

Пояснительная записка

к дипломной работе на тему:

**"Распознавание компьютерного текста на английском и русском языке с экрана монитора"**

Автор: Коробов Алексей

Группа: DLL-19

# Постановка задачи

Компании всё чаще используют гибридный подход работы для своих сотрудников. Это означает что количество удаленной работы выросло в разы. Одним из основных способов коммуникации сотрудников являются мессенджеры и звонки.

Среди программистов очень популярная практика это показать исходный код программы своему коллеге и попытаться решить проблему совместно. Иногда бывают ситуации, когда этот исходный код нужно проверить самостоятельно, попросив коллегу передать код, который показан на экране.

Так же бывают ситуации, когда показывают презентацию и предоставляются ссылки на какие-то материалы.

Или при просмотре какого-то видео сформировать конспект из представленного текст.

Подобные проблемы могут возникать во многих областях.

И все они сводятся к распознаванию текста на экране.

Задача, решаемая в данной работе, состоит в детекции и распознавании текста с изображения экрана монитора.

**Сформулируем критерии:**

* Распознавании компьютерного текста на английском и русском языке включая спецсимволы
* Сохранение последовательности слов
* Сохранение отступов от левого края в виде пробелов
* Фон может быть светлым и темным
* Различный шрифт и размер текста

# Анализ существующих решений

Рассмотрим несколько готовых решений.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Ссылка | Плюсы | Минусы |
| **Pytesseract** | [https://pypi.org/ project/pytesseract/](https://pypi.org/project/pytesseract/) | * Готовое решение для использования. | * Закрытое решение, невозможно расширять * Устаревшая * Качество распознавания не высокое * Не поддерживает русский язык |
| **EasyOcr** | [https://github.com/ JaidedAI/EasyOCR](https://github.com/JaidedAI/EasyOCR) | * Реализовано на PyTorch * Поддерживает много языков (включая русский) * Высокое качество распознавания | * Не умеет строить последовательный текст (выдает отдельные слова) * Пропускает спец символы например “{“ * Медленно работает |
| **keras\_ocr** | [https://github.com /faustomorales/keras-ocr](https://github.com/faustomorales/keras-ocr) | * Высокое качество распознавания | * Использует TensorFlow * Последнее обновление в декабре 2022 |

Для поставленной задачи больше всего подходит **EasyOcr**, но такие недостатки как – пропуск символов или отсутствие построения последовательности слов означает, что необходимо дообучать модель или изменять промежуточные процессы при распознавании.

Поэтому для решения поставленной задачи будут реализованы две модели машинного обучения – для детекции текста с картинки и распознавания модели с картинки.

# Использование данных

В качестве источника данных для обучения будет использоваться сгенерированный текст. Так же будут сгенерированы регионы позиции текста **- bounding box (bbox)**

|  |  |
| --- | --- |
| **Описание** | **Изображение** |
| Изображение с текстом, имеющее отступы и разный цвет текста |  |
| Смещение от начала координат |  |
| Различный фон |  |
| Различный шрифт |  |
| Помехи в виде горизонтальных или вертикальных линий |  |
| Использование больше текста |  |

# Методика реализации

## Выбор модели для детекции

Модели машинного обучения глубокой нейронной сети для детекции объектов на изображении бывают двух видов – однопроходные и двухпроходные. Так как однопроходные направлены на наиболее быстрый результат ценою качества, то выбираем двухпроходную, потому что для нашей задачи не требуется быстрого результата, но требуется высокое качество.

В статье Text Detection Forgot About Document OCR(1) сравниваются модели для детекции текста с электронных документов.

Сравниваются следующие модели:

* PAN
* DBNet
* CRAFT
* TextBPN++
* DBNet++
* DCLNet

Наилучшее качество среди представленных показывает - **TextBPN++** В статье Arbitrary Shape Text Detection via Boundary Transformer (2) описана реализация

Основной смысл данной сети — это обнаружение текста произвольной формы благодаря обнаружению границ текста при помощи итеративного оптимизированного преобразователя границ.

