Международный журнал прикладных наук и технологий "Integral"

Научная статья

Original article

УДК 004.85

РАСПОЗНАВАНИЕ ДОРОЖНЫХ ЗНАКОВ С ИСПОЛЬЗОВАНИЕМ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ

RECOGNITION OF ROAD SIGNS USING CONVOLUTIONAL NEURAL NETWORK

Стахеева Алина Алексеевна, магистрант, Северный (Арктический) федеральный университет им. М. В. Ломоносова, г. Архангельск, staheeva.a@narfu.ru

Крайников Александр Николаевич, магистрант, Северный (Арктический) федеральный университет им. М. В. Ломоносова, г. Архангельск, krajnikov.a@edu.narfu.ru

Вяткин Дмитрий Андреевич, старший преподаватель, Северный (Арктический) федеральный университет им. М. В. Ломоносова, г. Архангельск, d.vyatkin@narfu.ru

Stakheeva Alyona Alekseevna, Master's student, M. V. Lomonosov Northern (Arctic) Federal University, Arkhangelsk, staheeva.a@narfu.ru

Kraynikov Alexander Nikolaevich, Master's student, M. V. Lomonosov Northern (Arctic) Federal University, Arkhangelsk, krajnikov.a@edu.narfu.ru

Vyatkin Dmitry Andreevich, Senior Lecturer, M. V. Lomonosov Northern (Arctic) Federal University, Arkhangelsk, d.vyatkin@narfu.ru

Аннотация. Одной из важнейших задач автопилотирования автомобиля является распознавание дорожных знаков. В данной статье рассматривается возможность распознавания знаков ПДД на изображениях с помощью сверточных нейронных сетей. Для этого в данной работе был написан код для модели определения дорожных знаков на языке Python, а затем эта модель была обучена на большом наборе данных с изображениями различных дорожных знаков. Также был написан код для применения уже обученной модели к входным изображениям и вывода результата на экран. На основании тестирования модели на изображениях различной сложности был сделан вывод о качестве распознавания дорожных знаков обученной моделью.

Annotation. One of the most important tasks of autopiloting a car is the recognition of road signs. This article discusses the possibility of recognizing traffic signs in images using convolutional neural networks. To do this, in this paper, code was written for a model for determining road signs in Python, and then this model was tested on a large data set with images of various road signs. Code was also written to apply the already trained model to the input image and output the result to the screen. Based on testing the model on images of varying complexity, a conclusion was made about the quality of recognition of road signs by the trained model.

Ключевые слова: автопилотирование автомобиля, дорожные знаки, сверточные нейронные сети, машинное обучение, Python, распознавание на изображении.

Keywords: car autopiloting, road signs, convolutional neural networks, machine learning, Python, image recognition.

Постановка задачи и входные данные

Необходимо разработать программу на языке Python для определения дорожного знака по изображению. В качестве входных данных был использован общедоступный набор данных с изображениями 43 различных знаков дорожного движения. Данная задача будет решена с помощью различных методов компьютерного зрения и сверточных нейронных сетей. Так как нам необходимо

определять класс дорожного знака, то будет реализована именно задача классификации.

Рассмотрим данные, которые будут использоваться для обучения. Исходный набор данных представляет собой структурированный датасет с изображениями различных знаков правил дорожного движения. В данном датасете имеется более 35000 изображений 43 различных видов дорожных знаков. Пример изображения из этого набора данных показан на рисунке 1.

Рисунок 1 – Пример изображения из набора данных

В качестве входных данных кроме датасета нам также понадобится файл формата «csv», в котором прописаны номера классов для каждого знака и их названия соответственно. Как выглядит данный файл показано на рисунке 2.

labels.csv

```
ClassId,Name
0,Speed limit (20km/h)
1,Speed limit (30km/h)
2,Speed limit (50km/h)
3,Speed limit (60km/h)
4,Speed limit (70km/h)
5,Speed limit (80km/h)
6,End of speed limit (80km/h)
7,Speed limit (100km/h)
8,Speed limit (120km/h)
```

Рисунок 2 – Файл с номерами классов и названиями знаков

Проверять качество обученной модели и точность определения ею знаков ПДД будем на случайных снимках из интернета.

Для того чтобы классификация дорожных знаков на входных изображениях стала возможной нужно решить несколько задач. Первоочередная задача заключается в создании и обучении самой модели. Далее будет необходимо написать программу, которая обрабатывает входные изображения так, чтобы обученная модель могла с ними работать. А также надо выводить

получившийся результат в виде исходного изображения, на котором написано название дорожного знака и процентная вероятность того, что это он.

