

График прохождения практики

|  |  |
| --- | --- |
| Дата | Вид деятельности |
| 05.02.2024 | Получение задания на практику |
| 06.02.2024-  10.04.2024 | Выполнить задание |
| 11.04.2024-14.04.2024 | Подготовить и оформить отчет |

Введение

Производственная практика является неотъемлемой и важной частью учебного процесса. В ходе неё не только закрепляются теоретические знания, полученные во время обучения, но и отрабатываются необходимые практические умения для эффективной работы в профессиональной сфере, осваивается инструментарий для работы и взаимодействия в команде, оттачиваются коммуникативные навыки для общения с коллегами.

Данный документ представляет собой отчет по производственной практике, проходившей в период с 5 февраля по 14 апреля ООО «Синаптик», г. Киров.

1. Общие сведения о предприятии

Предприятие ООО «Синаптик» действует с 14 февраля 2013 года.

Физический адрес: Российская Федерация, Кировская область, город Киров, улица Карла Маркса, 18А.

Организационно правовая форма предприятия – Общество с ограниченной ответственностью.

Основным видом деятельности компании является разработка компьютерного программного обеспечения, консультационные услуги в данной области и другие сопутствующие услуги.

1. Описание выполненной работы

В ходе данной практики предстояло разработать программный модуль для распознавания государственного номера автомобиля.

В соответствии с заданием от организации во время прохождения практики необходимо выполнить следующие работы:

1. Разработать модуль
2. Протестировать модель на представленных примерах
   1. Обоснование выбора метода

Для реализации модуля будут использоваться сверточная нейронная сеть, так как она имеет ряд преимуществ:

1. **Автоматическое извлечение признаков**: Нейронные сети могут автоматически извлекать признаки из изображений на разных уровнях абстракции, начиная с низкоуровневых признаков, таких как грани и текстуры, и заканчивая более высокоуровневыми концепциями, такими как формы и объекты. Это позволяет модели самостоятельно находить характеристики объектов на изображении без необходимости вручную определять признаки.
2. **Гибкость**: Нейронные сети предоставляют гибкую платформу для обучения на разнообразных данных и задачах. Их архитектура может быть адаптирована для различных видов объектов и условий съемки.
3. **Обучение на больших наборах данных**: Современные нейронные сети обычно требуют больших объемов данных для обучения. Однако, когда доступны достаточно большие и разнообразные наборы данных, нейронные сети способны достичь высокой точности и обобщающей способности в распознавании объектов.
4. **Прогрессивные результаты**: С появлением новых архитектур и методов обучения, нейронные сети продолжают достигать новых рекордов в точности распознавания объектов на изображениях. Такие архитектуры, как сверточные нейронные сети и их вариации, показывают впечатляющие результаты в области компьютерного зрения.
5. **Автоматизация**: После обучения нейронной сети ее можно использовать для автоматического распознавания объектов на новых изображениях без необходимости ручной обработки. Это обеспечивает высокую эффективность и ускоряет процесс обработки изображений в различных приложениях.

Методы, такие как SVM и метод Виолы-Джонса, также имеют свои преимущества и недостатки. Но существует несколько причин, по которым они могут быть менее предпочтительными по сравнению с нейронными сетями для распознавания объектов на изображении:

1. **Ограниченные возможности в извлечении признаков**: SVM и метод Виолы-Джонса требуют ручной инженерии признаков, что может быть сложным и трудоемким процессом. В случае большого разнообразия объектов и условий съемки создание подходящих признаков может быть непростой задачей. В отличие от этого, нейронные сети способны автоматически извлекать признаки из данных, что делает их более гибкими и способными к обучению на разнообразных наборах данных.
2. **Требования к размеру обучающего набора данных**: SVM и метод Виолы-Джонса могут быть более чувствительны к размеру и разнообразию обучающего набора данных. Они могут показывать хорошие результаты только при наличии достаточного количества представительных примеров изображений объектов, что может быть проблематично для некоторых приложений.
3. **Производительность на сложных задачах**: В случае больших и сложных наборов данных или задач компьютерного зрения, например, сегментации объектов с высоким уровнем детализации или с распознаванием объектов в условиях переменного освещения, нейронные сети обычно показывают более высокую производительность по сравнению с SVM и методом Виолы-Джонса.
4. **Необходимость в переобучении**: SVM и метод Виолы-Джонса могут иметь склонность к переобучению на некоторых типах данных, особенно если обучающий набор данных недостаточно разнообразен или если не проводится должная настройка параметров алгоритмов.

