Hand Landmark Detection Experiments for SSD

Aug 9th, 2023

To, professor

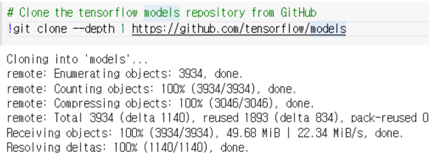
Index_

1. TensorFlow Object Detection Dependencies
2. Prepare Training Data
3. Training a model
4. Inference test images and Calculate mAP

1. TensorFlow Object Detection Dependencies

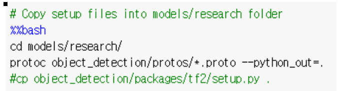
1. TensorFlow Object Detection Dependencies

1.



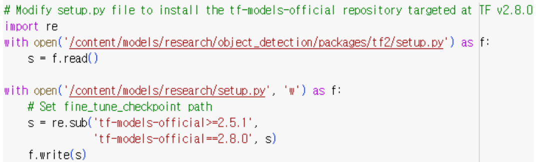
```
# Clone the tensorflow models repository from GitHub
!git clone --depth 1 https://github.com/tensorflow/models

Cloning into 'models'...
remote: Enumerating objects: 3934, done.
remote: Counting objects: 100% (3934/3934), done.
remote: Compressing objects: 100% (3046/3046), done.
remote: Total 3934 (delta 1140), reused 1893 (delta 834), pack-reused 0
Receiving objects: 100% (3934/3934), 49.68 MiB | 22.34 MiB/s, done.
Resolving deltas: 100% (1140/1140), done.
```

2.



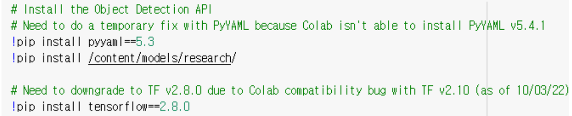
```
# Copy setup files into models/research folder
%%bash
cd models/research/
protoc object_detection/protos/*.proto --python_out=.
#cp object_detection/packages/tf2/setup.py .
```

3.



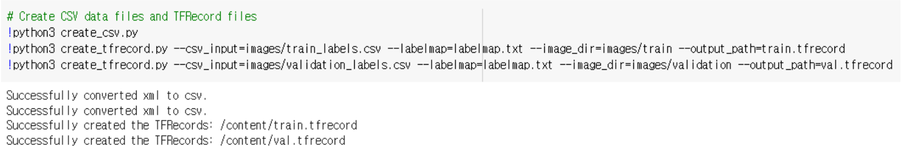
```
# Modify setup.py file to install the tf-models-official repository targeted at TF v2.8.0
import re
with open('/content/models/research/object_detection/packages/tf2/setup.py') as f:
    s = f.read()

with open('/content/models/research/setup.py', 'w') as f:
    # Set fine_tune_checkpoint path
    s = re.sub('tf-models-official>=2.5.1',
               'tf-models-official==2.8.0', s)
    f.write(s)
```

4.



```
# Install the Object Detection API
# Need to do a temporary fix with PyYAML because Colab isn't able to install PyYAML v5.4.1
!pip install pyyaml==5.3
!pip install /content/models/research/

# Need to downgrade to TF v2.8.0 due to Colab compatibility bug with TF v2.10 (as of 10/03/22)
!pip install tensorflow==2.8.0
```

5.

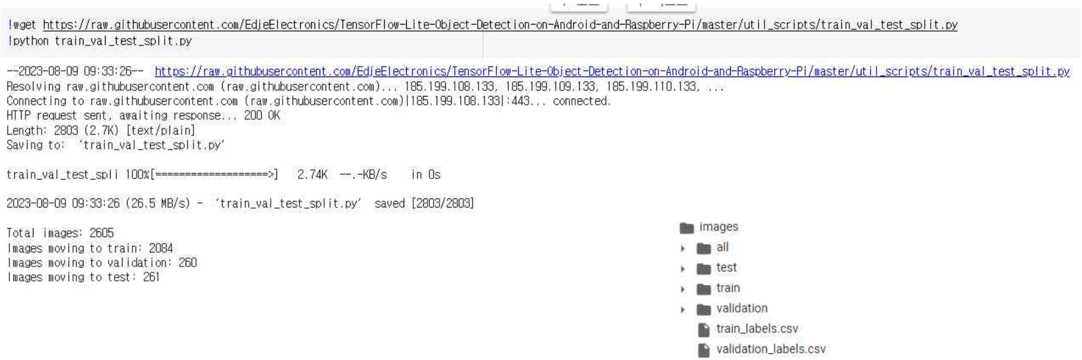


```
# Create CSV data files and TFRecord files
!python3 create_csv.py
!python3 create_tfrecord.py --csv_input=images/train_labels.csv --labelmap=labelmap.txt --image_dir=images/train --output_path=train.tfrecord
!python3 create_tfrecord.py --csv_input=images/validation_labels.csv --labelmap=labelmap.txt --image_dir=images/validation --output_path=val.tfrecord

Successfully converted xml to csv.
Successfully converted xml to csv.
Successfully created the TFRecords: /content/train.tfrecord
Successfully created the TFRecords: /content/val.tfrecord
```