Обучение модели

Для начала необходимо создать проект и загрузить в него входной набор данных с изображениями и файл формата «csv», в котором прописаны номера классов для каждого знака и их названия соответственно. У нас будет 2 файла с кодом, один для обучения модели, а второй для настройки входного изображения, тестирования уже обученной модели и вывода результата на экран. Далее рассмотрим создание и обучение модели. Для начала подключим все необходимые библиотеки (листинг 1).

Листинг 1 – Подключение библиотек для первого файла

```
# Подключение библиотек:
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import pandas as pd
import random
import cv2
import os
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.utils.np_utils import to_categorical
from keras.layers import Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
```

Далее опишем некоторые важные параметры (листинг 2). В параметр «path» записываем имеющийся набор данных с изображениями, в «labelFile» – файл с идентификационным номером класса и его названием. Далее прописываем числовые данные для самого процесса обучения, указываем размер изображений и процентную составляющую тестовой и валидационной выборок от всего датасета.

Листинг 2 – Описание параметров

```
path = 'myData' # набор данных с изображениями
labelFile = 'labels.csv' # файл с идентификационным номером класса знака
и его названием
batch_size_val = 20 # размер батчей
```

```
steps_per_epoch_val = 1000 # количество итераций
epochs_val = 16 # количество эпох
imageDimesions = (32,32,3) # размер изображения
testRatio = 0.2 # процент тестовой выборки от всего количества данных
validationRatio = 0.2 # процент валидационной выборки от всего количества
данных
```

Далее нужно импортировать изображения из всех папок и поместить их в один массив «images», а идентификатор класса для каждого изображения запишем в массив «classNo» (листинг 3).

Листинг 3 – Импортирование изображений

```
# Импортирование набора данных с изображениями
count = 0
images = []
classNo = []
myList = os.listdir(path)
print('Обнаружено классов:', len(myList))
noOfClasses=len(myList)
for x in range (0,len(myList)):
    myPicList = os.listdir(path+"/"+str(count))
    for y in myPicList:
        curImg = cv2.imread(path+"/"+str(count)+"/"+y)
        images.append(curImg)
        classNo.append(count)
    print(count, end =" ")
    count +=1
print(" ")
images = np.array(images)
classNo = np.array(classNo)
```

Далее разделим имеющийся набор данных на обучающую, валидационную и тестовую выборки (листинг 4). «X_train» в данном случае это массив изображений для обучения, а «y_train» это соответствующий идентификатор класса данного изображения.

Листинг 4 – Разбиение набора данных на выборки

```
# Разбиение набора данных на обучающую, валидационную и тестовую выборки
с указанием их процентного соотношения
X_train, X_test, y_train, y_test = train_test_split(images, classNo,
test_size=testRatio)
X_train, X_validation, y_train, y_validation = train_test_split(X_train,
y_train, test_size=validationRatio)
```

Выведем размеры обучающей, валидационной и тестовой выборок. На всякий случай сделаем проверку для каждой выборки на соответствие количества изображений и классов для них. А также сравним размеры

изображений в наборе данных с тем размером изображений, который будет у обучаемой модели. Все это представлено в листинге 5.

Листинг 5 – Проверка изображений

```
# Вывод размеров обучающей, валидационной и тестов ой выборок
print("Data Shapes:")
print("Train ", end = "")
print(X_train.shape,y_train.shape)
print("Validation ", end = "")
print(X_validation.shape,y_validation.shape)
print("Test ", end = "")
print(X_test.shape,y_test.shape)
# Проверка изображений для каждой выборки на соответствие количества
изображений и классов для них
assert(X_train.shape[0]==y_train.shape[0])
assert(X_validation.shape[0]==y_validation.shape[0])
assert(X_test.shape[0]==y_test.shape[0])
# Проверка размеров изображений в наборе данных с тем размером
изображений, который будет у обучаемой модели
assert(X_train.shape[1:]==(imageDimesions))
assert(X_validation.shape[1:]==(imageDimesions))
assert(X_test.shape[1:]==(imageDimesions))
```

Теперь прочитаем файл формата «csv», в котором прописаны номера классов для каждого дорожного знака и их названия соответственно (листинг 6).

Листинг 6 – Чтение «csv» файла

```
# Чтение файла с номерами классов для каждого дорожного знака и их
названиями соответственно
data = pd.read_csv(labelFile)
print("Data shape ", data.shape, type(data))
```

На данном этапе в консоль выводятся следующие результаты как показано на рисунке 3.

```
Обнаружено классов: 43
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
Data Shapes:
Train (22271, 32, 32, 3) (22271,)
Validation (5568, 32, 32, 3) (5568,)
Test (6960, 32, 32, 3) (6960,)
Data shape  (43, 2) <class 'pandas.core.frame.DataFrame'>
```

Рисунок 3 – Выведенные размеры

Предварительная обработка изображений представлена в листинге 7. Для начала преобразуем изображение в оттенки серого, стандартизируем контрастность и затем нормализуем значения так, чтобы вместо от 0 до 255 было от 0 до 1.

