Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет "Высшая школа экономики"

Образовательная программа "Прикладная математика"
бакалавр

ОТЧЕТ
по проектной работе

Распознавание протоколов

Выполнили студенты гр.БПМ-211

Кармаев Александр Андреевич
Родин Сергей Алексеевич
Индюченко Никита Андреевич

Руководитель проекта:

Доцент

Сластников Сергей Александрович

_____ _____
(оценка) (подпись)

_____
(дата)

Москва - 2022

Оглавление

Глава 1

Введение

В настоящее время все больше используются электронные системы хранения данных. Но многие организации все еще используют консервативные методы хранения информации на бумажных носителях. Благодаря современным технологиям стало возможно конвертировать печатные и рукописные документы в электронный формат. Обеспечение системами распознавания табличных данных позволяет значительно оптимизировать процесс работы и избавить сотрудников от ручного переноса данных с бумажных носителей на электронные.

1.1 Цель работы

Разработка и тестирование программы по переводу рукописно заполненных таблиц с протоколами различных соревнований в excel таблицу.

1.2 Предмет исследования

Различные инструменты распознавания образов и детектирования содержимого определенного вида печатных и рукописных документов на основе технологий Computer Vision.

1.3 Задачи

- Изучить технологии распознавания объектов, текста для обработки документов, выбрать наиболее подходящие для нашей цели инструменты
- Разработать алгоритмы обнаружения и распознавания таблицы в протоколе, используя выбранные выше технологии
- Протестировать полученные решения на разных типах входных данных для получения метрик точности распознавания

1.4 Планируемые результаты

- Реализация различных алгоритмов решения задачи и выявление наиболее эффективного способа считывания документов
- Сравнение результатов работы программы с печатными и с рукописными данными
- Определение наиболее подходящих параметров исходных изображений и настроек программы для улучшения распознавания

Глава 2

Теоретический материал

2.1 Что такое OCR

OCR (Optical character recognition) – это программа для перевода изображений печатного или рукописного текста в текстовые данные. Она используется при работе с отсканированными документами.

Технология OCR способна экспортировать формат и размер текста. Она применяется для преобразования копий документов в любой популярный электронный формат.

Например, при сканировании больших многостраничных документов в цифровое изображение OCR программа преобразует документ в редактируемый файл.

Для распознавания протоколов мы воспользовались языком программирования Python. Также мы рассмотрели возможности технологии OCR, которые совместимые с нашим инструментом – PyTesseract.

Мы использовали библиотеку Python-Tesseract, которая является оболочкой для Google Tesseract-OCR Engine.

2.2 Что такое нейронные сети

Основной замысел нейронной сети заключается в том, чтобы спроектировать множество объединённых между собой "клеток мозга"в глубине вычислительной машины для реализации возможности научить ее угадывать, анализировать вещи, законы и следовательно действовать почеловечески. Особенностью “мозга” компьютера является то, что ему не нужно задавать определённые шаги, действия, то есть программирование не является объектом обучения. Он сам освоится, как человеческий мозг.

Мы выделили два основных вида нейронных сетей, первый – рекуррентный, а второй – сверточный. Он абсолютно подходит для наших целей, то есть обработки изображения, в отличии от первого, который хорошо распознаёт тексты, потому что наши данные – это печатные или рукописные файлы.

2.2.1 TensorFlow

TensorFlow – открытая библиотека, созданная Google для численных вычислений. Написана на Python и C++ и признана на рынке одним из лучших фреймворков. TensorFlow отлично подходит для сложных проектов, таких как создание нейронных сетей. На ее основе можно создавать программы распознавания голоса и изображений.

Достоинства TensorFlow:

- Множество ресурсов и интернете, большая документация
- Наличие TensorBoard – средство отслеживания процесса обучения и визуализации результатов
- Поддержка распределенного обучения
- Популярна среди разработчиков, используется во многих технических компаниях
- Поддержка обслуживания моделей
- Использование различных мощностей компьютера (GPU, поддерживающие технологию CUDA, CPU и TPU)

Недостатки TensorFlow:

- Высокий порог вхождения для начинающих разработчиков. Фреймворк является весьма низкоуровневым, требует много изначально подготовленного кода, сложный процесс отладки
- Python является единственным языком, на который поддерживает TensorFlow

2.2.2 CUDA

CUDA – Это программная технология, которая позволяет параллельно распределить вычисления графического процессора, произведённые фирмой Nvidia. Функции, ускоренные при помощи CUDA, можно вызывать из различных языков, в том числе Python.

