SISTEM DETEKSI PLAT NOMOR MENGGUNAKAN *TENSORFLOW OBJECT DETECTION API* DENGAN MEMANFAATKAN *SSD MOBILENET V2*

LAPORAN KECERDASAN BUATAN

Disusun untuk Memenuhi Persyaratan Mata Kuliah Pengantar Kecerdasan Buatan
Program Studi Sarjana Terapan Teknik Informatika

Oleh :

Nama	NIM
1. Desty Nurul Anitsa	20090134
2. Maulana Alamsyah	20090137

PROGRAM STUDI SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA TEGAL
2022

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Pesatnya berkembangan teknologi saat ini membuat peningkatan teknologi dan penemuan teknologi baru dikembangkan dan ditemukan dengan cepat. Hal ini dilakukan untuk membantu mempermudah pekerjaan manusia dengan biaya yang cukup efisien. Salah satu teknologi yang sedang dilakukan pengembangan secara besar-besaran yaitu *Artificial Intelligence*.

Artificial Intelligence atau *AI* memiliki berbagai macam fungsi dan tujuan tergantung dari sistem yang akan dibuat. Salah satunya yaitu pendekteksian objek dan teks dari gambar atau video.

Object detection merupakan suatu teknik agar komputer bisa melihat dan mengidentifikasi objek yang ada baik dalam gambar ataupun video atau dapat diartikan juga sebagai perancangan suatu sistem agar sistem dapat melihat dan mengidentifikasi objek seperti manusia (Zhao et al., 2019). Dalam pembuatan ini diperlukan data-data pendukung yang nantinya data-data tersebut akan dilatih agar sistem dapat mengenali suatu objek. Contoh dari pemanfaatan teknologi ini yaitu pada pendeteksian objek plat nomor kendaraan.

Pada penelitian kali ini kami membuat sistem pendeteksi plat nomor menjadi teks, menggunakan framework *Tensorflow Object Detection* untuk melakukan training data dengan memanfaatkan model dari algoritma SSD Mobilenet V2 yang sudah dilatih oleh *Tensorflow*, hasil dari training data tersebut adalah pretrained model. Dari pretrained model tersebut ditraining data kembali dengan menggunakan dataset gambar yang sudah peneliti labeling. Hasil training data tersebut adalah model yang digunakan pada objek deteksi. Sistem ini dibantu dengan library OpenCV pada bahasa pemrogramanan python. Penelitian ini bertujuan untuk mengembangkan sistem deteksi plat nomor menjadi teks sebagai keamanan parkir

1.2 Rumusan Masalah

Dari latar belakang, dapat disimpulkan rumusan masalah sebagai berikut:

2.1.1 Melakukan training data dan testing data dengan dataset gambar plat nomor.

2.1.2 Membuat sistem pendeteksi plat nomor dikonversikan menjadi teks.

1.3 Tujuan

Berikut tujuan dibuatnya sistem deteksi plat nomor:

3.1.1 Mengembangkan sistem deteksi plat nomor menjadi teks sebagai keamanan parkir.

BAB II

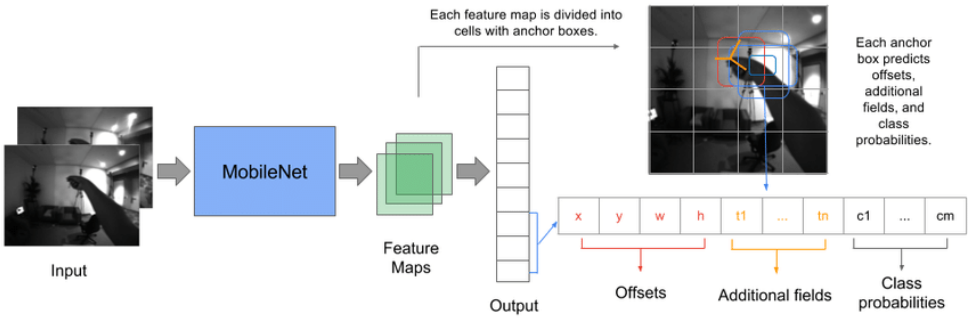
METODE YANG DIGUNAKAN

2.1 MobileNet

MobileNet, merupakan salah satu arsitektur *Convolutional Neural Network (CNN)* yang dapat digunakan untuk mengatasi kebutuhan akan sumber komputasi berlebih. Para peneliti dari Google membangun MobileNet atas kebutuhan arsitektur CNN yang dapat digunakan untuk ponsel maupun sistem tertanam.

