Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

РАБОТА ПРОВЕРЕНА
РЕЦЕНЗЕНТ,
_____
_____
«____»_____2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
ЗАВЕДУЮЩИЙ КАФЕДРОЙ,
Д.Ф.-М.Н., ДОЦЕНТ
_____С.А. ЗАГРЕБИНА
«____»_____2018 г.

Реализация алгоритма распознавания объектов на изображении

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2018.044.10.000 ПЗ

Нормоконтролер
к.ф.-м.н, доцент каф. МиКМ
_____Т.А. Макаровских
_____2018 г.

Руководитель
ст. преподаватель кафедры МиКМ
_____А.К. Богушов
_____2018 г.

Автор работы
студент группы ЕТ-416
_____В.В. Блащаневич
_____2018 г.

Челябинск, 2018

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»
Направление «Прикладная математика и информатика»

УТВЕРЖДАЮ
Заведующий кафедрой МиКМ
_____С.А. Загребина
_____ 2018 г.

ЗАДАНИЕ

на выпускную квалификационную работу студента

Блащаневича Владимира Владимировича

Группа ЕТ-416

1 Тема работы

Реализация алгоритма распознавания объектов на изображении утверждена приказом по университету от «____»_____ 2018 г. №_____.

2 Срок сдачи студентом законченной работы

«___» _____ 2018 г.

3 Исходные данные к работе

- Данные из открытых источников (Интернет) по различным алгоритмам распознавания объектов на изображении.

- Обучающая выборка, представленная фотографиями различных товаров, подлежащих классификации.

4 Содержание пояснительной записки (перечень подлежащих разработке вопросов)

- Сравнительный обзор различных подходов к классификации

объектов на изображении.

- Реализация подхода, наилучшим образом выполняющего поставленную задачу.
- Оптимизация полученного результата.

5 Графический материал

- Цели и задачи работы.
- Основные подходы к распознаванию объектов на изображении.
- Проектирование и обучение сверточной нейронной сети.
- Оптимизация полученной модели.
- Заключение.

КАЛЕНДАРНЫЙ ПЛАН

Наименование разделов выпускной квалификационной работы	Срок выполнения разделов работы	Отметка руководителя о выполнении
Изучение литературы по теме дипломной работы и поставленным задачам		
Структурирование и обработка исходных данных		
Реализация алгоритма распознавания объектов на изображении		
Анализ полученных результатов		
Проверка работы руководителем, исправление замечаний		
Оформление пояснительной записки		
Оформление графического материала		
Нормоконтроль		
Рецензирование, представление зав. кафедрой		
Защита выпускной квалификационной работы		

Дата выдачи задания «___» _____ 2018 г.

Руководитель работы _____/А.К. Богушов/

Задание принял к исполнению_____/В.В. Блащаневич/

АННОТАЦИЯ

Блащаневич В.В. Реализация алгоритма распознавания объектов на изображении – Челябинск: ЮУрГУ, ИЕТН; 2018, 52 с.
11 иллюстраций, библиографический список – 21 наименование, 3 приложений.

В выпускной квалификационной работе представлена реализация алгоритма распознавания объектов на изображении средствами сверточной нейронной сети. Также, произведен обзор существующих подходов к распознаванию объектов на изображении.

В результате была разработана и обучена модель сверточной нейронной сети, которая имеет точность распознавания 87% на отложенных данных из предоставленной выборки.

Оглавление

Введение

Современные технологии проникают во все сферы жизни, меняя при этом способы производства и реализации товаров и услуг. Часть этого прогресса достигнута благодаря машинному обучению. Оно позволяет ускорить рабочий процесс, а также сократить расходы в различных сферах деятельности. Технология компьютерного зрения является частью машинного обучения. С помощью нее осуществляются следующие процессы:

- системы управления процессами (промышленные роботы, автономные транспортные средства);
- системы видеонаблюдения (распознавание лиц);
- системы организации информации (например, для индексации баз данных изображений);
- системы моделирования объектов или окружающей среды (анализ медицинских изображений, топографическое моделирование);
- системы взаимодействия (например, устройства ввода для системы человеко-машинного взаимодействия);
- системы дополненной реальности (системы обучения, игры);
- вычислительная фотография (в мобильных устройствах с камерами).

Технологии компьютерного зрения находят применение в сфере ретейла, например, с помощью них производится распознавание товаров на изображении, с целью последующего учета.

У компьютерного зрения есть ряд преимуществ, окупающие большинство сложностей внедрения и работы:

1) стоимость внедрения и обслуживания технологии меньше, чем сумма, затрачиваемая на сотрудников, осуществляющих ту же работу;
2) высокая скорость получения результата;
3) точность распознавания объектов компьютера выше чем у человека [10].

Так как существует ряд подходов к распознаванию объектов на изображении **целью работы** – определить, какой из методов распознавания объектов на изображении наиболее эффективен в задаче распознавания товаров на фотографии витрины магазина, реализовать его, добиться точности распознавания приемлемой для бизнес процессов [7], а также оценить полученный результат.

Для этого необходимо решить следующие **задачи**:

- изучить особенности каждого из возможных алгоритмов решения, определить какой подход решает поставленную задачу с наибольшей точностью распознавания;
- убрать зашумленность данных (удалить изображения на которых объект отсутствует);
- скорректировать данные в обучающей выборке (привести исходные изображения к виду: разрешение 128 на 128 пикселя в градациях серого);
- разработать алгоритм распознавания объектов на изображении (разработать структуру сверточной нейронной сети, произвести ее обучение, оптимизировать гиперпараметры модели сети);
- написать скрипт, предсказывающий, с помощью обученной сети, наличие объекта на изображении.

Объект исследования – алгоритм распознавания объектов на изображении.

Предмет исследования – программа для распознавания товаров на витринах магазинов.

Информационная база исследования.

В качестве информационной базы исследования были использованы:

- статьи следующих авторов, опубликованные в открытых источниках (Интернет): Р. Давлеткалиев [4]; Е. Лабинтсев [8]; А. Шарма [9]; П. Нестеров [16].

- учебные пособия следующих авторов: С.И. Николенко [2]; Л. Рамальо [3]; А. Мюллер [7].

Работа состоит из трёх глав, введения, заключения, 20 источников в библиографическом списке, трех приложений.

Во введении обоснована актуальность выбранного исследования, определены цели и задачи работы, отмечены предмет и объект исследований, приведена информационная база и указано краткое содержание работы, выполняется постановка задачи, производится подготовка и описание входных данных, раскрывается специфика задачи.

В первой главе выполняется обзор доступных методов распознавания объектов на изображении, осуществляется выбор оптимального варианта.

Во второй главе реализуется выбранный алгоритм.

В третьей главе производится оптимизация полученного решения, делается вывод о применимости его на практике.

В заключении подведены итоги дипломной работы, приводится список реализованных задач.

В приложении А представлено техническое задание к выпускной квалификационной работе

В приложении Б представлен исходный код реализации алгоритма распознавания объекта на изображении.

В приложении В представлено руководство пользователя к выпускной квалификационной работе.

1 Существующие подходы классификации объектов на изображении

1.1 Контурный анализ

Товары на стеллажах складских помещений и витринах торговых залов расположены так, что можно выделить средствами компьютерной обработки некие очертания, которые будут однозначно идентифицировать этот товар. Благодаря этим очертаниям можно надежно распознавать товар на изображении.