Это не означает, что SVM и метод Виолы-Джонса не имеют применения в распознавании объектов на изображениях. Они могут быть эффективны в некоторых специфических сценариях и при условии, что их преимущества и недостатки учитываются при выборе подходящего метода для конкретной задачи. Однако в целом, современные нейронные сети, особенно сверточные, обычно являются более предпочтительным выбором для задач распознавания объектов на изображениях благодаря своей гибкости.

Таким образом, использование нейронных сетей для распознавания объектов на изображении обеспечивает эффективный и точный метод автоматической обработки изображений в различных областях, таких как медицина, автомобильная промышленность, безопасность и многое другое.

* 1. Описание модели

Модель представляет собой комбинацию сверточной нейронной сети (CNN), рекуррентной нейронной сети долгой краткосрочной памяти (LSTM) и функции свертки по времени (CTC) для распознавания текста на изображениях. CNN используется для извлечения признаков из изображения номерного знака, LSTM преобразует эти признаки в последовательность символов, а CTC используется для выравнивания последовательности символов с фактическим текстом номера.

Структура модели представлена на рисунке 1.

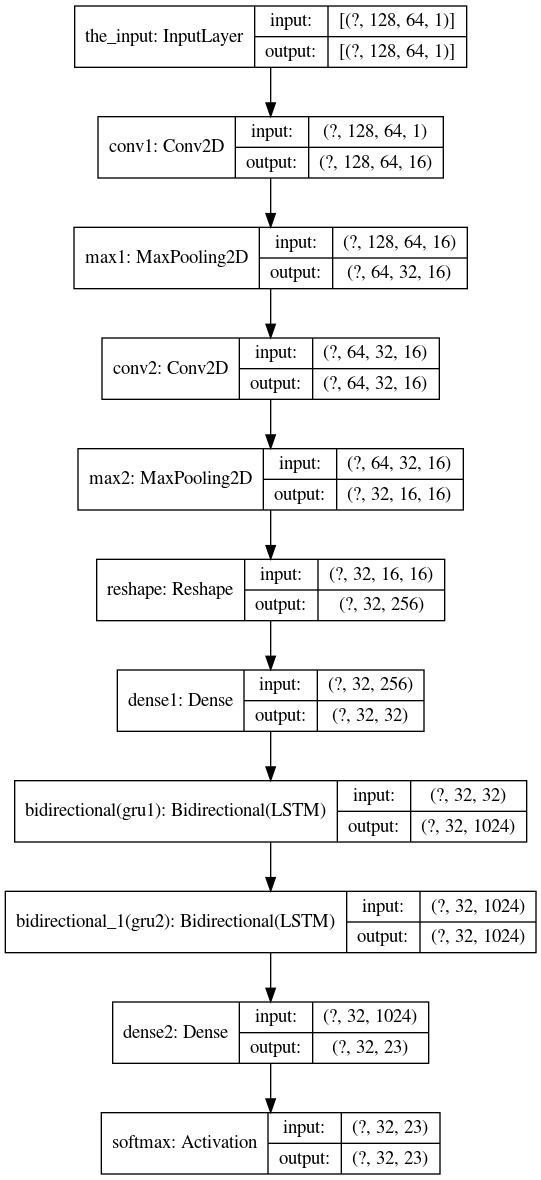
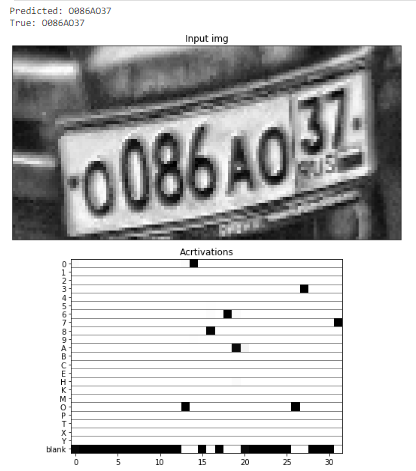


