УДК 004.869

***Д.А. Кудряшов, Д.Н. Ногтев, А.А. Суконщиков***
*Вологодский государственный университет*

# РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ БУТЫЛОК В ФАНДОМАТЕ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

В статье рассматриваются построение системы распознавания бутылок в фандомате на основе сверточной нейронной сети (СНИ). Для программной реализации СНИ используем язык Python, библиотеки OpenCV и модели СНИ EfficientNet-B0, EfficientNet-B3. Для обучения использовались 1650 изображений из Kaggle.com. Точность распознавания на обученной СНИ не менее 83 %.

Сверточная нейронная сеть, фандомат, бутылки, программа, система распознавания.

В настоящее время возникает ряд проблем при приеме бутылок в фандомате: в основном определяется бутылка по штрих-коду. В случае его отсутствия или повреждения бутылка не принимается. В данной статье рассмотрим решение данной проблемы. Сегодня разработан ряд методов и систем поиска и распознавания объектов на видеоизображениях, в которых широко применены СНИ. Поэтому решаем задачу на основе СНИ [1–3].

Для разработки программы реализующей распознавание бутылок необходимо выбрать язык программирования. Можно использовать Python, C++, Java, Matlab. Для реализации системы распознавания пластиковых бутылок на основе нейронной сети был выбран язык Python. Данный язык является интерпретируемым, простым в освоении, имеет большое количество библиотек, поддерживается на любой платформе, имеет довольно простой синтаксис. Программы, написанные при использовании языка Python, обладают высокой скоростью выполнения.

Этапом, следующим после выбора языка программирования, является подключение библиотек, основными функциями которых является работа с изображениями. Библиотекой, которая наиболее удовлетворяет требования данного проекта, является OpenCV. Отличительными особенностями данной библиотеки являются функции: перевод в оттенки серого, в черно-белое, а также определение контуров.

В данной работе используются модели нейронных сетей EfficientNet-B0 и EfficientNet-B3. Они относятся к одной семье архитектур, а отличаются только глубиной (количество слоев) и шириной (количество каналов в каждом слое). Для каждой модели использовалось обучение с сегментированными и с обычными изображениями. Из моделей была взята только сверточная часть, выделяющая характерные признаки.

В качестве данных были выбраны 10 видов бутылок: прозрачные пустые бутылки маленькие (до 0,5 л); прозрачные пустые бутылки средние (до 2,5 л); прозрачные пустые бутылки большие (до 6 л); бутылки мятые большие; бутылки мятые маленькие; бутылки с жидкостью; цветные пустые бутылки маленькие (до 0,5 л); цветные пустые бутылки средние (до 2,5 л); стеклянные бутылки; алюминиевые банки; отсутствие бутылки.

Исходными данными стали датасеты из Kaggle.com. Объем выборки – 1650 изображений. Для подготовки данных была разработана отдельная программа, которая перемещает изображения, указанные в файле sample_labels.csv, в отдельный каталог.

Основной фрагмент программы, распределяющей изображения, выглядит следующим образом:

```
for (int i = 0; i < mass_length; i++) {
name[i] = Mass[i].Substring(0, 16);
class_n[i] = Mass[i].Substring(17, 2);
Console.WriteLine(name[i] + " " + class_n[i]);
Directory.CreateDirectory("folder\\"+class_n[i]);
File.Copy("images\\" + name[i], "folder\\" + class_n[i]+"\\"+name[i], true);
}
```

Массив name[i] хранит названия файлов, а массив class_n[i] – названия группы, которые используются в архиве.

Метод Directory.CreateDirectory("folder\\"+class_n[i]) создает каталог с названием указанным в массиве class_n[i]. Метод File.Copy("images\\" + name[i], "folder\\" + class_n[i]+"\\"+name[i], true) сохраняет изображения в соответствующие каталоги.

Для обучения модели используются генераторы:

train_generator – генератор для обучения;

validation_data – генератор для проверки.

Генераторы нужны для преобразования изображений в вид, пригодный для обучения нейронной сети.

Метод ImageDataGenerator(rescale=1./255) делит каждый элемент изображения на 255. Все элементы изображения будут находиться в диапазоне от 0 до 255, такое преобразование необходимо для обучения.

