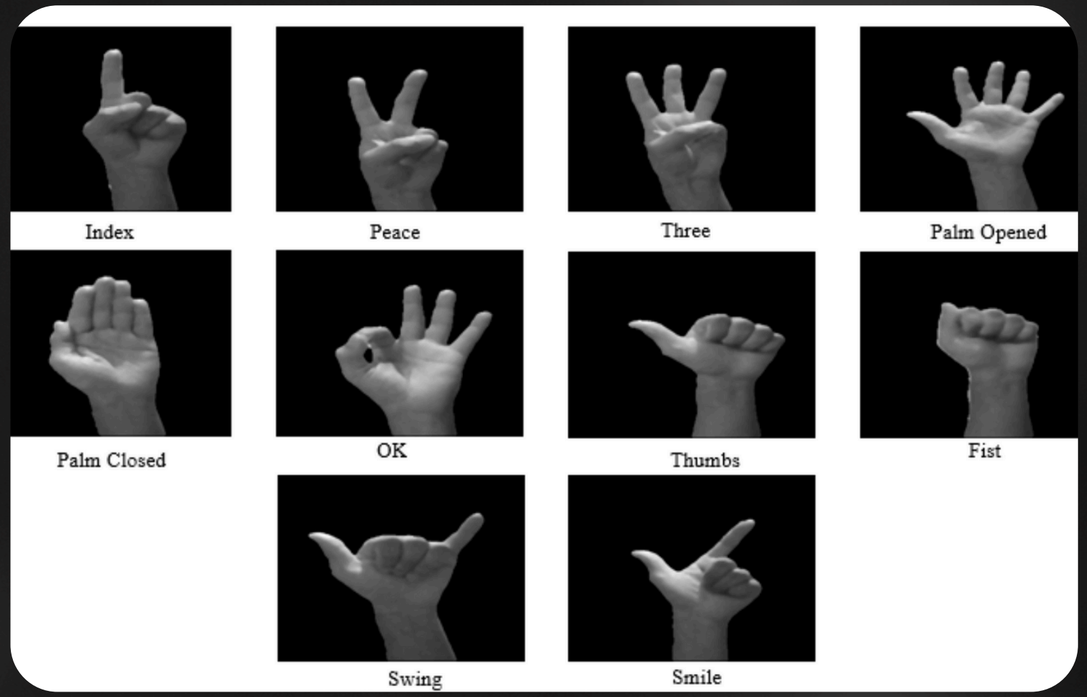
HAND GESTURE RECOGNITION SYSTEM

Team Members:

1-Yousef Elsayed Lashin
2-Mahmoud Osama Shokry
3-Mariam Tarek Mahmoud Abdelrahman
4-Romisaa Hassan Hassan
5-Fatma Ashraf Fawzy

Project Overview

The Hand Gesture Recognition System is a deep learning project aimed at enabling real-time interaction between humans and computers through hand gestures. By using computer vision techniques and neural networks, the system can recognize and classify different hand signs captured via camera. This technology has wide-ranging applications in fields such as virtual reality, human-computer interaction, and accessibility tools for individuals with disabilities. The project consists of several milestones including data collection, model training, real-time deployment, and system monitoring to ensure continuous improvement



vision

To create an intelligent and accessible hand gesture recognition system that bridges the gap between humans and machines, empowering seamless interaction in daily life and advanced digital environments

mission

Our mission is to design, develop, and deploy a reliable real-time gesture recognition system using deep learning and computer vision, providing innovative solutions for virtual reality, assistive technologies, and human-computer interaction

value

Innovation

Efficiency

Integrity

User-Centricity

Collaboration

what makes my project stand out

Our Hand Gesture Recognition System stands out for its real-time performance, high accuracy, and practical integration with real-world applications. The system uses deep learning and advanced computer vision techniques to recognize hand gestures quickly and efficiently. It is also designed with a user-friendly interface and can be easily connected to various technologies such as virtual reality systems, smart devices, and accessibility tools. This makes the project both innovative and impactful in everyday use

Milestone 1: Data Collection, Preprocessing, and Exploration

In the first milestone, we focused on building a custom dataset to our project's specific requirements. We collected images for different hand gestures manually, ensuring a variety of lighting conditions, hand positions, and backgrounds to increase the model’s robustness. After collection, we preprocessed the data by resizing the images, normalizing pixel values, and applying background removal techniques. We also used data augmentation methods such as rotation, flipping, and scaling to expand the dataset and improve model generalization. This stage was crucial in preparing high-quality data for training