Рисунок 1 – Структура модели

Примеры тестирования представлены на рисунке 2.



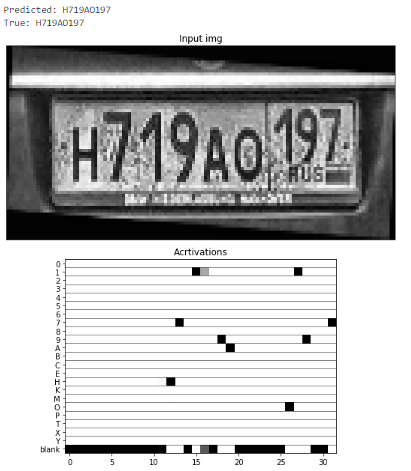


Рисунок 2 – Результаты тестирования

* 1. Разработка модуля.

Разработанный программный модель можно разбить на несколько частей:

1. **Пользовательский интерфейс**:
   * Скрипт загружает ранее разработанный с помощью PyQt6 макет пользовательского интерфейса из файла **.ui**.
   * Он отображает кнопки для выполнения действий, таких как загрузка видеофайла, разделение видео на кадры, выбор изображения и распознавание номерных знаков автомобилей.
   * В интерфейсе также есть метки для отображения изображений и распознанных номеров.

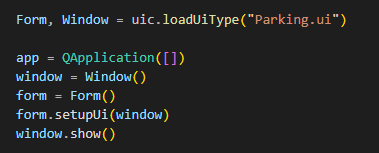


Рисунок 3 – Фрагмент кода

1. **Операции с базой данных**:
   * Скрипт взаимодействует с базой данных SQLite с названием **server.db**.
   * Создается таблица с названием **parking** для хранения номеров автомобильных номеров и количества посещений.
   * Функции **load\_file()** и **save\_file()** обрабатывают загрузку данных из базы данных и сохранение данных в нее соответственно.

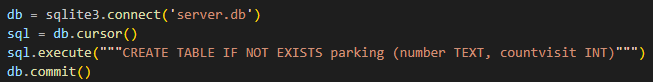


Рисунок 4 – Взаимодействие с БД

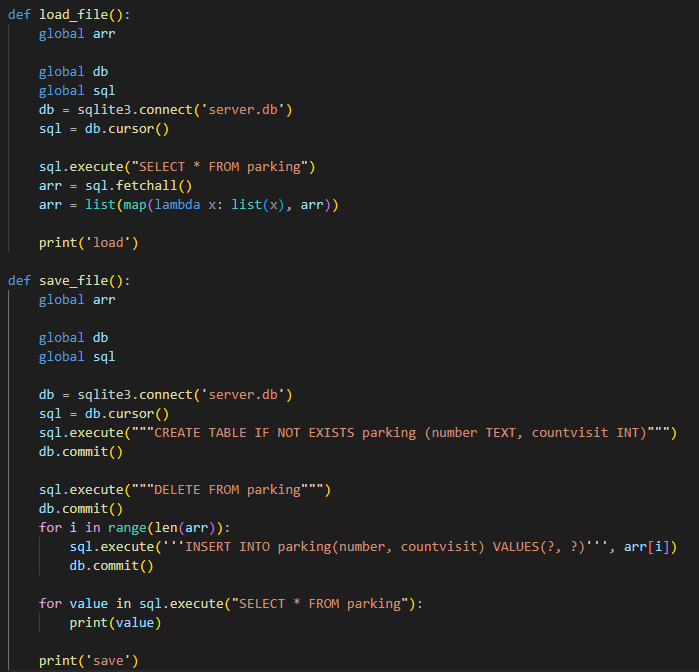


Рисунок 5 – Функции load\_file() и save\_file()

1. **Обработка видео**:
   * Скрипт позволяет загружать видеофайл и разделять его на кадры.
   * Функция **split()** использует MoviePy для разделения видео на кадры и сохранения их как изображения.