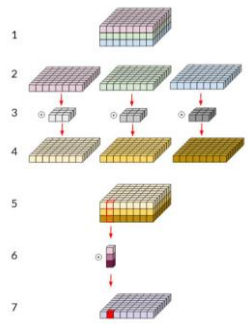
Perbedaan mendasar antara arsitektur MobileNet dan arsitektur CNN pada umumnya adalah penggunaan lapisan atau layer konvolusi dengan ketebalan filter yang sesuai dengan ketebalan dari input image. MobileNet membagi konvolusi menjadi depthwise convolution dan pointwise convolution. Arsitektur MobileNet memanfaatkan Batch Normalization (BN) dan Rectified-Linear unit (ReLU) untuk depthwise convolution dan pointwise convolution.

MobileNet dibangun di atas arsitektur jaringan yang efisien dengan menggunakan konvolusi yang dapat dipisahkan secara mendalam untuk menghasilkan Deep Neural Network yang ringan. Alur kerja MobileNet dapat dilihat pada Gambar 1



Gambar 1. Alur Kerja MobileNet

DSC menggantikan konvolusi standard dengan 2 tahap operasi: 1. *Depthwise Convolution* dimana setiap filter $DF \times DF$ hanya melakukan proses filter terhadap sebuah feature map input secara mendalam; 2. *Pointwise Convolution* yang merupakan 1×1 *convolution layer* yang digunakan untuk menggabungkan jalur informasi dari *depthwise layer*. Ilustrasi DSC dapat dilihat pada Gambar 2



Gambar 2. Ilustrasi DSC

DSC menyebabkan jalur konvolusi menjadi jauh lebih efesien dengan menggunakan parameter yang jauh lebih sedikit. Penggunaan parameter dan daya komputasi dapat dilihat pada Tabel 1

Tabel 1. Ukuran Parameter dan Komputasi pada Depthwise Separable Convolution

Layer *Standard Conv*	Ukuran Parameter	Ukuran Komputasi
Depthwise Separable	$F \times F \times C1+1 \times 1 \times C1 \times C2$	$F \times F \times DM \times DM \times C1+1 \times 1 \times C1 \times C2$

Reduksi ukuran komputasi yang didapat adalah sebagai berikut :

$$\frac{F \times F \times D_M \times D_M \times C_1 + C_1 \times C_2 \times D_N \times D_N}{F \times F \times D_M \times D_M \times C_1 \times C_2 \times D_N} = \frac{1}{N} + \frac{1}{F^2}$$

Dengan begitu, maka reduksi parameter menjadi :

$$\frac{F \times F \times C_1 + 1 \times 1 \times C_1 \times C_2}{F \times F \times D_M \times D_M \times C_1 \times C_2 \times D_N} = \frac{1}{C_2} + \frac{1}{F^2|}$$

2.2 Single Shot MultiBox Detector (SSD)

Perbedaan utama antara pelatihan detektor klasik seperti R-CNN dan pelatihan SSD adalah bahwa detektor objek klasik menggunakan proposal wilayah dan metode SSD menggunakan data *groundtruth* yang harus ditetapkan untuk hasil yang ditentukan dalam rangkaian hasil detektor yang jelas. Setelah penugasan ini ditentukan, *loss function* dilakukan dari ujung ke ujung. Metode SSD meakukan pencocokan terhadap objek dengan *default bounding box* melalui beragam skala dan rasio untuk setiap *feature map location*. Setiap elemen dari *feature map* memiliki sejumlah kotak yang saling berhubungan. Setiap bounding box dengan IoU lebih besar dari 0,5 dianggap cocok terhadap objek dan akan diproses.