Live Application

```
ıving images to: E:\new yousef\hand-gesture-recognition\data\raw
ebcam is accessible. Starting image capture...
ıpturing images for sign: hello
ıage 1 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_0.jpg
ıage 2 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_1.jpg
ıage 3 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_2.jpg
ıage 4 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_3.jpg
ıage 5 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_4.jpg
ıage 6 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_5.jpg
ıage 7 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_6.jpg
ıage 8 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_7.jpg
ıage 9 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_8.jpg
ıage 10 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_9.jpg
ıage 11 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_10.jpg
ıage 12 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_11.jpg
ıage 13 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_12.jpg
ıage 14 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_13.jpg
ıage 15 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_14.jpg
ıage 16 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_15.jpg
ıage 17 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_16.jpg
ıage 18 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_17.jpg
ıage 19 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_18.jpg
ıage 20 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_19.jpg
ıage 21 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_20.jpg
ıage 22 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_21.jpg
.
ıage 4997 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_4996.jpg
ıage 4998 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_4997.jpg
ıage 4999 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_4998.jpg
ıage 5000 captured: E:\new yousef\hand-gesture-recognition\data\raw\hello_4999.jpg
utput is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Milestone 2: Model Development and Training

In Milestone 2, we focused on developing and training a model for hand gesture recognition. We selected MobileNetV2, a pre-trained Convolutional Neural Network (CNN), and fine-tuned it for better performance on our hand gesture dataset. The model was trained using a batch size of 64 for 20 epochs, with both initial and fine-tuning phases to improve accuracy. We then optimized the model by adjusting hyperparameters like learning rates and trainable layers. The final deliverables include a trained model and an evaluation report with performance metrics such as accuracy, precision, recall, F1-score, and a confusion matrix.

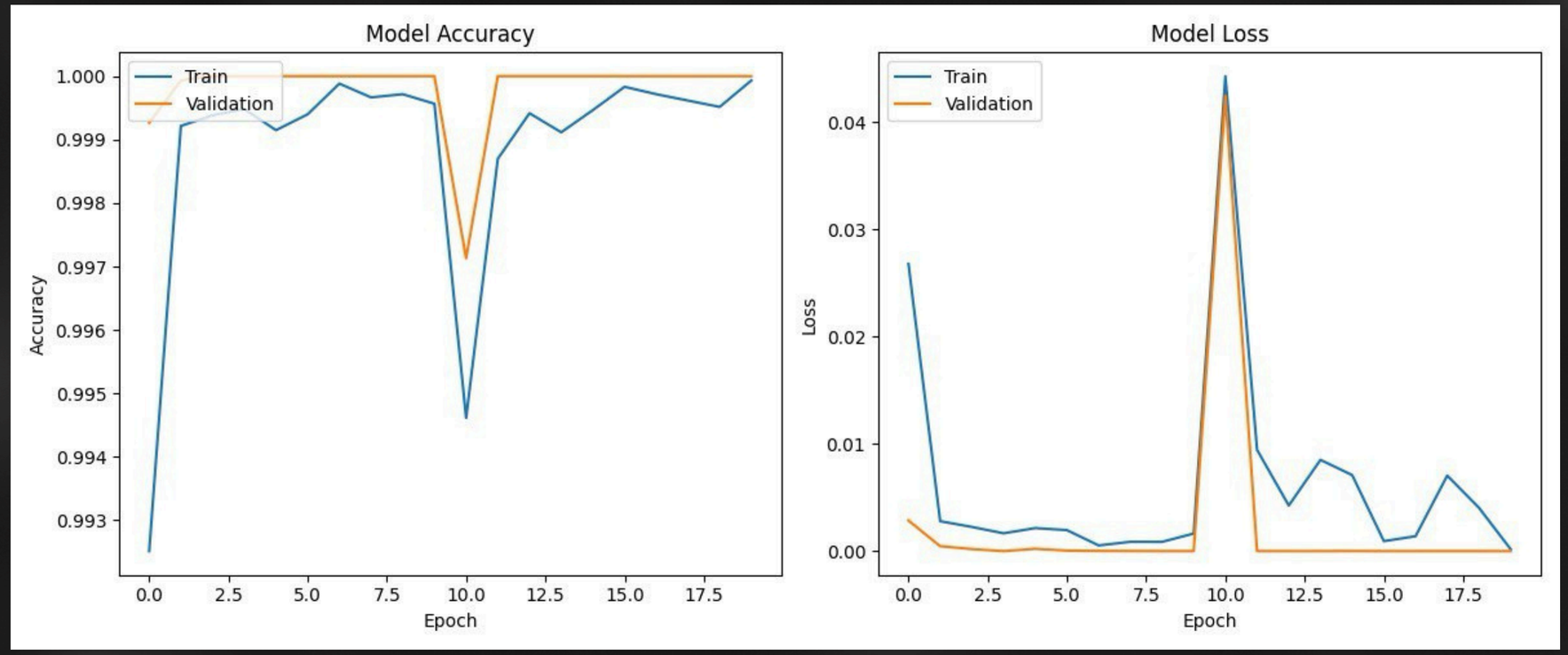
Live Application