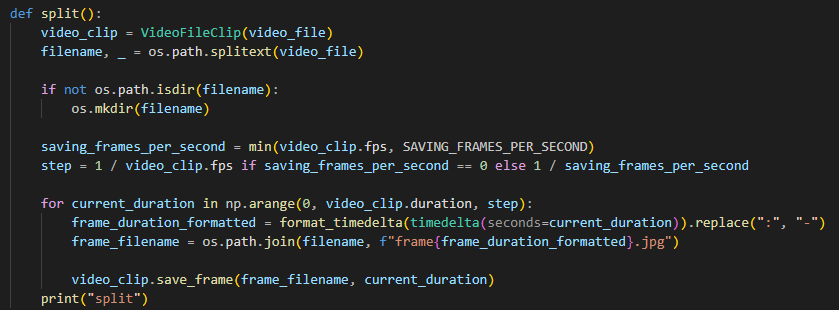


Рисунок 6 – Функция split()

1. **Обработка изображений и распознавание номерных знаков**:
   * Пользователи могут выбрать файл изображения с автомобильным номером.
   * Функция **carplate\_text()** выполняет распознавание номерного знака с использованием модели TensorFlow Lite.
   * Производится предварительная обработка изображения, обнаружение номерного знака, применение преобразований и распознавание текста номерного знака.
   * Распознанный текст номерного знака возвращается для отображения.
2. **Распознавание и обновление UI**:
   * Функция **recognize()** обновляет пользовательский интерфейс с распознанным номером автомобильного номера.
   * Также обновляется количество посещений в базе данных и рассчитывается процент скидки на основе количества посещений.



Рисунок 7 – Функция recognize()

2.2 Результат работы

На рисунке 8 представлено окно программы.

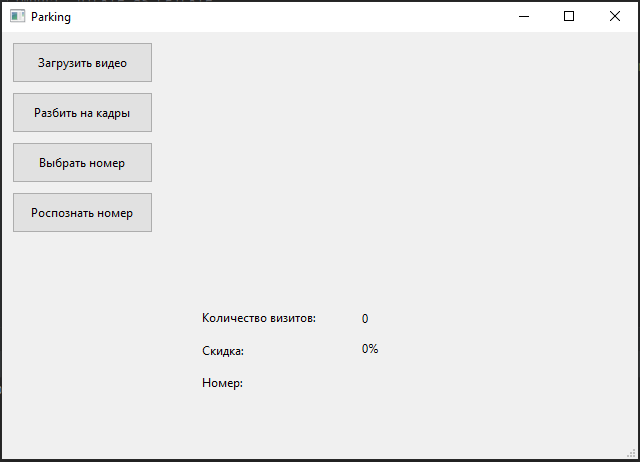


Рисунок 8 – Окно программы

После загрузки видеофайла и разбиения его на кадры можно выбрать один из полученных кадров.