2. Prepare Training Data

2. Split images into train, validation and test folders

- There are 2,605 image files extracted with OpenPose and 2,605 xml files using dark label.
- First, all files are in the images/all/ folder.
- Second, I organized the train, val, and test folders into images and csv files at a ratio of 8:1:1 through the train_val_test.py file.

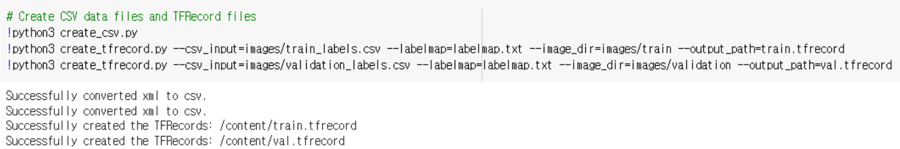


```
!wget https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/master/util_scripts/train_val_test_split.py
!python train_val_test_split.py
```

```
--2023-08-09 09:33:26--  https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/master/util_scripts/train_val_test_split.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2803 (2.7K) [text/plain]
Saving to: 'train_val_test_split.py'

train_val_test_spli 100%[===================>]   2.74K  --.-KB/s    in 0s

2023-08-09 09:33:26 (26.5 MB/s) - 'train_val_test_split.py' saved [2803/2803]

Total images: 2605
Images moving to train: 2084
Images moving to validation: 260
Images moving to test: 261
```

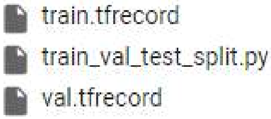
2. Creating tfrecord files from csv files & Labelmap

- After finishing Data Split, I converted xml files into csv files, and train.tfrecord files and val.I converted it into a tfrecord file.



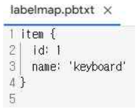
```
# Create CSV data files and TFRecord files
!python3 create_csv.py
!python3 create_tfrecord.py --csv_input=images/train_labels.csv --labelmap=labelmap.txt --image_dir=images/train --output_path=train.tfrecord
!python3 create_tfrecord.py --csv_input=images/validation_labels.csv --labelmap=labelmap.txt --image_dir=images/validation --output_path=val.tfrecord
```

```
Successfully converted xml to csv.
Successfully converted xml to csv.
Successfully created the TFRecords: /content/train.tfrecord
Successfully created the TFRecords: /content/val.tfrecord
```



```
train.tfrecord
train_val_test_split.py
val.tfrecord
```

