ADVANCED MACHINE LEARNING APPLICATIONS IN COMPUTER VISION: VISUALIZATION OF CNN ACTIVATION MAPS, REAL-TIME SENIOR CITIZEN IDENTIFICATION, AND SIGN LANGUAGE RECOGNITION

INTRODUCTION

THIS INTERNSHIP CENTERED AROUND THE DEVELOPMENT AND IMPLEMENTATION OF MACHINE LEARNING MODELS TO TACKLE SPECIFIC COMPUTER VISION CHALLENGES. THE PROJECT LEVERAGED POWERFUL TOOLS, INCLUDING OPENCV (CV2) AND TENSORFLOW, TO ACHIEVE ITS GOALS.

Visualize CNN Activation Maps : Implement techniques to analyze and interpret CNN activation maps, which helps in understanding how different parts of an image influence model predictions and improves insight into the model's decision-making process.

Real-Time Senior Citizen Detection System: Build a robust system to identify and classify individuals based on age and gender in real-time from video feeds, focusing on detecting senior citizens and logging relevant data efficiently.

Sign Language Detection Model : Design and train a model to recognize and interpret sign language gestures. This involved creating a user-friendly graphical interface for both image uploads and live video processing, operating within specific time constraints.

BACKGROUND

Machine learning and computer vision are rapidly evolving fields with significant impact across diverse domains. Each task in this project leverages these advancements to address specific challenges:

PROJECT FOCUS

This project utilized OpenCV for handling image and video processing tasks and TensorFlow for training and deploying deep learning models. By leveraging these tools, the project addressed the following challenges:

- **VISUALIZING CNN ACTIVATION MAPS:**

  Understanding how different regions of an image influence CNN predictions, thereby providing insights into the model's decision-making process.

- **REAL-TIME SENIOR CITIZEN DETECTION SYSTEM:**

  Identifying and classifying individuals based on age and gender from live video feeds, with a focus on detecting senior citizens and efficiently logging their data.

- **SIGN LANGUAGE DETECTION MODEL:**

  Recognizing and interpreting sign language gestures through a model trained on relevant data, complemented by a user-friendly interface for both image uploads and real-time video processing.

This background sets the stage for the subsequent sections of the report by explaining the technological advancements and tools used in the project, providing context for the implemented solutions and their impact.

LEARNING OBJECTIVES

The internship project focused on the application of machine learning and computer vision techniques to address specific challenges. The following learning objectives guided the development and implementation of the project tasks:

1. VISUALIZE CNN ACTIVATION MAPS:

Understand how different regions of an image activate various layers in a convolutional neural network (CNN).

2. REAL-TIME SENIOR CITIZEN DETECTION SYSTEM:

Implement a system that identifies and classifies individuals based on age and gender from video feeds, focusing on detecting senior citizens.

3. SIGN LANGUAGE DETECTION MODEL:

Train a model to recognize and interpret sign language gestures and build a user-friendly interface for both image upload and live video processing.

ACTIVITIES AND TASKS

VISUALIZING CNN ACTIVATION MAPS:

- **Activity:**

  Implemented techniques to visualize activation maps from CNN layers to understand how different parts of an image influence the CNN's predictions.

- **Tasks:**

  Loaded Pre-Trained Models: Used TensorFlow to load a pre-trained CNN model that was created and fine-tuned for the specific application.

  Extracted Intermediate Layer Outputs: Programmatically accessed and extracted outputs from intermediate layers of the CNN to analyze how features are processed.

  Visualized Activation Maps: Developed methods to visualize these activation maps, providing insights into which image regions and features are most influential for the CNN’s decisions.

ACTIVITIES AND TASKS

REAL-TIME SENIOR CITIZEN DETECTION SYSTEM:

- **Activity:**

  Developed a robust system capable of detecting and classifying individuals based on age and gender from live video feeds, with a specific focus on identifying senior citizens.

- **Tasks:**

  Integrated OpenCV for Video Processing: Utilized OpenCV to handle video capture and processing tasks, enabling real-time analysis of video streams.

