Обработка изображений рукописных документов методами компьютерного зрения

Т. М. Татарникова

Санкт-Петербургский государственный университет аэрокосмического приборостроения

tm-tatarn@yandex.ru

А. А. Шихотов

Санкт-Петербургский государственный университет аэрокосмического приборостроения

ashikhotov@bk

Аннотация. **В статье рассмотрено несколько способов к удалению клетки на листе бумаги с использованием библиотеки OpenCV языка Python: нахождение границ Кэнни, алгоритм преобразования Хафа, метод фильтрации изображения по цветовым характеристикам и комбинация методов. Предлагаемые способы и их комбинации являются важным этапом предобработки рукописного документа в задаче распознавания образов.**

Ключевые слова: рукописный документ, извлечение информации, шум, обработка изображений, компьютерное зрение

I. Введение

Анализ рукописных текстов в настоящее время рассматривается в нескольких приложениях: от оцифровки архивов и сортировки почтовых отправлений до идентификации личности и его настроения в момент написания текста.

Сама задача анализа рукописных текстов не является тривиальной, поскольку, несмотря на свою историческую значимость, рукописные тексты представляют собой сложный объект для анализа из-за разнообразия стилей письма, различных уровней четкости и наличия шумов.

В общем случае задача анализа рукописных документов относится к распознаванию образов, в которой этап предобработки является самым трудоемким, в частности, удаление шума, который возникает из-за различных факторов, таких как качество сканирования, условия освещения и физическое состояние документа.

Одной из частных проблем при обработке изображений рукописных документов является необходимость удаления рисунка клетки на клетчатой бумаге. Клетка может мешать распознаванию текста путём создания визуального шума, отвлекая внимание алгоритмов распознавания и ухудшая общую точность извлечения информации. Рассмотрено несколько способов удаления клетки на листе бумаги с использованием библиотеки OpenCV языка Python и непосредственно распознавание текст.

II. Предобработка изображения рукописного документа

Первый способ – применение алгоритма обнаружений границ Canny. Границами являются такие кривые на изображении, вдоль которых происходит резкое изменение яркости или других видов неоднородностей. Причинами возникновения границ являются изменение освещенности, цвета и глубины сцены.

В зависимости от требований к решаемой задаче возможно выделить большинство границ или только те границы, которые имеют четкое начертание. Это все становится возможным при изменении параметров порогового значения гистерезиса [2]. В результате применения данного алгоритма можно на некоторых изображениях добиться игнорирования клетки и выделения полезной информации, как показано на рис. 1. Ожидаемо, что вместе с клеткой пропадает некоторая часть рукописного текста с изображения, из-за чего данный способ может быть эффективно использован только в случае четкого изображения и толстых линий рукописного текста.

В случае архитектуры, основанной на выводе модели, сравниваются «логиты» учителя и ученика при одних и тех же входных данных. Схематично это представлено на рис. 1.

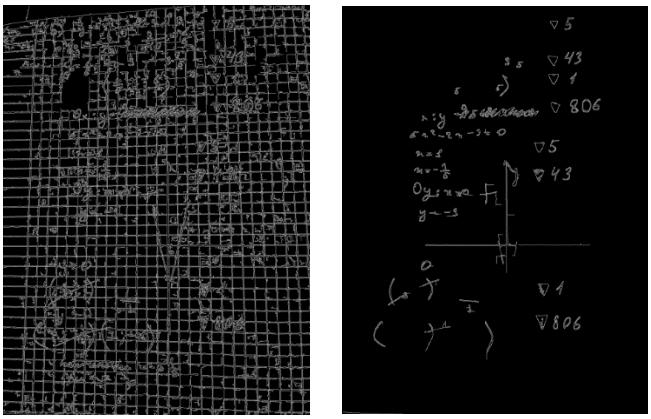

Рис. 1. Архитектура дистилляции знаний, основанная на выводе модели

Второй способ использует в дополнение к алгоритму Canny алгоритм преобразования Хафа для нахождения прямых линий клетки в границах. Этот способ полезен, когда клетка на листе бумаги или любые другие линии имеют четкие контуры, которые возможно убрать, но необходимо учесть факт потери качества полезных данных.