Для полной поддержки технологии CUDA, которую мы хотим использовать для обучения нашей модели, необходимо:

1. Наличие видеокарты Nvidia, которая поддерживает CUDA. В нашем случае – это видеокарта GTX1650
2. Поколение видеокарты должно быть не старее Kepler – у нас Pascal
3. Обновить драйвер до последней версии (выше 410.0). В нашем случае установлена версия 516.01
4. В зависимости от версии драйвера установить соответствующую версию CUDA. Для нас актуальна версия 11.7
5. Установить соответствующую версию CUDA Toolkit, которая связана с CUDA и версией драйвера вашей видеокарты
6. Установить и настроить библиотеку cuDNN, которая также связана с версией CUDA

2.2.3 TPU и Colaboratory

TPU – тензорный процессор, принадлежит к виду нейронных процессоров. Создан корпорацией Google для применения вместе с библиотекой TensorFlow. Представляет собой нейронный, который является интегральной схемой, содержит 4 чипа, по 2 ядра. На каждом ядре выполняется копия графа TF, обучение происходит параллельно на всех ядрах при использовании репликации.

Colaboratory – облачная система, подобная jupyter notebook, где можно создавать блокноты для написания и выполнения кода, созданная Google для разработок в области машинного обучения.

Для работы выделяется виртуальная машина, с предустановленными библиотеками, такими как TensorFlow и Keras для обучения собственных моделей.

Именно поэтому в командной работе мы воспользуемся Colaboratory, так как помимо удобности в реализации проекта, большим плюсом являются мощности серверов. Наша система позволяет воспользоваться всего лишь 6 TFlops. То есть облачный сервис способен в 30 раз быстрее обучить нашу нейронную сеть, чем настольный компьютер. Однако есть и свои минусы:

1. Данные нужно хранить на Google Drive, так как виртуальная машина предоставляется на протяжении 12 часов, после чего все данные удаляются. Однако, ничто не мешает запустить ещё одну виртуальную машину

2. Потери производительности могут быть связаны из-за низкой скорости интернета, которая связывает ПК и сервером, то есть является bottleneck'ом (узким местом нашей цепочки)

2.2.4 Keras

Keras – это вспомогательная библиотека, написанная на Python, которая запускается поверх TensoFlow и других библиотек ML. Была разработана компанией Google для ускорения проведения различных экспериментов. Keras поддерживает множество слоёв нейронных сетей, например, сверточных, тех которые использовались нами в программе. Фреймворк весьма хорош в задачах для работы с речью, текстами, изображениями и т.д.

Преимущества:

- Удобен и прост в изучении
- Достаточно мало весит для построения сложных моделей глубокого обучения
- Эффективное прототипирование
- Может обучаться на нескольких GPU
- Модули полностью конфигурируемы
- Поддержка TPU от Google, GPU от NVIDIA и Open-CL

Недостатки:

- Не всегда легко кастомизировать
- Ограничен бэкэндами CNTK, Tensorflow и Theano

В данном проекте для обучения сверточной нейронной сети, которая применялась для распознавания рукописных цифр, был выбран модуль Keras, так как он является функциональным, легким и поддерживает возможности TensorFlow.

2.3 Работа с изображениями

2.3.1 PIL

PIL (Python Imaging Library) предоставляет Python широкий спектр возможностей по работе с изображениями: повороты, изменения размера и другие полезные функции. Библиотека станет хорошим фундаментом для общей обработки и предобработки изображений, например, преобразование цветового пространства и фильтрация с набором встроенных ядер свертки.

2.3.2 OpenCV

OpenCV – библиотека компьютерного зрения, обладающая обширными возможностями, связанными с машинным обучением и распознаванием образов. Была разработана в корпорации Intel в 1999 году.

Использует множество языков программирования, например C++, Python, Java и др. В данный момент в разработке интерфейсы операций с высокой скоростью при помощи CUDA и OpenCL.

OpenCV-Python имеет Numpy, как встроенную библиотеку, которая представляет собой высокоскоростной инструментарий для числовых операций и работы с массивами.

