УДК 004.415

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ РАСПОЗНАВАНИЯ РУКОПИСНОГО ТЕКСТА НА ОСНОВЕ ГИБРИДНОЙ НЕЙРОСЕТЕВОЙ АРХИТЕКТУРЫ

Сафина Г.Ф.

к. ф.-м. н, доцент,

ФГБОУ ВО «Уфимский университет науки и технологий», Нефтекамский филиал,

Нефтекамск, Россия

Павлов Д.А.

студент,

ФГБОУ ВО «Уфимский университет науки и технологий», Нефтекамский филиал,

Нефтекамск, Россия

Аннотация

В статье описывается создание интеллектуальной системы для распознавания рукописного текста на русском языке. Рассматриваются проблемы, связанные с вариативностью почерков, отсутствием чёткой сегментации символов и шумами на изображениях. Предложена гибридная модель CRNN (свёрточная рекуррентная нейронная сеть), объединяющая извлечение признаков с помощью CNN и обработку последовательностей через двунаправленные слои LSTM. Приводится реализация системы на языке Python с использованием фреймворка PyTorch в среде Google Colab. Описаны этапы предобработки данных, формирования датасета, обучения модели и декодирования выходной последовательности. Проведена оценка качества распознавания с применением метрик CER и WER. Сравнение с классическими OCR-подходами показывает преимущества нейросетевого метода при работе с рукописными текстами.

Ключевые слова: распознавание рукописного текста, CRNN, LSTM, CTC loss, компьютерное зрение, Python, PyTorch, предобработка изображений, OCR.

DEVELOPMENT OF AN INTELLIGENT SYSTEM FOR HANDWRITTEN TEXT RECOGNITION BASED ON HYBRID NEURAL NETWORK ARCHITECTURE

Safina G.F.

PhD, Associate Professor,

Neftekamsk branch of the Ufa University of Science and Technology,

Neftekamsk, Russia

Pavlov D.A.

student,

Neftekamsk branch of the Ufa University of Science and Technology,

Neftekamsk, Russia

Abstract

The paper describes the creation of an intelligent system for handwritten text recognition in Russian. The problems associated with handwriting variability, lack of clear character segmentation, and image noise are discussed. A hybrid CRNN model (convolutional recurrent neural network) is proposed, combining feature extraction using CNN and sequence processing via bidirectional LSTM layers. The implementation in Python using the PyTorch framework in the Google Colab environment is presented. The stages of data preprocessing, dataset creation, model training, and output sequence decoding are described. The recognition quality is assessed using CER and WER metrics. Comparison with classical OCR approaches shows the advantages of the neural network method when processing handwritten texts.

Keywords: handwritten text recognition, CRNN, LSTM, CTC loss, computer vision, Python, PyTorch, image preprocessing, OCR.

Перевод рукописных документов в цифровой формат остаётся актуальной задачей для архивов, медицинских учреждений и государственных структур. Печатный текст распознаётся достаточно надёжно классическими OCR-системами (например, Tesseract), однако рукописный текст создаёт ряд трудностей: непостоянство формы и размера символов, соединение букв, наклон, шумы при сканировании. Традиционные методы сегментации и шаблонного сопоставления показывают на рукописях низкую точность. В последние годы предпочтение отдаётся нейросетевым подходам [1-5], которые обучаются непосредственно на изображениях и не требуют точного выделения каждого символа.

Целью данной работы является разработка интеллектуальной системы распознавания рукописного текста на русском языке с использованием гибридной архитектуры CRNN (Convolutional Recurrent Neural Network) и оценка её эффективности.

Анализ рукописных документов выявил несколько характерных проблем:

– символы одного автора могут различаться по высоте и ширине;
– соседние буквы часто сливаются, особенно в быстром почерке;
– многие буквы визуально близки («и» – «й», «ш» – «щ», «л» – «д»);
– фон документа содержит шумы, пятна, неравномерную яркость.

Эти особенности делают неэффективным классический конвейер «бинаризация → сегментация → выделение признаков → классификация». Ошибки на этапе сегментации быстро накапливаются, и итоговое распознавание становится неудовлетворительным. Выходом является применение end-to-end нейросетевых моделей, которые обрабатывают изображение целиком и выдают последовательность символов без явного разделения на символы.