Преобразование Хафа – это алгоритм, используемый для поиска геометрических фигур на изображении. На основе искомой формы выбирается определенное количество параметров, например для прямой линии $y=ax+c$ таких параметров два, и они уникальны для каждой конкретной прямой. Соответственно, все точки на этой линии будут иметь одну и ту же пару параметров. Определение формы фигуры происходит по параметрам, а конкретно по вычисленным значениям параметров каждого пикселя. Так происходит голосование за каждую пару параметров, полученных из пикселей изображения. Пара с наибольшим количеством голосов соответствует прямой линии. Затем эта линия строится с использованием параметров. После нахождения линии создается фильтр, который заменяет пиксели линии на смежные пиксели.

Преобразование Хафа – медленный алгоритм, поскольку происходит голосование за каждый пиксель и учитываются все голоса для построения линии [3]. Существует более быстрая в вычислительном отношении версия данного алгоритма – вероятностное преобразование Хафа, в которой требуется только определенное количество голосов пикселей, чтобы достичь консенсуса по параметрам линии. Это ускоряет вычисления, но точность снижается.

На реальном изображении нами были использованы обе версии алгоритма (рис. 2). Стоит отметить, что преобразование Хафа имеет смысл применять, когда изображение было сформировано в результате использования сканера или же оно расположено на идеально ровной поверхности и съёмка ведётся под прямым углом. В данном случае были найдены лишь некоторые линии, которые требуется убрать из изображения. Возможно уменьшение нижнего порога гистерезиса для прорисовки более детальной сетки, однако в данном случае возникает большое количество линий, которые некорректно описывают клетку листа.

При применении вероятностного преобразования Хафа наблюдается частично необходимый эффект, который самостоятельно неспособен убрать клетку с листа, однако он может являться дополнением к другим инструментам удаления шума из изображения. При изменении параметров длины линии возможно захватывание большего количества линий клетки, но при этом обязательно будет захват линий у полезных данных на изображении, что приводит к потере качества текста.

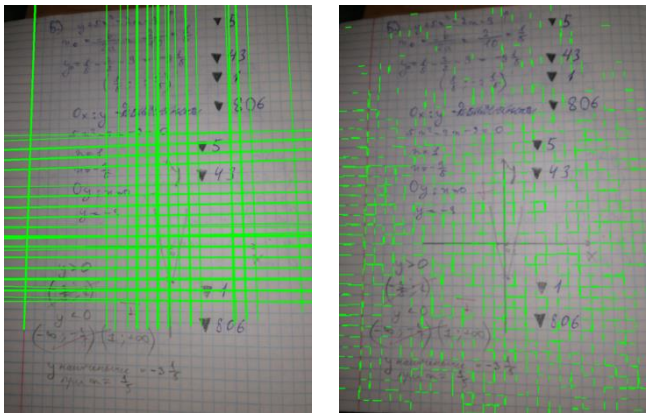

Рис. 2. Архитектура дистилляции знаний, основанная на внутренней структуре модели

Третий способ заключается в применении простого метода фильтрации изображения по цветовым характеристикам [4]. Так возможно выделение необходимого диапазона синего цвета чернил ручки, которым был написан текст. Для упрощения задачи лучше конвертировать RGB изображение в пространство HSV, где необходимо создать маску по цвету, насыщенности и яркости, а после применить маску к исходному изображению. В результате фильтр выделяет напрямую рукописный текст из изображения, как показано на рис. 3. Качество полученных данных не высокое из-за узкого диапазона цветового спектра, которые устранить возможно, но при этом под фильтр попадет клетка, что добавляет шума в изображение. Также имеется существенный недостаток фильтра: его невозможно применить универсально ко всем данным. В одном случае насыщенность клетки может совпадать с насыщенностью ручки, а в другом цвет ручки будет отличаться от примера, на котором был создан фильтр.