  Used a Pre-Trained Model for Age and Gender Prediction: Applied a custom pre-trained model (created by me) to predict age and gender from video frames. This model was trained on a relevant dataset to ensure accurate predictions.

  Implemented Logging Functionality: Developed a feature to log the age, gender, and time of visit for individuals identified as senior citizens, enabling efficient data collection and analysis.

ACTIVITIES AND TASKS

SIGN LANGUAGE DETECTION MODEL:

- **Activity:**

  Designed, trained, and implemented a model for recognizing sign language gestures, along with building a user-friendly graphical interface for interaction.

- **Tasks:**

  Collected and Pre-Processed Sign Language Data: Gathered a dataset of sign language gestures and performed preprocessing to prepare the data for model training.

  Trained a Model for Gesture Recognition: Developed and trained a custom model (created by me) to accurately recognize and classify sign language gestures from images and video.

  Developed a GUI with Tkinter: Created a graphical user interface using Tkinter to facilitate image uploads and real-time video processing, providing an accessible and interactive tool for users to engage with the sign language detection system.

SKILLS AND COMPETENCIES

MACHINE LEARNING

- **Proficiency in TensorFlow:** Gained expertise in using TensorFlow for developing and training machine learning models. This included:
    - **Model Training:** Experience in designing, training, and fine-tuning deep learning models for various tasks, such as image classification and gesture recognition.
    - **Inference and Deployment:** Skills in deploying trained models for inference, including optimizing models for performance and accuracy.
    - **Custom Model Development:** Successfully created and implemented custom pre-trained models for specific applications, including age and gender prediction and sign language gesture recognition..

SKILLS AND COMPETENCIES

COMPUTER VISION

- **Experience with OpenCV:** Developed proficiency in using OpenCV for a range of computer vision tasks, including:
    - **Video Capture and Processing:** Utilized OpenCV to handle real-time video feeds, including capturing, processing, and analyzing video frames.
    - **Feature Detection:** Applied OpenCV techniques for feature detection, object tracking, and image manipulation.
    - **Integration with Deep Learning Models:** Combined OpenCV's capabilities with TensorFlow models to create seamless workflows for real-time applications

SKILLS AND COMPETENCIES

SOFTWARE DEVELOPMENT

- **Python Programming:** Demonstrated strong programming skills in Python, which included:
    - **Script Development:** Writing and optimizing Python scripts for data processing, model training, and system integration.
    - **Error Handling and Debugging:** Proficient in debugging and optimizing code to enhance system performance and reliability.
- **GUI Development with Tkinter:** Developed user-friendly graphical interfaces using Tkinter, including:
    - **Image and Video Processing Interfaces:** Created intuitive interfaces for image uploads and real-time video processing, improving user interaction with the sign language detection system.

SKILLS AND COMPETENCIES

DATA HANDLING

- **Competency with Pandas:** Managed and analyzed data effectively using Pandas, including:
    - **Data Preparation:** Preprocessed and cleaned data to prepare it for model training and analysis.
    - **Data Analysis:** Conducted data analysis and visualization to gain insights and support decision-making, including logging and analyzing results from the senior citizen detection system.

EVIDENCE

Screenshots of Activation Maps

Description:

Visualizations showing how different layers of the CNN model activate in response to specific regions of input images. These images help illustrate which parts of the image are most influential for the model's predictions.

Original Input Image:

The original image used as input to the CNN model. Providing this image allows readers to see what the model is processing and understand the context for the activation maps.

Attachment:

image 1

image 2

EVIDENCE

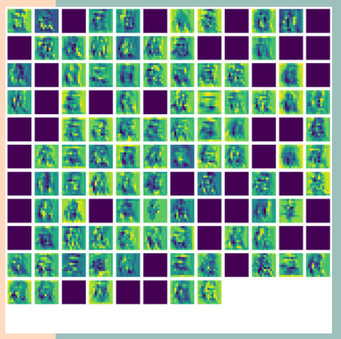
Screenshots of Activation Maps

Activation Maps

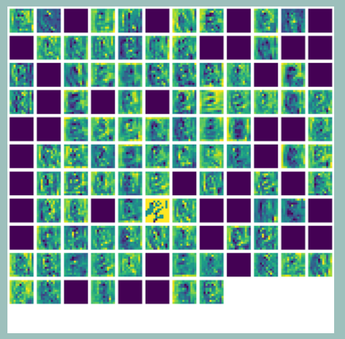
Visualizations of the activation maps from different layers of the CNN. These maps show which parts of the image were activated by the CNN filters.

