Import Libraries

```
import os #"os" module, which helps us work with files and folders on our computer.
import cv2 #use OpenCV, a powerful tool for working with images and videos.
import math # "math" module, which gives us access to mathematical functions like square roots and trigonometry.
import numpy as np #NumPy, which is a library for working with arrays and mathematical operations.
import mediapipe as mp #use the MediaPipe library, which is great for tasks like tracking hands or estimating poses in images or videos.
import tensorflow as tf #TensorFlow, a popular library for machine learning and deep learning. It helps us build and train neural networks for various tasks.
```

TensorFlow and Keras Setup

```
from tensorflow.keras.models import load_model #This function is used to load pre-trained neural network models saved in the Hierarchical Data Format (HDF5) file format.
from tensorflow.keras import layers, models, datasets #Contains classes for building different layers of a neural network, like dense layers, convolutional layers, etc.
from sklearn.model_selection import train_test_split #This function is part of the Scikit-learn library and is used to split datasets into training and testing subsets for model evaluation and validation.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

Sign Language Phrase Video Writer

```
file_paths = { # These are the keys representing different sign language phrases.
    "Hello": "hello_sign_language.avi",
    "Thank you": "thank_you_sign_language.avi",
    "I love you": "i_love_you_sign_language.avi" #These are the values, representing the file paths of videos for the respective sign language phrases.
} #This dictionary stores the names of sign language phrases as keys and the corresponding file paths of videos as values.
fourcc = cv2.VideoWriter_fourcc(*'XVID') #This line creates a codec used for writing videos with the XVID codec. The cv2.VideoWriter_fourcc function converts the four-character code 'XVID' into an integer that OpenCV can understand as a codec.
video_writers = {
    "Hello": cv2.VideoWriter(file_paths["Hello"], fourcc, 20.0, (640, 480)),
```

```
    "Thank you": cv2.VideoWriter(file_paths["Thank you"], fourcc,
20.0, (640, 480)),
    "I love you": cv2.VideoWriter(file_paths["I love you"], fourcc,
20.0, (640, 480)) #It specifies the file path, codec, frame rate (20
frames per second), and frame size (640x480 pixels). Similar
initialization is done for the other phrases as well.
}#This dictionary stores instances of cv2.VideoWriter objects for each
sign language phrase. Each video writer is initialized with the file
path, codec, frame rate, and frame size.
```

Hand Detection and Tracking with MediaPipe

```
mp_hands = mp.solutions.hands # This line imports the "hands" module
from the MediaPipe library and assigns it to the variable mp_hands.
MediaPipe provides a pre-trained model for hand detection and
tracking.
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=2,
min_detection_confidence=0.5, min_tracking_confidence=0.5) #This line
initializes a hand detection and tracking object using the Hands class
from the MediaPipe module.
imgSize = 300 #This line assigns the value 300 to the variable
imgSize. This variable likely represents the desired size (in pixels)
for the input image to be fed into the hand detection and tracking
model.
offset = 20 #This line assigns the value 20 to the variable offset.
This variable may represent an offset value used for some calculations
or adjustments in the subsequent code. Its exact purpose depends on
the context of the code that follows.
```

Real-time Sign Language Capture and Recognition

```
def capture_video():
    # Open a connection to the default camera
    cap = cv2.VideoCapture(0)

    # Check if the camera opened successfully
    if not cap.isOpened():
        print("Error: Could not open camera")
        return

    print("Press 'q' to quit the video capture.")
    print("Press 'h' to save a clip as 'Hello'.")
    print("Press 't' to save a clip as 'Thank you'.")
    print("Press 'l' to save a clip as 'I love you'.")