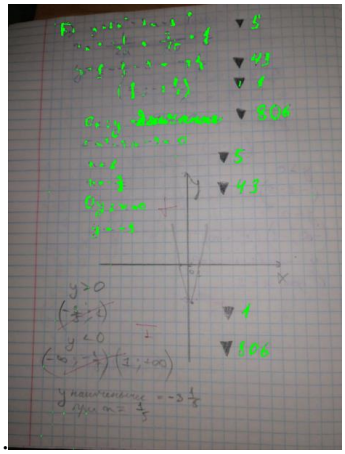

Рис. 3. Показатели достоверности и точности модели ученика [5]

Четвертый способ использует комбинацию методов. Примером может служить комбинация двух методом из библиотеки OpenCV. В первую очередь выполняется бинаризация изображения с использованием алгоритма выделения контуров изображений Adaptive Threshold, а после – морфологическая функция Morph Close для окончательного удаления шума на изображении [5],[6].

Adaptive Threshold() использует адаптивный подход к расчету порога яркости пикселя, что позволяет гибко выделять группы точек, находящиеся на границе образа. Принцип работы:

- На входе поступает исходное изображение.
- Объявляются два цикла, вложенные друг в друга, счётчики которых перебирают все значения по ширине и длине изображения соответственно.
- После перехода к очередной точке изображения программа определяет значение адаптивного порога для бинаризации.
- Затем проверяется условие, и если оно истинно, то точке результирующего изображения присваивается значение 1, во всех остальных случаях – 0. Таким образом алгоритм обрабатывает все точки изображения и формирует результат.

Функция Morph Close выполняет операцию замыкания на двоичном или сером изображении – позволяет удалить небольшие внутренние «дырки» и убрать зернистость по краям области.

Фактически оба метода используются с одной и той же целью – удаление шума, но их комбинация позволяет лучше справиться с данной задачей. Так в результате применения данного подхода было получено изображение, показанное на рис. 4.

Стоит отметить, что клетка исчезла, оставив после себя небольшое количество шума, которое слабо влияет на качество текста. Попытки изменения регулирующих параметров происходит ухудшение качества текста, потому при дальнейшей необходимости стоит использовать подходы с формированием маски клетки, которые будут исключать дальнейшее взаимодействие с текстом во избежание ухудшения результата.

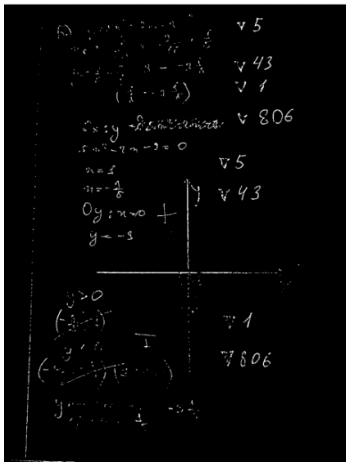

Рис. 4. Результат применения алгоритма Canny в совокупности с Morph Close

После небольшого анализа полученных результатов, было замечено, что наивысшее качество данных в тестовом изображении выдал алгоритм нахождения границ Canny, однако цифры были внутри полыми, что не годилось для дальнейшей обработки. Поэтому была применена морфологическая функция Morph Close, которая дала достаточно качественный результат (рис. 5).

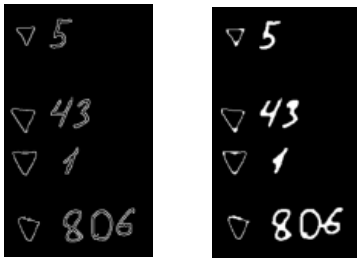

Рис. 5. Результат применения алгоритма Canny в совокупности с Morph Close

Сравнивая два комбинированных подхода на рис. 6, можно подытожить, что оба из них имеют место на существование. В частности, первый результат оставляет на изображении тонкие и слегка прерывистые линии цифр, а второй толстые и немного смазанные линии цифр. Выбор подхода зависит как минимум от тех данных, на которых производилось обучение нейронной сети. При достаточно качественной обучающей выборке разница в комбинированных подходах, как в данном примере, будет несущественной, что даёт право специалисту выбрать подход на его усмотрение. В данном случае было принято решение выбрать второй комбинированный подход в силу визуально более качественных данных.