Модуль предложения границ, состоящий из многослойных расширенных сверток, будет вычислять важную предварительную информацию (включая карту классификации, поле расстояний и поле направления) для создания грубых предложений границ, направляя оптимизацию граничного преобразователя. Модуль граничного преобразователя использует структуру энкодер-декодер, в которой энкодер построен из многоуровневых блоков преобразователя с остаточным соединением, а декодер представляет собой простую линейную многоуровневую нейронную сеть.

Но после нескольких попыток адаптировать модель под детекцию текста поставленной задачи, выясняется, что эта модель не подходит по следующим причинам:

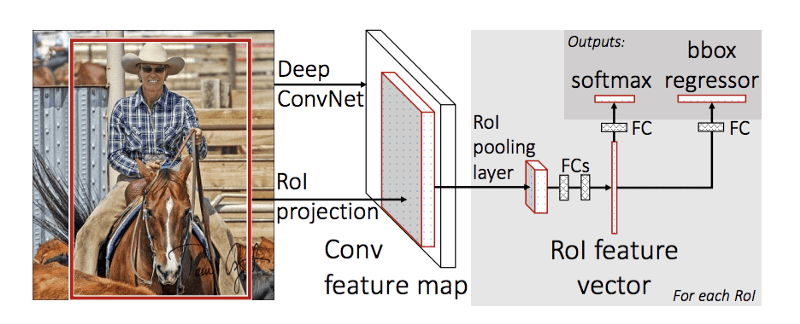
* Модель сильно ошибается на примерах с большим количеством слов
* Ошибается при поиске маленького шрифта
* Если строчки содержат только один символ, то модель пропускает такие символы

Такие недостатки данной сети приводит к тому что она не применима для данной задачи.

Поэтому для данной задачи требуется более простой подход для обнаружения текста.

Следующим кандидатом является **Fast R-CNN** (3), как более надежная модель для детекции объектов на изображении.

Основной смысл данной сети – изображение подается на вход сверточной нейронной сети и обрабатывается сверточной сетью. В итоге, имеем карту признаков и регионы потенциальных объектов. Координаты регионов потенциальных объектов преобразуются в координаты на карте признаков. Полученная карта признаков с регионами передается слою RoI (Region of Interest) polling слою. Здесь на каждый регион накладывается сетка размером HxW. Затем применяется MaxPolling для уменьшения размерности. Так, все регионы потенциальных объектов имеют одинаковую фиксированную размерность. Полученные признаки подаются на вход полносвязного слоя (Fully-conectedlayer), который передается двум другим полносвязным слоям. Первый с функцией активацией softmax определяет вероятность принадлежности классу, второй — границы (смещение) региона потенциального объекта.



Такая модель для детекции текста показывает хороший результат.

Дальнейшим развитием является модель **Faster R-CNN,** которая ускоряет модель Fast R-CNN за счет переиспользования карты объектов для расчета предложений регионов и классификации изображений.

То есть сверточная нейронная сеть используется для создания карты признаков изображения, которая одновременно используется для обучения сети предложений регионов и классификатора изображений. Благодаря этому совместному вычислению значительно улучшилась скорость обнаружения объектов.

## Реализация

В качестве реализации будем использовать сеть **Faster R-CNN** из официального источника pytorch модуля **torchvision. models. detection. faster\_rcnn. FasterRCNN** (4)

В качестве предобученой модели используется сеть ResNet50.

На вход подается изображение фиксированного размера **500x500px.** Это допущение необходимо для ускорения обучения детекции.

На выходе у сети меняем количество box\_predictor на 2 класса – фон и текст.

Генерируем текст.

Обучение происходит с использованием оптимизатора SGD

В качестве функции потерь используются различные показатели из стандартной реализации pytorch torchvision.

## Метрики

В качестве метрики используется площадь входного bbox и выходного по каждому блоку и считается MSE.