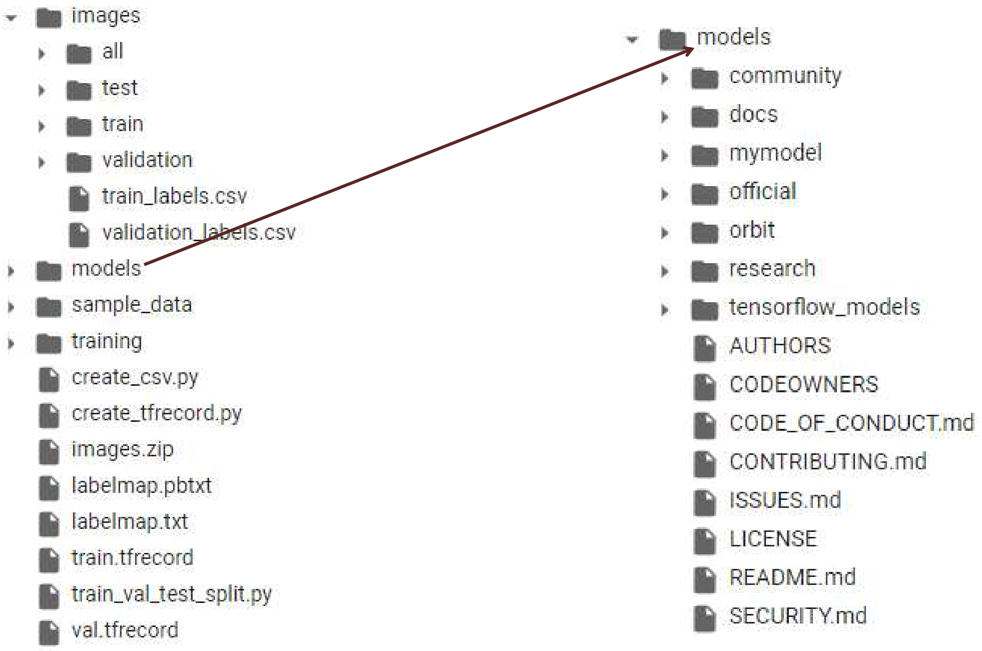
- Creating Labelmap
    - In labelmap.pbtxt, only one class called 'keyboard' was entered. For your information, the id starts with 1 (0 is background)



labelmap.pbtxt

```
item {
  id: 1
  name: 'keyboard'
}

```

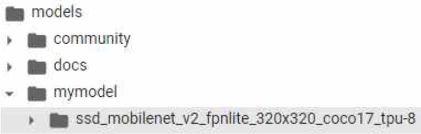
2. Model folder structure



```
images
    all
    test
    train
    validation
    train_labels.csv
    validation_labels.csv
models
sample_data
training
create_csv.py
create_tfrecord.py
images.zip
labelmap.pbtxt
labelmap.txt
train.tfrecord
train_val_test_split.py
val.tfrecord
```

```
models
    community
    docs
    mymodel
    official
    orbit
    research
    tensorflow_models
    AUTHORS
    CODEOWNERS
    CODE_OF_CONDUCT.md
    CONTRIBUTING.md
    ISSUES.md
    LICENSE
    README.md
    SECURITY.md
```

3. Training a model



```
models
  community
  docs
  mymodel
    ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8
```

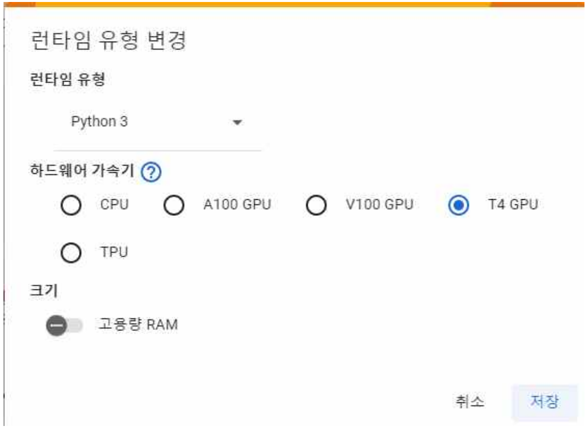
A pretrained model was chosen ‘ssd-mobilenet-v2-fpnlite-320’.

files description

- ckpt: The commonly talked ckpt file is the same as .ckpt-data, with only learned weights. (excluding deep learning models)
- Checkpoint file: This file is a binary file that stores weights, biases, gradients, etc.(consist of data-00000-of-00001 and ckpt-num.index)

3. Training a model

- I trained the model using Google Colab's T4 GPU.
- This model’s parameters are 30,000 of steps, batch size of 16.