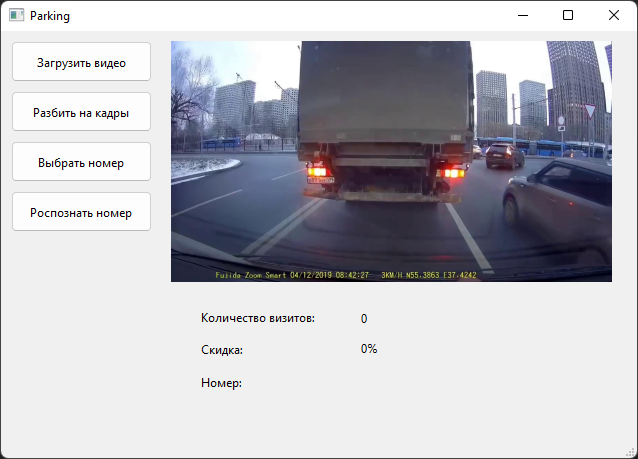


Рисунок 9 – Кадр выбран

Дальше можно распознать номер. Результат работы показан на рисунках 10-11.



Рисунок 10 – Определение номера

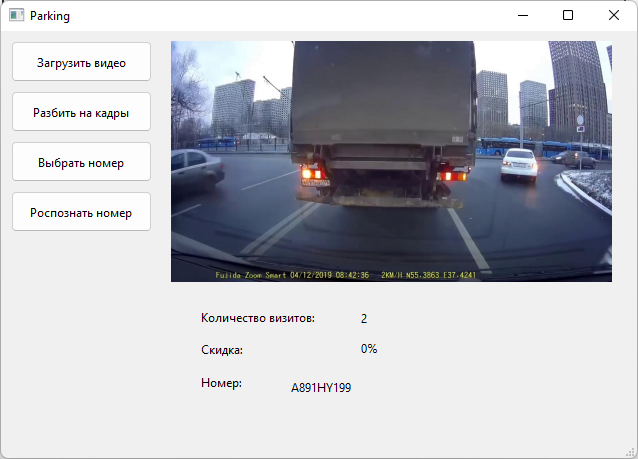


Рисунок 11 – Окно программы после определения

2.3 Тестирование

Для тестирования было подготовлено 100 изображений автомобилей в разных условиях. Примеры приведены на рисунке 12.





Рисунок 12 – Примеры тестовых изображений

Фрагмент итогов тестирования представлены в таблице 1.

Таблица 1 - Тестирование

|  |  |  |  |
| --- | --- | --- | --- |
| Изображение | Предсказанный номер | Эталонный номер | Распознано верно? |
| 1 | А123ВС78 | А123ВС78 | да |
| 2 | К456МО34 | К456МО34 | да |
| 3 | У789ХУ56 | У789ХУ56 | да |
| 4 | М012ОР78 | М012ОР78 | да |
| 5 | Т345НС72 | Т345НС12 | нет |
| 6 | А890СТ56 | А890СТ56 | да |
| 7 | М123КУ78 | М123КУ78 | да |
| 8 | О456СХ34 | О456СХ34 | да |
| 9 | А012МО78 | А012МО78 | да |
| 10 | У678ХС56 | У678ХС56 | да |

Диаграмма тестирования представлена на рисунке 13.



Рисунок 13 - Диаграмма

Заключение

В ходе выполнения задания был разработан и реализован программный проект, представляющий собой систему распознавания автомобильных номеров на изображениях. Программа успешно использует различные технологии, такие как TensorFlow Lite, OpenCV, SQLite, и другие, для решения задачи распознавания номеров.

Проект предоставляет пользователю интерфейс для загрузки видео и изображений, разделения видео на кадры, выбора конкретного кадра для распознавания, а также автоматического распознавания номеров на изображениях.

Основные функции проекта включают в себя обработку изображений с использованием алгоритмов компьютерного зрения, таких как алгоритмы Canny и Hough, а также использование нейронных сетей для распознавания символов на номерных знаках.

Благодаря разработанной программе пользователь получает возможность эффективно обрабатывать видео и изображения с автомобильными номерами, что может быть полезно для различных целей, включая контроль парковок, автоматизацию учета автомобилей и другие.

Для оценки качества работы программы была проведена серия тестов на сгенерированной тестовой выборке, результаты которых показали, что точность распознавания составляет более 80%, что свидетельствует о высокой эффективности и точности разработанной системы.