Принцип работы контурного анализа заключается в том, что изначально определяются эталонные контуры объекта, которые выделяются на заранее подготовленных изображениях товара. Затем, в процессе распознавания, на изображении стеллажа с товаром или витриной выделяются замкнутые контуры, в том числе и товаров, производится фильтрация и коррекция, чтобы в последствии сравнить найденные контуры с эталонными значениями и дать предсказание о наличии объекта на изображении.

Однако данный подход имеет ряд недостатков, распознавание средствами контурного анализа чувствительно к условиям съемки. Это означает, что для успешного распознавания товаров они не должны соприкасаться на полках, иначе произойдет слияние контуров, что в свою очередь недопустимо. Также нельзя допускать перекрытия одного товара другим, это приведет к потере части контура, в результате чего распознавание будет затруднено. Также стоит отметить, что при недостаточном освещении при съемке изображение будет зашумлено, что также пагубно сказывается на выделении контуров.

Таким образом, чтобы добиться стабильной работы алгоритма распознавания товаров средствами контурного анализа ретейлеру необходимо увеличить расходы на освещение складских помещений, менее рационально использовать площадь стеллажей и витрин, а также расставлять товары в соответствии со строгими правилами, которые позволят четко

выделять контур. С точки зрения ретейлера это невозможно, тем более при наличии альтернативных вариантов, данное решение не является оптимальным.

1.2 Гистограмма направленных градиентов

Суть подхода распознавания объекта на изображении средствами гистограммы направленных градиентов (HOG - Histogram of Oriented Gradients) заключается в том, что изображение разбивается на множество непересекающихся квадратных сегментов, каждый из которых образует некоторую локальную область. В каждой из этих областей ведется подсчет количества направлений градиентов, т. е. считаются интенсивности перепадов цветов в
разных направлениях, затем полученные значения нормализуются по блоку.

В результате данной операции выделяются признаки, присущие товару, так называемые HOG-дескрипторы. На этих дескрипторах обучается бинарный классификатор (результатом работы классификатора являются два ответа 0 или 1, в зависимости от того, был найден товар или нет). Наиболее эффективным классификатором для HOG является машина опорных векторов (SVM - Support Vector Machine).

Для данной задачи лучше подойдет линейная SVM. Выбор обусловлен тем, что она устойчива к переобучению, это особенно важно, т.к. размер дескриптора велик (около 1000 вещественных чисел на каждый
сканируемый участок), а тренировочных данных довольно мало.

Обучение осуществляется в несколько эпох. После каждой эпохи выявляются ложные срабатывания на тех изображениях обучающей выборки, на которых нет данного товара. В этих изображениях, отсканированых классификатором, выявляется участок, где найден объект (ложное срабатывание), и этот участок добавляется в тренировочную выборку в качестве сложного отрицательного примера. Через 4–5 эпох классификатор достигает своей максимальной точности и перестаёт улучшаться, далее идет переобучение.

Данный подход характерен быстротой как в обучении, так и в работе. Устойчив к изменению масштаба объекта. Однако он не удовлетворяет требованиям по точности распознавания товара (по имеющейся обучающей выборке не достигается 85% успешных распозновааний), поэтому неприменим к задаче.

1.3 Сверточная нейронная сеть

Свёрточная нейронная сеть (CNN - convolutional neural network) — специальная архитектура искусственных нейронных сетей, нацеленная на эффективное распознавание изображений, является частью технологий глубокого обучения (deep learning)[11].

Принцип работы CNN заключается в том, что на входной слой сети подается обработанное изображение (в нашем случае матрица чисел от 0 до 1 размера 128 на 128, что соответствует изображению 128 на 128 пикселя в градациях серого, где 0 абсолютно черный, а 1 абсолютный белый), на выходе сети работает полносвязный слой. Этот слой использует входные данные, обработанные промежуточными слоями, и выводит вектор длинной N, где N — число классов (товаров). Числа в векторе означают процент уверенности сети в том, что на изображении имеется объект класса (для каждого класса свое число).

Сверточная нейронная сеть содержит следующие виды слоев.

- Слой входных данных. В нем содержатся исходная информация об изображении. Ширина, высота, количество каналов соответствует размерности входных данных (в нашем случае размерность 128, 128, 1)).
- Слой свёртки умножает значения фильтра на исходные значения пикселей изображения (поэлементное умножение), после этой операции все эти умножения суммируются. Фильтр представляет собой матрицу весов нейронной сети. Результатом работы слоя являются преобразованные данные меньшей размерности.
- Слой активации. Скалярный результат работы каждого слоя свертки проходит через функцию активации. Она представляет собой некую

нелинейную функцию. Слой активации обычно логически объединяют со слоем свёртки. Функция нелинейности может быть любой по выбору исследователя, в качестве нее использовали функции типа гиперболического тангенса $(f(x) = \tanh(x))$ или сигмоиды $(f(x) = (1 + e^{-x})^{-1})$. Однако представленные функции требовали относительно больших вычислительных мощностей и в 2000х годах была предложена и исследована [13] новая функция активации — ReLU (сокращение от rectified linear unit), за счет упрощения вычислений был достигнут значительный прирост в скорости обучения [14]. Функция ReLU имеет следующий вид: $f(x) = \max(0, x)$. Она выполняет операцию отсечения отрицательной части скалярной величины. На момент середины 2018 года данная функция и её модификации (Noisy ReLU, Leaky ReLU и другие) являются наиболее часто используемыми функциями активации в глубоких нейросетях, в частности, в свёрточных [2].

- Слой пулинга выполняет операцию по понижению дискретизации пространственных размеров (ширины и высоты). Данный слой осуществляет нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. Наиболее употребительна при этом функция максимума. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в один пиксель, при этом выбирается пиксель, имеющий максимальное значение. Операция пулинга позволяет существенно уменьшить пространственный объём изображения. Пулинг интерпретируется так: если на предыдущей операции свёртки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. Также фильтрация уже ненужных деталей помогает не переобучаться и выявлять низкоуровневые признаки. Слой пулинга, как правило, вставляется после слоя свёртки перед слоем следующей свёртки.

- После нескольких прохождений свёртки изображения и уплотнения с помощью пулинга система перестраивается от конкретной сетки пикселей с высоким разрешением к более абстрактным картам признаков. В конце концов остаётся большой набор каналов, хранящих небольшое число данных, которые интерпретируются как самые абстрактные понятия, выявленные из исходного изображения. Эти данные объединяются и передаются на обычную полносвязную нейронную сеть, которая тоже может состоять из нескольких слоёв. При этом полносвязные слои уже утрачивают пространственную структуру пикселей и обладают сравнительно небольшой размерностью (по отношению к количеству пикселей исходного изображения).

В результате объединения этих слоев получается сверточная нейронная сеть. Данный алгоритм является одним из лучших алгоритмов по распознаванию и классификации изображений. Это обусловлено тем, что он относительно устойчив к повороту и сдвигу распознаваемого изображения. Также плохое освещение не так пагубно сказывается на результате распознавания, как в контурном анализе. Недостатком, в сравнении с другими методами, является тот факт, что для достижения необходимой точности распознавания нужно иметь обучающую выборку относительно большого объема. Однако этот минус не является критичным, т. к. для обучения имеется необходимая выборка.