Attachments:

activation map of image 1



activation map of image 2



By including the original image:

- Contextual Understanding: Readers can see which regions of the original image are being activated by the CNN filters, making it easier to understand the significance of the activation maps.
- Enhanced Analysis: It helps in analyzing how different layers of the model respond to various parts of the input image, which can be useful for debugging and improving model performance.

EVIDENCE

Senior Citizen Identification

Description:

This task involves developing a system to identify and classify individuals in a video feed based on their age and gender. The model predicts the age and gender of detected faces. The GUI determines if an individual is a senior citizen (age > 60) based on the model's age prediction and annotates the video feed with this information. Results are saved in a CSV file..

Code Snippets

Model Creation Code:

Code for creating and training the model for age and gender prediction. This example uses TensorFlow and Keras to build a simple CNN model.

EVIDENCE

Senior Citizen Identification

Code Snippets

```
import tensorflow as tf
import cv2
import numpy as np
from matplotlib import pyplot as plt
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dense,MaxPooling2D,Conv2D
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input,Activation,Add
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam,Adagrad,Adadelta,Adamax,RMSprop
```

```
def model(input_shape):
    inputs=Input(shape=input_shape)
    conv_1=Convolution(inputs,32)
    maxp_1=MaxPooling2D(pool_size=(2,2))(conv_1)
    conv_2=Convolution(maxp_1,64)
    maxp_2=MaxPooling2D(pool_size=(2,2))(conv_2)
    conv_3=Convolution(maxp_2,128)
    maxp_3=MaxPooling2D(pool_size=(2,2))(conv_3)
    conv_4=Convolution(maxp_3,256)
    maxp_4=MaxPooling2D(pool_size=(2,2))(conv_4)
    conv_5=Convolution(maxp_4,512)
    maxp_5=MaxPooling2D(pool_size=(2,2))(conv_5)
    flatten=Flatten()(maxp_5)
    dense_1=Dense(64,activation='relu')(flatten)
    dense_2=Dense(64,activation='relu')(flatten)
    drop_1=Dropout(0.2)(dense_1)
    drop_2=Dropout(0.2)(dense_2)
    output_1=Dense(1,activation='sigmoid',name='sex_out')(drop_1)
    output_2=Dense(1,activation='relu',name='age_out')(drop_2)
    model=Model(inputs=[inputs],outputs=[output_1,output_2])
    model.compile(loss=['binary_crossentropy',"mae"],optimizer="Adam",metrics=["accuracy","accuracy"])
    return model
```

```
History=Model.fit(X_train,y_train_2,batch_size=64,validation_data=(X_test,y_test_2),
                  epochs=250,callbacks=callback_list)
```

EVIDENCE

Senior Citizen Identification

GUI for Video Processing Code:

Code for the Tkinter GUI that handles video file upload, processes video frames, predicts age and gender using the model, determines senior citizen status, annotates the video feed, and saves results to a CSV file.