OpenCV-Python - это API Python для OpenCV, сочетающий в себе лучшие качества API OpenCV C++ и языка Python. Использование Numpy делает проще работу с другими библиотеками, работающими с ним, такими как SciPy и Matplotlib.

Глава 3

Практическая часть

3.1 Поворот картинки

Первая задача, которую нужно решить при обработке фотографии – это выявление и устранение наклона таблицы. Для этого используется метод суммирования строк.

Фотография – это двумерный массив, хранящий описание пикселей. При решении проблемы поворота мы работаем с уже преобразованным к бинарной форме изображением, следовательно, каждый пиксель может принимать значение 0 или 1. Мы можем суммировать значения в каждой строке и получить одномерный массив со значениями "насыщенности цвета" для каждой строки. Данный массив в коде программы сохраняется в переменную histogram.

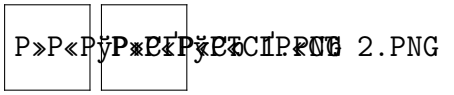

Рис. 3.1: Фотография с наклоном и без наклона

Далее, нужно определить дельту между максимальным и минимальным значениями в массиве. Чтобы определить угол, на который повернута таблица, перебираем углы от 0 до 360 с определенным шагом и запоминаем угол с максимальной дельтой. Это и будет угол, на который повернуто наше изображение.

```
score = np.sum((histogram[1:] - histogram[:-1]) ** 2, dtype=float)
best_angle = angles[scores.index(max(scores))]
```

Для поворота изображения в библиотеке OpenCV существуют функции getRotationMatrix(center, angle, scale) и warpAffine(img, M, size, ...).

```
M = getRotationMatrix(center, angle, scale)
```

center – центр изображения
angle – угол, на который изображение должно быть повернуто
scale – множитель увеличения/уменьшения размера изображения
M – матрица преобразования, применяемая к изображению

```
img = warpAffine(img_init, M, size,      )
```

img_init – исходное изображение
M – матрица преобразования
size – размер полученного изображения
img – преобразованное изображение

С помощью этих функций:

1. Находим матрицу преобразования
2. Считаем размер изображения после поворота, чтобы избежать обрезания краев
3. Применяем афинное преобразование к фотографии, чтобы развернуть ее в правильное положение

Таким образом, фотография готова для дальнейшего считывания таблицы.

3.2 Основной алгоритм (OpenCV + PyTesseract)

Алгоритм решения задачи делится на два ключевых этапа:

1. Разбиение таблицы на ячейки

2. Распознавание содержимого каждой ячейки

И первый пункт является самым объёмным, из-за универсального подхода к распознанию структуры таблицы и нужды предусматривать различные исключения на каждом этапе.

Основные функции, применяемые в программе (main.py) хранятся в модуле package.py

Первым делом считываем фото и производим предобработку исходной картинки для более точного распознавания -> reading_photo(...):

1. Считываем фото из файла -> cv2.imread(...)
2. Поворот картинки с помощью функции rotate_to_correct(...) по алгоритму описанному выше
3. Увеличивается размер фото минимум до 600 пикселей в высоту
4. Сводим фото к оттенкам серого
5. Размываем фото -> cv2.erode(...)
6. Избавляемся от шумов, сведением фото к двум цветам - чёрному(фон) и белому(текст) -> cv2.adaptiveThreshold(...)

Разбиение таблицы на ячейки:

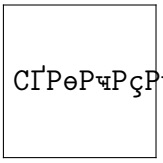

Рис. 3.2: Исходная таблица

1. Распознание вертикальных линий, формирующих таблицу
-> vertical_lines_detection(...).

Основой для выполнения конкретной задачи является функция, которая распознаёт на фото прямоугольники с размером ядра, задающим вертикальную линию - минимальная ширина по оси X.

```
cv2.getStructuringElement(cv2.MORPH_RECT, (1, kernel_len))
```

Рис. 3.3: Вертикальные линии

2. Распознание горизонтальных линий, формирующих таблицу
-> horizontal_lines_detection(...).

Данная часть выполняется аналогично функцией, которая распознаёт на фото прямоугольники с размером ядра, задающим горизонтальную линию - минимальная ширина по оси У.

```
cv2.getStructuringElement(cv2.MORPH\_RECT, (kernel\_len, 1))
```

Рис. 3.4: горизонтальные линии

3. Объединение горизонталей и вертикалей, для формирования цельной таблицы без содержимого -> combining_horizontals_and_verticals(...).