Листинг 7 – Функция обработки изображений

```
# Функция обработки изображений
def preprocessing(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) # превращение изображения
в серое
    img =cv2.equalizeHist(img) # выравнивание контрастности изображения
    img = img/255 # нормализуем значения так, чтобы вместо от 0 до 255
было от 0 до 1
    return img
```

Добавим изображениям глубину, аугментируем их и отобразим примеры таких дополнительных изображений (листинг 8).

Листинг 8 – Добавление глубины изображениям, их аугментация и отображение нескольких полученных изображений

```
# Аугментация изображений
X_train=np.array(list(map(preprocessing,X_train)))
X_validation=np.array(list(map(preprocessing,X_validation)))
X_test=np.array(list(map(preprocessing,X_test)))
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1],
X_train.shape[2], 1)
X_validation = X_validation.reshape(X_validation.shape[0],
X_validation.shape[1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1],
X_test.shape[2], 1)
dataGen = ImageDataGenerator(width_shift_range = 0.1, height_shift_range
= 0.1, zoom_range = 0.2, shear_range = 0.1, rotation_range = 10)
dataGen.fit(X_train)
batches = dataGen.flow(X_train, y_train,batch_size=20)
X_batch, y_batch = next(batches)
```

Далее преобразуем наши выборки с номерами классов в двоичные матрицы классов (листинг 9).

Листинг 9 – Преобразование выборок с номерами классов

```
# Преобразуем вектор класса выборок в двоичную матрицу классов
y_train = to_categorical(y_train,noOfClasses)
y_validation = to_categorical(y_validation,noOfClasses)
y_test = to_categorical(y_test,noOfClasses)
```

Примеры некоторых изображений после аугментации показаны на рисунке 4.

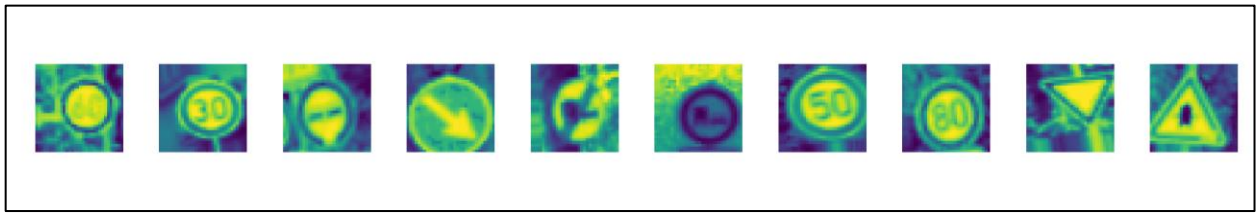

Рисунок 4 – Изображения после аугментации

Напишем функцию создания модели для сверточной нейронной сети (листинг 10).

Листинг 10 – Создание модели сверточной нейронной сети

```
# Модель сверточной нейронной сети
def myModel():
    no_Of_Filters=60
    size_of_Filter=(5,5)
    size_of_Filter2=(3,3)
    size_of_pool=(2,2)
    no_Of_Nodes = 500
    model= Sequential()

model.add((Conv2D(no_Of_Filters,size_of_Filter,input_shape=(imageDimesion
s[0],imageDimesions[1],1),activation='relu')))
    model.add((Conv2D(no_Of_Filters, size_of_Filter, activation='relu')))
    model.add(MaxPooling2D(pool_size=size_of_pool))
    model.add((Conv2D(no_Of_Filters//2,
size_of_Filter2,activation='relu')))
    model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2,
activation='relu')))
    model.add(MaxPooling2D(pool_size=size_of_pool))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(no_Of_Nodes,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(noOfClasses,activation='softmax'))

model.compile(Adam(lr=0.001),loss='categorical_crossentropy',metrics=['ac
curacy'])
    return model
```

Теперь приступим непосредственно к обучению нашей модели (листинг 11).

Листинг 11 – Обучение модели

```
# Обучение модели
model = myModel()
print(model.summary())
history =
model.fit_generator(dataGen.flow(X_train,y_train,batch_size=batch_size_va
l), steps_per_epoch=steps_per_epoch_val, epochs=epochs_val,
validation_data=(X_validation,y_validation), shuffle=1)
```

Опытным путем было выяснено лучшее значение коэффициента «epochs_val», которое отвечает за количество итераций в обучении модели. Для данного набора данных было решено остановиться на «epochs_val» равному 16. В итоге обученная модель имеет следующие значения метрик: «Loss = 0.049», «Accuracy = 0.985».