Например:

|  |  |
| --- | --- |
| Входной Bbox | Предсказанный Bbox |
|  |  |

Площадь считается как:

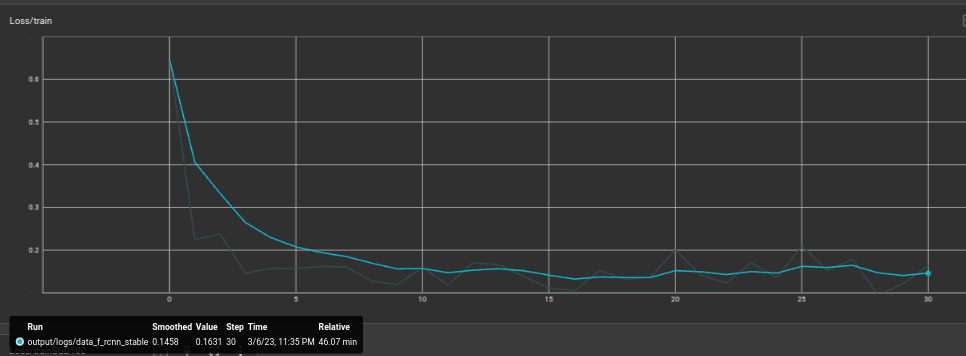
MSE это Mean squared error – средняя квадратичная ошибка, считается как:

где, n – количество блоков bbox.

Для нашего примера получаем:

Чем ниже будет число, тем выше точность детекции.

Результат:

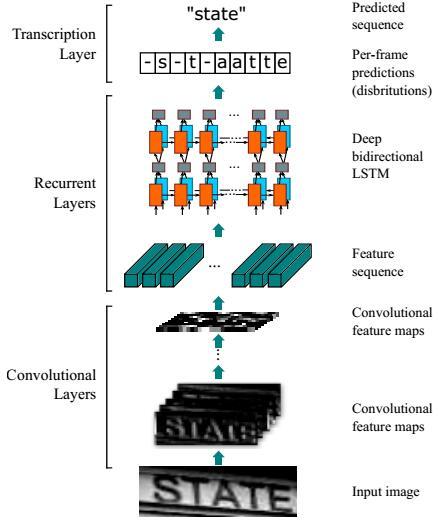


На графике видно, что функция потерь стабильно снизилась до минимума

## Выбор модели для распознавания

Для распознавания выбираем модель – CRNN (Convolutional recurrent neural network) (5)

В этой сети последовательно применяются операции свертки и макс-пулинга для извлечения полезных признаков. Затем они подаются в рекуррентный блок со стробированием (GRU/LSTM) для захвата временной информации. Выходные данные сети представляют собой сигмовидные оценки, которые указывают на несколько активных событий в текстовой последовательности.



Сетевая структура объединяет CNN + RNN

* Размер окна в Max pooling равен 1 \* 2, что гарантирует, что предлагаемая функция имеет горизонтальную длину, что способствует распознаванию более длинного текста;
* обучение CNN + RNN сложнее, поэтому добавляется BatchNorm, чтобы помочь модели сойтись;

**Преимущество**

* Возможно сквозное обучение;
* Нет необходимости выполнять операции сегментации символов и горизонтального масштабирования, только вертикальное масштабирование до фиксированной длины, и он может распознавать последовательности любой длины;
* Он может обучать модели на основе словарей и произвольные модели, не основанные на словарях;
* Скорость обучения высокая, а модель имеет небольшой размер.

## Реализация

Сеть имеет следующее представление:

|  |  |
| --- | --- |
| CNN | |
| (conv1) | Conv2d(3, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1)) |
| (relu1) | ReLU(inplace=True) |
| (pool1) | MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False) |
| (conv2) | Conv2d(64, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1)) |
| (relu2) | ReLU(inplace=True) |
| (pool2) | MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False) |
| (conv3\_1) | Conv2d(128, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1)) |
| (bn3) | BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True) |
| (relu3\_1) | ReLU(inplace=True) |
| (conv3\_2) | Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1)) |
| (relu3\_2) | ReLU(inplace=True) |
| (pool3) | MaxPool2d(kernel\_size=(2, 2), stride=(2, 1), padding=(0, 1), dilation=1, ceil\_mode=False) |
| (conv4\_1) | Conv2d(256, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1)) |
| (bn4) | BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True) |
| (relu4\_1) | ReLU(inplace=True) |
| (conv4\_2) | Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1)) |
| (relu4\_2) | ReLU(inplace=True) |
| (pool4) | MaxPool2d(kernel\_size=(2, 2), stride=(2, 1), padding=(0, 1), dilation=1, ceil\_mode=False) |
| (conv5) | Conv2d(512, 512, kernel\_size=(2, 2), stride=(1, 1)) |
| (bn5) | BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True) |
| (relu5) | ReLU(inplace=True) |
| RNN | |
| BidirectionalLSTM | |
| (lstm1) | LSTM(512, 256, bidirectional=True) |
| (embedding1) | Linear(in\_features=512, out\_features=256, bias=True) |
| BidirectionalLSTM | |
| (lstm2) | LSTM(256, 256, bidirectional=True) |
| (embedding2) | Linear(in\_features=512, out\_features=102, bias=True) |