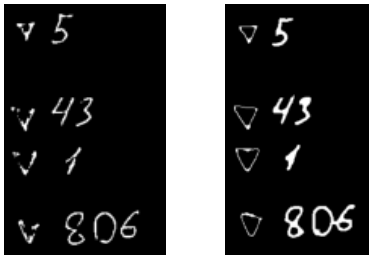

Рис. 6. Сравнение двух комбинированных подходов

В результате применения функций Find Contours() и Bounding Rect() из библиотеки OpenCV и применения простой фильтрации по размеру удалось извлечь цифры из изображения прямоугольниками, как показано на рис. 7.

Функция findContours**()** позволяет обнаруживать контуры на бинарном изображении. Она идентифицирует контуры, анализируя интенсивность пикселей изображения и соединяя соседние белые пиксели для формирования непрерывной границы. Функция boundingRect**()** используется для нахождения прямоугольника, который полностью окружает контур или группу точек. Она возвращает четыре значения: координату x, координату y, ширину и высоту ограничивающего прямоугольника. Эти значения можно использовать для рисования прямоугольника вокруг контура или обрезки изображения до ограничивающего прямоугольника, чтобы сосредоточиться на конкретной области интереса.

Рис. 7. Извлечение цифр из изображения

Далее полученные цифры были преобразованы к размеру 28 на 28 пикселей в соответствии с обучающей выборкой для нейронной сети. Пример полученного изображения цифры показан на рис. 8.

Рис. 8. Преобразование к размеру 28 на 28 пикселей

Для распознавания цифр была использована свёрточная нейронная сеть. Данный тип нейронных сетей отлично справляется с задачей компьютерного зрения, выделяя характерные черты объектов на изображении и проводя успешную классификацию. Также свёрточные нейросети достаточно легко обучаются, что позволяет не прибегать к необходимости использовать большие вычислительные ресурсы.

Архитектура нейронной сети имеет вид, представленный на рис. 9. В качестве функции ошибки использовалась категориальная кроссэнтропия, а в качестве оптимизатора был использован Adam. Качество модели оценивалось по метрике точности. Обучение модели длилось 10 эпох и было произведено на датасете mnist, который содержит рукописные цифры. Итоговая точность модели составляет 98.97 %.

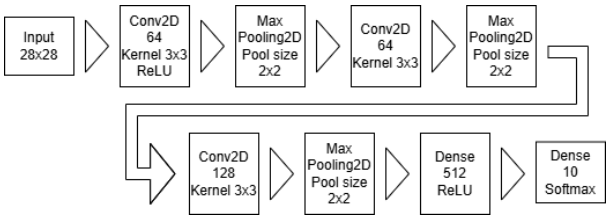

Рис. 9. Архитектура свёрточной нейронной сети

CNN включают следующие слои: сверточные (Conv), пулинга (Pooling) и полносвязные (Dense).

Сверточные слои изучают пространственные иерархии шаблонов, сохраняя пространственные отношения. Например, первый сверточный слой может изучать базовые признаки, такие как ребра, а второй сверточный слой может изучать шаблоны, состоящие из базовых признаков, изученных в предыдущем слое. И так далее, пока не будут изучены очень сложные шаблоны. Это позволяет сверточным нейронным сетям эффективно изучать все более сложные и абстрактные визуальные концепции.

Слои пулинга упрощают информацию, собранную сверточным слоем, и создают сжатую версию содержащейся в них информации. Существует несколько способов сжатия информации, но распространенным способом является MaxPooling, который сохраняет максимальное значение из тех, что находятся во входном окне, в качестве значения.