Code Snippets

```python
import tkinter as tk
from tkinter import filedialog
import cv2
import tensorflow as tf
import numpy as np
from datetime import datetime
import pandas as pd

# Load saved model
model = tf.keras.models.load_model('Age_Sex_Detection.keras')

# Initializing CSV file
csv_file = 'results.csv'
with open(csv_file, 'w') as file:
    file.write('Age,Gender,Senior Citizen,Time of Visit\n')

def preprocess_face(face):
    face = cv2.resize(face, (48, 48))
    face = face.astype('float32') / 255.0  # Normalize
    face = np.expand_dims(face, axis=0)
    return face

def predict_age_gender(face):
    face = preprocess_face(face)
    predictions = model.predict(face)
    print(predictions)
    age = int(np.round(predictions[1][0]))
    gender_prob = predictions[0][0][0]
    gender = 'Male' if gender_prob > 0.5 else 'Female'
    return age, gender
```

```python
def detect_faces(image):
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    return faces

def process_video(file_path):
    cap = cv2.VideoCapture(file_path)
    if not cap.isOpened():
        print("Error: Could not open video file.")
        return

    while True:
        ret, frame = cap.read()
        if not ret:
            print("End of video or error reading frame.")
            break

        faces = detect_faces(frame)
        print(f"Detected faces: {faces}")

        for (x, y, w, h) in faces:
            face = frame[y:y+h, x:x+w]
            age, gender = predict_age_gender(face)
            senior_citizen = age > 60
            senior_status = 'Yes' if senior_citizen else 'No'

            # Debug: Print data before writing
            print(f"Writing to CSV: Age={age}, Gender={gender}, Senior Citizen={senior_status}")
```

EVIDENCE

Senior Citizen Identification

GUI for Video Processing Code:

Code Snippets

```python
def detect_faces(image):
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    return faces

def process_video(file_path):
    cap = cv2.VideoCapture(file_path)
    if not cap.isOpened():
        print("Error: Could not open video file.")
        return

    while True:
        ret, frame = cap.read()
        if not ret:
            print("End of video or error reading frame.")
            break

        faces = detect_faces(frame)
        print(f"Detected faces: {faces}")

        for (x, y, w, h) in faces:
            face = frame[y:y+h, x:x+w]
            age, gender = predict_age_gender(face)
            senior_citizen = age > 60
            senior_status = 'Yes' if senior_citizen else 'No'

            # Debug: Print data before writing
            print(f"Writing to CSV: Age={age}, Gender={gender}, Senior Citizen={senior_status}")
```

```python
            # Save results to CSV
            time_of_visit = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            with open(csv_file, 'a') as file:
                file.write(f"{age},{gender},{senior_status},{time_of_visit}\n")

                # Draw rectangle and text on frame
                cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
                cv2.putText(frame, f'Age: {age}, Gender: {gender}, Senior Citizen: {senior_status}',
                            (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

        cv2.imshow('Video Feed', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

def upload_video():
    file_path = filedialog.askopenfilename(title="Select Video File", filetypes=[("Video Files", "*.mp4;*.avi")])
    print(f"Selected file: {file_path}")  # Debug: Print selected file path
    if file_path:
        process_video(file_path)
# GUI Setup
root = tk.Tk()
root.title("Video Age and Gender Detection")
upload_button = tk.Button(root, text="Upload Video", command=upload_video)
upload_button.pack(padx=20, pady=20)

# Run the GUI loop
root.mainloop()
```

EVIDENCE

Sign Language Detection

Description:

In this task, we aim to develop a machine learning model for recognizing sign language gestures. The system needs to be operational in real-time and should be able to recognize specific sign language signs either from images or video feeds. A graphical user interface (GUI) facilitates the upload of images and the display of results, enhancing user interaction.

Code Snippets

Model Creation Code:

The sign language detection model aims to classify images of sign language gestures into predefined categories. It uses a Convolutional Neural Network (CNN) to extract features from images and predict the corresponding sign language gestures. The model is trained on a dataset of sign language images and can be used for both real-time prediction and static image classification.

EVIDENCE

Sign Language Detection

GUI Code snippet:

Main window

```
# The main window
root = tk.Tk()
root.title("Sign Language Prediction")

# Image upload button
upload_image_button = tk.Button(root, text="Upload Image", command=load_image)
upload_image_button.pack(pady=10)

# Video upload button
upload_video_button = tk.Button(root, text="Upload Video", command=process_video)
upload_video_button.pack(pady=10)

# The GUI event loop
root.mainloop()
```