런타임 유형 변경

런타임 유형

Python 3

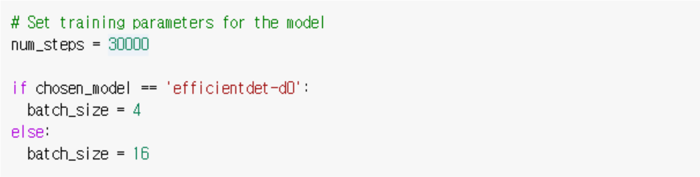
하드웨어 가속기

○ CPU ○ A100 GPU ○ V100 GPU ● T4 GPU

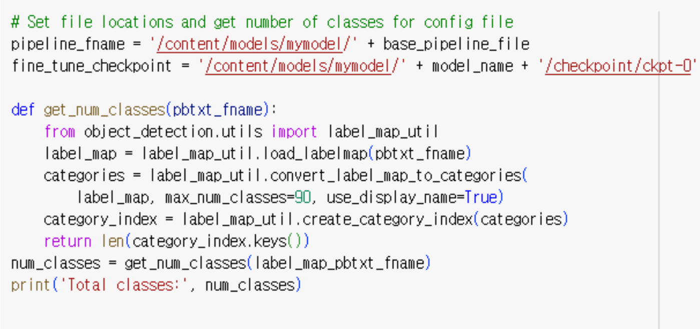
○ TPU

크기

고용량 RAM



```
# Set training parameters for the model
num_steps = 30000

if chosen_model == 'efficientdet-d0':
  batch_size = 4
else:
  batch_size = 16
```

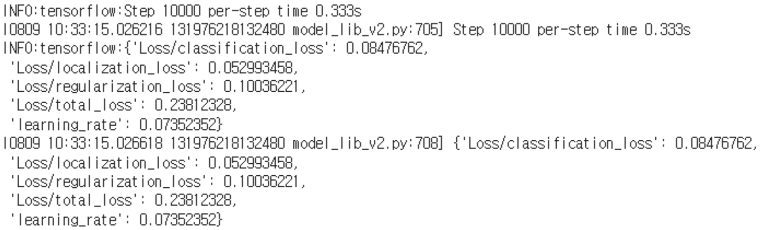


```
# Set file locations and get number of classes for config file
pipeline_fname = '/content/models/mymodel/' + base_pipeline_file
fine_tune_checkpoint = '/content/models/mymodel/' + model_name + '/checkpoint/ckpt-0'

def get_num_classes(pbtxt_fname):
    from object_detection.utils import label_map_util
    label_map = label_map_util.load_labelmap(pbtxt_fname)
    categories = label_map_util.convert_label_map_to_categories(
        label_map, max_num_classes=90, use_display_name=True)
    category_index = label_map_util.create_category_index(categories)
    return len(category_index.keys())
num_classes = get_num_classes(label_map_pbtxt_fname)
print('Total classes:', num_classes)
```

```
Total classes: 1
```

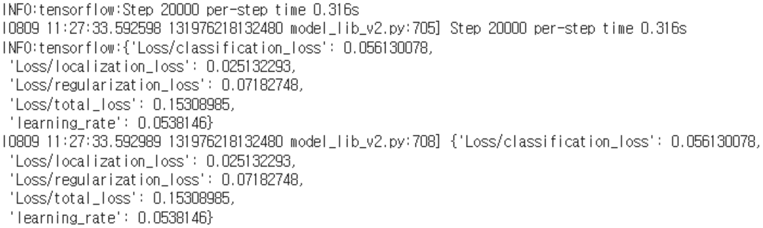
3. Training a model

- Step 10,000



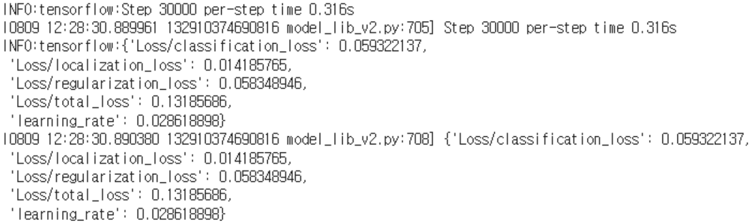
```
INFO:tensorflow:Step 10000 per-step time 0.333s
I0809 10:33:15.026216 131976218132480 model_lib_v2.py:705] Step 10000 per-step time 0.333s
INFO:tensorflow:{'Loss/classification_loss': 0.08476762,
 'Loss/localization_loss': 0.052993458,
 'Loss/regularization_loss': 0.10036221,
 'Loss/total_loss': 0.23812328,
 'learning_rate': 0.07352352}
I0809 10:33:15.026618 131976218132480 model_lib_v2.py:708] {'Loss/classification_loss': 0.08476762,
 'Loss/localization_loss': 0.052993458,
 'Loss/regularization_loss': 0.10036221,
 'Loss/total_loss': 0.23812328,
 'learning_rate': 0.07352352}
```

