ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Використання згорткових нейронних мереж для розв'язання задачі розпізнавання реєстраційних автомобільних номерів

Автор:

Валентин ДЕМЧЕНКО
студ. гр. КІУКІ-21-5

Керівник:

Наталія БОЛОГОВА
доц. каф.ЕОМ

Мета

Метою кваліфікаційної роботи є вивчення технологій комп'ютерного зору та написання програми, що здійснює сегментацію та розпізнавання символів реєстраційного автомобільного номера на зображенні.

За даними Укравтопром, продажі нових легковиків в Україні у 2024 році оцінено у понад 125 млрд гривень, це на 18% більше, ніж у 2023 році. Понад три чверті від всіх легковиків, що у 2023 році поповнили вітчизняний автопарк були вживані авто, ввезені з-за кордону, за 12 місяців 2024 року українці придбали понад 222,1 тис. таких автомобілів.

Транспортна інфраструктура розвивається швидкими темпами, і необхідність контролювати рух транспорту зростає.

Розпізнавання реєстраційних номерних знаків – це одна із задач ідентифікації транспортних засобів. Системи розпізнавання реєстраційних номерів по зображенню і відео використовуються в багатьох сферах, пов'язаних з транспортом, наприклад, на автобазах, підприємствах, стоянках ТЗ, автозаправних станціях, станціях технічного обслуговування, біля терміналів платних доріг, при автоматичній фото- і відеофіксації порушень на трасах і т. д.

KOBI software

INNI

В області контролю дорожнього руху теж використовуються технології Computer Vision.

В Україні розроблено декілька апаратно-програмних комплексів для розпізнавання автомобільних державних номерних знаків. Найпоширенішими є «Kobi Software», та «INNI Tech».

Kobi Software (ZetPro VMS) – система від української компанії Kobi Software призначена для розпізнавання державних номерних знаків транспортних засобів у режимі реального часу.

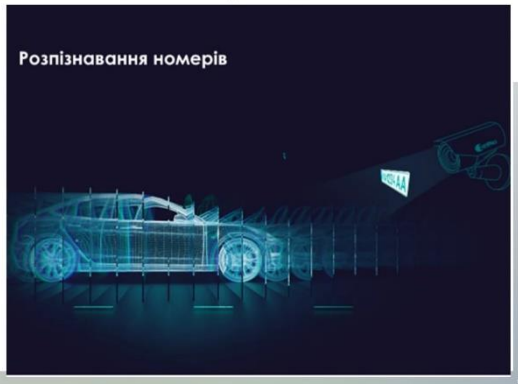
Призначена для використання на паркінгах, платних дорогах, АЗС та об'єктах критичної інфраструктури, де важливо відстежувати в'їзд і виїзд транспортних засобів.

INNI Tech – українська розробка, яка використовує технології штучного інтелекту та згорткових нейронних мереж для точного розпізнавання номерних знаків. INNI Tech підтримує обробку відео та фото в реальному часі, дозволяючи реалізувати проєкти на базі камер спостереження або мобільних пристроїв. Система широко використовується в міському моніторингу, контролі трафіку та автоматизованих системах паркування.

Проблеми систем розпізнавання

До основних проблем систем розпізнавання об'єктів на зображеннях і відеофайлах відносять наступні:

- низька якість зображення (причинами можуть бути зламана камера, невірні установки знімальної апаратури, велика відстань від камери до об'єкта, неправильна обробка фото та ін.);
- погане освітлення, наявність тіней;
- розмитість зображення (ця проблема найчастіше присутня в системах, які проводять відеофіксацію рухомих об'єктів);
- наявність предмета, який повністю або частково закриває собою досліджуваний об'єкт.



5

Основні причини вибору CNN:

1) Спеціалізація на обробці зображень.
2) Автоматичне виділення ознак.
3) Стійкість до спотворень.
4) Наявність готових архітектур.
5) Висока точність.

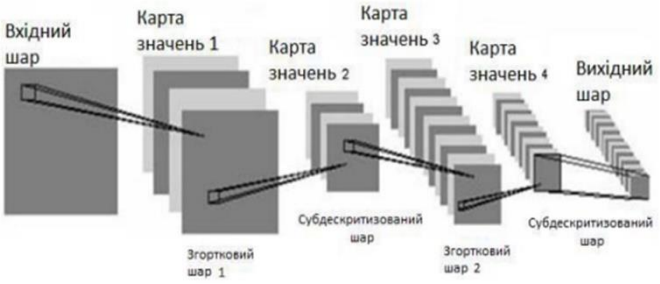


Рисунок 1 – Схема роботи згорткової мережі

6

Інструменти реалізації проєкту

Основним інструментом реалізації проєкту було обрано мову програмування Python.

Для написання коду використовувалося інтерактивне середовище Jupyter Notebook.