```
# Define the model using MobileNetV2
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))
base_model.trainable = False

model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Live Application



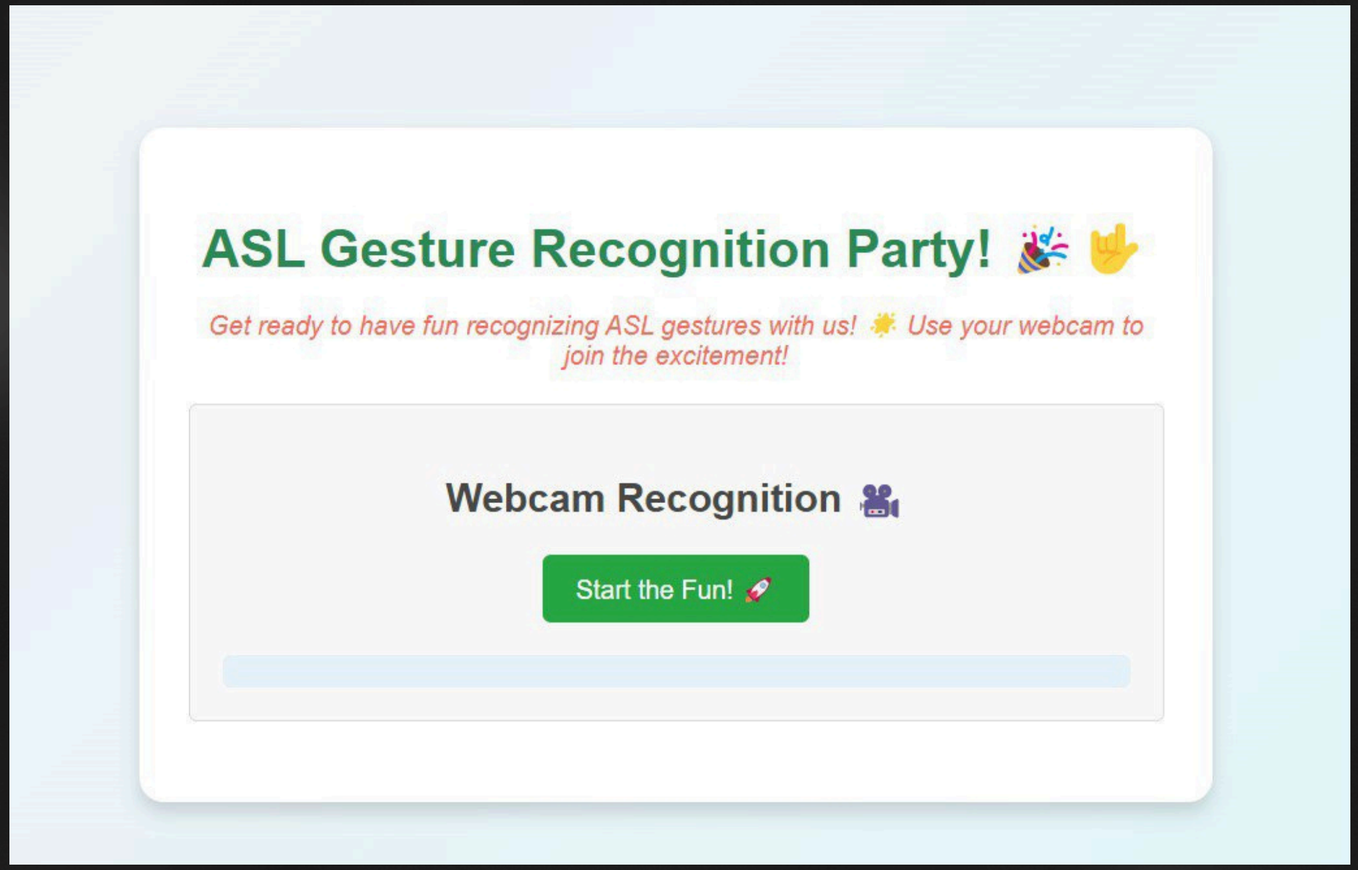
Live Application

```
Epoch 1/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 83s 84ms/step - accuracy: 0.9689 - loss: 0.0992 - val_accuracy: 0.9993 - val_loss: 0.0029
Epoch 2/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 79s 84ms/step - accuracy: 0.9993 - loss: 0.0029 - val_accuracy: 0.9999 - val_loss: 4.6392e-04
Epoch 3/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 79s 84ms/step - accuracy: 0.9994 - loss: 0.0027 - val_accuracy: 1.0000 - val_loss: 1.9476e-04
Epoch 4/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 79s 85ms/step - accuracy: 0.9994 - loss: 0.0015 - val_accuracy: 1.0000 - val_loss: 7.4194e-07
Epoch 5/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 80s 85ms/step - accuracy: 0.9990 - loss: 0.0032 - val_accuracy: 1.0000 - val_loss: 2.2067e-04
Epoch 6/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 81s 87ms/step - accuracy: 0.9991 - loss: 0.0028 - val_accuracy: 1.0000 - val_loss: 5.0851e-05
Epoch 7/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 80s 86ms/step - accuracy: 0.9999 - loss: 3.9933e-04 - val_accuracy: 1.0000 - val_loss: 1.9974e-05
Epoch 8/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 81s 86ms/step - accuracy: 0.9998 - loss: 6.1865e-04 - val_accuracy: 1.0000 - val_loss: 1.1788e-05
Epoch 9/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 81s 86ms/step - accuracy: 0.9997 - loss: 8.0016e-04 - val_accuracy: 1.0000 - val_loss: 8.3646e-07
Epoch 10/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 82s 87ms/step - accuracy: 0.9996 - loss: 0.0014 - val_accuracy: 1.0000 - val_loss: 4.1890e-06
Epoch 1/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 138s 138ms/step - accuracy: 0.9866 - loss: 0.1144 - val_accuracy: 0.9971 - val_loss: 0.0424
Epoch 2/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 127s 136ms/step - accuracy: 0.9984 - loss: 0.0116 - val_accuracy: 1.0000 - val_loss: 1.9297e-06
Epoch 3/10
...
Epoch 9/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 127s 136ms/step - accuracy: 0.9994 - loss: 0.0064 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 10/10
937/937 ━━━━━━━━━━━━━━━━━━━━ 129s 137ms/step - accuracy: 1.0000 - loss: 2.0000e-04 - val_accuracy: 1.0000 - val_loss: 4.8768e-
```