- Step 20,000



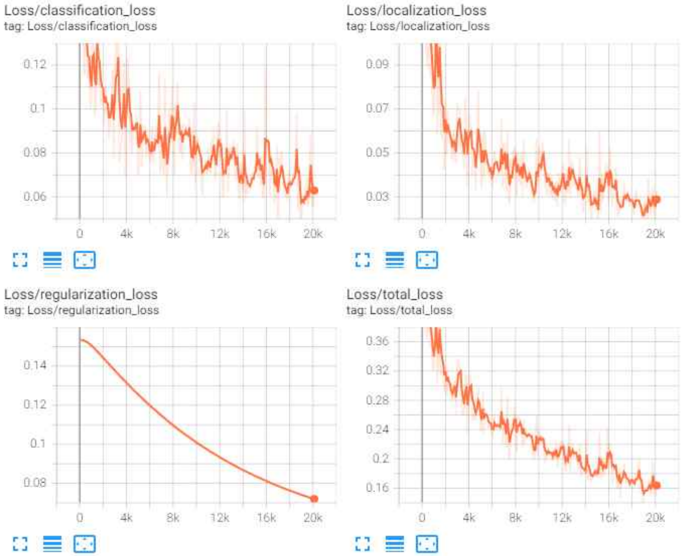
```
INFO:tensorflow:Step 20000 per-step time 0.316s
I0809 11:27:33.592598 131976218132480 model_lib_v2.py:705] Step 20000 per-step time 0.316s
INFO:tensorflow:{'Loss/classification_loss': 0.056130078,
 'Loss/localization_loss': 0.025132293,
 'Loss/regularization_loss': 0.07182748,
 'Loss/total_loss': 0.15308985,
 'learning_rate': 0.0538146}
I0809 11:27:33.592989 131976218132480 model_lib_v2.py:708] {'Loss/classification_loss': 0.056130078,
 'Loss/localization_loss': 0.025132293,
 'Loss/regularization_loss': 0.07182748,
 'Loss/total_loss': 0.15308985,
 'learning_rate': 0.0538146}
```

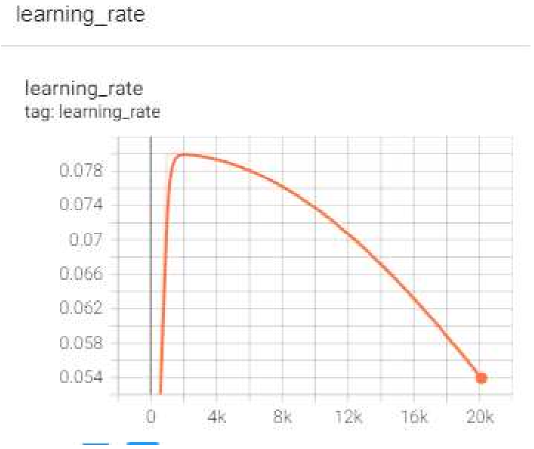
- Step 30,000



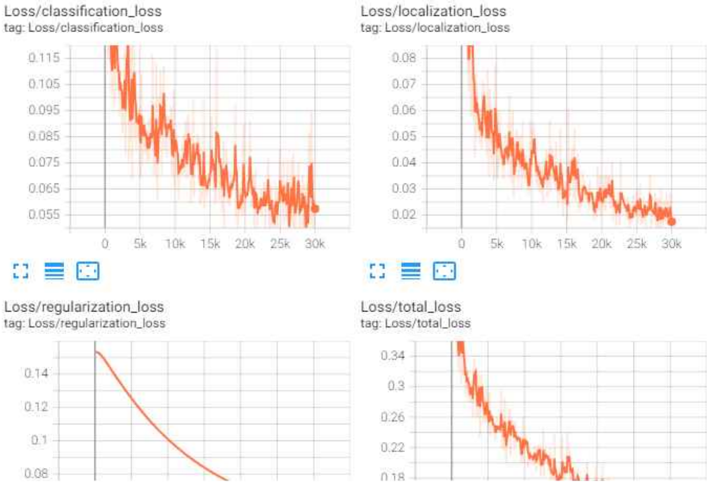
```
INFO:tensorflow:Step 30000 per-step time 0.316s
I0809 12:28:30.889961 132910374690816 model_lib_v2.py:705] Step 30000 per-step time 0.316s
INFO:tensorflow:{'Loss/classification_loss': 0.059322137,
 'Loss/localization_loss': 0.014185765,
 'Loss/regularization_loss': 0.058348946,
 'Loss/total_loss': 0.13185686,
 'learning_rate': 0.028618898}
I0809 12:28:30.890380 132910374690816 model_lib_v2.py:708] {'Loss/classification_loss': 0.059322137,
 'Loss/localization_loss': 0.014185765,
 'Loss/regularization_loss': 0.058348946,
 'Loss/total_loss': 0.13185686,
 'learning_rate': 0.028618898}
```