Применяем функцию, которая наложит полученные ранее фото вертикальных и горизонтальных линий в равном соотношении прозрачности.

```
cv2.addWeighted(vertical\_lines, 0.5, horizontal\_lines, 0.5, 0.0)
```

Рис. 3.5: Совмещенные линии

4. Находим контуры таблицы, полученной на предыдущем шаге, таким образом мы получим контуры всех прямоугольников таблицы - наши будующие ячейки -> cv2.findContours(...).

Используем режим группировки CV_RETR_TREE — группирует контуры в многоуровневую иерархию и метод упаковки CV_CHAIN_APPROX_SIMPLE — склеивает все горизонтальные, вертикальные и диагональные контуры.

5. Сортируем контуры таблицы по порядку их расположения в таблице (сверху вниз) -> sort_contours(...).

6. Формируем из контуров список ячеек с рамками -> form_cells(...).

Данный этап программы можно разделить на подпункты, ведь именно здесь происходит обработка всевозможных исключений, который позволят исключить из списка контуров ненужные - те, которые были выявлены или из-за ошибки программы, или из-за заранее известного фактора:

(a) Имея список контуров, переводим их в список прямоугольников задающих координаты ячеек таблицы -> cv2.boundingRect(...), и добавляем в список лишь те, значения ширины и высоты которых не превосходят половины от размеров всего фото - такие ячейки будут представлять из себя контур всей таблицы, а она не является ячейкой таблицы, которую стоит рассматривать

(b) Удаляем ячейки, которые лежат внутри других - они определенно лишние, такого не должно быть в таблице. Подобное может возникнуть из-за большого разрешения содержимого ячеек, которые могут сами быть приняты за таковые, так как в них могут распознаваться вертикали и горизонтали

(c) Проверяем лежат ли две ячейки в одном столбце и в одной строке - погрешность, из-за черты, которую человек мог провести в какой-то строке. И мы удаляем нижнюю из этих двух, а верхнюю ячейку расширяем, добавляя высоту нижней

Рис. 3.6: Выделение ячеек

7. Разбиваем список рамок на ячейки по строкам -> rows(...).

До тех пор, пока поле не отличается больше, чем его собственное (высота + среднее значение / 2), поле находится в том же ряду. Как только разница в высоте становится больше текущей (высота + среднее значение /2) мы знаем, что начинается новая строка.

8. Сортируем строки в каждом столбце (слева направо) -> rows_sort(...).

Распознавание значений в ячейках -> recognition(...)

1. Рассматриваем циклично содержимое каждой ячейки
2. Подгоняем содержимое для лучшего распознавания расширением и размыванием -> cv2.dilate(...) cv2.erode(...)
3. Анализируем содержимое с помощью технологий компьютерного зрения и формируем список значений на соответствующих позициях -> pytesseract.image_to_string(...)

Язык для распознавания содержимого устанавливаем русский: lang='rus'

Также делаем дополнительную обработку ячейки, если её значение пустует. Это может означать, что ячейка содержит единственный символ, который функция pytesseract.image_to_string(...) не способна распознать при базовой конфигурации. Поэтому необходимо условие с дополнительным считыванием при следующих параметрах:

lang= 'rus'

config= '–psm 10 –oem 1 -c tessedit_char_whitelist=123456789+-'

Последним пунктом, конечно же, является запись матрицы значений в excel файл -> data_to_excel(...).

Вспомогательное программное обеспечение

Программа написана на языке программирования Python с использованием различных его модулей:

- Matplotlib.pyplot – модуль для визуализации данных: выводим распознанные горизонтали и вертикали, саму таблицу, ячейки с рамками
- NumPy – необходим для упрощения работа со списками данных
- Pandas – библиотека применяется для записи таблицы распознанных значений в excel документ
- Scipy – модуль применяется в функции поворота картинки

3.3 Алгоритм на TensorFlow

Данный способ считывания таблицы с фотографии представляет собой обучение и использование сверточной нейронной сети с помощью tensorflow и Keras для определения положения и границ таблицы и применение pytesseract для непосредственного извлечения данных из нее.

Основные составляющие программы:

1. Подготовка датасета для обучения нейронной сети
2. Описание модули и ее обучение
3. Применение модели для распознавания таблиц на фотографиях

Подготовка датасета для обучения сверточной нейронной сети

Для обучения сверточной нейронной сети, которая должна находить таблицу на фотографии, используется датасет под названием “Marmot Dataset”. Он содержит изображения и их описания в формате xml.