Полностью связанный слой помогает сопоставить представление между входными данными и меткой класса.

В качестве функции ошибок использовалась категориальная кроссэнтропия (CCE):

$$CCE\left(\vec{p}_{true}, \vec{p}_{pred}\right) = \sum_{i=1}^{K} p_{true}[i] \ln\left(p_{pred}[i]\right), \qquad (1)$$

где $\vec{p}_{true}$ – вероятности, создаваемые моделью; $\vec{p}_{pred}$ – опорные ответы (метки класса); K – количество классов.

Как видно, в векторе $\vec{p}_{true}$ только один элемент равен единице, а остальные равны нулю. Для него первый множитель будет равен единице, а второй будет логарифмом предсказанной вероятности для правильного класса.

В качестве оптимизатора использовался Adam. Оптимизатор позволяет улучшить производительность модели глубокого обучения. Adam (адаптивная оценка момента) – это алгоритм оптимизации, который является расширением стохастического градиентного спуска для обновления весов сети во время обучения. Оптимизатор Adam обновляет скорость обучения для каждого веса сети индивидуально.

Качество модели оценивалось по метрике точности:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \qquad (2)$$

где *TP* (True Positive) – предсказание модели положительного класса объекта совпадает с фактической меткой класса; *FP* (False Positive) – ошибка первого рода, модель неверно классифицировала объект как положительный класс, когда он на самом деле принадлежит отрицательному классу; *FN* (False Negative) – ошибка II типа, модель неверно классифицировала отрицательный объект как положительный; *TN* (True Negative) – модель классифицировала объект как отрицательный, хотя на самом деле он таковым является.

Обучение модели длилось 10 эпох и проводилось на наборе данных MNIST, содержащем рукописные цифры. Финальная точность модели составила 98,97 %.

III. Заключение

Сравнивая четыре подхода по обработке изображений с целью сохранения качества рукописного текста и удаления шума можно выделить несколько важных моментов.

Во-первых, каждый из рассмотренных методов обладает регулирующими параметрами, которые позволяют балансировать между требуемым качеством рукописного текста и наличием шума.

Во-вторых, стоит выделять характерные черты рукописного текста в документе, такие как цвет и насыщенность текста, а также качество освещённости документа. При обработке единичных образцов данные характеристики могут послужить созданию простейшего и эффективного фильтра. Конечно, в массовой автоматизации обработки изображений персональный подход не обладает должной гибкостью, однако характерные черты всё ещё могут быть полезны при составлении алгоритма массовой обработки изображений.

В-третьих, в частном случае может преобладать качество у отдельного подхода, но лучшим решением для общих случаев может послужить комбинация нескольких подходов, которые могут в совокупности представлять многоэтапный процесс обработки, учитывающий множество особенностей обрабатываемой группы документов.

Список литературы

[1] Bradski G., Kaehler A. Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media Inc, 2008. 6 p.

[2] He K., Zhang X., Ren S. and Sun J. Deep residual learning for image recognition // IEEE conference on computer vision and pattern recognition. 2016. P. 770–778.

[3] Wu B., Wan A., Yue X. and Keutzer K. Squeezeseg: Convolutional neural nets with recurrent crf for real-time roadobject segmentation from 3d lidar point cloud // IEEE International Conference on Robotics and Automation (ICRA)/ 2018. P. 1887–1893,

[4] Татарникова Т.М., Бимбетов Ф., Богданов П.Ю. Выявление аномалий сетевого трафика методом глубокого обучения // Известия СПбГЭТУ ЛЭТИ. 2021. № 4. С. 36-41.

[5] Suprun A., Tatarnikova T., Sikarev I., Shmeleva A. Detection of malicious program for the android platform by deep training // Lecture Notes in Networks and Systems. 2022. Vol. 387. P. 31-38, DOI: 10.1007/978-3-030-93872-7_3.