Также для наглядного понимания успешности обучения модели для определения класса дорожного знака отобразим графики точности обучения и потерь для модели в зависимости от номера эпохи (листинг 12). График точности обучения модели показан на рисунке 5, а график потерь – на рисунке 6.

Листинг 12 – Графики с результатами обучения модели

```
# Графики результатов обучения модели
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training','validation'])
plt.title('acurracy')
plt.xlabel('epoch')
plt.show()
score = model.evaluate(X_test,y_test,verbose=0)
print('Test Loss:',score[0])
print('Test Accuracy:',score[1])
```

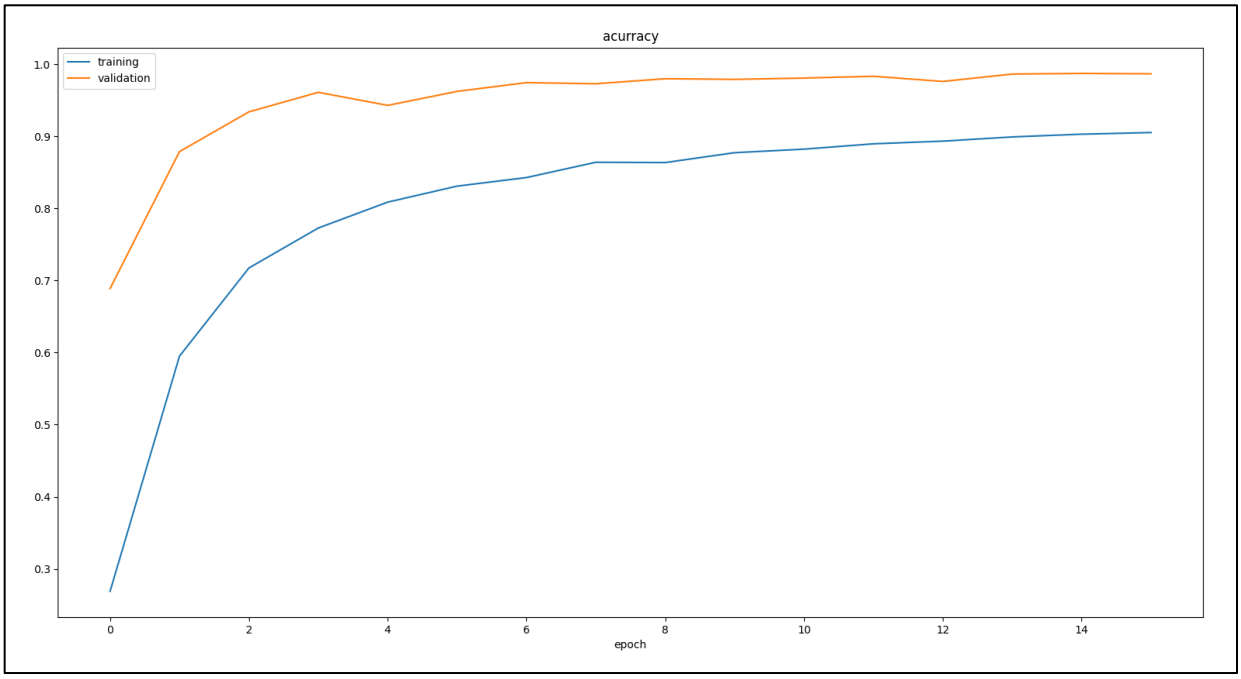

Рисунок 5 – Точность обучения модели в зависимости от номера эпохи

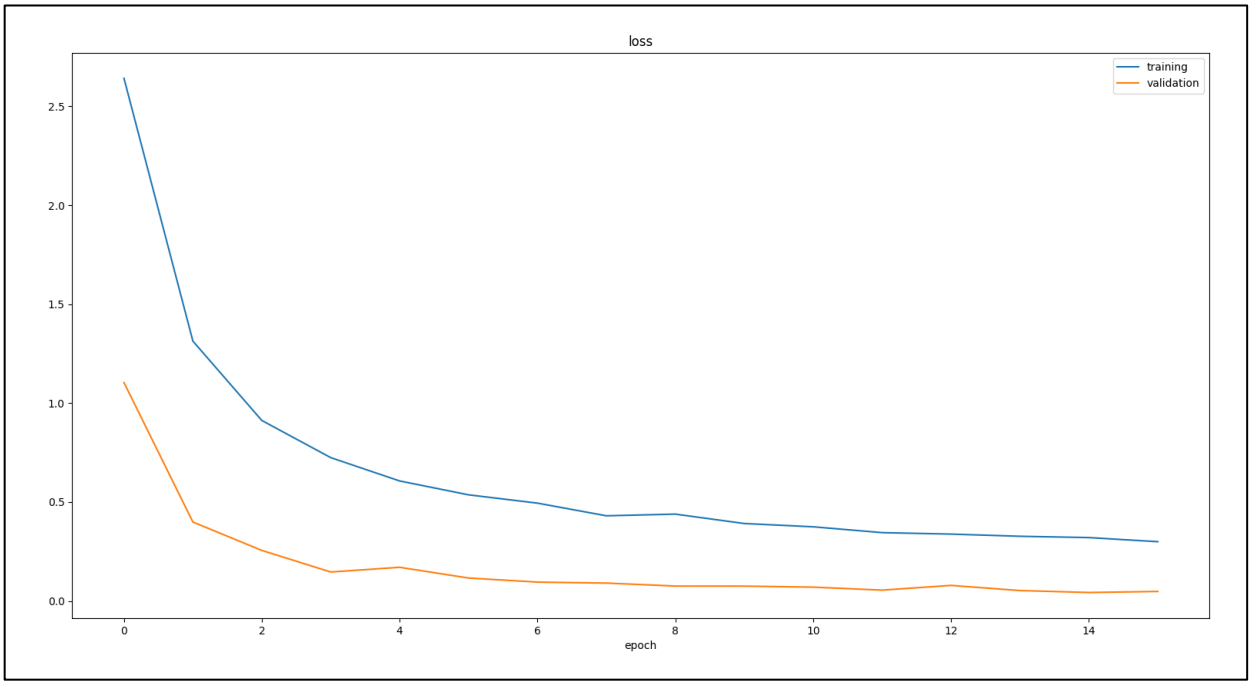

Рисунок 6 – Потери при обучении модели в зависимости от номера эпохи

Осталось только сохранить нашу обученную модель (листинг 13).

Листинг 13 – Сохранение обученной модели

```
# Сохранение модели
model.save('model_v16')
```

Теперь у нас есть модель для определения дорожных знаков, которая была обучена на большом наборе данных.

Определение дорожных знаков с помощью обученной модели

Создадим второй файл с кодом для настройки входного изображения, тестирования уже обученной модели и вывода результата на экран. Для начала подключим все необходимые библиотеки (листинг 14).

Листинг 14 – Подключение библиотек для второго файла

```
# Подключение библиотек
from tensorflow import keras
import cv2
import numpy as np
```

Добавим некоторые важные параметры (листинг 15). Параметр «threshold» обозначает порог точности обнаружения дорожного знака, при котором знак будет считаться определенным правильно. Параметр «font» нужен для возможности добавления текста прямо на изображение.

Листинг 15 – Добавление параметров

```
threshold = 0.6 # порог точности обнаружения дорожного знака для
срабатывания
font = cv2.FONT_HERSHEY_SIMPLEX # переменная для добавления текста на
изображение
```

Необходимо загрузить обученную модель (листинг 16).

Листинг 16 – Загрузка обученной модели

```
# Загрузка обученной модели
model = keras.models.load_model('model_v16')
```

Далее напишем функцию, чтобы при вызове которой по идентификационному номеру выводилось само название дорожного знака (листинг 17).

Листинг 17 – Функция для вывода названия дорожного знака по его идентификационному номеру

```