Milestone 3: Real-Time Gesture Recognition and Deployment

In Milestone 3, the goal is to implement a real-time gesture recognition system. Using Flask, the app captures hand gestures through images or live camera feeds and processes them using the trained model. The model predicts the gesture and returns a label with confidence. This milestone focuses on deploying the system, allowing users to interact with it through a web interface or by using real-time camera input

Live Application



End Users

- people with disabilities: Helping them interact with digital devices using simple hand gestures.
- VR/AR users: Enabling gesture-based control in immersive environments.
- Smart home users: Using gestures to control devices hands-free.
- Educators & Presenters: Allowing them to flip slides or control systems without touching a devices

Features

- Real-time recognition: Instant response to hand gestures.
- Custom dataset support: Trained on manually collected gestures for better accuracy.
- Lightweight model: Based on MobileNetV2, optimized for fast performance.
- Easy integration: Can be connected to other systems like smart TVs, virtual environments, or accessibility tools.
- User-friendly interface: Simple and intuitive design for all user types

Data Structure

Our dataset was custom-built and organized to support supervised learning. Each image file followed a structured naming pattern to include the gesture label . All images were stored in a single directory, and class labels were extracted from the filenames. The dataset included five gesture classes: hello, yes, no, i love you, and thank you. To ensure balanced learning, we applied preprocessing techniques such as resizing, normalization, and data augmentation.

Data Flow

- Image Input: Hand gesture images are captured or loaded from the custom dataset.
- Preprocessing: Images are resized, normalized, and background noise is reduced.
- Data Augmentation: Techniques like rotation and flipping are applied to increase dataset diversity.
- Model Input: Preprocessed images are fed into the MobileNetV2-based model.
- Prediction: The model classifies the gesture into one of the five predefined classes.
- Output: The predicted gesture is displayed or used to trigger an action (e.g., flip slide, control device

Challenges We Faced

During the development of the Hand Gesture Recognition System, we encountered several challenges:

- Custom dataset collection: Gathering high-quality gesture images manually was time-consuming and required consistent lighting and background settings.
- Data imbalance: Some gestures had more images than others, which affected the model's performance and required additional augmentation.
- Background noise: Removing background clutter to focus on the hand was tricky and affected data quality.
- Model tuning: Finding the right training settings and fine-tuning the model to improve accuracy without overfitting took several trials.
- Performance on different conditions: Ensuring the model works well under various lighting and hand positions was a key challenges

Conclusion

Our Hand Gesture Recognition System demonstrates the power of deep learning and computer vision in creating real-time, touchless interaction between humans and machines. By building a custom dataset, designing an efficient model, and focusing on user-centered features, we delivered a system that is both practical and innovative. Despite the challenges faced, we successfully developed a solution with real-world applications in accessibility, smart environments, and virtual reality. This project lays the foundation for future enhancements and broader gesture-based technologies.

THANK YOU