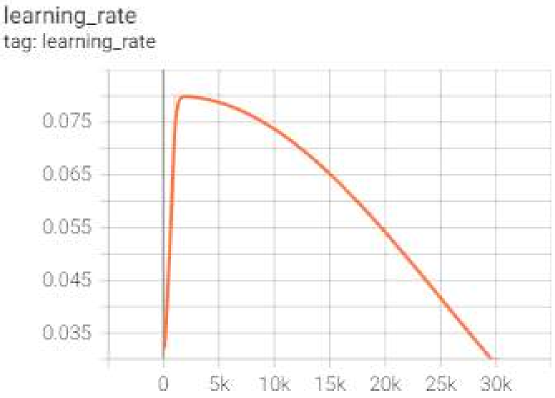
3. Tensor Board – until 20,000 steps



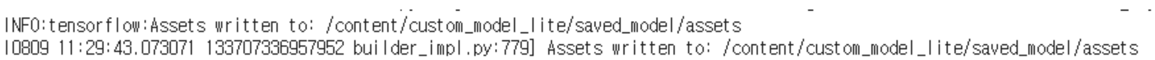


3. Tensor Board – until 30,000 steps

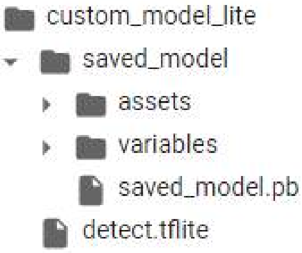




3. Convert Model to TensorFlow Lite



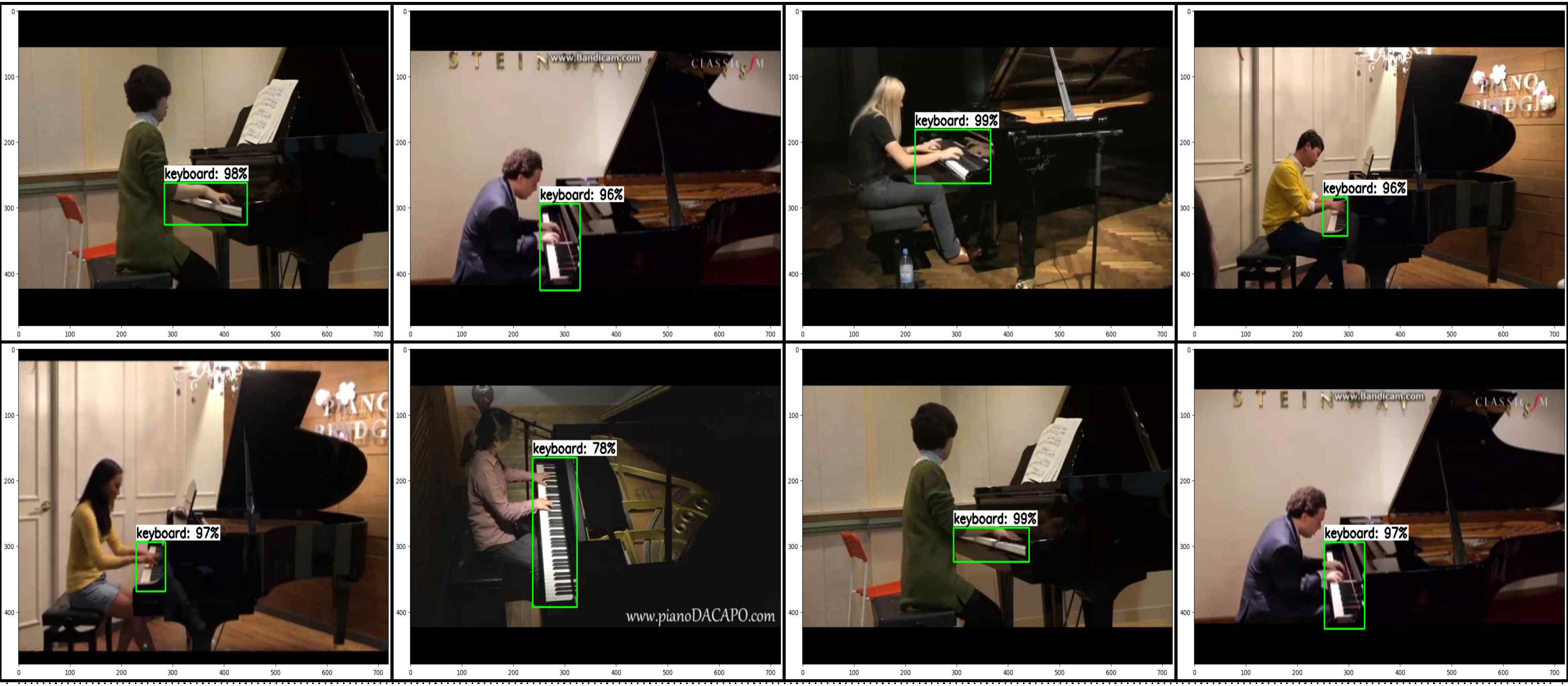
```
INFO:tensorflow:Assets written to: /content/custom_model_lite/saved_model/assets
I0809 11:29:43.073071 133707336957952 builder_impl.py:779] Assets written to: /content/custom_model_lite/saved_model/assets
```



```
custom_model_lite
  saved_model
    assets
    variables
    saved_model.pb
  detect.tflite
```