BAB III

IMPLEMENTASI SISTEM

1. Mendownload arsitektur *My SSD MobileNet*
   Membuat variable untuk menyimpan file/url yang dibutuhkan.

```
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/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
```

2. Membuat folder path
   Dalam folder path terdapat key yang berisikan jalur folder yang akan disimpan.

```
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')
 }
```

3. Membuat file
   Dalam file, terdapat key untuk menyimpan file pada jalur yang sudah ditentukan.

```
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)
}
```

4. Memastikan *OS* yang digunakan
   OS yang digunakan adalah posix(linux). Maka akan membuat path yang telah disimpan pada variable path

```
print(os.name)

#os yang digunakan posix, maka akan dibuat direktori path
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

5. Mendownload tf model garder, model yang sudah disediakan oleh *tensorflow* File yang akan di download/clone akan di simpan pada folder tensorflow/model/research/object_detection.

```
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

```
Cloning into 'Tensorflow/models'...
remote: Enumerating objects: 74222, done.
remote: Counting objects: 100% (308/308), done.
remote: Compressing objects: 100% (174/174), done.
remote: Total 74222 (delta 146), reused 274 (delta 125), pack-reused 73914
Receiving objects: 100% (74222/74222), 580.23 MiB | 32.33 MiB/s, done.
Resolving deltas: 100% (52585/52585), done.
```

6. Install Framwork *Tensorflow Object Detection* dan Install Protobuf Untuk mengInstall Tensorflow Object Detection menggunakan serialisai dari file yang .proto menjadi .py

```
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=.
    && cp object_detection/packages/tf2/setup.py . && python -m pip install .
```

7. Verifikasi library yang dibutuhkan VERIFICATION_SCRIPT menyimpan jalur dari file model_builder_tf2_test.py

```
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}
```

8. Mengimport library object_detection dan mendownload pretrained model *SSD MobileNet*

```
#test library
import object_detection
```

```
#download pretrained model ssd mobile net
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'}
```

9. Membuat Label Map
   Membuat label yang sesuai dengan labeling

```
labels = [{'name':'platNomor', 'id':1}]

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')
```

10. Menduplikat file pretrained model ke folder training

```
if os.name =='posix':
    !cp {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
if os.name == 'nt':
    !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
```

11. Mengimport library dan membuat variable config
    Konfigurasi pada pipeline training mendefinisikan model mana dan parameter apa yang akan digunakan untuk proses training. API deteksi objek tensorflow menggunakan berkas protobuf untuk mengkonfigurasi proses training dan evaluasi

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

```
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
```

```
config
```

12. Membaca konfigurasi

```
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
```

13. Memberi nilai pada pipeline_config

```
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'test.record')]
```

14. Install OpenCV

```
!pip install opencv-python-headless==4.1.2.30
```

15. Mengcopy dataset yang sudah dilabeling dari roboflow file .record

Dataset yang sudah dilabeling di copy pada Tensorflow/workspace/annotations/train.record Tensorflow/workspace/annotations/test.record

```
!cp '/content/gdrive/MyDrive/KULIAH DESTY/SMT 4/PROJEK AI-SISTEM PARKIR/FIX/train.record' 'Tensorflow/workspace/annotations/train.record'
```

```
!cp '/content/gdrive/MyDrive/KULIAH DESTY/SMT 4/PROJEK AI-SISTEM PARKIR/FIX/test.record' 'Tensorflow/workspace/annotations/test.record'
```

16. Training model

Pada training model menggunakan dataset yang telah dilabeling yaitu data yang ada pada folder tensorflow/workspace/annotation/train.record. Pada saat proses training, sistem manghasilkan chekpoint(file ckpt) yang dibuat otomatis oleh tf.

```
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'])
```

```
print(command)
```

```
python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/models/my_ssd_mobnet --pipeline_con
```

```
!{command}
```

17. Load model dari checkpoint

```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
```

```
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-11')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections

configs['model']
```