Выводы по главе один

В результате проведенного сравнительного обзора основных подходов к распознаванию объектов на изображении было выяснено, что алгоритм, основанный на сверточной нейронной сети, имеет явное превосходство над остальными в требуемых метриках. В решении данной задачи выбор был сделан в пользу него, так как точность, которую он позволяет достичь, достаточна для задач бизнеса [7]. Данный алгоритм применим к задаче распознавания товаров на фотографии витрины магазина.

2 Реализация алгоритма распознавания объектов на изображении средствами сверточной нейронной сети

Первым этапом разработки алгоритма является очистка данных от шумов (удаление некорректных изображений). Этот процесс выполняется вручную, т. к. программными средствами добиться нужного качества выборки не представляется возможным. В результате была получена выборка из 10 классов (видов номенклатур), для каждого класса в выборке подготовлено 100 изображений, этого будет достаточно для обучения сверточной нейронной сети [2].

Следующим этапом требуется преобразовать исходные изображения в массив данных, размер которого будет зависеть от 4 параметров: ширины и высоты изображения, числа каналов (в нашем случае один канал (т. к. изображения в градациях серого), в случае использования изображения в RGB палитре было бы 3), а также количества изображений в выборке.

Для реализации данной задачи был использован язык программирования Python, т. к. он удобен и имеет стандартную библиотеку работы с изображениями PIL. Также для работы с путями файловой системы необходим модуль glob, для быстрой работы с массивами импортируем библиотеку Numpy, также для преобразования числовых значений класса в категориальное кодирование нам понадобится функция to_categorical из библиотеки Keras. Для этого их необходимо импортировать (листинг 1).

```
from PIL import Image
from glob import glob
import numpy as np
import keras
from keras.utils import to_categorical
```

Листинг 1 – Импортирование библиотек

Т.к. преобразование исходных данных будет производиться не только при обучении, но и при предсказании на новых данных, следует написать функцию для этой операции (листинг 2). Эта функция получает входные

параметры, которые описывают требуемую высоту и ширину изображения, а также директорию, в которой находятся папки с изображениями классов. С помощью модуля glob мы сначала получаем перечень этих папок, а затем поочередно считываем каждое изображение в папке, преобразуя его цветовую палитру в градации серого, изменяем его размер на требуемый. Также делим значение каждого пикселя на 255, чтобы его значение было в интервале от 0 до 1 включительно. Эта операция нужна, т. к. сверточная нейронная сеть может работать только с числами из этого интервала [2]. В результате работы функция возвращает два массива. Первый массив содержит преобразованные изображения, второй содержит принадлежность изображения к классам. Он необходим т. к. обучение сверточных нейронных сетей относится к категории обучения с учителем [7]. Т. е. при обучении сеть заранее знает правильный ответ и в зависимости от него корректирует свои веса.

```
def get_preprocessing_data(width, height, dir):
    img_gray_data = np.array([], dtype=np.uint8)
    img_class = np.array([], dtype=np.uint8)

    adress_sourse = glob(dir)

    for num, one_class in enumerate(adress_sourse):
        for id, img in enumerate(glob(one_class+'*.jpg')):
            img_sourse = Image.open(img)
            img_gray = img_sourse.convert("L")
            img_gray = img_gray.resize((width, height))
            img_gray_data = np.append(img_gray_data,
list(img_gray.getdata()))
            img_class = np.append(img_class, num)

    img_gray_data = img_gray_data/255
    img_gray_data = np.reshape(img_gray_data, (-1,
```

```
img_gray.height, img_gray.width, 1))
    img_class = to_categorical(img_class)


    return img_gray_data, img_class
```

Листинг 2 – Функция преобразования исходных изображений в массив данных

Завершающим этапом предобработки является вызов функции и сохранение средствами библиотеки Numpy в файлы полученные массивы (листинг 3).

```
img_gray_data, img_class = get_preprocessing_data(128, 128,
'raw_data/*/')


np.save('data/sample.npy', img_gray_data)
np.save('data/class_label.npy', img_class)
```

Листинг 3 – Вызов функции и сохранение ее вывода

В результате исходные изображения преобразуются в пригодные для обучения данные (рис. 1, 2).

Рисунок 1 – Исходное изображение до преобразования

Рисунок 2 – Исходное изображение после преобразования

Для конвертации из RGB палитры в градации серого использовалась формула [15]:

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B,$$

где Y значение от 0 до 255 отражающее градацию серого, а R, G, B — красный, зеленый, синий канал соответственно.

Третий этап разработки включает в себя проектирование нейронной сети и ее обучение на заранее подготовленных изображениях. Любая архитектура нейронной сети по сути является чередованием слоев свертки, активации и пулинга. На момент 2018 года наиболее употребляемая архитектура сверточной нейронной сети включает в себя 3 слоя свертки и 2 слоя пулинга [2,4,7] (рис. 3).

Рисунок 3 – Архитектура наиболее употребляемой сверточной нейронной сети

Для решения поставленной задачи для построения сверточной нейронной сети была использована библиотека Keras. Она эффективна, а также позволяет с максимальной наглядностью конструировать нейронные сети [1]. Также для быстрой работы массивов импортируем библиотеку Numpy, для разделения выборки на обучающую и тестовую импортируем функцию `train_test_split` из библиотеки sklearn (листинг 4).

```
import keras
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten, Input
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
from keras.models import model_from_json
from sklearn.model_selection import train_test_split
import numpy as np
```

Листинг 4 – Импортирование библиотек для построения и обучения сверточной нейронной сети

Следующим этапом производим загрузку выборки и ответов о принадлежности изображений выборки к классам, а также разделяем выборку на обучающую и тестовую. Это нужно для того, чтобы оценивать обобщающую способность модели (если на тестовой выборке точность распознавания значительно ниже чем на обучающей выборке, то мы делаем вывод, что наша модель переобучилась, т. е. просто заучила входные данные, а не выявила в них некоторые признаки объекта) (листинг 5).

```
X = np.load('data/sample.npy')
y = np.load('data/class_label.npy')


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.1, random_state=47)
```

Листинг 5 – Загрузка и разделение на обучающую и тестовую выборку данных

Теперь, когда у нас есть подготовленные данные, необходимо построить сеть, которая будет на них обучаться (листинг 6). Как было сказано ранее, для построения была использована наиболее употребляемая схема сети (рис. 3). В начале, устанавливаем гиперпараметры `batch_size`, `epochs`. Они означают размер подвыборки, количество эпох обучения. Далее производим инициализацию сверточной нейронной сети. Т.к. мы будем использовать эту операцию неоднократно, напишем для нее функцию. Также внутри функции добавляем слои в соответствии со схемой. Помимо слоев, указанных в схеме, мы также добавляем `Dropout`. Это основной метод борьбы с переобучением, его суть заключается в том, что в процессе обучения указанная доля нейронов не задействованы и решение принимается на основе оставшихся. В процессе обучения состав и количество задействованных нейронов меняется.

