, vol. PP, pp. 1–1, 03 2021.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 06 2018, pp. 4510–4520.

[5] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," 2021.

[6] "Original hagrid data set," 2022. [Online]. Available: https://www.kaggle.com/datasets/kapitanov/hagrid

[7] "Hagrid sample 30k 384p," 2022. [Online]. Available: https://www.kaggle.com/datasets/innominate817/hagrid-sample-30k-384p

[8] "Final project link," 2024. [Online]. Available: https://tinyurl.com/3bsnve88