В генераторах используется метод flow_from_ directory – поток данных из каталога, в котором лежат файлы: tain_dir – каталог, в котором лежат файлы; target_size – размер изображения; batch_size – размер выборки; class_mode – режим классов.

Функция модели показана ниже:

```
model_name='EfficientNetB3'
base_model=tf.keras.applications.EfficientNetB3 (include_top=False,
weights="imagenet",input_shape=img_shape, pooling='max')
x=base_model.output
x=keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
x = Dense(256, kernel_regularizer = regularizers.l2(l = 0.016),activity_regularizer=regularizers.l1(0.006),
bias_regularizer=regularizers.l1(0.006)
,activation='relu')(x)
x=Dropout (rate=.45, seed=123)(x)
output=Dense(class_count, activation='softmax')(x)
model=Model(inputs=base_model.input, outputs=output)
model.compile(Adamax(learning_rate=.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Библиотеки, которые использовались для работы: tensorflow и sklearn. Для работы с данными использовались библиотеки: pandas и numpy. Для работы с графическими данными использовались библиотеки: matplotlib.pyplot и seaborn. Пример тренировочной выборки показан на рисунке 1.

Тип выбранной нейронной сети: EfficientNet.

EfficientNet – это архитектура сверточной нейронной сети и метод масштабирования, который равномерно масштабирует все измерения глубины/ ширины/разрешения с использованием составного коэффициента.

Размер входной формы EfficientNetB3 составил 150×150.

Количество каналов составило 3. Глубина нейронной сети 4.

Тип весов – 'imagenet'.

Размер батча – количество строк данных, подающихся на вход модели за одну итерацию. С помощью него определяется количество входных данных для расчета градиента, необходимого для обратного распространения ошибки. В нашем случае он составил 20.

Число эпох. Одна эпоха – это полное прохождение всех обучающих данных через нейронную сеть один раз. В нашем случае 40.

Длина последовательности 0,016.

Структура алгоритма показана на рисунке 2.

*Рис. 1. Тренировочная выборка*

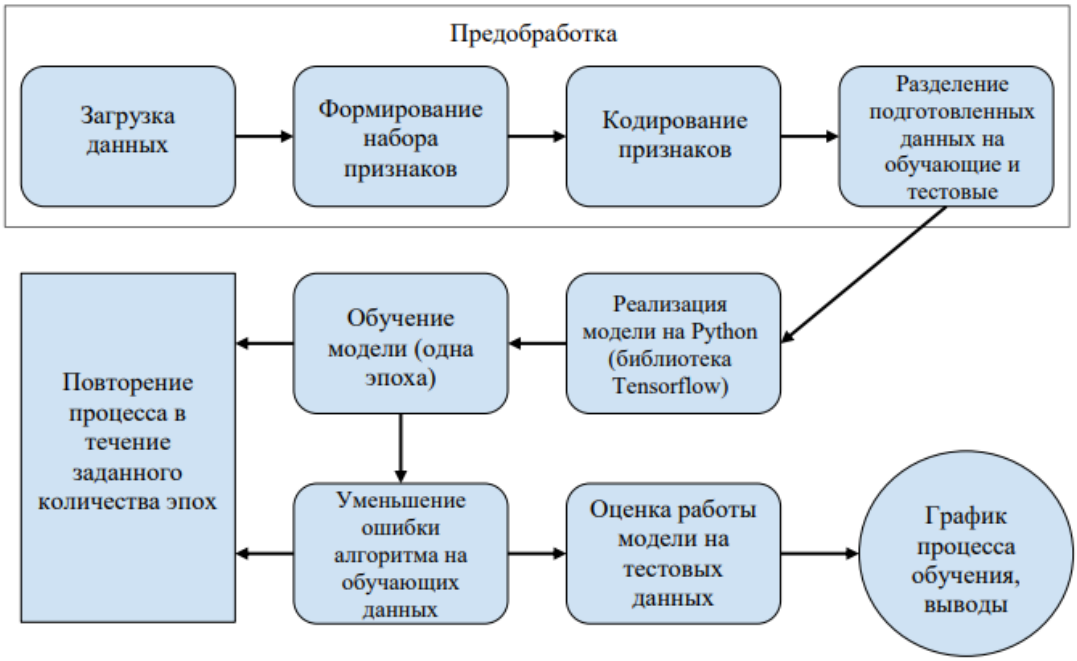


*Рис. 2. Структура алгоритма*

Для оценки качества модели использовались точность, F1, Jaccard score. Точность зависит от количества всех изображений, по которым классификатор принял правильное решение.