После добавления всех слоев модель необходимо скомпилировать и запустить процесс обучения. При этом мы указываем, что функция потерь считается как категориальная кроссэнтропия [17], а метрика качества модели это процент точно опознанных объектов на изображении.

```
batch_size = 15

epochs = 12

num_classes = y.shape[1]


def get_cnn_model(X):

     class_model = Sequential()

     class_model.add(Conv2D(32, kernel_size=(3, 3),
activation='linear', padding='same', input_shape=(X.shape[1],
X.shape[2], 1)))

     class_model.add(LeakyReLU(alpha=0.1))

     class_model.add(MaxPooling2D((2, 2), padding='same'))

     class_model.add(Dropout(0.25))

     class_model.add(Conv2D(64, (3, 3), activation='linear',
padding='same'))

     class_model.add(LeakyReLU(alpha=0.1))

     class_model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

     class_model.add(Dropout(0.25))

     class_model.add(Conv2D(128, (3, 3), activation='linear',
padding='same'))

     class_model.add(LeakyReLU(alpha=0.1))

     class_model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

     class_model.add(Dropout(0.4))

     class_model.add(Flatten())

     class_model.add(Dense(128, activation='linear'))

     class_model.add(LeakyReLU(alpha=0.1))

     class_model.add(Dropout(0.3))

     class_model.add(Dense(num_classes, activation='softmax'))


     class_model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adam(), metrics=['accuracy'])


     return class_model
```

```
class_model = get_cnn_model(X)

classificator_train = class_model.fit(X_train, y_train,
batch_size=batch_size, epochs=epochs, verbose=1,
validation_data=(X_test, y_test))


test_eval = class_model.evaluate(X_test, y_test, verbose=0)

print('loss value:', test_eval[0])

print('accuracy:', test_eval[1])
```

Листинг 6 – Построение и обучение сверточной нейронной сети

После того как модель обучилась, ее необходимо сохранить. Сама структура сети и веса полученные в процессе обучения сохраняются в два отдельных файла `product_classification.json` и `product_classification.h5py` соответственно (листинг 7).

```
class_model.save("product_classification.h5py")
model_json = class_model.to_json()
with open("product_classification.json", "w") as json_file:
                    json_file.write(model_json)
```

Листинг 7 – Сохранение структуры и весов сверточной нейронной сети

Последним этапом разработки является написание скрипта, который на основе полученной сверточной нейронной сети предсказывает наличие объекта на новом изображении (т. е. определяет какие товары из заданных присутствуют на фотографии полки магазина).

Для реализации данной задачи требуется библиотеки Keras, Numpy. Также импортируем функцию `get_preprocessing_data` (листинг 8).

```
import keras

import numpy as np

from keras.models import model_from_json

from preprocessing import get_preprocessing_data
```

Листинг 8 – Импортирования библиотек для построения предсказания наличия объекта на изображении

После того, как все необходимые библиотеки импортированы, загружаем ранее обученную сверточную нейронную сеть (листинг 9).

```
json_file = open('product_classification.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
loaded_model.load_weights("product_classification.h5py")


loaded_model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
```

Листинг 9 – Загрузка и компилирование модели сверточной нейронной сети

Для построения прогноза наличия товара на фотографии витрины магазина необходимо, используя ранее написанную функцию `get_preprocessing_data`, преобразовать фотографии, загруженные для прогнозирования (помещенные в папку predict корневого каталога). Далее с помощью метода `predict` модели получаем вектор предсказания вероятности наличия каждой из номенклатур на фотографии (листинг 10). В результате будет выведен вектор с предсказанием наличия объектов на изображении (рис. 4).

```
img_gray_data, _ = get_preprocessing_data(128, 128, 'predict/')
print(loaded_model.predict(img_gray_data))
```

Листинг 10 – Преобразование исходных изображений для предсказаний и построение прогноза

Рисунок 4 – Вывод программы предсказывающей наличие объекта на изображении

Выводы по главе два

На основании информации, полученной в первой главе, был разработан алгоритм распознавания объектов на изображении. В его основе лежит технология сверточной нейронной сети, как следствие в ходе разработки была построена и обучена сверточная нейронная сеть. Также был реализован процесс очистки и предобработки данных. В результате, разработанный алгоритм дал точность распознавания на отложенной выборке 87%, что подтвердило вывод из первой главы о состоятельности сверточных нейронных сетей в бизнес задачах.

3 Оптимизация гиперпараметров сверточной нейронной сети

3.1 Выбор оптимального размера изображения обучающей выборки

Для выполнения данной задачи имеется обучающая выборка, которая представляет собой изображения множества товаров (10 штук), которые подлежат распознаванию. Однако исходные изображения имеют разрешение 720 на 1280 пикселей при этом их цветовая палитра RGB. Данное разрешение слишком велико для обучения сверточной нейронной сети, т.к. велика вероятность переобучения, а также операция обучения будет требовать относительно больших вычислительных мощностей. При данном объеме выборки наибольшая точность распознавания достигается при разрешениях 64 на 64 и 128 на 128 пикселей [2, 7]. Очевидно, что один из вариантов (64 на 64 пикселя) даст прирост к скорости обучения и распознавания, ввиду меньшего размера, однако опытным путем было получено заключение о том, что обучение на данной выборке с шириной и высотой изображения равных 64 пикселя не дает той точности распознавания, которая считается приемлемой в бизнес процессах [7]. На скриншоте процесса обучения видно, что точность, которая была достигнута на отложенной выборке равна 78% (рис. 5).

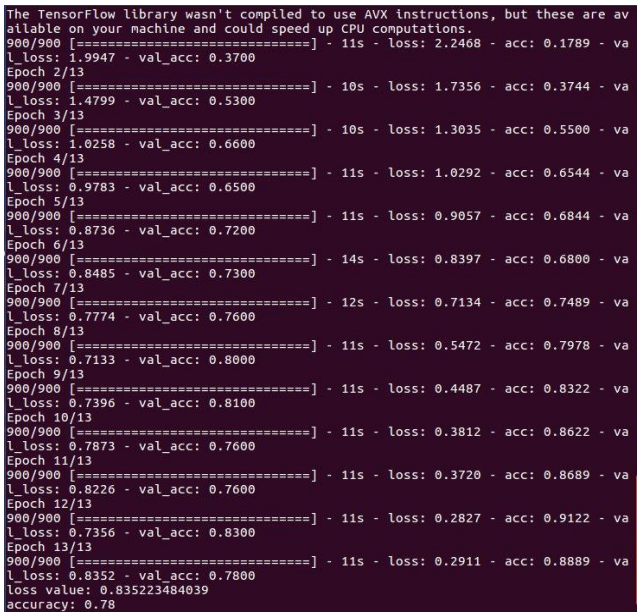

Рисунок 5 – Результат обучения на изображениях 64 на 64 пикселя

Далее было произведено обучение при параметрах ширины и высоты изображения равным 128 пикселям. В данном случае была достигнута требуемая точность распознавания на обучающей выборке 87% (рис. 6).