В качестве входных данных подается картинка произвольной длины, но имеющая фиксированный размер высоты в **32px**.

Чтобы выровнить изображение по высоте используется стандартное масштабирование из библиотеки OpenCV – resize.

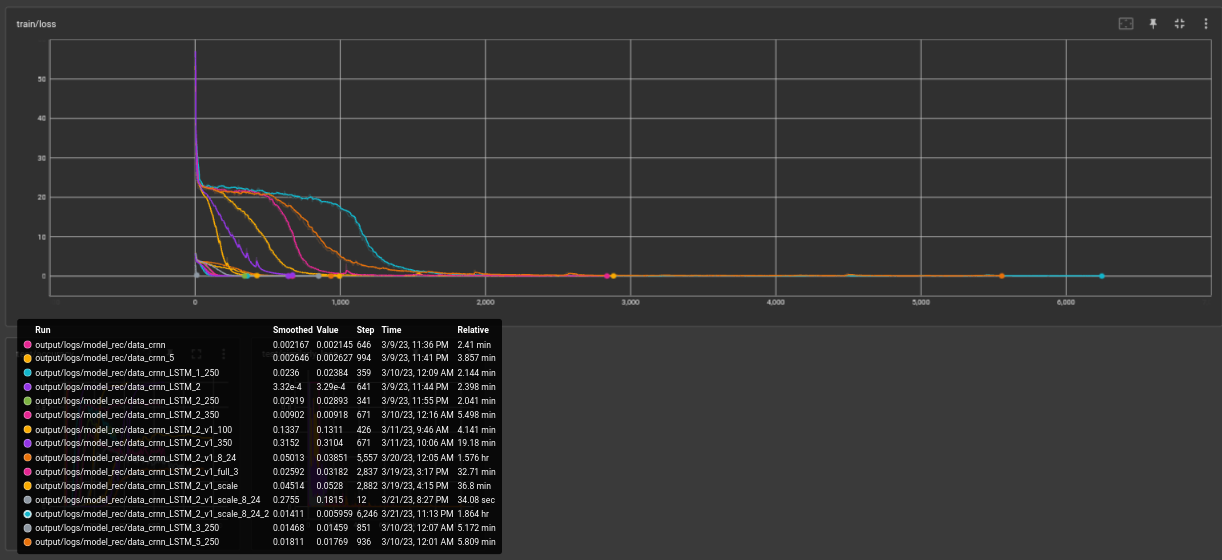
Для ускорения обучения, чтобы на вход сети подавать не одно изображение, а несколько (батч картинок), для этого картинка преобразуется по длине к одному размеру, добавляя в конец тензора **-1**.

В качестве функции потери используется CTC Loss (Connectionist Temporal Classification Loss) (5). На вход необходимо подавать количество символов для обучения. В нашем случае количество символов **101** (русские буквы, английские буквы, спецсимволы и цифры)