EVIDENCE

Sign Language Detection

GUI Codesnippet:

```python
def load_image():
    """Load and predict the uploaded image if within operational hours."""
    if is_within_operational_hours():
        file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png")])
        if file_path:
            try:
                image = cv2.imread(file_path)
                if image is None:
                    messagebox.showerror("Error", "Unable to load image.")
                    return

                # Convert BGR to RGB
                image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

                # Detect hand landmarks
                hand_landmarks = detect_hand_landmarks(image)

                if hand_landmarks:
                    # Draw landmarks on the image
                    image_with_landmarks = draw_hand_landmarks(image, hand_landmarks)

                    # Get bounding box around the palm
                    palm_coordinates = get_palm_coordinates(hand_landmarks, image)
                    if palm_coordinates:
                        x, y, w, h = palm_coordinates
                        palm_area = image[y:y+h, x:x+w]

                        # Check if the palm_area is valid before resizing
                        if palm_area.size > 0:
                            palm_area = cv2.resize(palm_area, (28, 28))
                            palm_area = cv2.cvtColor(palm_area, cv2.COLOR_BGR2GRAY)
                            palm_area = palm_area.reshape(1, 28, 28, 1) / 255.0

                            # Predict the letter
                            prediction = model.predict(palm_area)
                            predicted_class = np.argmax(prediction, axis=1)
                            letter = class_labels[predicted_class[0]]

                            # Create a new top-level window to display the image and result
                            top = tk.Toplevel()
                            top.title("Image with Prediction")

                            # Display the image with landmarks
                            img_display = cv2.cvtColor(image_with_landmarks, cv2.COLOR_BGR2RGB)
                            img_display = Image.fromarray(img_display).resize((400, 400))  # Resize for display
                            img_tk = ImageTk.PhotoImage(img_display)

                            # Display the image
                            label_img = tk.Label(top, image=img_tk)
                            label_img.image = img_tk  # Keep a reference to avoid garbage collection
                            label_img.pack(pady=10)

                            # Display the prediction result
                            tk.Label(top, text=f"Predicted Letter: {letter}", font=('Helvetica', 16)).pack(pady=10)
                else:
                    messagebox.showwarning("No Hands Detected", "No hand landmarks detected in the image.")

            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
    else:
        messagebox.showwarning("Time Restriction", "The model is not operational outside of 6 PM to 10 PM.")
```

EVIDENCE

Sign Language Detection

GUI Code snippets

```python
def process_video():
    """Load and process video, detecting hands, predicting letters, and drawing landmarks."""
    if is_within_operational_hours():
        file_path = filedialog.askopenfilename(filetypes=[("Video Files", "*.mp4;*.avi")])
        if file_path:
            try:
                cap = cv2.VideoCapture(file_path)
                while cap.isOpened():
                    ret, frame = cap.read()
                    if not ret:
                        break

                    # Check if the frame is valid
                    if frame is None:
                        continue

                    # Detect hand landmarks
                    hand_landmarks = detect_hand_landmarks(frame)

                    if hand_landmarks:
                        # Draw landmarks on the frame
                        frame = draw_hand_landmarks(frame, hand_landmarks)

                        # Get bounding box around the palm
                        palm_coordinates = get_palm_coordinates(hand_landmarks, frame)
                        if palm_coordinates:
                            x, y, w, h = palm_coordinates
                            palm_area = frame[y:y+h, x:x+w]
                            # Check if the palm_area is valid before resizing
                            if palm_area.size == 0:
                                continue

                            palm_area = cv2.resize(palm_area, (28, 28))
                            palm_area = cv2.cvtColor(palm_area, cv2.COLOR_BGR2GRAY)
                            palm_area = palm_area.reshape(1, 28, 28, 1) / 255.0

                            # Predict the letter
                            prediction = model.predict(palm_area)
                            predicted_class = np.argmax(prediction, axis=1)
                            letter = class_labels[predicted_class[0]]

                            # Display the prediction on the frame
                            cv2.putText(frame, f'Predicted Letter: {letter}', (10, 30),
                                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)
                    # Add exit instruction
                    cv2.putText(frame, 'Press "q" to exit', (10, frame.shape[0] - 10),
                                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)
                    # Show the video frame with detection and prediction
                    cv2.imshow('Video', frame)
                    # Exit if 'q' is pressed
                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break

                cap.release()
                cv2.destroyAllWindows()
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
    else:
        messagebox.showwarning("Time Restriction", "The model is not operational outside of 6 PM to 10 PM.")
```

CHALLENGES AND SOLUTIONS

Challenge 1: Integrating Real-Time Video Processing with Machine Learning Model for Accurate Age and Gender Detection

- **Description:** The primary challenge was integrating a machine learning model with real-time video feeds to accurately detect age and gender. This task involved not only ensuring the model's accuracy but also managing the complexities of video processing and frame extraction.