Таблица 1

**Результаты тестирование нейронных сетей без сегментации**

| Классификатор | Точность | F1 | Jaccard score |
|---|---|---|---|
| EfficientNet-B3 | 83 % | 81 % | 83 % |

Тестирование программной системы показано в таблице 2.

Таблица 2

**Тестирование программной системы**

| Класс изображения | Классифицировано верно | Классифицировано неверно | Тест пройден |
|---|---|---|---|
| 1. Прозрачные пустые бутылки маленькие (до 0,5 л) | 55 | 1 | Да |
| 2. Прозрачные пустые бутылки средние (до 2,5 л) | 55 | 3 | Да |
| 3. Прозрачные пустые бутылки большие (до 6 л) | 44 | 2 | Да |
| 4. Бутылки мятые большие | 57 | 3 | Да |
| 5. Бутылки мятые малые | 51 | 2 | Да |
| 6. Бутылки с жидкостью | 59 | 5 | Да |
| 7. Цветные пустые бутылки маленькие (до 0,5 л) | 51 | 1 | Да |
| 8. Цветные пустые бутылки средние (до 2,5 л) | 57 | 1 | Да |
| 9. Стеклянные бутылки | 51 | 4 | Да |
| 10. Алюминиевые банки | 52 | 3 | Да |
| 11. Отсутствие бутылки | 51 | 1 | Да |

Для нахождения лучших результатов обучающая выборка разбивалась в соотношении 3 к 1 для каждого класса изображений.

На 80 % из них модель обучалась, на 20 % – тестировалась. Для обучения было решено установить 25 эпох, так как дальнейшее увеличение количества эпох не приносило ощутимого роста точности распознавания.

Нейронную сеть размещаем на сервере Apache. При загрузке изображения для классификации, используем специальную страницу на сервере. Тестирование удаленной классификации на сервере представлено в таблице 3.

Таблица 3

**Тест-кейс № 1. Удаленная классификация на сервере**

| Действие | Ожидаемый результат |
|---|---|
| Войти на сервер | Должна открыться стартовая страница системы с возможностью выбора классификатора |
| Выбрать классификатор | Должна открыться отдельная страница классификатора, на которой должна присутствовать форма для загрузки изображений |
| Нажать кнопку выбора файла –> выбрать изображение на диске –> нажать кнопку «Загрузить» | Изображение должно быть загружено на сервер, пользователь должен быть перенесен на страницу результата распознавания, на странице должно присутствовать соответствующее изображение с меткой класса |

После загрузки изображения классификатор назначает класс для изображения и выводит его на экран.

Рассмотрим фрагмент определения класса:

```
filename = '../uploads/last.png'
img = image.load_img(filename,
target_size=(img_height, img_width))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
pred = loaded_model.predict(x)[0]
```

Здесь filename определяет имя последнего загруженного файла.

img = image.load_img(filename, target_size=(img_height, img_width)) используется для преобразования изображения в пригодный для классификатора вид.

pred = loaded_model.predict(x)[0] выводит вещественное число от 0 до 1 определяющее класс.

Представление веб-интерфейса состоит из следующих частей:

1. Авторизация пользователя (форма для ввода логина и пароля пользователя).

2. Основной вид компаний рассылки (содержит все компании и темы, управляющие элементы для создания и редактирования компаний, тем, сообщений и для отображения статистических данных).

3. Система всплывающих окон (добавление и редактирование компаний, тем и сообщений и отображение статистики).

При первом попадании пользователя на страницу веб-интерфейса система показывает форму авторизации. Пользователь вводит логин и пароль и нажимает на кнопку «Войти». При проверке введенной информации система или предоставляет дальнейший доступ к веб-интерфейсу, показывая основной вид компаний, или запрещает доступ и возвращает пользователя на форму для авторизации.

Разработанная программа предоставляет более удобную для чтения сводку производительности модели в конце каждой эпохи. Она также предоставляет удобную функцию, которая позволяет вам установить количество эпох для обучения, пока не появится сообщение с вопросом, хотите ли вы остановить обучение в текущей эпохе, введя H, или ввести целое число, которое будет определять, сколько еще эпох нужно запустить, до того как сообщение появляется снова. Это очень полезно, если вы обучаете модель и решаете, что метрики удовлетворительны, и хотите закончить обучение модели досрочно. Обратите внимание, что обратный вызов всегда возвращает вашу модель с весами, установленными для той эпохи, которая имела самую высокую производительность по отслеживаемой метрике (точность или точность проверки).

Обратный вызов первоначально отслеживает точность обучения и будет корректировать скорость обучения на основе этого до тех пор, пока точность не достигнет заданного пользователем порогового уровня. Как только этот уровень точности обучения будет достигнут, обратный вызов переключается на мониторинг потерь проверки и регулирует скорость обучения на основе этого.

Обратный вызов печатает обучающие данные в виде электронной таблицы. Указав csv_path, вы можете создать файл csv с этими данными. Это полезно, когда вы выполняете настройку гиперпараметров, чтобы иметь возможность сравнивать данные обучения из предыдущих прогонов с данными текущего прогона. Созданный CSV-файл имеет отметку времени. Если форма имя-месяц=день-год-час-минута-секунды. обратный вызов имеет форму:

callbacks=[LRA(model, base_model, patience, stop_patience, threshold,factor, dwell, batches, initial_epoch, epochs, ask_epoch )];

• наша модель – my_model;

• base_model – это имя вашей base_model, если пользователь проводит трансферное обучение;

• после 1 эпохи без улучшений скорость обучения будет снижена;

• после 3-х последовательных корректировок скорости обучения без улучшения метрики обучение прекращается;

• как только точность обучения достигает 90 %, обратный вызов корректирует скорость обучения на основе потери проверки;

• при настройке скорости обучения новая скорость обучения составляет 0,5 X скорость обучения;

• если значение метрики текущей эпохи не улучшилось, загружаются веса для предыдущей эпохи и скорость обучения снижается;

• 85 пакетов данных выполняются для завершения эпохи;

• начальная эпоха равна 0;

• тренировка на 20 эпох;

• после пятой эпохи пользователя спросят, хотите ли вы остановить обучение, введя H, или ввести целое число, обозначающее, сколько еще эпох должно пройти, прежде чем вам будет предложено снова, или ввести T, чтобы сделать base_model=trainable.

Полученное качество моделей является достаточно хорошим для доказательства работоспособности системы. Дальнейшее улучшение качества возможно за счет увеличения объема обучающей выборки и более точного подбора гиперпараметров модели.

С точки зрения скорости работы система распознавания показала приемлемый результат. Весь процесс обработки кадра может быть выполнен за разумное время на бюджетном по меркам вычислительных центров оборудовании.

**Литература**

1. Ногтев, Д. Н. Система для обратного торгового автомата, использующая нейронные сети / Д. Н. Ногтев // Молодые исследователи – регионам : материалы Международной научной конференции.– Вологда : ВоГУ, 2022. – Т. 1. – С. 124–125.

2. Кудряшов, Д. А. Применение нейросетевых технологий в задачах идентификации дефектов на пиломатериалах / Д. А. Кудряшов, А. А. Суконщиков // Вестник ВоГУ. – 2023. – № 3. –С. 28–31.

3. Суконщиков, А. А. Мультиагентные интеллектуальные системы и сети / А. А. Суконщиков, А. Н. Швецов. – Вологда : ВоГУ, 2019. – 171 с.

***D.A. Kudryashov, D.N. Nogtev, A.A. Sukonshchikov***
*Vologda State University*

**DEVELOPMENT OF BOTTLE RECOGNITION SYSTEM IN FANDOMAT BASED ON NEURAL NETWORKS**

The article discusses the construction of a bottle recognition system in a fandomat based on a convolutional neural network (CNN). For software implementation of SNI we use the Python language, OpenCV libraries, and CNN models EfficientNet-B0, EfficientNet-B3. 1650 images from Kaggle.com were used for training. The recognition accuracy on the trained CNN is at least 83 %.

Convolutional neural network, fandomat, bottles, program, recognition system.