У розробці було використано Python 3.13.0, який надає доступ до широкого спектру бібліотек для аналізу даних і побудови моделей машинного навчання. Серед найважливіших:

- NumPy – бібліотека для високопродуктивних математичних операцій, що базується на багатовимірних масивах та реалізована на мові C;
- PIL (Python Imaging Library) – бібліотека для роботи з растровими зображеннями, яка підтримує формати JPEG, PNG, BMP, GIF тощо, і забезпечує можливості масштабування, обробки фільтрами та малювання;
- Keras – популярна високорівнева бібліотека для створення і навчання моделей штучних нейронних мереж, сумісна з TensorFlow і має вбудовані набори даних;
- OpenCV (Open Source Computer Vision Library) – потужна платформа для комп'ютерного зору, яка включає алгоритми обробки, фільтрації, сегментації зображень та багато іншого.

7

Формат номерного знаку

Вид реєстраційних номерних знаків (НЗ) України – це класифікація державних номерних знаків транспортних засобів за формою, кольором, призначенням і категорією власника згідно з технічними стандартами та нормативами, прийнятими в Україні.

- формат номерного знаку:
- стандартний зразок (після 2004 року): АА 0000 АА;
- перші дві літери – код регіону реєстрації (напр., АХ або КХ – Харків);
- далі – чотири цифри;
- останні дві літери – серія.

Формат має вигляд дві літери, чотири цифри, 2 літери

Рисунок 2 – Формат номерного знаку

8

Структура алгоритму

Алгоритм розпізнавання державного номерного знака складається з наступних етапів:

а) навчання нейронної мережі;
б) отримання знімка НЗ;
в) сегментація НЗ на окремі символи;
ґ) розпізнавання символів нейронною мережею.

9

Архітектура нейронної мережі

Для реалізації задачі розпізнавання символів на зображеннях автомобільних номерів було використано набір даних Train_Cells. Джерелом доступу – Kaggle, який належить компанії Google.

Рисунок 3 – Частина навчального набору

Реалізований у роботі алгоритм розпізнавання символів ґрунтується на архітектурі нейронної мережі LeNet.

Для навчання моделі було використано генератори зображень з бібліотеки Keras, які забезпечили завантаження та попередню обробку даних із набору Train_Cells.

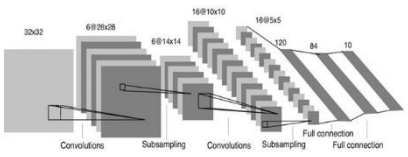


Рисунок 4 – Мережа LeNet

10

Робота алгоритму

Для тестування роботи алгоритму йому на вхід було подано знімок номерної пластини, представлений на рисунку 5.

Рисунок 5 – Знімок номерного знака

Алгоритм сегментації виділив такі символи на вхідному зображенні (рисунок 6).

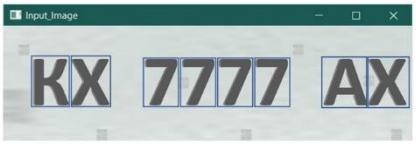


Рисунок 6 – Контури символів

На рисунку 7 представлені зображення символів (з доданою контрастністю та зміненим розміром) та результат роботи нейронної мережі – розпізнаний номер транспортного засобу.

Рисунок 7 – Результат роботи нейромережі

ВИСНОВКИ

У кваліфікаційній роботі було досліджено основні методи розпізнавання об'єктів на зображеннях, зокрема особливості проектування та навчання згорткових нейронних мереж. Практична частина роботи включала реалізацію процедури сегментації зображень та створення спеціалізованої ЗНМ для розпізнавання символів на державних номерних знаках транспортних засобів. Для реалізації проекту було обрано мову програмування Python з використанням бібліотек Keras для роботи з нейронними мережами та OpenCV для обробки зображень.

В якості основного інструменту сегментації був застосований метод Thresholding, тоді як для розпізнавання символів використана оптимізована архітектура LeNet з алгоритмом оптимізації Adamax. Навчання мережі проводилось на вибірці з 20 000 зображень, що дозволило досягти високої точності розпізнавання. Отримані результати демонструють ефективність згорткових нейронних мереж для розв'язання вузькоспеціалізованих задач комп'ютерного зору, зокрема у сфері автоматичного розпізнавання номерних знаків. Висока продуктивність та адаптивність таких мереж відкривають широкі перспективи для їх застосування в різних галузях, де необхідна обробка та аналіз візуальних даних.