Таким образом, разработанный программный проект представляет собой важный шаг в области распознавания автомобильных номеров, что может быть полезно для широкого спектра приложений, связанных с автомобильной безопасностью, учетом транспортных средств и другими задачами.

Приложение А

Код модуля

import os

from datetime import timedelta

from pathlib import Path

import matplotlib.pyplot as plt

import cv2

#import \_pickle as cPickle

import numpy as np

import re

import imutils

import pytesseract

import pytesseract as tess

tess.pytesseract.tesseract\_cmd = r'Tesseract-OCR\tesseract.exe'

from PyQt6 import uic

from PyQt6.QtGui import QPixmap

from PyQt6.QtWidgets import QApplication, QFileDialog, QMessageBox

from imutils import contours

from moviepy.editor import VideoFileClip

import json

import sqlite3

import itertools

import tensorflow as tf

from skimage.feature import canny

from skimage.transform import hough\_line, hough\_line\_peaks

from skimage.transform import rotate

from skimage.color import rgb2gray

import matplotlib.gridspec as gridspec

SAVING\_FRAMES\_PER\_SECOND = 1

Form, Window = uic.loadUiType("Parking.ui")

app = QApplication([])

window = Window()

form = Form()

form.setupUi(window)

window.show()

video\_file = ''

image\_file = ''

result = ''

arr=[]

db = sqlite3.connect('server.db')

sql = db.cursor()

db = sqlite3.connect('server.db')

sql = db.cursor()

sql.execute("""CREATE TABLE IF NOT EXISTS parking (number TEXT, countvisit INT)""")

db.commit()

def format\_timedelta(td):

    result = str(td)

    try:

        result, ms = result.split(".")

    except ValueError:

        return result + ".00".replace(":", "-")

    ms = round(int(ms) / 10000)

    return f"{result}.{ms:02}".replace(":", "-")

def load\_file():

    global arr

    global db

    global sql

    db = sqlite3.connect('server.db')

    sql = db.cursor()

    sql.execute("SELECT \* FROM parking")

    arr = sql.fetchall()

    arr = list(map(lambda x: list(x), arr))

    print('load')

def save\_file():

    global arr

    global db

    global sql

    db = sqlite3.connect('server.db')

    sql = db.cursor()

    sql.execute("""CREATE TABLE IF NOT EXISTS parking (number TEXT, countvisit INT)""")

    db.commit()

    sql.execute("""DELETE FROM parking""")

    db.commit()

    for i in range(len(arr)):

        sql.execute('''INSERT INTO parking(number, countvisit) VALUES(?, ?)''', arr[i])

        db.commit()

    for value in sql.execute("SELECT \* FROM parking"):

        print(value)

    print('save')

def load():

    global video\_file

    video\_file= QFileDialog.getOpenFileName()

    path=Path(video\_file[0])

    video\_file=path.name

    print("load")

def split():

    video\_clip = VideoFileClip(video\_file)

    filename, \_ = os.path.splitext(video\_file)

    if not os.path.isdir(filename):

        os.mkdir(filename)

    saving\_frames\_per\_second = min(video\_clip.fps, SAVING\_FRAMES\_PER\_SECOND)

    step = 1 / video\_clip.fps if saving\_frames\_per\_second == 0 else 1 / saving\_frames\_per\_second

    for current\_duration in np.arange(0, video\_clip.duration, step):

        frame\_duration\_formatted = format\_timedelta(timedelta(seconds=current\_duration)).replace(":", "-")

        frame\_filename = os.path.join(filename, f"frame{frame\_duration\_formatted}.jpg")

        video\_clip.save\_frame(frame\_filename, current\_duration)

    print("split")

def check\_format(variable):

    pattern = r'^[A-Z]\d{3}[A-Z]{2}\d{3}$'

    pattern2 = r'^[A-Z]\d{3}[A-Z]{2}\d{2}$'

    if re.match(pattern, variable) or re.match(pattern2, variable):

        return True

    else:

        return False

def choose():

