Автоматизация решения CAPTCHA в текстовом формате

Студент 5.306М группы: Лаптев А. В.
Научный руководитель: Калачев А. В.

27 апреля 2025 г.

Цель и задачи работы

Целью работы является: разработать, реализовать и протестировать программу для автоматизированного распознавания текстовых CAPTCHA.

Задачи, которые необходимо решить для достижения цели:

1. изучить принципы реализации текстовых CAPTCHA на основе открытых источников (допустимые символы, применяемые искажения);
2. выбрать архитектуру нейронной сети, наиболее подходящую для распознавания CAPTCHA;
3. подготовить датасет изображений CAPTCHA с учётом возможных искажений;
4. обучить нейронную сеть с достаточной точностью;
5. протестировать модель на тестовом наборе данных и оценить её эффективность.

Современная реализация текстовых CAPTCHA

Текстовые CAPTCHA обычно состоят из букв и цифр. Зачастую используются символы латинского алфавита (как прописные, так и строчные) и цифры от 0 до 9. Рекомендуемый набор символов в генераторах на некоторых CRM платформах выглядит следующим образом: ABCDEFGHJKLMNPQRSTWXYZ23456789. Длина последовательности символов обычно составляет от 4 до 8 символов.

Для усложнения автоматического распознавания текстовые CAPTCHA подвергаются различным искажениям:

1. геометрические искажения;
2. перекрытие символов;
3. добавление шума;
4. сложные фоны;
5. нелинейные искажения.

Подготовка датасета с изображениями

Для эффективного обучения необходимо, чтобы набор данных соответствовал следующим требованиям:

1. достаточное количество изображений для каждого символа;
2. разнообразие данных, включающее:
    1. различные углы наклона символов;
    2. вариативность написания символов и их искажения;
    3. наличие побочных визуальных элементов;
    4. использование различных шрифтов.
3. переменная длина последовательностей символов.

Для генерации синтетических CAPTCHA использовалась библиотека captcha на языке python. Данная библиотека поддерживает генерацию изображений с пользовательскими шрифтами и различными эффектами искажений, что исключает необходимость привлечения дополнительных инструментов.

Блок кода для генерации CAPTCHA

```python
from captcha.image import ImageCaptcha
from random import randint, shuffle
import numpy as np
import os
from textcaptcha.preprocessing_image import preprocessing_image

def generate_image(path_to_file: str, alphabet: list,
    number_of_start:int, number_of_captcha: int,
    size_of_image: tuple) -> list:
    # Генерация текстовых captcha
    text = ImageCaptcha(size_of_image[0], size_of_image[1],
    ['./fonts/arial.ttf', './fonts/comic.ttf', './fonts/cour.ttf',
    './fonts/georgia.ttf'])
    # Структура возвращаемого списка:
    # [filename, label, (width, height)]
    filenames = []
    for _ in range(number_of_start, number_of_captcha):
        captcha_text = [alphabet[randint(0, len(alphabet) - 1)]
        for _ in range(randint(4, 7))]
        shuffle(captcha_text)
        text.write(''.join(captcha_text),
        f'{path_to_file}/{"".join(captcha_text)}.png')
        filenames.append(
            [f'{path_to_file}/{"".join(captcha_text)}.png',
            ''.join(captcha_text)]
        )
    return filenames
```

```python
if __name__ == '__main__':
    # Алфавит допустимых символов
    alphabet = 'ABCDEFGHJKLMNPQRSTWXYZ023456789'
    # Создание директории для хранения полноценных
    # синтетических текстовых captcha
    path_to_dataset = '../datasets/captcha'
    if not os.path.isdir(path_to_dataset):
        os.mkdir(path_to_dataset)
    # Создаем датасет из нужного полноценных
    # синтетических captcha длиной от 4 до 7
    # символов размером 250x60
    filenames = generate_image(path_to_dataset,
    list(alphabet), 0, 100000, (250, 60))

    # Предобработка изображений
    preprocessing_image(filenames)

    # Для отладки без создания датасета с нуля
    numpy_data = np.array(filenames, dtype=object)
    np.save('data.npy', numpy_data)
```

Предобработка датасета с изображениями

После создания изображений все они прошли этапы предобработки, направленные на улучшение качества данных и повышение эффективности обучения модели.