ДОДАТОК Б

Вихідний код

```
import keras as K
від PIL import ImageEnhance import matplotlib.pyplot as plt
import cv2

TrainingGenerator =
K.preprocessing.image.ImageDataGenerator().flow_from_directory(
'Train_Cells', target_size=(48, 48), batch_size=32,
shuffle=True) ValidationGenerator =
K.preprocessing.image.ImageDataGenerator().flow_from_directory(
'Train_Cells', target_size=(48, 48), batch_size=32,
shuffle=True) AugGenerator =
K.preprocessing.image.ImageDataGenerator(
width_shift_range=0.1, height_shift_range=0.1, zoom_range=0.05,
brightness_range=(0.5,2.0)).flow_from_directory(
'Train_Cells', target_size=(48, 48), batch_size=32,
shuffle=True) Model = K.Sequential()
Model.add(K.layers.Conv2D(filters=64, kernel_size=(3,3),
activation="relu",input_shape=(48, 48, 3),
kernel_initializer="he_normal"))
Model.add(K.layers.Conv2D(filters=64,kernel_size=(3,3),
activation="relu", kernel_initializer="he_normal"))
Model.add(K.layers.BatchNormalization())
Model.add(K.layers.AveragePooling2D())Model.add(K.layers.Conv2D(
filters=128, kernel_size=(3,3),
activation="relu", kernel_initializer="he_normal"))

Model.add(K.layers.Conv2D(filters=128, kernel_size=(3,3),
activation="relu", kernel_initializer="he_normal"))
Model.add(K.layers.BatchNormalization())
Model.add(K.layers.AveragePooling2D())
Model.add(K.layers.Flatten())Model.add(K.layers.Dense(units=120,
activation="relu",
kernel_regularizer=K.regularizers.l1_l2(l1=1e-4, l2=1e-5),
kernel_initializer="he_normal"))
Model.add(K.layers.BatchNormalization())
Model.add(K.layers.Dropout(0.2))Model.add(K.layers.Dense(units=8
4, activation="relu",
kernel_regularizer=K.regularizers.l1_l2(l1=1e-4, l2=1e-5),
kernel_initializer="he_normal"))
Model.add(K.layers.BatchNormalization())
Model.add(K.layers.Dropout(0.2))Model.add(K.layers.Dense(units=2
1, activation="softmax"))
Model.compile(optimizer=K.optimizers.Adamax(learning_rate=1e-2),
loss=K.losses.categorical_crossentropy) Model.fit_generator(
TrainingGenerator, steps_per_epoch=480, epochs=100,
```

```
validation_data=ValidationGenerator, validation_steps=96)
Model.fit_generator( AugGenerator, steps_per_epoch=480,
epochs=100,
validation_data=ValidationGenerator, validation_steps=96)

def Image_Segmentation(Image_Name: str, out_size=48): IMAGE =
cv2.imread(Image_Name)
IMAGE_Gray_Colour = cv2.cvtColor(IMAGE, cv2.COLOR_BGR2GRAY)
_, Threshold = cv2.threshold (IMAGE_Gray_Colour, 125, 255,
cv2.THRESH_BINARY)
IMG_Erode = cv2.erode(Threshold, np.ones((3, 3), np.uint8),
iterations=1) Contours, Hierarchy = cv2.findContours(IMG_Erode,
cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
IMAGE_Copy = IMAGE.copy() Character_Images = []
for i, contour in enumerate(Contours):
(x, y, w, h) = cv2.boundingRect(contour) if Hierarchy[0][i][3]
== 0:
cv2.rectangle(IMAGE_Copy, (x, y), (x + w, y + h), (70, 0, 0), 1)
cropped_character = IMAGE_Gray_Colour [y: y + h, x: x + w]
size_max = max (w, h)
square_character = 255 * np.ones(shape=[size_max, size_max],
dtype=np.uint8)
if w > h:
y_pos = size_max//2 - h//2
square_character[y_pos:y_pos + h, 0:w] = cropped_character elif
w < h:
x_pos = size_max//2 - w//2
square_character[0:h, x_pos:x_pos + w] = cropped_character else:
square_character = cropped_character
Character_Images.append((x, w, cv2.resize(square_character,
(100, 150), interpolation=cv2.INTER_AREA)))
Character_Images.sort(key=lambda x: x[0], reverse=False) for j
in range(len(Character_Images) - 1):

One_Character = Image.fromarray(Character_Images[i][2])
One_Character.save(f"Symbols_for_NNW\symbol_{i}.bmp")
return Character_Images


Image_Segmentation("test_number.png")CarNumberSymbols =
"Symbols_for_NNW" PredictionCarNumber = ""
AllLetters = "ABCEMKMPTYX" FIGURE = plt.figure(figsize=(16,8))
SubPlots = FIGURE.subplots(1, 6) for s в SubPlots
s.axis("off")i = 0
for file in os.listdir(CarNumberSymbols):
image = Image.open(os.path.join(CarNumberSymbols, file)) image =
image.resize((48, 48), Image.BICUBIC)
ImageWithFilter =
ImageEnhance.Contrast(image).enhance(100).convert("L")
SubPlots[i].imshow(ImageWithFilter, cmap="gray")
NewImage = np.asarray(np.dstack((ImageWithFilter,
```

```
ImageWithFilter, ImageWithFilter)), dtype=np.uint8)
p = np.argmax(Model.predict(np.array([np.array(NewImage)]))))) if
p < 10:
PredictionCarNumber += str(p) else:
PredictionCarNumber += AllLetters[p-10] i += 1
plt.show()
print("Car number:") print(*PredictionCarNumber)
```