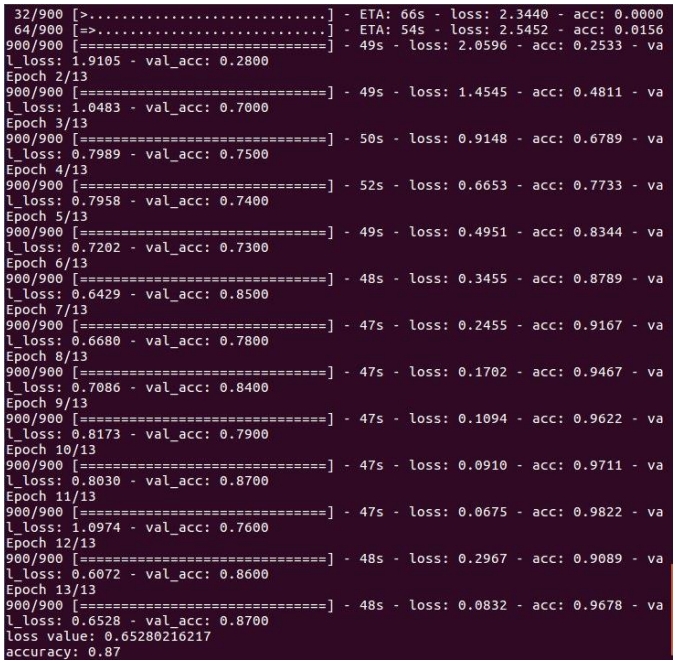

Рисунок 6 – Результат обучения на изображениях 128 на 128 пикселя

Таким образом, с точки зрения точности распознавания, разрешение 128 на 128 превосходит 64 на 64.

3.2 Оптимизация размера подвыборки и количества эпох

В связи с тем, что гиперпараметры `batch_size` (размер подвыборки) и `epochs` (количество эпох) взаимосвязаны, необходимо перебрать все возможные варианты сочетаний этих параметров и выбрать из них то, которое имеет наименьший показатель функции потерь на отложенной выборке, при условии достижения требуемой точности. Этот вариант гиперпараметров будет иметь наибольшую обобщающую способность. Для нахождения оптимальных гиперпараметров необходимо выбрать интервал поиска. В соответствии с рекомендациями [18], а также проведенными начальными приближениями, были выбраны интервалы $[10, 35]$ для `batch_size`, и $[10, 16]$ для `epochs`. Используя полученные интервалы, можно программно осуществить перебор вариантов гиперпараметров с

целью нахождения оптимальных (листинг 11). На основе данных полученных в результате работы програмы (таб. 1), можно сделать вывод о том, что наибольшая обобщающая способность достигается при параметрах `batch_size` равным 15 и `epochs` равным 12 (рис. 7).

```
welaury@welaury:~/Рабочий стол/Ссылка на programdiplom/product_classificator/nomenclature_classificator$ python t
rain.py
Using TensorFlow backend.
Split:  (900, 128, 128, 1) (100, 128, 128, 1) (900, 10) (100, 10)
train.py:28: UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(32, kernel_size=(3, 3), activatio
n="linear", padding="same", input_shape=(128, 128,..., kernel_regularizer=<keras.reg...)`
  class_model.add(Conv2D(32, W_regularizer=l2(l2_lambda), kernel_size=(3, 3), activation='linear', padding='same'
, input_shape=(X.shape[1], X.shape[2], 1)))
Train on 900 samples, validate on 100 samples
Epoch 1/12
2018-06-08 23:29:19.432090: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't com
piled to use SSE4.1 instructions, but these are available on your machine and could speed up CPU computations.
2018-06-08 23:29:19.432139: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't com
piled to use SSE4.2 instructions, but these are available on your machine and could speed up CPU computations.
2018-06-08 23:29:19.432159: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't com
piled to use AVX instructions, but these are available on your machine and could speed up CPU computations.
900/900 [==============================] - 85s - loss: 2.0319 - acc: 0.2456 - val_loss: 1.2993 - val_acc: 0.6100
Epoch 2/12
900/900 [==============================] - 78s - loss: 1.0747 - acc: 0.6267 - val_loss: 0.8763 - val_acc: 0.7300
Epoch 3/12
900/900 [==============================] - 76s - loss: 0.6674 - acc: 0.7533 - val_loss: 0.6739 - val_acc: 0.7900
Epoch 4/12
900/900 [==============================] - 72s - loss: 0.4413 - acc: 0.8522 - val_loss: 0.6204 - val_acc: 0.8500
Epoch 5/12
900/900 [==============================] - 75s - loss: 0.3044 - acc: 0.8822 - val_loss: 0.5387 - val_acc: 0.8400
Epoch 6/12
900/900 [==============================] - 75s - loss: 0.1883 - acc: 0.9289 - val_loss: 0.5193 - val_acc: 0.8200
Epoch 7/12
900/900 [==============================] - 84s - loss: 0.1193 - acc: 0.9656 - val_loss: 0.5694 - val_acc: 0.8600
Epoch 8/12
900/900 [==============================] - 83s - loss: 0.0517 - acc: 0.9833 - val_loss: 0.6545 - val_acc: 0.8400
Epoch 9/12
900/900 [==============================] - 83s - loss: 0.0871 - acc: 0.9689 - val_loss: 0.7033 - val_acc: 0.8600
Epoch 10/12
900/900 [==============================] - 86s - loss: 0.0511 - acc: 0.9833 - val_loss: 0.8196 - val_acc: 0.8500
Epoch 11/12
900/900 [==============================] - 84s - loss: 0.0498 - acc: 0.9811 - val_loss: 0.6888 - val_acc: 0.8300
Epoch 12/12
900/900 [==============================] - 84s - loss: 0.0412 - acc: 0.9900 - val_loss: 0.6517 - val_acc: 0.8800
loss value: 0.651681718826
accuracy: 0.88
```

Рисунок 7 – Процесс обучения сверточной нейронной сети с оптимальными параметрами `batch_size` и `epochs`

Таблица 1 – Значения функции потерь при различных значениях `batch_size` и `epochs`

Batch_size / epochs	10	15	20	25	30	35
10	0.686	0.844	0.771	0.78	0.641	0.706
12	0.702	0.65	1.050	0.865	0.678	0.731
14	0.966	0.813	0.815	0.695	0.764	0.810
16	0.811	0.801	0.692	0.814	0.808	0.697

```
import numpy as np
import keras
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten, Input
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
from keras.models import model_from_json
from train import get_cnn_model

X = np.load('data/sample.npy')
y = np.load('data/class_label.npy')
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15, random_state=47)
num_classes = y.shape[1]
batch_size = list(range(10,40,5))
epochs = list(range(10,20,2))
score_cv = []

for epoch in epochs:
    for batch in batch_size:
        class_model = get_cnn_model(X)
        classificator_train = class_model.fit(X_train,
y_train, batch_size=batch, epochs=epoch, verbose=1,
validation_data=(X_test, y_test))
        test_eval = class_model.evaluate(X_test, y_test,
verbose=0)
        score_cv.append([epoch, batch, test_eval[0],
test_eval[1]])
```

Листинг 11 – Обучение сверточной нейронной сети с гиперпараметрами `batch_size` и `epochs` из заданного интервала

Выводы по главе три

Результатом оптимизации гиперпараметров сверточной нейронной сети является повышение точности распознавания на отложенной выборке, что в совокупности с относительно низким значением функции потерь позволяет сделать вывод о том, что обученная модель имеет удовлетворительную обобщающую способность. Этот факт делает алгоритм применимым в задаче распознавания товаров на фотографии витрины магазина.