В качестве базовой модели выбрана архитектура CRNN [1], которая хорошо зарекомендовала себя в задачах распознавания текста на изображениях. Она состоит из трёх основных блоков:

– свёрточный блок (CNN). Несколько слоёв Conv2D с активацией ReLU и max-пулингом. Из входного изображения размером 128×32 пикселя извлекаются пространственные признаки (линии, углы, фрагменты символов). На выходе формируется карта признаков размера (C × H' × W'), которая затем преобразуется в последовательность векторов (W' векторов размерности C·H');

– рекуррентный блок (RNN). Два двунаправленных слоя LSTM со 256 скрытыми нейронами в каждом направлении. Они обрабатывают полученную последовательность слева направо и справа налево, что позволяет учитывать контекст до и после текущего элемента. На каждом шаге сеть выдаёт распределение вероятностей по символам словаря;

– функция потерь CTC. Поскольку разметка не содержит точного указания, какой участок изображения какому символу соответствует, используется Connectionist Temporal Classification [2]. CTC вычисляет вероятность всех возможных выравниваний между входной последовательностью и целевой текстовой строкой. Это позволяет обучать модель без ручной сегментации.

Архитектура была реализована на Python с использованием библиотеки PyTorch 2.0 [4,5]. Ниже приведён фрагмент листинга, описывающего построенную модель.

```
class CRNN(nn.Module):
    def __init__(self, num_classes):
        super().__init__()
        self.cnn = nn.Sequential(
nn.Conv2d(1, 64, 3, padding=1), nn.ReLU(), nn.MaxPool2d(2,2),
nn.Conv2d(64,128,3, padding=1), nn.ReLU(), nn.MaxPool2d(2,2),
nn.Conv2d(128,256,3, padding=1), nn.ReLU(),
nn.Conv2d(256,256,3, padding=1), nn.ReLU(),
nn.MaxPool2d((2,1)),
```

```
     nn.Conv2d(256,512,3, padding=1), nn.BatchNorm2d(512),
nn.ReLU(),
     nn.Conv2d(512,512,3, padding=1), nn.ReLU(),
nn.MaxPool2d((2,1)),
              )
     self.rnn = nn.LSTM(512, 256, num_layers=2,
bidirectional=True, batch_first=True)
     self.fc = nn.Linear(256*2, num_classes)
         def forward(self, x):
     features = self.cnn(x)                  # [B, C, H, W]
     batch, C, H, W = features.size()
     features = features.view(batch, C*H, W)  # [B, feat_len, W]
     features = features.permute(0,2,1)       # [B, W, feat_len]
     out, _ = self.rnn(features)              # [B, W, 512]
     out = self.fc(out)                       # [B, W,
num_classes]
     return out
```

Листинг 1 – Фрагмент описания модели

Для обучения требовался набор изображений слов на русском языке. Поскольку готовых размеченных рукописных датасетов для кириллицы недостаточно, был применён синтез: текст писался стандартным русским шрифтом с последующим добавлением искажений (поворот, размытие, шум).

Использовался шрифт Cyrillic.ttf. Генерация выполнялась в Python с помощью библиотеки Pillow. Все изображения приводились к оттенкам серого, масштабировались до высоты 32 пикселя (ширина варьировалась, но затем фиксировалась 128 пикселей путём добавления белых полей). Нормализация: значения пикселей делились на 255, приводясь к диапазону [0,1].

Словарь символов включал: строчные и прописные русские буквы, цифры, знаки препинания, а также специальный токен CTC blank (индекс 0). Всего 82 класса.

Для загрузки данных реализован класс OCRDataset, который по пути к изображению и текстовой метке возвращает тензор изображения и целевую последовательность индексов. При помощи DataLoader из PyTorch формировались батчи с динамической паддингом меток (для CTC необходим список целей разной длины).

Обучение проводилось в среде Google Colab с активацией GPU Tesla T4. Параметры: оптимизатор Adam, скорость обучения 0.001, размер батча 32, количество эпох – 50. Функция потерь – torch.nn.CTCLoss. Дополнительно применялась аугментация: случайное изменение яркости, небольшой поворот (до ±5°), добавление гауссовского шума.

На рисунке 1 показан график изменения функции потерь на тренировочной и валидационной выборках. Значение loss монотонно снижалось с 18 до 2.5, что свидетельствует об успешной сходимости модели. Переобучение не наблюдалось благодаря валидации и аугментации.