Примеры сгенерированных и предобработанных CAPTCHA приведены на рисунке ниже:

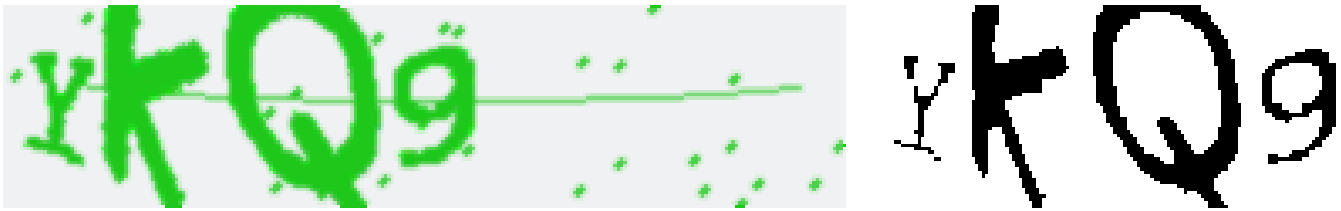

Рис. 1: Изображение CAPTCHA и результат обработки.

Выбор эффективной модели

Для распознавания текста с переменной длиной последовательности в задачах CAPTCHA наиболее часто применяются следующие архитектуры нейронных сетей:

1. оптическое распознавание символов (OCR);
2. рекуррентные сверточные нейронные сети (CRNN);
3. архитектуры последовательного обучения (Seq-to-Seq).

На начальных этапах экспериментов Seq-to-Seq-модель показала наилучшие результаты среди рассмотренных вариантов. В отличие от OCR- и CRNN-моделей, данная архитектура смогла достичь более высокой точности распознавания последовательностей символов, что обусловлено применением механизма внимания. Дальнейшая работа с моделью была сосредоточена на её оптимизации и улучшении параметров обучения.

Seq-to-Seq модель. Функция потерь

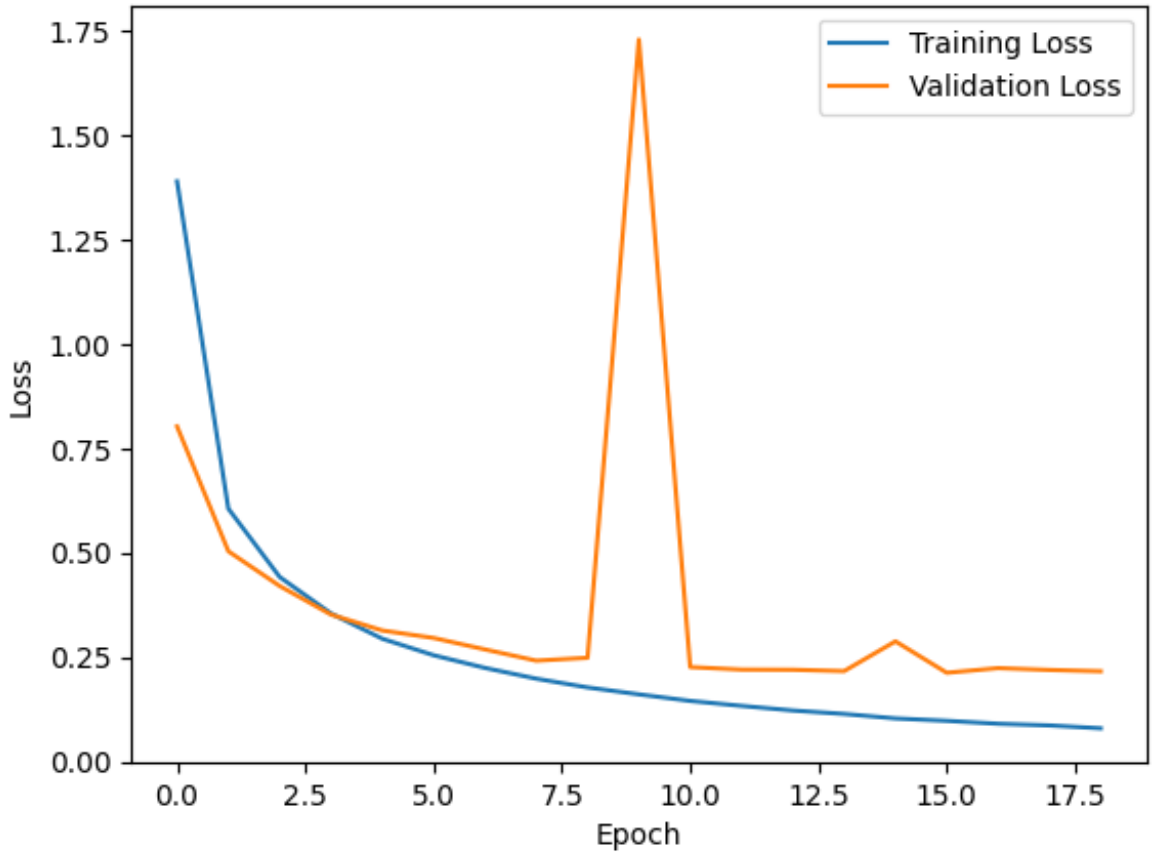

Рис. 2: График изменения значений функции потерь.