    global image\_file

    image\_file = QFileDialog.getOpenFileName()

    image\_file = image\_file[0]

    form.label\_6.setPixmap(QPixmap(image\_file))

    form.label\_6.setScaledContents(True)

    print("choose")

def carplate\_text():

    global image\_file

    image0 = cv2.imread(image\_file)

    image\_height, image\_width, \_ = image0.shape

    image = cv2.resize(image0, (1024, 1024))

    image = image.astype(np.float32)

    paths = './model\_resnet.tflite'

    interpreter = tf.lite.Interpreter(model\_path=paths)

    interpreter.allocate\_tensors()

    input\_details = interpreter.get\_input\_details()

    output\_details = interpreter.get\_output\_details()

    X\_data1 = np.float32(image.reshape(1, 1024, 1024, 3))

    interpreter.set\_tensor(input\_details[0]['index'], X\_data1)

    interpreter.invoke()

    detection = interpreter.get\_tensor(output\_details[0]['index'])

    net\_out\_value2 = interpreter.get\_tensor(output\_details[1]['index'])

    net\_out\_value3 = interpreter.get\_tensor(output\_details[2]['index'])

    net\_out\_value4 = interpreter.get\_tensor(output\_details[3]['index'])

    img = image0

    razmer = img.shape

    img2 = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

    img3 = img[:, :, :]

    number = 0

    while number < len(detection[0][number]) and detection[0, number, 0] > 0.9:

        number = number + 1

    box\_x = int(detection[0, number, 0] \* image\_height)

    box\_y = int(detection[0, number, 1] \* image\_width)

    box\_width = int(detection[0, number, 2] \* image\_height)

    box\_height = int(detection[0, number, 3] \* image\_width)

    cv2.rectangle(img2, (box\_y, box\_x), (box\_height, box\_width), (230, 230, 21), thickness=5)

    plt.imshow(img2)

    plt.xticks([]), plt.yticks([])  # Hides the graph ticks and x / y axis

    plt.show()

    net\_out\_value3

    image = image0[box\_x:box\_width, box\_y:box\_height, :]

    img2 = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

    grayscale = rgb2gray(image)

    edges = canny(grayscale, sigma=3.0)

    out, angles, distances = hough\_line(edges)

    h, theta, d = out, angles, distances

    angle\_step = 0.5 \* np.diff(theta).mean()

    d\_step = 0.5 \* np.diff(d).mean()

    bounds = [np.rad2deg(theta[0] - angle\_step),

              np.rad2deg(theta[-1] + angle\_step),

              d[-1] + d\_step, d[0] - d\_step]

    \_, angles\_peaks, \_ = hough\_line\_peaks(out, angles, distances, num\_peaks=20)

    angle = np.mean(np.rad2deg(angles\_peaks))

    angle

    if 0 <= angle <= 90:

        rot\_angle = angle - 90

    elif -45 <= angle < 0:

        rot\_angle = angle - 90

    elif -90 <= angle < -45:

        rot\_angle = 90 + angle

    if abs(rot\_angle) > 20:

        rot\_angle = 0

    rotated = rotate(image, rot\_angle, resize=True) \* 255

    rotated = rotated.astype(np.uint8)

    rotated1 = rotated[:, :, :]

    if rotated.shape[1] / rotated.shape[0] < 2:

        minus = np.abs(int(np.sin(np.radians(rot\_angle)) \* rotated.shape[0]))

        rotated1 = rotated[minus:-minus, :, :]

        print(minus)

    lab = cv2.cvtColor(rotated1, cv2.COLOR\_BGR2LAB)

    l, a, b = cv2.split(lab)

    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))

    cl = clahe.apply(l)

    limg = cv2.merge((cl, a, b))

    final = cv2.cvtColor(limg, cv2.COLOR\_LAB2BGR)

    letters = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'E', 'H', 'K', 'M', 'O', 'P', 'T', 'X',