    # Loop to continuously capture frames from the camera
    while True:
        # Capture frame-by-frame
        success, frame = cap.read()
```

```
        # Check if frame was captured successfully
        if not success:
            print("Error: Could not read frame")
            break

        # Convert the frame from BGR to RGB for MediaPipe processing
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Process the frame with MediaPipe hand tracking
        results = hands.process(frame_rgb)

        # Initialize the hand detection result as None
        hand = None

        # Draw hand landmarks and connections on the frame
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                mp.solutions.drawing_utils.draw_landmarks(
                    frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)
                # Retrieve bounding box and hand data
                # Convert normalized landmarks to pixel coordinates
                h, w, _ = frame.shape
                x_min = min(landmark.x for landmark in
hand_landmarks.landmark) * w
                y_min = min(landmark.y for landmark in
hand_landmarks.landmark) * h
                x_max = max(landmark.x for landmark in
hand_landmarks.landmark) * w
                y_max = max(landmark.y for landmark in
hand_landmarks.landmark) * h
                x, y, w, h = int(x_min), int(y_min), int(x_max -
x_min), int(y_max - y_min)

                hand = {'bbox': (x, y, w, h)}

        # If a hand is detected, process the frame
        if hand:
            # Get the bounding box coordinates
            x, y, w, h = hand['bbox']

            # Skip processing if width or height is zero
            if w == 0 or h == 0:
                continue

            # Create a white background image for padding
            imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255

            # Make sure the bounding box coordinates do not exceed
frame dimensions
```

```
            x_start = max(0, x - offset)
            y_start = max(0, y - offset)
            x_end = min(frame.shape[1], x + w + offset)
            y_end = min(frame.shape[0], y + h + offset)

            # Crop the frame to include only the hand
            imgCrop = frame[y_start: y_end, x_start: x_end]

            # Check if the cropped image is empty
            if imgCrop.shape[0] == 0 or imgCrop.shape[1] == 0:
                continue

            # Calculate the aspect ratio of the cropped hand image
            aspectRatio = h / w

            # Resize the cropped hand image based on the aspect ratio
            if aspectRatio > 1:
                k = imgSize / h
                wCal = math.ceil(k * w)
                imgResize = cv2.resize(imgCrop, (wCal, imgSize))
                wGap = math.ceil((imgSize - wCal) / 2)
                imgWhite[:, wGap: wCal + wGap] = imgResize
            else:
                k = imgSize / w
                hCal = math.ceil(k * h)
                imgResize = cv2.resize(imgCrop, (imgSize, hCal))
                hGap = math.ceil((imgSize - hCal) / 2)
                imgWhite[hGap: hCal + hGap, :] = imgResize

            # Display the cropped and white background images
            cv2.imshow('ImageCrop', imgCrop)
            cv2.imshow('ImageWhite', imgWhite)

            # Save the frames as video clips based on user input
            if key == ord('h'):
                video_writers["Hello"].write(frame)
                print("Saved 'Hello' clip.")
            elif key == ord('t'):
                video_writers["Thank you"].write(frame)
                print("Saved 'Thank you' clip.")
            elif key == ord('l'):
                video_writers["I love you"].write(frame)
                print("Saved 'I love you' clip.")

        # Display the frame with landmarks
        cv2.imshow("Sign Language Capture", frame)

        # Check for user input
        key = cv2.waitKey(1) & 0xFF
```

```
        # Exit the loop if 'q' is pressed
        if key == ord('q'):
            break

    # Release the camera and close the window
    cap.release()
    cv2.destroyAllWindows()

    # Release all video writers
    for writer in video_writers.values():
        writer.release()

    # Close the MediaPipe hand tracking instance
    hands.close()

# Call the function to capture video
capture_video()
```

```
Press 'q' to quit the video capture.
Press 'h' to save a clip as 'Hello'.
Press 't' to save a clip as 'Thank you'.
Press 'l' to save a clip as 'I love you'.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
```

```
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
```

```
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
```

```
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
```

```
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Hello' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
```

```
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
```

```
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
```

```
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'Thank you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
```

```
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
```

```
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
Saved 'I love you' clip.
```

Gesture Recognition Dataset Creation from Video Files

```python
# Function to extract frames from video files and create dataset
def load_video_data(file_paths, label_map):
    data = []
    labels = []

    for label, file_path in file_paths.items():
        cap = cv2.VideoCapture(file_path)
        frames = []

        while True:
            ret, frame = cap.read()
            if not ret:
                break
            # Resize frame to 64x64 and convert to RGB
            frame = cv2.resize(frame, (64, 64))
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frames.append(frame)

        cap.release()
        # Convert list of frames to numpy array and append to data
        data.extend(frames)
        labels.extend([label_map[label]] * len(frames))

    data = np.array(data)
    labels = np.array(labels)
    return data, labels
```

```
# Define the file paths for each gesture
file_paths = {
    "Hello": "hello_sign_language.avi",
    "Thank you": "thank_you_sign_language.avi",
    "I love you": "i_love_you_sign_language.avi"
}