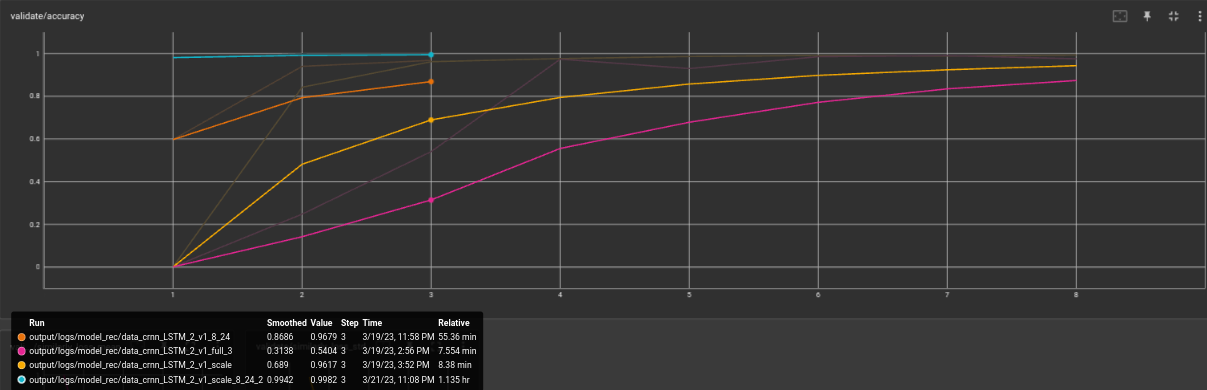
## Метрики

В качестве метрики используется accuracy – отношение количество правильно распознанных слов к общему количеству слов:

Было проведено множество экспериментов с изменением входных параметров:



Наилучшее качество даёт результат при обучении на 105000 изображений текста, с различным шрифтом от 12px до 24px на различном фоне и различным шрифтом. Качество на валидационной выборке составляет **0.9982**



# Постобработка результатов

После получения распознанного текста с картинки необходимо поставить слова в правильном порядке и добавить отступы.

## Сортировка Bbox

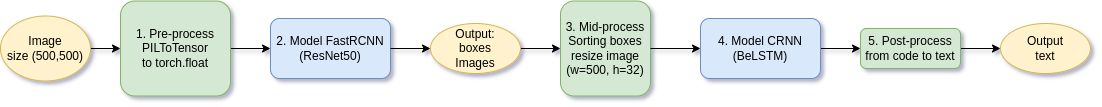
Сортировка должна происходить сверху вниз и слева на права.

## Добавление отступов слева

Отступы добавляются в виде пробелов от самого левого блока. Количество пробелов определяется от среднего количества пикселей на символ в каждом блоке.

# Общий процесс обработки и получения результатов

Последовательность обработки изображения выглядит следующим образом:



%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%223%22%20target%3D%227%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%223%22%20value%3D%221.%20Pre-process%26lt%3Bbr%26gt%3BPILToTensor%26lt%3Bbr%26gt%3Bto%20torch.float%22%20style%3D%22rounded%3D1%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%23d5e8d4%3BstrokeColor%3D%2382b366%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%2270%22%20y%3D%22192.5%22%20width%3D%22110%22%20height%3D%2295%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%224%22%20value%3D%22%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%225%22%20target%3D%223%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%225%22%20value%3D%22Image%26lt%3Bbr%26gt%3Bsize%20(500%2C500)%22%20style%3D%22ellipse%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%23fff2cc%3BstrokeColor%3D%23d6b656%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22-50%22%20y%3D%22207.5%22%20width%3D%2290%22%20height%3D%2265%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%226%22%20value%3D%22%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%2215%22%20target%3D%228%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%227%22%20value%3D%222.%20Model%20FastRCNN%26lt%3Bbr%26gt%3B(ResNet50%26lt%3Bspan%20style%3D%26quot%3Bbackground-color%3A%20initial%3B%26quot%3B%26gt%3B)%26lt%3B%2Fspan%26gt%3B%22%20style%3D%22rounded%3D1%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%23dae8fc%3BstrokeColor%3D%236c8ebf%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22220%22%20y%3D%22210%22%20width%3D%22120%22%20height%3D%2260%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%228%22%20value%3D%223.%20Mid-process%26lt%3Bbr%26gt%3BSorting%20boxes%26lt%3Bbr%26gt%3Bresize%20image%20(w%3D500%2C%20h%3D32)%22%20style%3D%22rounded%3D1%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%23d5e8d4%3BstrokeColor%3D%2382b366%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22510%22%20y%3D%22192.5%22%20width%3D%2290%22%20height%3D%2295%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%229%22%20value%3D%22%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%2210%22%20target%3D%2212%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2210%22%20value%3D%224.%20Model%26amp%3Bnbsp%3BCRNN%20(BeLSTM)%22%20style%3D%22rounded%3D1%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%23dae8fc%3BstrokeColor%3D%236c8ebf%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22650%22%20y%3D%22210%22%20width%3D%22120%22%20height%3D%2260%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2211%22%20value%3D%22%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%2212%22%20target%3D%2213%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2212%22%20value%3D%225.%20Post-process%26lt%3Bbr%26gt%3Bfrom%20code%20to%20text%22%20style%3D%22rounded%3D1%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%23d5e8d4%3BstrokeColor%3D%2382b366%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22810%22%20y%3D%22220%22%20width%3D%22100%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2213%22%20value%3D%22Output%26lt%3Bbr%26gt%3Btext%22%20style%3D%22ellipse%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%23fff2cc%3BstrokeColor%3D%23d6b656%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22950%22%20y%3D%22215%22%20width%3D%2290%22%20height%3D%2250%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2214%22%20value%3D%22%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%227%22%20target%3D%2215%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%3E%3CmxPoint%20x%3D%22340%22%20y%3D%22240%22%20as%3D%22sourcePoint%22%2F%3E%3CmxPoint%20x%3D%22510%22%20y%3D%22240%22%20as%3D%22targetPoint%22%2F%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2215%22%20value%3D%22Output%3A%26lt%3Bbr%26gt%3Bboxes%26lt%3Bbr%26gt%3BImages%22%20style%3D%22ellipse%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%23fff2cc%3BstrokeColor%3D%23d6b656%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22380%22%20y%3D%22215%22%20width%3D%2290%22%20height%3D%2250%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2216%22%20value%3D%22%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3BentryX%3D0%3BentryY%3D0.5%3BentryDx%3D0%3BentryDy%3D0%3B%22%20edge%3D%221%22%20source%3D%228%22%20target%3D%2210%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%3E%3CmxPoint%20x%3D%22610%22%20y%3D%22240%22%20as%3D%22sourcePoint%22%2F%3E%3CmxPoint%20x%3D%22652%22%20y%3D%22240%22%20as%3D%22targetPoint%22%2F%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