Заключение

По результатам теоретических и практических исследований подходов к распознаванию объектов на изображении можно сделать следующие выводы.

Для задачи распознования товаров на фотографии витрины магазина наиболее применим алгоритм на основе сверточной нейронной сети, при этом выборка из 100 изображений каждого класса достаточна для обучения.

Для эффективной работы над задачей была использована библиотека Keras, а также написан предобработчик исходных изображений. В качестве схемы сверточной нейронной сети была использована наиболее употребляемая архитектура [2,4,7]. Обучение производилось на персональном компьютере с процессором Intel core i7 и видеокартой Nvidia GT 750M на операционной системе Windows 8, время обучения 16 минут. При подобранных оптимальных параметрах (размер изображения 128 на 128 пикселей, `batch_size` и `epochs` 15 и 12 соответственно), была достигнута точность распознавания 87% на отложенной выборке, при этом значение функции потерь (0.65) указывает на высокую обобщающую способность модели.

Данное решение востребовано в области логистики и менеджмента ретейлеров продукции различного характера. Полученная точность делает полученное решение применимым на практике.

Библиографический список

1. Keras Documentation // https://keras.io [Электронный ресурс]. – Дата доступа 25.02.2018.

2. Николенко, С.И. Глубокое обучение. Погружение в мир нейронных сетей / С.И. Николенко. – М.: Питер, 2017. – 480 с.

3. Рамальо, Л. Python. К вершинам мастерства / Л. Рамальо. – М.: ДМК-Пресс, 2016. – 768 с.

4. Что такое свёрточная нейронная сеть // https://habr.com/post/309508 [Электронный ресурс]. Дата доступа 25.02.2018.

5. Выделение и описание контуров // http://wiki.technicalvision.ru/index.php/Выделение_и_описание_контуров [Электронный ресурс]. – Дата доступа 25.02.2018.

6. NumPy Reference // https://docs.scipy.org/doc/numpy-1.13.0/reference [Электронный ресурс]. – Дата доступа 25.02.2018.

7. Мюллер, А. Введение в машинное обучение с помощью Python / А. Мюллер. – М.: O'Reilly, 2017. – 480 с.

8. Метрики в задачах машинного обучения // https://habr.com/company/ods/blog/328372 [Электронный ресурс]. – Дата доступа 25.02.2018.

9. Convolutional Neural Networks in Python with Keras // https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python [Электронный ресурс]. – Дата доступа 25.02.2018.

10. History of computer vision contests won by deep CNNs on GPU // http://people.idsia.ch/~juergen/computer-vision-contests-won-by-gpu-cnns.html [Электронный ресурс]. – Дата доступа 25.02.2018.

11. Свёрточная нейронная сеть // https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть [Электронный ресурс]. – Дата доступа 25.02.2018.

12. Ключевые рекомендации по глубокому обучению // http://datareview.info/article/eto-nuzhno-znat-klyuchevyie-rekomendatsii-po-

glubokomu-obucheniyu-chast-2 [Электронный ресурс]. – Дата доступа 25.02.2018.

13. Deep Sparse Rectifier Neural Networks // https://docs.google.com/viewer?docex=1&url=http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf [Электронный ресурс]. – Дата доступа 25.02.2018.

14. Rectified linear units improve restricted Boltzmann machines // https://docs.google.com/viewer?docex=1&url=http://machinelearning.wustl.edu/mlpapers/paper_files/icml2010_NairH10.pdf [Электронный ресурс]. – Дата доступа 25.02.2018.

15. Image Module // https://pillow.readthedocs.io/en/5.1.x/reference/Image.html?highlight=Image [Электронный ресурс]. – Дата доступа 25.02.2018.

16. Обзор топологий глубоких сверточных нейронных сетей // https://habr.com/company/mailru/blog/311706 [Электронный ресурс]. – Дата доступа 25.02.2018.

17. Available loss functions // https://keras.io/losses [Электронный ресурс]. – Дата доступа 25.02.2018.

18. A Walkthrough of Convolutional Neural Network—Hyperparameter Tuning // https://towardsdatascience.com/a-walkthrough-of-convolutional-neural-network-7f474f91d7bd [Электронный ресурс]. – Дата доступа 25.02.2018.

19. Dropout — метод решения проблемы переобучения в нейронных сетях // https://habr.com/company/wunderfund/blog/330814 [Электронный ресурс]. – Дата доступа 25.02.2018.

20. How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras // https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras [Электронный ресурс]. – Дата доступа 25.02.2018.

Приложения

ПРИЛОЖЕНИЕ А

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра математического и компьютерного моделирования

Реализация алгоритма распознавания объектов на изображении

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2018.044.10.000 ПЗ

Нормоконтролер
к.ф.-м.н, доцент каф. МиКМ
_____Т.А. Макаровских
_____2018 г.

Руководитель
ст. преподаватель кафедры МиКМ
_____А.К. Богушов
_____2018 г.

Автор работы
студент группы ЕТ-416
_____В.В. Блащаневич
_____2018 г.

Челябинск, 2018

1. ВВЕДЕНИЕ

1.1. Наименование программного изделия

Полное наименование программы – «программное решение для распознавания товаров на витрине магазина». Краткое наименование программы – «ПО для распознавания товара на витрине».

1.2. Область применения

Программа предназначена для использования ретейлерами в ходе инвентаризации и мониторинга товаров.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1. Документ, на основании которого ведется разработка

Разработка ведется на основании задания на выпускную квалифицированную работу бакалавра по теме «Реализация алгоритма распознавания объектов на изображении».

2.2. Организация утвердившая этот документ, и дата его утверждения Задание утверждено руководителем работы, д.э.н., профессором кафедры МиКМ А.К. Богушовым.

2.3. Наименование темы разработки – «Распознавание товаров в сфере ретейла».

3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

Разработка является частью задания на выпускную квалификационную работу и позволит значительно сократить временные затраты сотрудников ретейла, а также сократит финансовые затраты на специальное оборудование.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ

4.1. Требования к функциональным характеристикам

4.1.1. Состав выполняемых функций

4.1.1.1. Распознавание товара на загруженном изображении

Функция распознавания товара должна обеспечивать прием изображения формата 128 на 128 пикселя в градациях серого, формировать на выход прогноз, на основе обученной сверточной нейронной сети, в виде вектора длины, равной количеству классов, где элементы вектора это

вероятность нахождения объекта на изображения в виде чисел от 0 до 1 включительно.

4.1.1.2. Обновление весов и структуры сверточной нейронной сети при добавлении новых классов

Функция обновления весов и структуры должна обеспечивать актуальность данных о классах в нейронной сети, т.е. при добавлении новых классов (товаров), обучаться заново.

4.1.1.3. Преобразование исходных изображений в данные пригодные для обучения

Так как исходные изображения невозможно напрямую подать на вход нейронной сети, функция преобразование исходных изображений формирует на их основе данные пригодные для обучения.

4.1.2 Организация входных и выходных данных

Входные данные хранятся в виде массива данных изображений на диске ПК.

Выходные данные выводятся через интерфейс командной строки ПК в виде отчетов.