# Вывод названия дорожного знака по его идентификационному номеру
def getClassName(classNo):
    if   classNo == 0: return 'Speed limit (20km/h)'
    elif classNo == 1: return 'Speed limit (30km/h)'
    elif classNo == 2: return 'Speed limit (50km/h)'
    elif classNo == 3: return 'Speed limit (60km/h)'
    elif classNo == 4: return 'Speed limit (70km/h)'
    elif classNo == 5: return 'Speed limit (80km/h)'
    elif classNo == 6: return 'End of speed limit (80km/h)'
    elif classNo == 7: return 'Speed limit (100km/h)'
    elif classNo == 8: return 'Speed limit (120km/h)'
    elif classNo == 9: return 'No passing'
    elif classNo == 10: return 'No passing for vechiles over 3.5 metric
tons'
    elif classNo == 11: return 'Right-of-way at the next intersection'
    elif classNo == 12: return 'Priority road'
    elif classNo == 13: return 'Yield'
    elif classNo == 14: return 'Stop'
    elif classNo == 15: return 'No vechiles'
    elif classNo == 16: return 'Vechiles over 3.5 metric tons prohibited'
    elif classNo == 17: return 'No entry'
    elif classNo == 18: return 'General caution'
    elif classNo == 19: return 'Dangerous curve to the left'
    elif classNo == 20: return 'Dangerous curve to the right'
    elif classNo == 21: return 'Double curve'
    elif classNo == 22: return 'Bumpy road'
    elif classNo == 23: return 'Slippery road'
    elif classNo == 24: return 'Road narrows on the right'
    elif classNo == 25: return 'Road work'
    elif classNo == 26: return 'Traffic signals'
    elif classNo == 27: return 'Pedestrians'
    elif classNo == 28: return 'Children crossing'
    elif classNo == 29: return 'Bicycles crossing'
```

```
    elif classNo == 30: return 'Beware of ice/snow'
    elif classNo == 31: return 'Wild animals crossing'
    elif classNo == 32: return 'End of all speed and passing limits'
    elif classNo == 33: return 'Turn right ahead'
    elif classNo == 34: return 'Turn left ahead'
    elif classNo == 35: return 'Ahead only'
    elif classNo == 36: return 'Go straight or right'
    elif classNo == 37: return 'Go straight or left'
    elif classNo == 38: return 'Keep right'
    elif classNo == 39: return 'Keep left'
    elif classNo == 40: return 'Roundabout mandatory'
    elif classNo == 41: return 'End of no passing'
    elif classNo == 42: return 'End of no passing by vechiles over 3.5
metric tons'
```