В каждом описании хранятся:

1. координаты и размеры таблицы
2. координаты и размеры столбцов таблицы

Пример изображения из датасета:

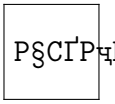

Рис. 3.7:

Пример описания таблицы:

Рис. 3.8:

Пример описания столбцов:

Рис. 3.9:

На аннотациях к фотографиям с таблицами содержится информация о расположении самой таблицы и о расположении ее столбцов.
В аннотациях расположение таблицы закодировано шестнадцатеричным числом, поэтому нам нужно сначала его декодировать с помощью функции hex2double.

```
def hex2double(hexcode):
```

hexcode – шестнадцатеричное число
return координаты расположения таблицы

Для определения границ таблицы по описанию из датасета используется функция return_bboxes_v1.

```
def return_bboxes_v1(xml, width, height):
```

xml – путь к файлу в формате xml
width, height – размеры изображения
return bboxes – координаты границ таблицы

Аналогично выглядит функция определения границ столбцов таблицы.

```
def return_bboxes_extended(xml):
```

xml - путь к файлу в формате xml
return width, height – размеры изображения
bboxes – координаты границ таблицы

Далее, с помощью функции create_mask создается маска таблицы, на которой показано ее расположение на фотографии, а также маска столбцов. Функция записывает маски изображений в отдельные папки.

```
def create_mask(img_path, list_of_bboxes, dim, mode='table'):
```

img_path – путь к изображению
list_of_bboxes – список, хранящий границы таблицы и столбцов таблицы
mode – параметр работы функции: создание маски таблицы или ее столбцов

Примеры масок для изображения:

Рис. 3.10: Маски таблицы и ее столбцов

Теперь изображения, маски и размеры изображений сохраняются в отдельный датафрейм и записываются в csv файл для последующего удобного обращения к ним. Для этого используется функция compute_masks. Функция записывает и сохраняет csv файл с информацией об изображениях.

```
def compute_masks(images_dir):
```

images_dir – путь к папке с изображениями

Описание модели и обучение

Для обучения модели потребуются подготовленные ранее данные. Для этого считываем их из созданного csv файла в переменную train_df. Создаем tensorflow-датасет для последующей загрузки в модель в переменной dataset.

Для подготовки изображений используется функция _parse_function.

```
def _parse_function(image, mask, colmask):
```

image – путь к изображению
mask – путь к маске с таблицей
colmask – путь к маске со столбцом

Загружает изображения в память, нормализует их, изменяет размер. Возвращает изображения и словарь с масками.

Создаем датасет для передачи модели для обучения, сохраняем его в переменную train_dataset.
Для проектирования и обучения модели машинного обучения используется библиотека TensorFlow.

Для увеличения эффективности модели используется заранее пред обученная модель компьютерного зрения “DenseNet”. Исключение (или

dropout) нейронной сети равно 0.6 для избежания переобучения.

Определение слоев нейронной сети задается классами

```
TableConvLayer(Layer)
ColumnConvLayer(Layer)
```

Создание модели происходит в функции build_tablenet.

```
def build_tablenet():
    return model
```

Запуск обучения модели:

```
model.fit(
    train_dataset, epochs=100,
    steps_per_epoch= STEPS_PER_EPOCH,
    validation_data=val_dataset,
    validation_steps= VALIDATION_STEPS,
    callbacks=callbacks
)
```

Применение модели к фотографиям для распознавания таблиц

Теперь, когда нейронная сеть обучена для распознавания таблиц нужно применить ее на конкретной фотографии.

Для этого загружаем изображение с помощью функции из библиотеки PIL Image.open(image_filename), нормализуем его, изменяем размер.

С помощью модели получаем маску изображения, на которой показано расположение таблицы:

```
tab_mask, _ = model.predict(np_image)
```

Находим границы прямоугольника по этой маске и вырезаем его по этим координатам на исходном изображении. Таким образом мы вырезали из всей фотографии лишь нужную нам часть – таблицу.

```
image_orig = image_orig
x, y, w, h = cv2.boundingRect(tab_mask)
```

```
tab = image_orig.crop((x, y, x + w, y + h))
```

Теперь к выделенной части фотографии применяем функцию из библиотеки pytesseract – image_to_string(tab), которая считывает текст с изображения и возвращает строку с текстом.

```
text = pytesseract.image_to_string(tab)
```

Для последующей работы с текстом используется функция text_to_ecxel. Далее приведено описание ее работы.

