ДОДАТОК А

Результати тренування моделі

Таблиця А.1 – Результати тренування моделі

epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision (B)	metrics/recall(B)	metrics/mAP50 (B)	metrics/mAP50-95(B)	val/box_loss	val/cls_loss	val/dfl_loss
1	1,265	1,236	1,239	0,905	0,880	0,930	0,566	1,410	0,851	1,305
2	1,266	0,788	1,257	0,948	0,904	0,939	0,572	1,384	0,673	1,316
3	1,246	0,733	1,245	0,962	0,907	0,956	0,607	1,309	0,632	1,271
4	1,218	0,694	1,235	0,968	0,922	0,960	0,636	1,258	0,554	1,233
5	1,197	0,657	1,220	0,956	0,931	0,963	0,641	1,240	0,545	1,228
6	1,171	0,632	1,202	0,965	0,932	0,971	0,639	1,229	0,537	1,227
7	1,150	0,605	1,194	0,974	0,928	0,967	0,644	1,225	0,499	1,216
8	1,132	0,586	1,181	0,977	0,938	0,971	0,656	1,193	0,500	1,200
9	1,123	0,572	1,176	0,971	0,942	0,971	0,671	1,171	0,481	1,183
10	1,098	0,557	1,165	0,962	0,945	0,975	0,671	1,175	0,474	1,182
11	1,089	0,481	1,184	0,973	0,944	0,976	0,663	1,181	0,480	1,193
12	1,074	0,462	1,177	0,981	0,943	0,977	0,674	1,171	0,455	1,188
13	1,056	0,453	1,169	0,985	0,944	0,981	0,684	1,150	0,452	1,174
14	1,044	0,437	1,162	0,986	0,945	0,982	0,686	1,142	0,447	1,169
15	1,029	0,427	1,156	0,979	0,949	0,981	0,692	1,135	0,436	1,163
16	1,013	0,414	1,151	0,987	0,950	0,983	0,689	1,135	0,436	1,162
17	1,003	0,403	1,138	0,984	0,953	0,982	0,697	1,139	0,419	1,168
18	0,984	0,388	1,127	0,981	0,958	0,983	0,696	1,118	0,412	1,149
19	0,971	0,381	1,116	0,983	0,956	0,983	0,700	1,121	0,405	1,151
20	0,960	0,371	1,112	0,989	0,954	0,984	0,701	1,119	0,401	1,149