18. Deteksi gambar

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

```
category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
print(category_index)
```

```
{1: {'id': 1, 'name': 'platNomor'}}
```

```
IMAGE_PATH = '/content/gdrive/MyDrive/KULIAH DESTY/SMT 4/PROJEK AI-SISTEM PARKIR/FIX/Image/20220530_160602.jpg'
```

```
IMAGE_PATH
```

```
'/content/gdrive/MyDrive/KULIAH DESTY/SMT 4/PROJEK AI-SISTEM PARKIR/FIX/Image/20220530_160602.jpg'
```

```
image_np = np.array(img) #gambar yang digunakan untuk deteksi diubah menjadi aray

#mengubah array ke bentuk tensor dengan tipe data berupa float32
input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()
count = 0
for box,score,cls in zip(detections['detection_boxes'],detections['detection_scores'],detections['detection_classes']):
    if score >= 0.3 and score <= 1.0:
        count += 1

count
viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.0,
            agnostic_mode=False,
            line_thickness=4,
            groundtruth_box_visualization_color='black',
            skip_scores=False,
            skip_labels=False,
            skip_track_ids=False)

# plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
# plt.show()
```

```
img = cv2.imread(IMAGE_PATH) #cv2 untuk membaca gambar
image_np = np.array(img) #gambar yang digunakan untuk deteksi diubah menjadi aray

#mengubah array ke bentuk tensor dengan tipe data berupa float32
input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)
```

```
num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()
count = 0
for box,score,cls in zip(detections['detection_boxes'],detections['detection_scores'],detections['detection_classes']):
    if score >= 0.5 and score <= 1.0:
        viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.5,
            agnostic_mode=False,
            line_thickness=20, #ketebalan box 20
            groundtruth_box_visualization_color='black',
            skip_scores=False,
            skip_labels=False,
            skip_track_ids=False)



plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

19. Install OCR

```
!pip install easyocr
#instal ocr(buah perangkat lunak yang mengubah teks dalam format berkas citra atau
# gambar ke dalam format teks yang bisa dibaca dan disunting oleh aplikasi komputer)
```

```
import easyocr
```

```
detection_threshold = 0.5 #minimal deteksi yang akan di ambil gambarnya
```

```
image = image_np_with_detections
norm_img = np.zeros((image.shape[0], image.shape[1]))
# lower = np.array([0,0,0])
# upper = np.array([179,100,130])
# image = cv2.inRange(image, lower, upper)
image = cv2.normalize(image, norm_img, 0, 255, cv2.NORM_MINMAX)
image = cv2.threshold(image, 100, 255, cv2.THRESH_BINARY)[1]
image = cv2.GaussianBlur(image, (1, 1), 0)
scores = list(filter(lambda x: x> detection_threshold, detections['detection_scores']))
boxes = detections['detection_boxes'][:len(scores)]
classes = detections['detection_classes'][:len(scores)]
```

```
width = image.shape[1]
height = image.shape[0]
```

```
#Penerapan Filter Recognition Optic Image untuk mengambil text dari gambar
for idx, box in enumerate(boxes):
    print(box)
    roi = box*[height, width, height, width]
    print(roi)
    region = image[int(roi[0]):int(roi[2]),int(roi[1]):int(roi[3])]
    reader = easyocr.Reader(['id'])
    ocr_result = reader.readtext(region)
    print(ocr_result)
```

```
deteksi = cv2.cvtColor(region, cv2.COLOR_BGR2RGB)
plt.imshow(deteksi)
```

```
<matplotlib.image.AxesImage at 0x7f358b4be7d0>
```


```
norm_img = np.zeros((image.shape[0], image.shape[1]))
image = cv2.normalize(image, norm_img, 0, 255, cv2.NORM_MINMAX)
image = cv2.threshold(image, 100, 255, cv2.THRESH_BINARY)[1]
image = cv2.GaussianBlur(image, (1, 1), 0)
print(image)
```

```
print(deteksi)
```

```
for result in ocr_result:
    print(np.sum(np.subtract(result[0][2],result[0][1])))
    print(result[1])
```

```
region_threshold = 0.02 #ketebalan huruf
```

```
# membuat function print plate hasil deteksi
def filter_text(region, ocr_result, region_threshold):
    rectangle_size = region.shape[0]*region.shape[1]

    plate = []

    for result in ocr_result:
        length = np.sum(np.subtract(result[0][1], result[0][0]))
        height = np.sum(np.subtract(result[0][2], result[0][1]))

        if length*height / rectangle_size > region_threshold:
            plate.append(result[1])
    return plate
```

```
arr = filter_text(region, ocr_result, region_threshold) #hasil dari fungsi flter teks adalah berupa array
plat = ' '.join(arr) #menggabungkan array dengan pemisah spasi
plat[0:10] #mencetak string pada plat mulai dari index ke 0 sampai index ke 10
```

```
'6 2377 BDF'
```