1. В полученной строке избавляемся от лишних символом в начале и конце. Затем делим ее на несколько строк, соответствующих строкам таблицы с помощью метода split() используя в качестве разделителя \n
2. Каждую полученную строку делим на слова, аналогичным образом, используя в качестве разделителя пробел
3. Зная общую структуру протокола, мы можем записать полученные данные в нужные нам ячейки
4. Полученное разбиение на ячейки организуем в датафрейм и записываем в excel файл

Глава 4

Результаты

4.1 Оценка эффективности каждого подхода реализации программы

Эффективность работы программы на конкретном фото протокола рассчитывается, как отношение количества правильно считанных ячеек таблицы, к количеству идеально распознанных ячеек в процентах.

В работе представлено два алгоритма решения поставленной задачи:

1. Использование OpenCV для считывания структуры таблицы и PyTesseract для считывания данных
2. Использование нейронной сети, обученной с помощью TensorFlow и Keras и считывание данных с помощью PyTesseract

Применив эти алгоритмы к тестовым фотографиям, мы получили следующие результаты.

4.1.1 OpenCV + PyTesseract

Эффективность работы с печатными данными:

0 - Всевозможные конфигурации параметров функции image_to_string() не позволили распознать крестики и нолики (записанные буквами X и O соответственно)

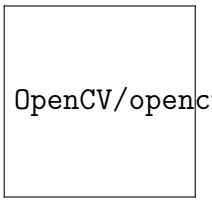

Рис. 4.1: Исходник

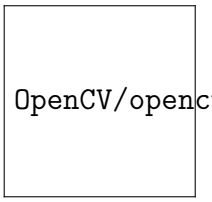

Рис. 4.2: Результат

1 - эффективность 100% - вместо крестиков и ноликов было решено использовать плюсики и минусы, их система способна распознать

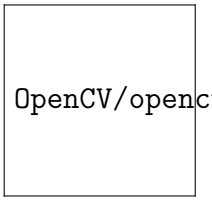

Рис. 4.3: Исходник

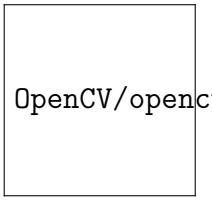

Рис. 4.4: Результат

2 - эффективность 97.2% - 3 ячейки считаны неправильно

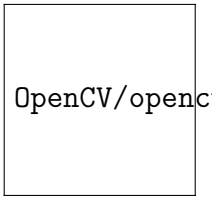

Рис. 4.5: Исходник

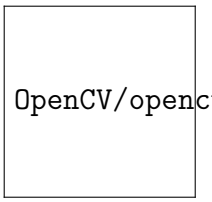

Рис. 4.6: Результат

Эффективность работы с данными, введенными с графического планшета:

1 - эффективность 98.6% - 2 ячейки не считаны (максимальная эффективность, возможная при рукописном заполнении таблицы - сделано перебором рукописных символов)

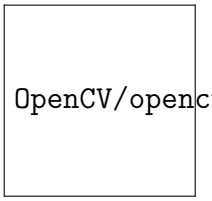

Рис. 4.7: Исходник

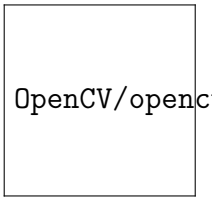

Рис. 4.8: Результат

2 - эффективность 62.9% - 2 ячейки не считаны (не улучшалось содержимое до хороших показателей эффективности)

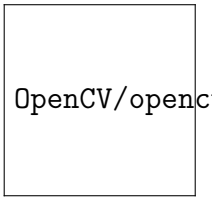

Рис. 4.9: Исходник

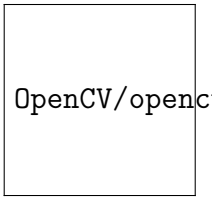

Рис. 4.10: Результат

Эффективность работы с данными, заполненными от руки:

1 - эффективность 57%

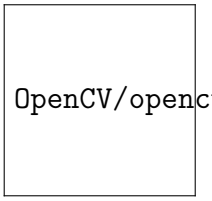

Рис. 4.11: Исходник

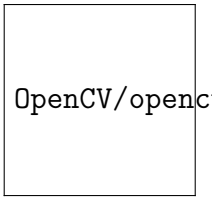

Рис. 4.12: Результат

Вывод: Программа, написанная с полным опором на модуль OpenCV показала внушительный результат. Она идеально считала структуру, приведенных таблиц и с хорошей эффективностью смогла считать содержимое таблицы. Печатные символы считываются идеально. А рукописные символы, записанные на графическом планшете, считываются немного лучше, чем те, что пишутся ручкой на распечатке с дальнейшим сканированием. Также примечательно то, что рукописи можно довести до такого состояния, чтобы программа могла их считывать почти со стопроцентной эффективностью.

4.1.2 TensorFlow + PyTesseract

Эффективность работы с печатными данными:

0

Рис. 4.13: Исходник

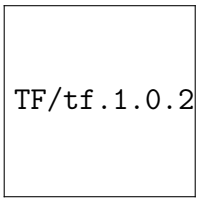

Рис. 4.14: Результат

1

Рис. 4.15: Исходник

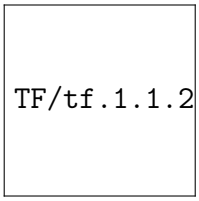

Рис. 4.16: Результат

2

Рис. 4.17: Исходник

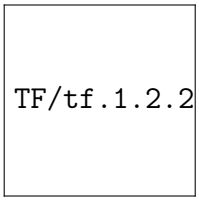

Рис. 4.18: Результат

Эффективность работы с данными, введенными с графического планшета:

1

Рис. 4.19: Исходник

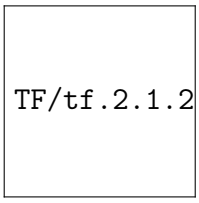

Рис. 4.20: Результат

2

Рис. 4.21: Исходник

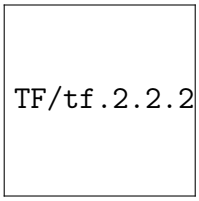

Рис. 4.22: Результат

Эффективность работы с данными, заполненными от руки:

1

Рис. 4.23: Исходник

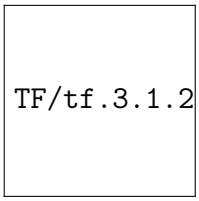

Рис. 4.24: Результат

Вывод: При обработке изображений с помощью алгоритма на Keras/TensorFlow теряется структура изображения и нет смысла говорить об эффективности. Данные, заполненные от руки не считались ни в одном случае.

4.2 Итоги

Сравнив результаты работы двух способов решения, поставленной перед нами задачи, можно понять, что алгоритм на OpenCV уже очень хорошо справляется со своей задачей. Однако алгоритм, реализованный через TensorFlow, нуждается в доработке, поэтому он уступает первому.

Были получены следующие результаты:

- Разработан алгоритм распознавания с использованием OpenCV + PyTesseract
- Разработан алгоритм распознавания с использованием Keras/TensorFlow + PyTesseract
- Проведено сопоставление эффективности работы двух алгоритмов на основе разных инструментов
- Определены наилучшие методологии работы распознавания на основе используемых инструментов

4.3 Будущее проекта

Приведённые в проекте программы имеет внушительные перспективы по доработке и улучшению качества точности распознавания:

- Доработка алгоритма детекции разметки таблицы при помощи технологий OpenCV:
  - Улучшить способ считывания некачественных фотографий протоколов. Например, если фото имеет большое количество помех: тени и грязная бумага
  - Научить программу достраивать таблицу при отсутствии на фото её кусочков, например, уголка или границ таблицы
  - Создание возможность для считывания таблиц любых типов, например, если имеются ячейки, растянутые на несколько столбцов

- Проектирование собственной нейронной сети, обученной на специализированном датасете в качестве замены PyTesseract, что должно значительно улучшить распознавание рукописных символов, особенно специальных, вроде крестика и нолика
- При глубинном изучении возможностей такого крупного пакета, как TensorFlow, возможно значительное улучшение работы программы при кардинальной её доработке и составлении новых тренировочных данных для модели распознавания таблиц

4.4 Примечания

Ссылка на репозиторий проекта: `https://github.com/Alex-Karma/ML_Project`