- **Solution:**

  - **Efficient Video Frame Handling:** Implemented a robust video frame extraction pipeline using OpenCV to ensure smooth and continuous capture of video frames. This involved optimizing the frame extraction process to avoid bottlenecks that could affect real-time performance.
  - **Asynchronous Processing:** Adopted asynchronous processing methods to separate video capture from model inference. This approach allowed the video feed to run continuously while the model processed frames in parallel, thus minimizing delays and ensuring timely detection.

CHALLENGES AND SOLUTIONS

Challenge 2: Developing a GUI for Sign Language Detection with Mediapipe Hand Landmark Integration

- **Description:** Creating a GUI for sign language detection using Mediapipe's hand landmark model presented challenges related to integrating real-time hand tracking with the graphical interface. The GUI needed to support live video input and display recognized signs effectively.

- **Solution:**

  - **Integration with Mediapipe:** Integrated Mediapipe’s hand landmark detection into the GUI to enable real-time hand tracking. This involved using the Mediapipe library to process video frames and extract hand landmarks, which were then used for sign language recognition.
  - **GUI Development:** Used Tkinter to develop a responsive and intuitive GUI. The interface included features for live video display and feedback on recognized signs. Ensured the GUI was capable of handling real-time updates from Mediapipe without lag.

OUTCOMES AND IMPACT

Activation Map Visualization

- **Enhanced Understanding:** The visualization of CNN activation maps has significantly improved the understanding of how different image regions influence the CNN filters. By observing these activation maps, we gained insights into which parts of an image are most important for the model's predictions. This understanding helps in refining model architectures and improving the interpretability of complex models.
- **Model Behavior Insight:** The ability to visualize activation maps aids in diagnosing issues with model performance, such as identifying why certain features are not being detected. This can lead to better model adjustments and more informed decisions during the model development phase.

OUTCOMES AND IMPACT

Senior Citizen Identification

- **Robust System Development:** The implementation of a real-time system for age and gender detection proved successful, demonstrating the capability to handle live video feeds and perform accurate predictions. This system is particularly valuable for applications in surveillance and customer analytics, where monitoring and analyzing demographic data in real-time can enhance service delivery and security measures.
- **Real-World Applications:** The system's ability to identify senior citizens can be utilized in various contexts, such as targeted marketing, customer service, and safety measures in public spaces. The inclusion of a data logging feature for recording age, gender, and visit time contributes to valuable data collection and analysis, supporting decision-making processes in commercial and security applications.

OUTCOMES AND IMPACT

Sign Language Detection

- **Functional System Creation:** The development of a sign language recognition system, complete with a graphical user interface (GUI), provides a practical tool for real-time sign language interpretation. This system facilitates better communication for users who rely on sign language, thus improving accessibility and interaction in diverse settings.
- **Improved Accessibility:** By integrating a user-friendly GUI for both image uploads and live video, the system becomes more accessible to non-expert users. This contributes to broader adoption and usability in educational settings, communication aids, and public services, enhancing inclusivity for individuals who use sign language as their primary mode of communication.

Overall, these outcomes demonstrate a significant impact on both the technical understanding of machine learning models and their practical applications in real-world scenarios. The ability to visualize model behavior, develop real-time identification systems, and create accessible communication tools highlights the practical and societal benefits of advanced machine learning technologies.

CONCLUSIONE

The internship offered invaluable experience in applying machine learning to practical computer vision problems. By successfully working on activation map visualizations, real-time age and gender detection, and sign language recognition, I developed a solid skill set in advanced programming, machine learning, and GUI development.

This experience greatly enhanced my understanding of computer vision technologies and their real-world applications. The knowledge and skills acquired will be instrumental in my future projects and professional development in the field.