В листинге 18 представлен код, который выполняет следующие действия:

– чтение входного изображения;

– обработка входного изображения;

– добавление на входное изображение таких слов как «Class» и «Probability»;

– предсказание отношения дорожного знака на входном изображении к каждому классу;

– поиск максимального предсказанного результата в массиве;

– поиск индекса максимального элемента в массиве (класса дорожного знака);

– вывод максимально возможного предсказанного результата и его индекса в консоль;

– сравнение порогового значения для срабатывания с предсказанным;

– в случае успешного обнаружения знака: добавление на входное изображение номера класса и его название, а также процентную вероятность предсказания знака;

– вывод на экран входного изображения с написанными на нем результатами;

– закрытие окна с результатом с помощью нажатия клавиши «esc».

Листинг 18 – Код для работы со входным изображением, предсказания дорожного знака и вывода результата на экран

```
# Основная часть программы
while True:
    imgOriginal = cv2.imread('60gg.jpg') # загрузка входного изображения,
на котором необходимо определить знак
    img = np.asarray(imgOriginal) # превращение изображения в массив
    img = cv2.resize(img, (32, 32)) # изменение размера изображения
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # превращение изображения
в серое
    img = cv2.equalizeHist(img) # выравнивание контрастности изображения
    img = img/255 # нормализуем значения так, чтобы вместо от 0 до 255
было от 0 до 1
    img = img.reshape(1, 32, 32, 1) # изменение формы массива
    cv2.putText(imgOriginal, "Class:", (20, 35), font, 0.75, (0, 0, 255),
2, cv2.LINE_AA) # добавление на исходное изображение слова "Класс"
    cv2.putText(imgOriginal, "Probability:", (20, 75), font, 0.75, (255,
0, 0), 2, cv2.LINE_AA) # добавление на исходное изображение слова
"Вероятность"
    predictions = model.predict(img) # применение обученной модели к
изображению
    probabilityValue = np.amax(predictions) # поиск максимальной точности
отношения дорожного знака к одному из классов
    # Поиск максимально возможного предсказанного результата и его
индекса
    max_el = 0
    ind_max = 0
    for i in range(0, 43):
        if predictions[0,i] > max_el:
            max_el = predictions[0,i]
            ind_max = i
    assert(probabilityValue==max_el)
    print('Максимальный элемент:', max_el, '\nИндекс максимального
элемента:', ind_max)
    if probabilityValue > threshold: # сравнение предсказанной точности
отбаружения знака с пороговым значением для срабатывания
        cv2.putText(imgOriginal, str(ind_max)+" =
"+str(getClassName(ind_max)), (100,35), font, 0.75, (0,0,255), 2,
cv2.LINE_AA) # добавление на исходное изображение номера класса и его
названия
        cv2.putText(imgOriginal, str(round(max_el*100,2))+"%", (160,75),
font, 0.75, (255,0,0), 2, cv2.LINE_AA) # добавление на исходное
изображение вероятность определения знака в процентах
    cv2.imshow("Result image", imgOriginal) # вывод на экран входного
изображения с написанными на нем результатами
    # Закрытие окна с результатом с помощью нажатия клавиши «esc»
    if cv2.waitKey(0) & cv2.waitKey(10) == 27:
        break
cv2.destroyAllWindows()
```

При распознавания дорожного знака с помощью обученной модели представлен на рисунке 7.

Рисунок 7 – Пример распознавания дорожного знака

Тестирование обученной модели

Проверим нашу обученную модель на различных изображениях, чтобы оценить ее работоспособность.