**Image** – на вход подается изображение (для обучения использовался размер 500х500px)

**1. Pre-process** – изображение обрабатывается – преобразуется в тензор, элеменды приводятся к типу float

**2. Model FastRCNN** – на вход получает изображение, а на выход список изоражений по регионам, содержащие текст, и сами регионы

**3. Mid-process** – каждое изображение масштабируется до размера w=500px, h=32px

**4. Model CRNN (BeLSTM)** – массив изображений подается на вход сети CRNN, на выходе получаем вероятности символов согласно входному алфавиту. Из этих вероятностей получаем максимальные значения, и по индексам получаем текст.

**5. Post-process** – последним этапом является пост обработка, которая сортирует блоки и добавляет отступы от левого края.

# Примеры результатов

|  |  |
| --- | --- |
| Исходное изображение и результат детекции | Результат распознавания |
|  |  |
|  |  |
|  |  |
|  |  |

# Вывод

Текущая реализация распознавания текста с картинки имеет высокий потенциал.

Как видно на примерах хорошо справляется с изображениями, на которых изображен исходный код. Без труда распознаются одинокие символы и слова.

Но при этом при большом количестве тексте на изображении модель детекции начинает выдавать больше регионов чем есть по факту. Поэтому модель распознавания выдает не верный результат.

Дальнейшие шаги для улучшения:

* Попробовать использовать модель CRAFT для улучшения детекции текста
* Улучшить постобработку, в случае когда блоки «наезжают» друг на друга, текст может быть распознан не правильно или по два раза одна и та же буква. Можно попробовать объединять два таких блока перед распознаванием или соединять буквы после распознавания.
* Увеличить размер входной картинки.

# Источники

1. *Text Detection Forgot About Document OCR (arxiv.org) - https://arxiv.org/pdf/2210.07903.pdf.* 2023.

2. *Arbitrary Shape Text Detection via Boundary Transformer https://arxiv.org/abs/2205.05320*. 2022.

3. *Fast R-CNN - https://arxiv.org/abs/1504.08083*. 2015.

4. *FASTER R-CNN in Torchvision - https://pytorch.org/vision/main/models/faster\_rcnn.html*. б.д.

5. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks - https://www.cs.toronto.edu/~graves/icml\_2006.pdf