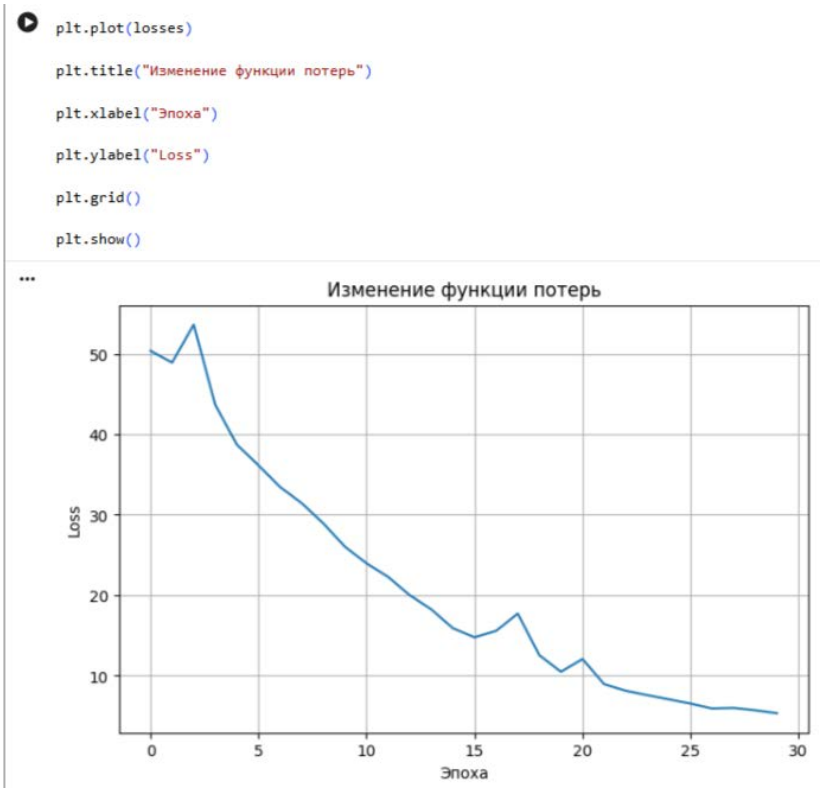

Рисунок 1 – График изменения функции потерь
в процессе обучения (авторская разработка)

Для оценки использовались метрики:

– CER (Character Error Rate) – минимальное количество замен, удалений и вставок символов, делённое на общее число символов в эталонном тексте;

– WER (Word Error Rate) – аналогично на уровне слов.

Тестирование проводилось на 200 изображениях, не участвовавших в обучении (разные почерки, включая реальные рукописные образцы, отсканированные с разрешением 300 dpi). Результаты в метриках: CER –12,4%; WER–28,7%.

Анализ ошибок показал, что большинство из них связано с неразличимостью похожих символов («ш» и «щ», «т» и «г») или с плохим качеством исходного изображения (низкая контрастность).

В качестве базового сравнения использовался Tesseract OCR v5 (режим --psm 7 – одна строка текста). На том же наборе тестовых рукописных изображений Tesseract показал CER = 34.1%, WER = 58.6%. Это значительно хуже, чем у предложенной модели. Причина – Tesseract ориентирован на печатные шрифты и не адаптирован к связному рукописному тексту без чёткой сегментации. Разработанная CRNN-модель благодаря рекуррентным слоям лучше улавливает контекст и вариативность почерка.

В ходе работы создана интеллектуальная система распознавания рукописного текста на русском языке, основанная на гибридной архитектуре CRNN. Система реализована на Python/PyTorch, использует GPU-ускорение в Google Colab, включает этапы предобработки, генерации датасета, обучения с CTC loss и декодирования.

Эксперименты показали, что модель достигает CER 12,4% на тестовых рукописных образцах, что существенно превосходит классический Tesseract (34,1%). Основные ошибки связаны со схожестью отдельных букв и недостаточным объёмом реальных рукописных данных. В перспективе предполагается расширить датасет за счёт реальных архивных документов,

добавить языковую модель для постобработки и рассмотреть трансформерные архитектуры (Vision Transformer + CTC).

Библиографический список:

1. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс; пер. с англ. –М.: Техносфера, 2012. – 1104 с.

2. Круглов, В. В. Искусственные нейронные сети. Теория и практика: учебное пособие для вузов / В. В. Круглов, В. В. Борисов. – 3-е изд., стер. – М.: Горячая линия–Телеком, 2017. – 382 с.

3. Паклин Н. Б. Интеллектуальный анализ данных как важнейший инструмент формирования интеллектуального капитала организаций / Н. Б. Паклин, В. И. Орешков. – СПб: Питер, 2019. – 464 с.

4. Баянов Э.И., Сафина Г.Ф. Аанализ байесовского подхода в машинном обучении / В сборнике: Достижения и приложения современной информатики, математики и физики. Материалы IX Всероссийской научно-практической конференции. – Уфа, 2021. – С. 7-10.

5. Основы программирования на языке Python: учеб. пособие / под ред. А. Ю. Богачева. – Краснодар: РЭА, 2023. – 135 с.