Обученная модель смогла со 100% вероятностью правильно определить дорожный знак на изображениях относительно хорошего качества, где знак занимает почти все пространство. С изображениями, где знак занимает почти все пространство изображения, но качество гораздо хуже, модель справилась так же хорошо, однако понизилась вероятность определения. То есть можно сделать вывод, что понижение качества изображение не играет большой роли для определения знака, однако пока это качество не станет критически плохим.

С фотографиями, где знак отдален от наблюдателя, модель справилась гораздо хуже. Обученная модель успешно определила только 33% знаков, остальные же либо определены неправильно, либо вообще не определились. Этот результат вполне логичен, так как изображения в наборе данных, по которым обучалась модель, обрезаны так, что на них виден только сам знак, и поэтому модели сложно работать с изображениями, где много лишнего. В дальнейшем можно попытаться исправить это тем, что на изображении будет находиться дорожный знак, затем изображение нужно будет обрезать так, чтобы

обнаруженный знак занимал почти все пространство изображения, и только потом применять нашу обученную модель.

Для изображений, на которых дорожный знак имеет некий разворот или угол наклона, результаты определения класса с помощью модели средние. Обученная модель успешно определила только 60% знаков, остальные же либо определены неправильно, либо вообще не определились.

После просмотра результатов определения знаков ПДД для изображений с плохими погодными условиями, в темное время суток и с небольшой испорченностью знака можно сделать вывод, что если данные условия проявляются лишь немного, то модель вполне успешно справляется со своей задачей. Однако если из-за таких условий скрыты почти все очертания знака или потеряны его основные элементы, то определить знак будет невозможно или модель это сделает с очень маленькой вероятностью.

Модель плохо работает с изображениями, на которых находятся сразу несколько знаков, а все дело в том, что в коде прописан поиск только одного наибольшего значения массива, поэтому данная модель может предсказать один наиболее явный дорожный знак. Для исправления этой ситуации нужно перед задачей классификации сначала решать задачу определения и уже по всем найденным знакам ПДД определять их принадлежность.

Со знаками, на которых что-то нарисовано, модель ведет себя по-разному в зависимости от уровня изменения знака. То есть наша модель может справляться только с небольшими изменениями самих знаков.

После всех протестированных изображений можно сделать вывод, что данная модель успешно определяет знаки ПДД, однако сильнее всего на ее работоспособность влияют наличие большого пространства вокруг знака или других знаков, очень плохие погодные условия, которые не позволяют рассмотреть очертания знака и какие-нибудь рисунки на самом знаке.

Однако есть еще такой фактор, как величина обучающей выборки для каждого из знаков. Для изображений, у которых была относительно небольшая

выборка, даже при хорошем входном изображении вероятность их определения не высокая, так что количество изображений в наборе данных тоже играет большую роль.

Заключение

В ходе работы на основе большого набора данных была создана и обучена сверточная нейронная сеть для классификации дорожных знаков на входных изображениях. В качестве работы с исходными данными была выполнена аугментация изображений, их преобразование в градации серого и выравнивание гистограммы. Точность определения итоговой модели на тестовой части набора данных в среднем равна 0.985, что является высоким показателем. Так что можно сделать вывод, что сверточные нейронные сети подходят для распознавания и классификации дорожных знаков изображениях.

Слабым местом обученной модели являются изображения с большим пространством вокруг самого дорожного знака, наличие нескольких знаков, плохие погодные условия, дополнительные элементы на самом знаке и знаки, имеющие менее 400 исходных экземпляров в наборе данных. Однако проблемы с отдаленностью дорожного знака от наблюдателя, наличие нескольких элементов на изображении и малое количество исходных изображений для обучения вполне решаемы с помощью написания дополнительных функций.

Литература

1. Traffic Sign Classification [Электронный ресурс]: [офиц. сайт] / CV Zone – Электрон. дан. – Режим доступа: https://www.computervision.zone/courses/traffic-sign-classification/, свободный (дата обращения: 09.08.2023). – Загл. с экрана.
2. Archive Meta Data [Электронный ресурс]: [офиц. сайт] / Electronic research data archive – Электрон. дан. – Режим доступа: https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html, свободный (дата обращения: 09.08.2023). – Загл. с экрана.