4.2. Требования к надежности

4.2.1. Требования к надежному функционированию

Система должна нормально функционировать при бесперебойной работе ЭВМ. При возникновении сбоя в работе аппаратуры восстановление нормальной работы программы должно производиться после:

1) перезагрузки операционной системы;

2) повторного выполнения действий, потерянных после сбоя.

4.2.2. Контроль входной и выходной информации

Система должна контролировать входные данные и не допускать обучения с данными неправильной размерности.

4.2.3 Время восстановления после отказа

Время восстановления после отказа должно состоять из:

1) Времени запуска пользователем системы;

2) Времени повторного ввода потерянных данных.

4.3. Условия эксплуатации

Система должна храниться в виде двух копий: резервной и рабочей.

4.4. Система должна работать на следующем иди совместимом с ним оборудовании:

1) персональный компьютер с процессором Pentium и выше;

2) фотографирующее устройство.

4.5. Требования к информационной и программной совместимости

4.5.1. Требования к информационным структурам на входе и выходе.

Требования к информационным структурам на входе и выходе определены в п. 4.1.2.

4.5.2. Требования к языкам программирования

Система должна быть разработана на языке программирования Python с использованием библиотеки Keras.

4.5.3. Требования к программным средствам, используемым программой

Для работы системы необходима операционная система Windows XP и выше или Linux.

4.6. Требования к маркировке и упаковке

Диски с резервным и рабочими экземплярами программы должны иметь маркировку, состоящую из надписи «Программа распознавания товаров на витрине», надписи «резервная» или «рабочая», даты последней перезаписи программы. На упаковке должны быть указаны условия транспортировки и хранения диска.

4.7. Требования к транспортированию и хранению

Условия транспортирования и хранения диска должны соответствовать п. 4.6.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Состав программной документации должен включать следующее документы:

1) технический проект программы по ГОСТ 19.404 -79;

2) описание программы по ГОСТ 19.402-78;

3) текст программы по ГОСТ 19:401.-78;

4) руководство пользователя.

6. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Разработка программы должна выполняться по следующим этапам:

1) Разработка, согласование и утверждение технического проекта программы – 5 недель;

2) Разработка рабочего проекта с комплексным тестированием – 8 недель;

3) Приемка-сдача с исправлением обнаруженных недостатков в программе – 2 недели;

4) Внедрение.

7. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

Программа должна считаться годной, если она удовлетворяет всем пунктам технического задания.

ПРИЛОЖЕНИЕ Б

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра математического и компьютерного моделирования

Реализация алгоритма распознавания объектов на изображении

ТЕКСТ ПРОГРАММЫ
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2018.044.10.000 ПЗ

Нормоконтролер
к.ф.-м.н, доцент каф. МиКМ
_____Т.А. Макаровских
_____2018 г.

Руководитель
ст. преподаватель кафедры МиКМ
_____А.К. Богушов
_____2018 г.

Автор работы
студент группы ЕТ-416
_____В.В. Блащаневич
_____2018 г.

Челябинск, 2018

1. Преобразование исходных изображений в массив изображений 128 на 128 пикселей в градациях серого, для последующей обработки сверточной нейронной сетью

```
from PIL import Image
from glob import glob
import numpy as np
import keras
from keras.utils import to_categorical


def get_preprocessing_data(width, height, dir):
    img_gray_data = np.array([], dtype=np.uint8)
    img_class = np.array([], dtype=np.uint8)

    adress_sourse = glob(dir)

    for num, one_class in enumerate(adress_sourse):
        for id, img in enumerate(glob(one_class+'*.jpg')):
            img_sourse = Image.open(img)
            img_gray = img_sourse.convert("L")
            img_gray = img_gray.resize((width, height))
          img_gray_data = np.append(img_gray_data,
list(img_gray.getdata()))
            img_class = np.append(img_class, num)

    img_gray_data = img_gray_data/255
    img_gray_data = np.reshape(img_gray_data, (-1,
img_gray.height, img_gray.width, 1))
    img_class = to_categorical(img_class)

    return img_gray_data, img_class

img_gray_data, img_class = get_preprocessing_data(128, 128,
'raw_data/*/')
```

```
np.save('data/sample.npy', img_gray_data)
np.save('data/class_label.npy', img_class)
```

2. Формирование тренировочной и тестовой выборки, обучение сверточной нейронной сети, сохранение полученных результатов

```
import numpy as np
import keras
from sklearn.model_selection import train_test_split
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten, Input
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
from keras.models import model_from_json


X = np.load('data/sample.npy')
y = np.load('data/class_label.npy')

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15, random_state=47)

print('Split:
',X_train.shape,X_test.shape,y_train.shape,y_test.shape)

batch_size = 15
epochs = 12
num_classes = y.shape[1]

def get_cnn_model(X):
     class_model = Sequential()
     class_model.add(Conv2D(32, kernel_size=(3, 3),
activation='linear', padding='same', input_shape=(X.shape[1],
X.shape[2], 1)))
     class_model.add(LeakyReLU(alpha=0.1))
     class_model.add(MaxPooling2D((2, 2), padding='same'))
     class_model.add(Dropout(0.25))
```

```
    class_model.add(Conv2D(64, (3, 3), activation='linear',
padding='same'))
    class_model.add(LeakyReLU(alpha=0.1))
    class_model.add(MaxPooling2D(pool_size=(2, 2),
padding='same'))
    class_model.add(Dropout(0.25))
    class_model.add(Conv2D(128, (3, 3), activation='linear',
padding='same'))
    class_model.add(LeakyReLU(alpha=0.1))
    class_model.add(MaxPooling2D(pool_size=(2, 2),
padding='same'))
    class_model.add(Dropout(0.4))
    class_model.add(Flatten())
    class_model.add(Dense(128, activation='linear'))
    class_model.add(LeakyReLU(alpha=0.1))
    class_model.add(Dropout(0.3))
    class_model.add(Dense(num_classes, activation='softmax'))

    class_model.compile(loss=keras.losses.categorical_crossentr
opy, optimizer=keras.optimizers.Adam(), metrics=['accuracy'])

    return class_model

class_model = get_cnn_model(X)

classificator_train = class_model.fit(X_train, y_train,
batch_size=batch_size, epochs=epochs, verbose=1,
validation_data=(X_test, y_test))

test_eval = class_model.evaluate(X_test, y_test, verbose=0)
print('loss value:', test_eval[0])
print('accuracy:', test_eval[1])


class_model.save("product_classification.h5py")
```

```
model_json = class_model.to_json()
with open("product_classification.json", "w") as json_file:
     json_file.write(model_json)
```

3. Получение прогноза нахождения объекта на новых входных данных

```
import keras
import numpy as np
from keras.models import model_from_json
from preprocessing import *


json_file = open('product_classification.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
loaded_model.load_weights("product_classification.h5py")

loaded_model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adam(),metrics=['accuracy'])

img_gray_data, _ = get_preprocessing_data(128, 128, 'predict/')

print(loaded_model.predict(img_gray_data))
```

ПРИЛОЖЕНИЕ В

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра математического и компьютерного моделирования

Реализация алгоритма распознавания объектов на изображении

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2018.044.10.000 ПЗ

Нормоконтролер
к.ф.-м.н, доцент каф. МиКМ
_____Т.А. Макаровских
_____2018 г.