Seq-to-Seq модель. Точность предсказаний

Таблица 1: Точность предсказаний для последовательностей от 4 до 7 символов.

Длина последовательности	Точность распознавания
4 символа	0.9305
5 символов	0.7450
6 символов	0.4575
7 символов	0.1915

Seq-to-Seq модель. Матрица ошибок

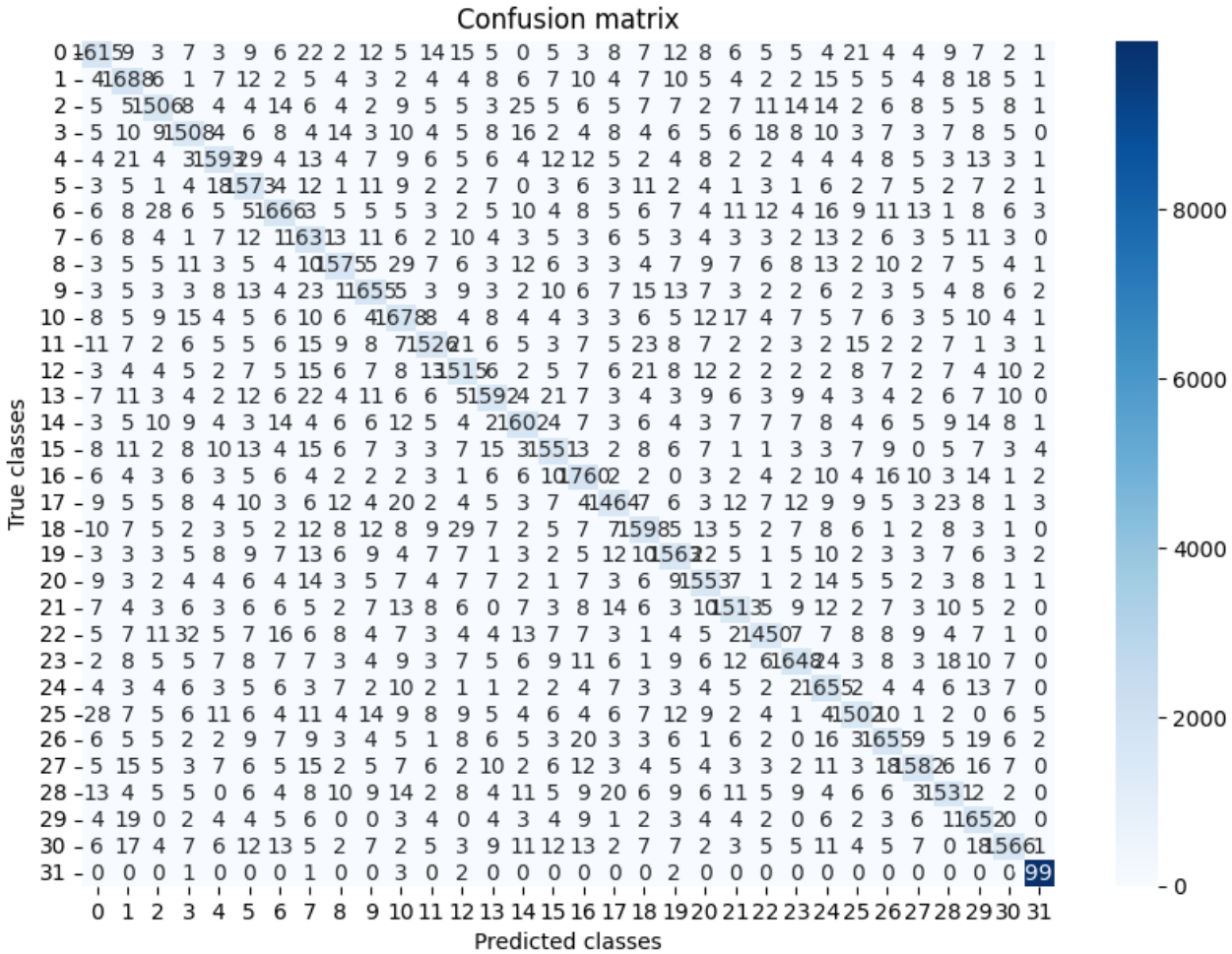

Рис. 3: Матрица ошибок для обученной модели.

Заключение

В ходе выполнения работы были решены задачи:

1. определен набор допустимых символов и искажений для создания датасета с текстовыми CAPTCHA;
2. выбрана модель нейронной сети Seq-to-Seq для распознавания текста с CAPTCHA;
3. подготовлен датасет из 100 000 изображений для обучения и тестирования модели нейронной сети;
4. обучена модель на изображениях с различной длиной последовательностей символов;
5. протестирована работа модели на тестовых изображениях.

В результате выполнения работы была создана модель, для распознавания текстовых CAPTCHA различной длины последовательности символов.