3. Как работает сверточная нейронная сеть (CNN) [Электронный ресурс]: [офиц. сайт] / Neurohive – Электрон. дан. – Режим доступа: https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/, свободный (дата обращения: 09.08.2023). – Загл. с экрана.
4. Расширяем датасет с помощью Data Augmentation в Tensorflow [Электронный ресурс]: [офиц. сайт] / Python school – Электрон. дан. – Режим доступа: https://python-school.ru/blog/data-augmentation/, свободный (дата обращения: 09.08.2023). – Загл. с экрана.
5. Обзор Keras для TensorFlow [Электронный ресурс]: [офиц. сайт] / Хабр – Электрон. дан. – Режим доступа: https://habr.com/ru/articles/482126/, свободный (дата обращения: 09.08.2023). – Загл. с экрана.
6. Слои Keras: параметры и свойства keras 5 [Электронный ресурс]: [офиц. сайт] / Python Ru – Электрон. дан. – Режим доступа: https://pythonru.com/biblioteki/sloi-keras-parametry-i-svojstva-keras-5, свободный (дата обращения: 09.08.2023). – Загл. с экрана.
7. Шпаргалка по OpenCV — Python [Электронный ресурс]: [офиц. сайт] / Tproger – Электрон. дан. – Режим доступа: https://tproger.ru/translations/opencv-python-guide/, свободный (дата обращения: 09.08.2023). – Загл. с экрана.
8. NumPy, часть 1: начало работы [Электронный ресурс]: [офиц. сайт] / Python 3 для начинающих – Электрон. дан. – Режим доступа: https://pythonworld.ru/numpy/1.html, свободный (дата обращения: 09.08.2023). – Загл. с экрана.
9. Split Your Dataset With scikit-learn's train_test_split() [Электронный ресурс]: [офиц. сайт] / Real Python – Электрон. дан. – Режим доступа: https://realpython.com/train-test-split-python-data/#application-of-train_test_split, свободный (дата обращения: 09.08.2023). – Загл. с экрана.
10. ML_CNN — about the batch size , epochs , steps , steps_per_epoch , iteration [Электронный ресурс]: [офиц. сайт] / Medium – Электрон. дан. – Режим

доступа: https://medium.com/@pinyuanhuang/ml-cnn-about-the-batch-size-epochs-steps-steps-per-epoch-iteration-6d6f9b4621f7, свободный (дата обращения: 09.08.2023). – Загл. с экрана.

References

1. Traffic Sign Classification [Electronic resource]: [official. website] / CV Zone – Electron. dan. – Access mode: https://www.computervision.zone/courses/traffic-sign-classification /, free (accessed: 09.08.2023). – Blank from the screen.
2. Archive Meta Data [Electronic resource]: [official. website] / Electronic research data archive – Electron. dan. – Access mode: https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html , free (accessed: 09.08.2023). – Blank from the screen.
3. How the convolutional neural network (CNN) works [Electronic resource]: [ofic. website] / Neurohive – Electron. dan. – Access mode: https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set /, free (accessed: 09.08.2023). – Blank from the screen.
4. Expanding the dataset using Data Augmentation in Tensorflow [Electronic resource]: [ofic. website] / Python school – Electron. dan. – Access mode: https://python-school.ru/blog/data-augmentation /, free (accessed: 09.08.2023). – Blank from the screen.
5. Review of Keras for TensorFlow [Electronic resource]: [official. website] / Habr – Electron. dan. – Access mode: https://habr.com/ru/articles/482126 /, free (accessed: 09.08.2023). – Blank from the screen.
6. Keras layers: parameters and properties of keras 5 [Electronic resource]: [ofic. website] / Python Ru – Electron. dan. – Access mode: https://pythonru.com/biblioteki/sloi-keras-parametry-i-svojstva-keras-5 , free (accessed: 09.08.2023). – Blank from the screen.
7. Cheat sheet on OpenCV — Python [Electronic resource]: [official. website] / Tproger – Electron. dan. – Access mode: https://tproger.ru/translations/opencv-python-guide /, free (accessed: 09.08.2023). – Blank from the screen.

8. NumPy, part 1: getting started [Electronic resource]: [ofic. website] / Python 3 for beginners – Electron. dan. – Access mode: https://pythonworld.ru/numpy/1.html , free (accessed: 09.08.2023). – Blank from the screen.
9. Split Your Dataset With scikit-learn's train_test_split() [Electronic resource]: [ofic. website] / Real Python – Electron. dan. – Access mode: https://realpython.com/train-test-split-python-data/#application-of-train_test_split , free (accessed: 09.08.2023). – Blank from the screen.
10. ML_CNN — about the batch size , epochs , steps , steps_per_epoch , iteration [Electronic resource]: [official. website] / Medium – Electron. dan. – Access mode: https://medium.com/@pinyuanhuang/ml-cnn-about-the-batch-size-epochs-steps-steps-per-epoch-iteration-6d6f9b4621f7 , free (accessed: 09.08.2023). – Blank from the screen.

Для цитирования: Стахеева А.А., Крайников А.Н., Вяткин Д.А. Распознавание дорожных знаков с использованием сверточной нейронной сети// Научный сетевой журнал «Столыпинский вестник» №8/2023.