Руководитель
ст. преподаватель кафедры МиКМ
_____А.К. Богушов
_____2018 г.

Автор работы
студент группы ЕТ-416
_____В.В. Блащаневич
_____2018 г.

Челябинск, 2018

1. Общие сведения о программном решении для распознавания товаров на витрине магазина

«ПО для распознавания товара на витрине» – программа, предназначенная для использования ретейлерами в ходе инвентаризации и мониторинга товаров. Использование программы позволит значительно сократить временные затраты сотрудников ретейла, а также сократит финансовые затраты на специальное оборудование. На текущий момент программа предоставляет решение следующих задач:

- распознавание товара на загруженном изображении;
- обновление весов и структуры сверточной нейронной сети при добавлении новых классов;
- преобразование исходных изображений в данные пригодные для обучения;

2. Интерфейс программного решения для распознавания товаров на витрине магазина

Интерфейс программы представлен командной строкой.

3. Работа в программном решении для распознавания товаров на витрине магазина

3.1 Преобразование исходных изображений в пригодные для обучения данные

Для выполнения данной задачи необходимо запустить скрипт `preprocessing.py` из корневого каталога программы (рис. 8). Скрипт автоматически отсканирует все папки внутри директории `raw_data` корневого катологa на предмет наличия изображений и преобразует их в массив входных данных в соответствии с требованиями.

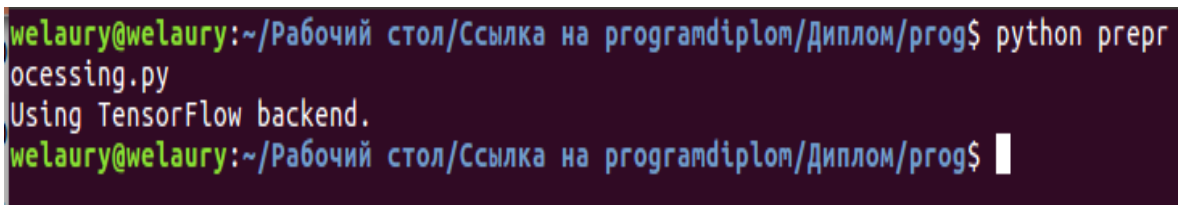

Рисунок 8 – Запуск преобразования изображений из командной строки

3.2 Обучение сверточной нейронной сети

Данная функция выполняется путем запуска скрипта `train.py` из корневого каталога программы.

Сразу после запуска скрипт выведет структуру модели сверточной нейронной сети, которая будет обучена на выборке (рис. 9).

```
welaury@welaury:~/PythonDan/classificator/train$ python Un_CNN_classification.pyUsing TensorFlow backend.
Split:  (2144, 128, 128, 1) (239, 128, 128, 1) (2144, 3) (239, 3)
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 128, 128, 32)      320
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 128, 128, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 64, 64, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 32, 32, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 16, 16, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 32768)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               4194432
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 387
=================================================================
Total params: 4,287,491
Trainable params: 4,287,491
Non-trainable params: 0
```

Рисунок 9 – Запуск обучения сверточной нейронной сети

Процесс обучения включает в себя 12 эпох, по прошествии каждой эпохи пользователю выводятся текущие показатели точности алгоритма распознавания и значение функции потерь как обучающей выборке, так и на отложенной (рис. 10).

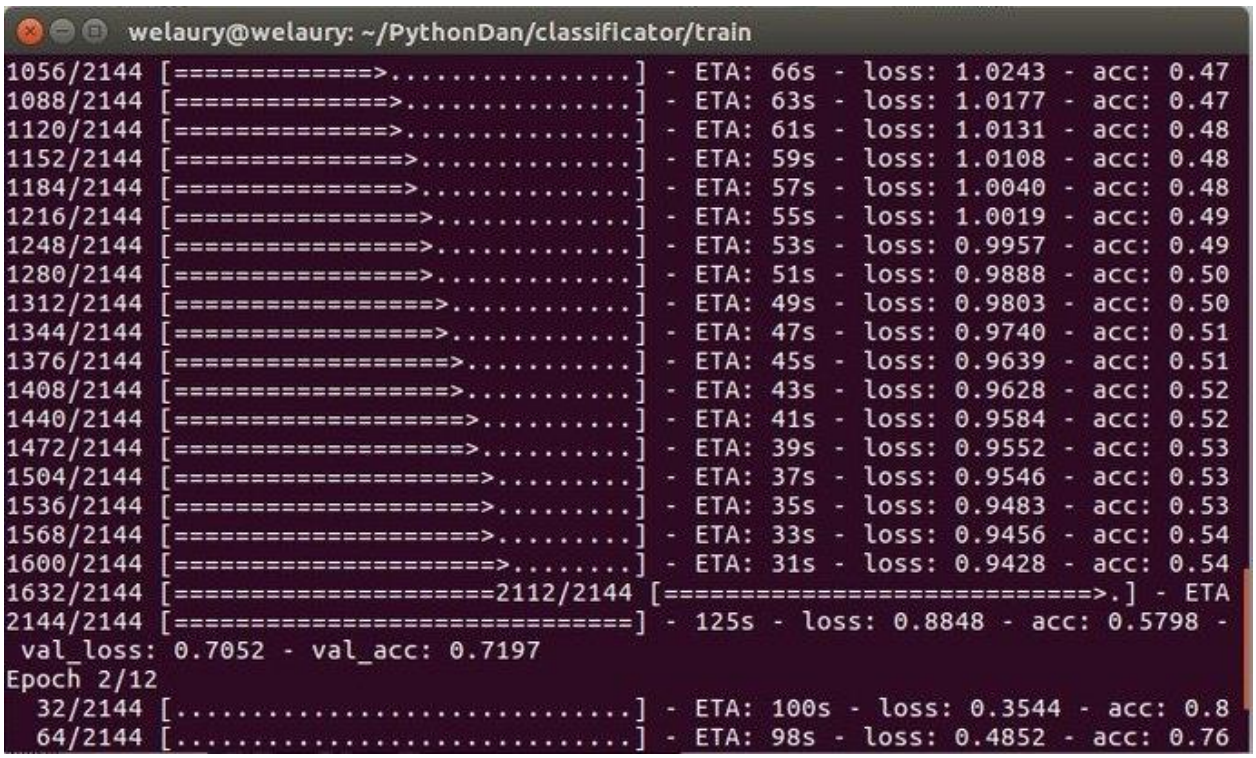

Рисунок 10 – Процесс обучения сверточной нейронной сети

Результат обучения сохраняется в два файла: файл `product_classification.h5py` содержит веса сети, а файл `product_classification.json` представляет собой ее структуру.

3.3. Предсказание наличия объекта на изображении

Чтобы воспользоваться этой функцией, нужно обеспечить нахождение фотографий витрин с товарами в папке `predict` корневого каталога, а также запустить скрипт `predict.py`. Результатом работы данного скрипта является массив векторов с предсказаниями. В данном случае была взята одна фотография витрины с товаром принадлежащему восьмому классу (рис. 11).

Рисунок 11 – Пример работы скрипта предсказания наличия товара на изображении витрины

21.