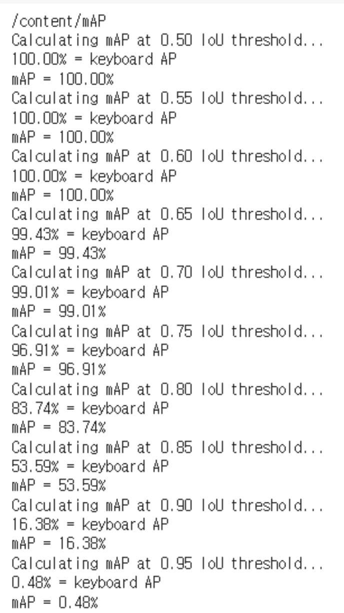
4. Inference test images and Calculate mAP

4. Inference test images – min_conf_threshold=0.5

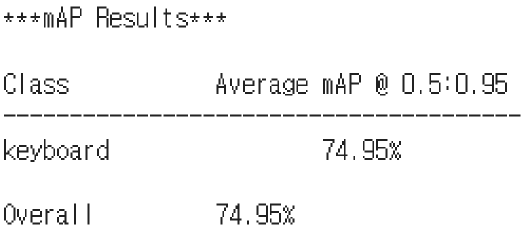


4. Inference test images – min_conf_threshold=0.5 – Failed

4. Calculate mAP



```
/content/mAP
Calculating mAP at 0.50 IoU threshold...
100.00% = keyboard AP
mAP = 100.00%
Calculating mAP at 0.55 IoU threshold...
100.00% = keyboard AP
mAP = 100.00%
Calculating mAP at 0.60 IoU threshold...
100.00% = keyboard AP
mAP = 100.00%
Calculating mAP at 0.65 IoU threshold...
99.43% = keyboard AP
mAP = 99.43%
Calculating mAP at 0.70 IoU threshold...
99.01% = keyboard AP
mAP = 99.01%
Calculating mAP at 0.75 IoU threshold...
96.91% = keyboard AP
mAP = 96.91%
Calculating mAP at 0.80 IoU threshold...
83.74% = keyboard AP
mAP = 83.74%
Calculating mAP at 0.85 IoU threshold...
53.59% = keyboard AP
mAP = 53.59%
Calculating mAP at 0.90 IoU threshold...
16.38% = keyboard AP
mAP = 16.38%
Calculating mAP at 0.95 IoU threshold...
0.48% = keyboard AP
mAP = 0.48%
```



```
***mAP Results***

Class          Average mAP @ 0.5:0.95
-----------------------------------------
keyboard                 74.95%

Overall        74.95%
```