ДОДАТОК Б

Лістинг програми

```
import cv2
import numpy as np
import pandas as pd
from scipy.interpolate import interp1d
from sort.sort import Sort
from ultralytics import YOLO
import easyocr
import string
import csv
import ast
class ObjectTracker:
def __init__(self):
self.mot_tracker = Sort()
def track_objects(self, detections):
# Відстежування виявлених об'єктів
track_ids = self.mot_tracker.update(np.asarray(detections))
return track_ids
class LicensePlateDetector:
def __init__(self, model_path):
self.model = YOLO(model_path)
self.reader = easyocr.Reader(['en'], gpu=False)
def detect_license_plates(self, frame):
# Виявлення номерних знакі в кадрі
return self.model(frame)[0]
def read_license_plate(self, license_plate_crop):
# Ідентифікації тексту номерного знака з зображення
detections = self.reader.readtext(license_plate_crop)
for detection in detections:
bbox, text, score = detection
text = text.upper().replace(' ', '')
if self.license_complies_format(text):
return self.format_license(text), score
return None, None
@staticmethod
def license_complies_format(text):
# Перевірка, чи відповідає текст номерного знака необхідному
формату
if len(text) != 7:
return False
if (text[0] in string.ascii_uppercase or text[0] in
LicensePlateDetector.dict_int_to_char.keys()) and \
(text[1] in string.ascii_uppercase or text[1] in
LicensePlateDetector.dict_int_to_char.keys()) and \
(text[2] in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
or text[2] in LicensePlateDetector.dict_char_to_int.keys()) and
\
(text[3] in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
or text[3] in LicensePlateDetector.dict_char_to_int.keys()) and
\
```

```
(text[4] in string.ascii_uppercase or text[4] in
LicensePlateDetector.dict_int_to_char.keys()) and \
(text[5] in string.ascii_uppercase or text[5] in
LicensePlateDetector.dict_int_to_char.keys()) and \
(text[6] in string.ascii_uppercase or text[6] in
LicensePlateDetector.dict_int_to_char.keys()):
return True
else:
return False
@staticmethod
def format_license(text):
# Застосування правил відображення символів до тексту номерного
знака з відхиленнями
license_plate_ = ''
mapping = {0: LicensePlateDetector.dict_int_to_char,
1: LicensePlateDetector.dict_int_to_char,
4: LicensePlateDetector.dict_int_to_char,
5: LicensePlateDetector.dict_int_to_char,
6: LicensePlateDetector.dict_int_to_char,
2: LicensePlateDetector.dict_char_to_int,
3: LicensePlateDetector.dict_char_to_int}
for j in [0, 1, 2, 3, 4, 5, 6]:
if text[j] in mapping[j].keys():
license_plate_ += mapping[j][text[j]]
else:
license_plate_ += text[j]
return license_plate_
dict_char_to_int =
{'O': '0', 'I': '1', 'J': '3', 'A': '4', 'G': '6', 'S': '5'}
dict_int_to_char =
{'0': 'O', '1': 'I', '3': 'J', '4': 'A', '6': 'G', '5': 'S'}
class UtilityFunctions:
@staticmethod
def get_car(license_plate, vehicle_track_ids):
# Отримання координат автомобіля та ідентифікатору на основі
координат номерного знака
x1, y1, x2, y2, score, class_id = license_plate
foundIt = False
for j in range(len(vehicle_track_ids)):
xcar1, ycar1, xcar2, ycar2, car_id = vehicle_track_ids[j]
if x1 > xcar1 and y1 > ycar1 and x2 < xcar2 and y2 < ycar2:
car_indx = j
foundIt = True
break
if foundIt:
return vehicle_track_ids[car_indx]
return -1, -1, -1, -1, -1
@staticmethod
def write_csv(results, output_path):
# Запис результатів у файл CSV
with open(output_path, 'w') as f:
```

```
f.write('{},{},{},{},{},{},{}\n'.format('frame_nmr', 'car_id',
'car_bbox', 'license_plate_bbox', 'license_plate_bbox_score',
'license_number', 'license_number_score'))
for frame_nmr in results.keys():
for car_id in results[frame_nmr].keys():
if 'car' in results[frame_nmr][car_id].keys() and \
'license_plate' in results[frame_nmr][car_id].keys() and 'text'
in \ results[frame_nmr][car_id]['license_plate'].keys():
f.write('{},{},{},{},{},{},{}\n'.format(frame_nmr, car_id, '[{}
{} {} {}]'.format( results[frame_nmr][car_id]['car']['bbox'][0],
results[frame_nmr][car_id]['car']['bbox'][1],
results[frame_nmr][car_id]['car']['bbox'][2],
results[frame_nmr][car_id]['car']['bbox'][3]), '[{} {} {}
{}]'.format(results[frame_nmr][car_id]['license_plate']['bbox'][
0], results[frame_nmr][car_id]['license_plate']['bbox'][1],
results[frame_nmr][car_id]['license_plate']['bbox'][2],
results[frame_nmr][car_id]['license_plate']['bbox'][3]),
results[frame_nmr][car_id]['license_plate']['bbox_score'],
results[frame_nmr][car_id]['license_plate']['text'],
results[frame_nmr][car_id]['license_plate']['text_score'])
)
f.close()
@staticmethod
def read_csv(input_path):
# Зчитування результатів з файлу CSV
with open(input_path, 'r') as file:
reader = csv.DictReader(file)
data = list(reader)
return data
@staticmethod
def interpolate_bounding_boxes(data):
# Інтерполяція відсутніх обмежувальних рамкок в даних
frame_numbers = np.array([int(row['frame_nmr']) for row in
data])
car_ids = np.array([int(float(row['car_id'])) for row in data])
car_bboxes = np.array([list(map(float, row['car_bbox'][1:-
1].split())) for row in data])
license_plate_bboxes = np.array([list(map(float,
row['license_plate_bbox'][1:-1].split())) for row in data])
interpolated_data = []
unique_car_ids = np.unique(car_ids)
for car_id in unique_car_ids:
frame_numbers_ = [p['frame_nmr'] for p in data if
int(float(p['car_id'])) == int(float(car_id))]
