task4

January 31, 2025

[1]:
```
import kaggle
import zipfile
import os

# Set dataset path
dataset_name = "gti-upm/leapgestrecog"
output_dir = "leapgestrecog"

# Download dataset
kaggle.api.dataset_download_files(dataset_name, path=output_dir, unzip=True)

# Verify downloaded files
print("Dataset downloaded successfully!")
print("Contents:", os.listdir(output_dir))
```

```
Dataset URL: https://www.kaggle.com/datasets/gti-upm/leapgestrecog
Dataset downloaded successfully!
Contents: ['leapgestrecog', 'leapGestRecog']
```

[1]:
```
import os
import cv2
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 tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define dataset path
dataset_path = "leapgestrecog/leapGestRecog"

# Image parameters
IMG_SIZE = 64  # Resize images
BATCH_SIZE = 32

# Data Augmentation and Loading
datagen = ImageDataGenerator(
    rescale=1./255, validation_split=0.2
```

```
)

train_data = datagen.flow_from_directory(
    dataset_path, target_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE,␣
 ↪class_mode='categorical', subset='training'
)

val_data = datagen.flow_from_directory(
    dataset_path, target_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE,␣
 ↪class_mode='categorical', subset='validation'
)

# Build CNN Model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(train_data.num_classes, activation='softmax')
])

# Compile Model
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Train Model
model.fit(train_data, validation_data=val_data, epochs=10)

# Save Model
model.save("gesture_recognition_model.h5")

print("Model training complete and saved.")
```

```
2025-01-31 14:23:58.762464: I tensorflow/core/util/port.cc:153] oneDNN custom
operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn them
off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-01-31 14:23:58.784207: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
E0000 00:00:1738333438.797828  237115 cuda_dnn.cc:8310] Unable to register cuDNN
```

```
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1738333438.802167  237115 cuda_blas.cc:1418] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
2025-01-31 14:23:58.827246: I tensorflow/core/platform/cpu_feature_guard.cc:210]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI AVX512_BF16
AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate
compiler flags.

Found 16000 images belonging to 10 classes.
Found 4000 images belonging to 10 classes.

/home/narasima/anaconda3/envs/tensor_hc/lib/python3.12/site-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
I0000 00:00:1738333446.340761  237115 gpu_device.cc:2022] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 5563 MB memory:  -> device: 0,
name: NVIDIA GeForce RTX 4060 Laptop GPU, pci bus id: 0000:64:00.0, compute
capability: 8.9

Epoch 1/10

/home/narasima/anaconda3/envs/tensor_hc/lib/python3.12/site-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1738333448.229765  237317 service.cc:148] XLA service 0x7fefd400a930
initialized for platform CUDA (this does not guarantee that XLA will be used).
Devices:
I0000 00:00:1738333448.229813  237317 service.cc:156]   StreamExecutor device
(0): NVIDIA GeForce RTX 4060 Laptop GPU, Compute Capability 8.9
2025-01-31 14:24:08.253105: I
tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR
crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1738333448.348158  237317 cuda_dnn.cc:529] Loaded cuDNN version
90300

  5/500             17s 35ms/step - accuracy:
0.1029 - loss: 2.2992
```

```
I0000 00:00:1738333450.512860  237317 device_compiler.h:188] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.

470/500            1s 42ms/step -
accuracy: 0.6813 - loss: 0.9191

2025-01-31 14:24:29.120110: I
external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_78', 4 bytes spill stores, 4 bytes spill loads


500/500            27s 48ms/step -
accuracy: 0.6924 - loss: 0.8873 - val_accuracy: 0.6917 - val_loss: 2.2446
Epoch 2/10
500/500            23s 46ms/step -
accuracy: 0.9738 - loss: 0.0656 - val_accuracy: 0.6750 - val_loss: 1.9097
Epoch 3/10
500/500            20s 39ms/step -
accuracy: 0.9808 - loss: 0.0478 - val_accuracy: 0.7610 - val_loss: 2.2451
Epoch 4/10
500/500            20s 39ms/step -
accuracy: 0.9797 - loss: 0.0377 - val_accuracy: 0.7067 - val_loss: 1.9851
Epoch 5/10
500/500            20s 40ms/step -
accuracy: 0.9839 - loss: 0.0348 - val_accuracy: 0.7815 - val_loss: 1.8372
Epoch 6/10
500/500            23s 46ms/step -
accuracy: 0.9840 - loss: 0.0284 - val_accuracy: 0.7750 - val_loss: 1.6039
Epoch 7/10
500/500            21s 41ms/step -
accuracy: 0.9845 - loss: 0.0324 - val_accuracy: 0.7558 - val_loss: 1.8425
Epoch 8/10
500/500            20s 40ms/step -
accuracy: 0.9853 - loss: 0.0257 - val_accuracy: 0.7303 - val_loss: 1.8607
Epoch 9/10
500/500            23s 46ms/step -
accuracy: 0.9861 - loss: 0.0307 - val_accuracy: 0.7750 - val_loss: 1.5128
Epoch 10/10
500/500            20s 39ms/step -
accuracy: 0.9876 - loss: 0.0227 - val_accuracy: 0.7730 - val_loss: 1.7498

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')`.

Model training complete and saved.
```

[1]:
```
import cv2
import numpy as np
from tensorflow.keras.models import load_model

# Load trained model
model = load_model("gesture_recognition_model.h5")

# Class labels (update with actual labels from your dataset)
class_labels = ["Fist", "Palm", "Thumbs Up", "Thumbs Down", "Other"]

# Open webcam
cap = cv2.VideoCapture(0)

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

    # Preprocess frame
    img = cv2.resize(frame, (64, 64))
    img = img.astype("float32") / 255.0
    img = np.expand_dims(img, axis=0)

    # Predict gesture
    prediction = model.predict(img)
    gesture = class_labels[np.argmax(prediction)]

    # Display result
    cv2.putText(frame, f"Gesture: {gesture}", (10, 50), cv2.
 ↪FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    cv2.imshow("Hand Gesture Recognition", frame)

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

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

```
2025-01-31 15:18:06.947226: I tensorflow/core/util/port.cc:153] oneDNN custom
operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn them
off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-01-31 15:18:07.213575: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
```

```
E0000 00:00:1738336687.313526    3204 cuda_dnn.cc:8310] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1738336687.340653    3204 cuda_blas.cc:1418] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
2025-01-31 15:18:07.582830: I tensorflow/core/platform/cpu_feature_guard.cc:210]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI AVX512_BF16
AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate
compiler flags.
I0000 00:00:1738336693.241563    3204 gpu_device.cc:2022] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 5563 MB memory:  -> device: 0,
name: NVIDIA GeForce RTX 4060 Laptop GPU, pci bus id: 0000:64:00.0, compute
capability: 8.9
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.
[ WARN:0@7.881] global cap_v4l.cpp:999 open VIDEOIO(V4L2:/dev/video0): can't
open camera by index
[ERROR:0@7.882] global obsensor_uvc_stream_channel.cpp:158 getStreamChannelGroup
Camera index out of range
```

[ ]: