gesture_recognizer_train

April 2, 2024

1 Hand gesture recognition model training approach

Edited from: Hand gesture recognition model customization guide

Original page in Colab

Original page on GitHub

This notebook shows the end-to-end process of customizing a gesture recognizer model for recognizing some common hand gestures in the HaGRID dataset.

1.1 Prerequisites

Install the MediaPipe Model Maker package.

```
[ ]: !pip install --upgrade pip
     !pip install mediapipe-model-maker
```

Import the required libraries.

```
[ ]: from google.colab import files
     import os
     import tensorflow as tf
     assert tf.__version__.startswith('2')

     from mediapipe_model_maker import gesture_recognizer

     import matplotlib.pyplot as plt
     %matplotlib inline
```

1.2 Preparing Dataset

The dataset for gesture recognition in model maker requires the following format: `<dataset_path>/<label_name>/<img_name>.*`. In addition, one of the label names (`label_names`) must be `none`. The `none` label represents any gesture that isn't classified as one of the other gestures.

This example uses a partial dataset sample.

```
[ ]: from google.colab import drive
     drive.mount('/content/drive')
```

[ ]:
```
# !wget https://github.com/Shadowfax221/553.806_Capstone_HandGesture/blob/main/
 ↪minor_dataset.zip
!unzip /content/drive/MyDrive/Colab/dataset_boxed2.zip
```

Verify the dataset by printing the labels. There should be 15 gesture labels, with one of them being the `none` gesture.

[ ]:
```
dataset_path = "dataset_boxed2"
print(dataset_path)
labels = []
for i in os.listdir(dataset_path):
  if os.path.isdir(os.path.join(dataset_path, i)):
    labels.append(i)
print(labels)
```

To better understand the dataset, plot a couple of example images for each gesture.

[ ]:
```
NUM_EXAMPLES = 5

for label in labels:
  label_dir = os.path.join(dataset_path, label)
  example_filenames = os.listdir(label_dir)[:NUM_EXAMPLES]
  fig, axs = plt.subplots(1, NUM_EXAMPLES, figsize=(10,2))
  for i in range(NUM_EXAMPLES):
    axs[i].imshow(plt.imread(os.path.join(label_dir, example_filenames[i])))
    axs[i].get_xaxis().set_visible(False)
    axs[i].get_yaxis().set_visible(False)
  fig.suptitle(f'Showing {NUM_EXAMPLES} examples for {label}')

plt.show()
```

1.3 Modeling

The workflow consists of 4 steps which have been separated into their own code blocks.

Load the dataset

Load the dataset located at `dataset_path` by using the `Dataset.from_folder` method. When loading the dataset, run the pre-packaged hand detection model from MediaPipe Hands to detect the hand landmarks from the images. Any images without detected hands are ommitted from the dataset. The resulting dataset will contain the extracted hand landmark positions from each image, rather than images themselves.

The `HandDataPreprocessingParams` class contains two configurable options for the data loading process: * `shuffle`: A boolean controlling whether to shuffle the dataset. Defaults to true. * `min_detection_confidence`: A float between 0 and 1 controlling the confidence threshold for hand detection.

Split the dataset: 80% for training, 10% for validation, and 10% for testing.

[ ]:
```
data = gesture_recognizer.Dataset.from_folder(
    dirname=dataset_path,
    hparams=gesture_recognizer.HandDataPreprocessingParams()
)
train_data, rest_data = data.split(0.8)
validation_data, test_data = rest_data.split(0.5)
```

Train the model

Train the custom gesture recognizer by using the create method and passing in the training data, validation data, model options, and hyperparameters. For more information on model options and hyperparameters, see the Hyperparameters section below.

[ ]:
```
hparams = gesture_recognizer.HParams(export_dir="exported_model")
options = gesture_recognizer.GestureRecognizerOptions(hparams=hparams)
model = gesture_recognizer.GestureRecognizer.create(
    train_data=train_data,
    validation_data=validation_data,
    options=options
)
```

Evaluate the model performance

After training the model, evaluate it on a test dataset and print the loss and accuracy metrics.

[ ]:
```
loss, acc = model.evaluate(test_data, batch_size=1)
print(f"Test loss:{loss}, Test accuracy:{acc}")
```

Export to Tensorflow Lite Model

After creating the model, convert and export it to a Tensorflow Lite model format for later use on an on-device application. The export also includes model metadata, which includes the label file.

[ ]:
```
model.export_model()
!ls exported_model
```

[ ]:
```
files.download('exported_model/gesture_recognizer.task')
```

1.4 Display Result

[ ]:

[ ]:

[ ]:

[ ]: