Lampiran 3 Kode Sumber Deteksi Abjad BISINDO.ipynb

```
# 0. Setup Paths
import os
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME =
'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/202007
11/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow',
'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow',
'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow',
'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow',
'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }
files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'],
TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'],
LABEL_MAP_NAME)
}
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
```

```
            !mkdir {path}

# Download TF Models Pretrained Models from Tensorflow Model Zoo
and Install TFOD

if os.name=='nt':
    !pip install wget
    import wget
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'],
'research', 'object_detection')):
    !git clone https://github.com/tensorflow/models
{paths['APIMODEL_PATH']}
# Clone the tensorflow models repository from GitHub
!pip uninstall Cython -y # Temporary fix for "No module named
'object_detection'" error
# Copy setup files into models/research folder
%%bash
cd /content/Tensorflow/models/research
protoc object_detection/protos/*.proto --python_out=.
# Modify setup.py file to install the tf-models-official
repository targeted at TF v2.8.0
import re
with
open('/content/Tensorflow/models/research/object_detection/package
s/tf2/setup.py') as f:
    s = f.read()

with open('/content/Tensorflow/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)
# Install the Object Detection API (NOTE: This block takes about
10 minutes to finish executing)

# 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/Tensorflow/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
!pip install tensorflowjs==3.18.0
!pip install numpy==1.22.0
```

```
# Install CUDA version 11.0 (to maintain compatibility with TF
v2.8.0)
!pip install tensorflow_io==0.23.1
!wget
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu180
4/x86_64/cuda-ubuntu1804.pin
!mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-
pin-600
!wget
http://developer.download.nvidia.com/compute/cuda/11.0.2/local_ins
tallers/cuda-repo-ubuntu1804-11-0-local_11.0.2-450.51.05-
1_amd64.deb
!dpkg -i cuda-repo-ubuntu1804-11-0-local_11.0.2-450.51.05-
1_amd64.deb
!apt-key add /var/cuda-repo-ubuntu1804-11-0-local/7fa2af80.pub
!apt-get update && sudo apt-get install cuda-toolkit-11-0
!export LD_LIBRARY_PATH=/usr/local/cuda-
11.0/lib64:$LD_LIBRARY_PATH
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'],
'research', 'object_detection', 'builders',
'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}
import object_detection
if os.name =='posix':
    !wget {PRETRAINED_MODEL_URL}
    !mv {PRETRAINED_MODEL_NAME+'.tar.gz'}
{paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf
{PRETRAINED_MODEL_NAME+'.tar.gz'}
if os.name == 'nt':
    wget.download(PRETRAINED_MODEL_URL)
    !move {PRETRAINED_MODEL_NAME+'.tar.gz'}
{paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf
{PRETRAINED_MODEL_NAME+'.tar.gz'}

# Create Label Map

labels = [
    {'name': 'Abjad A', 'id': 1},
    {'name': 'Abjad B', 'id': 2},
    {'name': 'Abjad C', 'id': 3},
    {'name': 'Abjad D', 'id': 4},
    {'name': 'Abjad E', 'id': 5},
    {'name': 'Abjad F', 'id': 6},
    {'name': 'Abjad G', 'id': 7},
```

```
    {'name': 'Abjad H', 'id': 8},
    {'name': 'Abjad I', 'id': 9},
    {'name': 'Abjad J', 'id': 10},
    {'name': 'Abjad K', 'id': 11},
    {'name': 'Abjad L', 'id': 12},
    {'name': 'Abjad M', 'id': 13},
    {'name': 'Abjad N', 'id': 14},
    {'name': 'Abjad O', 'id': 15},
    {'name': 'Abjad P', 'id': 16},
    {'name': 'Abjad Q', 'id': 17},
    {'name': 'Abjad R', 'id': 18},
    {'name': 'Abjad S', 'id': 19},
    {'name': 'Abjad T', 'id': 20},
    {'name': 'Abjad U', 'id': 21},
    {'name': 'Abjad V', 'id': 22},
    {'name': 'Abjad W', 'id': 23},
    {'name': 'Abjad X', 'id': 24},
    {'name': 'Abjad Y', 'id': 25},
    {'name': 'Abjad Z', 'id': 26}
]

with open(files['LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')



# Split Data
!wget https://raw.githubusercontent.com/hanntoo/split-
data/main/train_test_split.py
!python train_test_split.py

# Create TF records
if not os.path.exists(files['TF_RECORD_SCRIPT']):
    !git clone https://github.com/nicknochnack/GenerateTFRecord
{paths['SCRIPTS_PATH']}
!python {files['TF_RECORD_SCRIPT']} -x
{os.path.join(paths['IMAGE_PATH'], 'train')} -l
{files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'],
'train.record')}
!python {files['TF_RECORD_SCRIPT']} -x
{os.path.join(paths['IMAGE_PATH'], 'test')} -l {files['LABELMAP']}
-o {os.path.join(paths['ANNOTATION_PATH'], 'test.record')}
```

```
# Train the model
%load_ext tensorboard
%tensorboard --logdir
'/content/Tensorflow/workspace/models/my_ssd_mobnet/train'
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research',
'object_detection', 'model_main_tf2.py')
command = "python {} --model_dir={} --pipeline_config_path={} --
num_train_steps=10000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])
!{command}

# Evaluate the Model
%tensorboard --logdir
'/content/Tensorflow/workspace/models/my_ssd_mobnet/eval' --
port=6008
command = "python {} --model_dir={} --pipeline_config_path={} --
checkpoint_dir={}".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'],
paths['CHECKPOINT_PATH'])
!{command}
```