```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,
MaxPooling2D, Flatten, Dense, Dropout
from sklearn.model_selection import train_test_split
import cv2


def load_dataset(data_dir, img_size=(224, 224)):
    images = []
    labels = []
    class_names = os.listdir(data_dir)

    for label, class_name in enumerate(class_names):
        class_dir = os.path.join(data_dir, class_name)
        for img_name in os.listdir(class_dir):
            img_path = os.path.join(class_dir, img_name
)
            img = cv2.imread(img_path)
            if img is not None:
                img = cv2.resize(img, img_size)  #
Resize to match the input size
                images.append(img)
                labels.append(label)

    return np.array(images), np.array(labels),
class_names


# Load dataset
data_dir = "data"  # Update this with the path to your
dataset
X, y, class_names = load_dataset(data_dir)

# Normalize images
X = X / 255.0

# Split dataset
X_train, X_val, y_train, y_val = train_test_split(X, y
, test_size=0.2, random_state=42)

# Convert labels to categorical
y_train = tf.keras.utils.to_categorical(y_train,
num_classes=len(class_names))
y_val = tf.keras.utils.to_categorical(y_val,
num_classes=len(class_names))


model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(
224, 224, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(class_names), activation='softmax')
])

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

model.summary()


history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=10,  # Adjust based on your requirements and
 dataset size
    batch_size=32
)

# Save the model for later use
model.save("cricket_shot_model.h5")
```

```python
# Load the trained model
model = tf.keras.models.load_model("cricket_shot_model.
h5")

# Initialize the camera
cap = cv2.VideoCapture(0)  # 0 is typically the default
 camera

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Preprocess the frame for the model
    resized_frame = cv2.resize(frame, (224, 224))
    normalized_frame = resized_frame / 255.0
    input_data = np.expand_dims(normalized_frame, axis=
0)

    # Make a prediction
    predictions = model.predict(input_data)
    predicted_class = class_names[np.argmax(predictions
)]

    # Display the predicted class on the frame
    cv2.putText(frame, f'Predicted Shot: {
predicted_class}', (10, 50),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0
), 2)
    cv2.imshow('Cricket Shot Recognition', frame)

    # Press 'q' to exit the live feed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```