# Define label map for gestures
label_map = {"Hello": 0, "Thank you": 1, "I love you": 2}

# Load data
data, labels = load_video_data(file_paths, label_map)
```

Gesture Recognition Convolutional Neural Network Model Training

```
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(data, labels,
test_size=0.2, random_state=42)
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64,
3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(3, activation='softmax')  # 3 output classes
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=2, validation_data=(X_test,
y_test))
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print(f"Test accuracy: {test_acc:.4f}")

Epoch 1/2
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.2937 - loss:
61.1086 - val_accuracy: 0.9659 - val_loss: 0.3389
Epoch 2/2
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.9945 - loss:
0.1636 - val_accuracy: 1.0000 - val_loss: 5.3920e-05
Test accuracy: 1.0000
```

Sign Language Gesture Classifier Model Saved

```
model.save("sign_language_gesture_classifier.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

Long Short-Term Memory (LSTM) Model Training and Evaluation for Gesture Recognition

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.model_selection import train_test_split

# Define the shape of your input data
timesteps = X_train.shape[1]  # Number of time steps in input
sequences
features = X_train.shape[2]   # Number of features in each time step
num_classes = len(np.unique(y_train))  # Number of unique classes in
labels

# Define the LSTM model
model = Sequential([
    LSTM(units=64, input_shape=(timesteps, features)),  # LSTM layer
with 64 units
    Dense(units=num_classes, activation='softmax')  # Output layer
with softmax activation
])

# Compile the model
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Print model summary
model.summary()
```

```
Model: "sequential_7"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━
┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━
┩
│ lstm_3 (LSTM)                        │ (None, 64)                  │
33,024 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
│ dense_11 (Dense)                     │ (None, 3)                   │
```

```
195 │
└──────────────────────────────────────┴──────────────────────────────┴──────────
──────┘

 Total params: 33,219 (129.76 KB)

 Trainable params: 33,219 (129.76 KB)

 Non-trainable params: 0 (0.00 B)
```

"Real-time Sign Language Gesture Recognition with Pre-trained Model

```
# Load the pre-trained model
model = load_model("sign_language_gesture_classifier.h5")

# Define label map for gestures
label_map = {0: "Hello", 1: "Thank you", 2: "I love you"}

# Function to preprocess the frame
def preprocess_frame(frame):
    # Resize frame to 64x64 and convert to RGB
    frame = cv2.resize(frame, (64, 64))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # Normalize pixel values
    frame = frame / 255.0
    # Add an extra dimension to represent batch size (1, height,
width, channels)
    frame = np.expand_dims(frame, axis=0)
    return frame

# Open a connection to the default camera (usually the first camera on
your system)
cap = cv2.VideoCapture(0)

# Check if the camera opened successfully
if not cap.isOpened():
    print("Error: Could not open camera")
else:
    print("Press 'q' to quit real-time detection.")

# Loop to continuously capture frames from the camera
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Check if frame was captured successfully
    if not ret:
        print("Error: Could not read frame")
        break
```

```
    # Pre-process the frame
    preprocessed_frame = preprocess_frame(frame)

    # Predict gesture using the model
    predictions = model.predict(preprocessed_frame)

    # Get the predicted label index
    predicted_label_index = np.argmax(predictions)

    # Get the predicted label
    predicted_label = label_map[predicted_label_index]

    # Display the predicted label on the frame
    cv2.putText(frame, f"Gesture: {predicted_label}", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

    # Display the captured frame in a window
    cv2.imshow("Sign Language Detection", frame)

    # Check for user input
    key = cv2.waitKey(1) & 0xFF

    if key == ord('q'):
        # Exit the loop if 'q' is pressed
        break

# Release the camera and close the window
cap.release()
cv2.destroyAllWindows()
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

```
Press 'q' to quit real-time detection.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
```