               'Y']

    def decode\_batch(out):

        ret = []

        for j in range(out.shape[0]):

            out\_best = list(np.argmax(out[j, 2:], 1))

            out\_best = [k for k, g in itertools.groupby(out\_best)]

            outstr = ''

            for c in out\_best:

                if c < len(letters):

                    outstr += letters[c]

            ret.append(outstr)

        return ret

    paths = './model1\_nomer.tflite'

    interpreter = tf.lite.Interpreter(model\_path=paths)

    interpreter.allocate\_tensors()

    input\_details = interpreter.get\_input\_details()

    output\_details = interpreter.get\_output\_details()

    img = final

    img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    img = cv2.resize(img, (128, 64))

    img = img.astype(np.float32)

    img /= 255

    img1 = img.T

    img1.shape

    X\_data1 = np.float32(img1.reshape(1, 128, 64, 1))

    input\_index = (interpreter.get\_input\_details()[0]['index'])

    interpreter.set\_tensor(input\_details[0]['index'], X\_data1)

    interpreter.invoke()

    net\_out\_value = interpreter.get\_tensor(output\_details[0]['index'])

    pred\_texts = decode\_batch(net\_out\_value)

    pred\_texts

    fig = plt.figure(figsize=(10, 10))

    outer = gridspec.GridSpec(2, 1, wspace=10, hspace=0.1)

    ax1 = plt.Subplot(fig, outer[0])

    fig.add\_subplot(ax1)

    ax2 = plt.Subplot(fig, outer[1])

    fig.add\_subplot(ax2)

    return pred\_texts[0]

def recognize():

    load\_file()

    global result

    result=carplate\_text().upper()

    if result=='':

        QMessageBox.information(None, "Ошибка распознания", "НЕ РАСПОЗНАНО")

    elif check\_format(result)!=True:

        QMessageBox.information(None, "Ошибка формата", "Результат:"+result)

    else:

        form.label\_7.setText(result)

        count = 0;

        for i in range(len(arr)):

            if arr[i][0] == result:

                count += 1

        if count == 0:

            arr.append([result, 1])

        else:

            for i in range(len(arr)):

                if arr[i][0] == result:

                    arr[i][1] += 1

        for i in range(len(arr)):

            if arr[i][0] == result:

                form.label\_2.setText(str(arr[i][1]))

                if (arr[i][1]<5):

                    form.label\_5.setText('0%')

                elif (arr[i][1]>=5 and arr[i][1]<10):

                    form.label\_5.setText('5%')

                elif (arr[i][1]>=10 and arr[i][1]<20):

                    form.label\_5.setText('10%')

                else:

                    form.label\_5.setText('15%')

    save\_file()

    print("recognize")

form.pushButton.clicked.connect(load);

form.pushButton\_2.clicked.connect(split);

form.pushButton\_3.clicked.connect(choose);

form.pushButton\_4.clicked.connect(recognize);

app.exec()

Приложение Б

(справочное)

Список используемых источников

1. Habr Mask R-CNN: архитектура современной нейронной сети для сегментации объектов в изображениях [Электронный ресурс]: <https://habr.com/post/421299>
2. Habr Алгоритм распознавания номера на изображении с низкой вероятностью ошибки второго рода [Электронный ресурс]: <https://habr.com/ru/articles/501436/>
3. GitHub tflite\_avto\_num\_recognation [Электронный ресурс]: <https://github.com/sovse/tflite_avto_num_recognation/tree/main>
4. Habr Что такое свертка нейронная сеть [Электронный ресурс]: https://habr.com/post/309508
5. Автомаршал [Электронный ресурс]: уч. веб-сайт. 2014. – Режим доступа: http://avtomarshal
6. SL Трафик. Безопасный город [Электронный ресурс]: оф. веб-сайт – Режим доступа: <http://goal.ru/>
7. Evaluating image segmentation models [Электронный ресурс] / Blog Jeremy Jordan – Режим доступа: <https://www.jeremyjordan.me/evaluating-image-segmentation-models>