car_mask = car_ids == car_id
car_frame_numbers = frame_numbers[car_mask]
car_bboxes_interpolated = []
license_plate_bboxes_interpolated = []
first_frame_number = car_frame_numbers[0]
for i in range(len(car_bboxes[car_mask])):
frame_number = car_frame_numbers[i]
car_bbox = car_bboxes[car_mask][i]
license_plate_bbox = license_plate_bboxes[car_mask][i]
```

```
if i > 0:
prev_frame_number = car_frame_numbers[i - 1]
prev_car_bbox = car_bboxes_interpolated[-1]
prev_license_plate_bbox = license_plate_bboxes_interpolated[-1]
if frame_number - prev_frame_number > 1:
frames_gap = frame_number - prev_frame_number
x = np.array([prev_frame_number, frame_number])
x_new = np.linspace(prev_frame_number, frame_number,
num=frames_gap, endpoint=False)
interp_func = interp1d(x, np.vstack((prev_car_bbox, )), axis=0,
kind='linear')
interpolated_car_bboxes = interp_func(x_new)
interp_func = interp1d(x, np.vstack((prev_license_plate_bbox,
license_plate_bbox)), axis=0, kind='linear')
interpolated_license_plate_bboxes = interp_func(x_new)
car_bboxes_interpolated.extend(
interpolated_car_bboxes[1:])
license_plate_bboxes_interpolated.extend(
interpolated_license_plate_bboxes[1:])
car_bboxes_interpolated.append(car_bbox)
license_plate_bboxes_interpolated.append(license_plate_bbox)
for i in range(len(car_bboxes_interpolated)):
frame_number = first_frame_number + i
row = {}
row['frame_nmr'] = str(frame_number)
row['car_id'] = str(car_id)
row['car_bbox'] = ' '.join(map(str, car_bboxes_interpolated[i]))
row['license_plate_bbox'] = ' '.join(map(str,
license_plate_bboxes_interpolated[i]))
if str(frame_number) not in frame_numbers_:
row['license_plate_bbox_score'] = '0'
row['license_number'] = '0'
row['license_number_score'] = '0'
else:
original_row = [p for p in data if
int(p['frame_nmr']) == frame_number and int(float(p['car_id']))
== int(float(car_id))][0] row['license_plate_bbox_score'] =
original_row['license_plate_bbox_score'] if
'license_plate_bbox_score' in original_row else
'0'row['license_number'] = original_row['license_number'] if
'license_number' in original_row else
'0'row['license_number_score'] =
original_row['license_number_score'] if 'license_number_score'
in original_row else '0'
interpolated_data.append(row)
return interpolated_data
class Visualizer:
def __init__(self, video_path):
self.video_path = video_path
self.cap = cv2.VideoCapture(video_path)
def visualize_results(self):
# Візуалізація результатів
results = pd.read_csv('./output_interpolated.csv')
```

```
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
fps = self.cap.get(cv2.CAP_PROP_FPS)
width = int(self.cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(self.cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
out = cv2.VideoWriter('./output.mp4', fourcc, fps, (width,
height))
license_plate = {}
for car_id in np.unique(results['car_id']):
max_ = np.amax(results[results['car_id'] ==
car_id]['license_number_score'])
license_plate[car_id] = {'license_crop': None,
'license_plate_number':
results[(results['car_id'] == car_id) &
(results['license_number_score'] ==
max_)]['license_number'].iloc[0]}
self.cap.set(cv2.CAP_PROP_POS_FRAMES, results[(results['car_id']
== car_id) & (results['license_number_score'] ==
max_)]['frame_nmr'].iloc[0])
ret, frame = self.cap.read()
x1, y1, x2, y2 = ast.literal_eval(results[(results['car_id'] ==
car_id) & (results['license_number_score'] ==
max_)]['license_plate_bbox'].iloc[0].replace('[ ',
'[').replace(' ', ' ').replace(' ', ' ').replace(' ', ','))
license_crop = frame[int(y1):int(y2), int(x1):int(x2), :]
license_crop = cv2.resize(license_crop, (int((x2 - x1) * 400 /
(y2 - y1)), 400))
license_plate[car_id]['license_crop'] = license_crop
frame_nmr = -1
self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
ret = True
while ret:
ret, frame = self.cap.read()
frame_nmr += 1
if ret:
df_ = results[results['frame_nmr'] == frame_nmr]
for row_indx in range(len(df_)):
# Відображення виявлених номерних знаків
x1, y1, x2, y2 =
ast.literal_eval(df_.iloc[row_indx]['license_plate_bbox'].replac
e('[ ', '[').replace(' ', ' ').replace(' ', ' ').replace(' ',
',','))
cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0,
0, 255), 12)
license_crop =
license_plate[df_.iloc[row_indx]['car_id']]['license_crop']
H, W, _ = license_crop.shape
# Підписування виявлених номерних знаків
label_text =
license_plate[df_.iloc[row_indx]['car_id']]['license_plate_numbe
r']
cv2.putText(frame, label_text, (int(x1), int(y1) - 10),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 255, 0), 10)
out.write(frame)
```

```
frame = cv2.resize(frame, (1280, 720))
out.release()
self.cap.release()
```

ДОДАТОК В

Демонстраційний матеріал

№ документа	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК.502840.042ПЗ	Пояснювальна записка	А4, 67 с.
		Інші документи	
2		Результати тренування	
		моделі	А4, 1 с.
3		Лістинг програми	А4, 6 с.
4		Демонстраційний матеріал	А4, 18 с.

Змін.	Арк	Номер докум.	Підп.	Дата	ГЮИК.502840.042ВД			
Розробив		Головань Д.С.			Розроблення програмного модуля ідентифікації номерних знаків автотранспортних засобів Відомість кваліфікаційної роботи	Літер	Аркуш	Арк
Перевірив		Аллахвераноv Р.Ю.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Норм.контр		Демська Н.П.						
Затвердив		Невлюдов І.Ш.						