**Подготовка данных датасета CROHME 2019   
для обучения нейронной сети**

Исмагулов Милан Ерикович

Научный руководитель: Мельников Андрей Витальевич

**Аннотация**

Правильная подготовка данных датасета является одним из ключевых, если не самым ключевым этапом, сопровождающим процесс обучения нейросети. Некорректная подготовка данных и создание классов объектов могут привести к тому, что модель обучится на ошибочных числовых метках и станет непригодной для использования. В данной статье рассматривается процесс правильной подготовки данных датасета CROHME 2019, предназначенного для решения задач распознавания сложных рукописных математических выражений и последующего преобразования в одну из популярных систем компьютерной верстки математических выражений, таких как LaTeX или MathML. Статья охватывает следующие важнейшие этапы: загрузка и препроцессинг изображений, парсинг аннотаций LaTeX-выражений, связывание изображений и аннотаций, масштабирование диапазона значений из [0; 255] в [0; 1], создание словаря токенайзера и подготовка самого токенайзера, а также проверка полноты созданного словаря.

**Ключевые слова:** Подготовка данных, Датасет CROHME 2019, Рукописные математические выражения, Распознавание символов, Компьютерная верстка, LaTeX, Токенайзер, Машинное обучение, Препроцессинг изображений, Парсинг аннотаций, Масштабирование данных, Нейронные сети, Обработка изображений, Словарь токенайзера.

**Abstract**

Proper dataset preparation is one of the key, if not the most crucial, stages accompanying the neural network training process. Incorrect data preparation and object class creation can lead to the model being trained on erroneous numerical labels, rendering it useless. This article examines the process of properly preparing the CROHME 2019 dataset, which is designed to solve the problems of recognizing complex handwritten mathematical expressions and subsequently converting them into one of the popular systems for typesetting mathematical expressions, such as LaTeX or MathML. The article covers the following essential stages: image loading and preprocessing, parsing of LaTeX expression annotations, linking images and annotations, scaling the range of values from [0; 255] to [0; 1], creating a tokenizer dictionary and preparing the tokenizer itself, and checking the completeness of the created dictionary.

**Keywords:** Data preparation, CROHME 2019 dataset, Handwritten mathematical expressions, Symbol recognition, Typesetting, LaTeX, Tokenizer, Machine learning, Image preprocessing, Annotation parsing, Data scaling, Neural networks, Image processing, Tokenizer dictionary.

**Краткий обзор датасета CROHME**

Датасет CROHME результат многолетних наработок для одноименного конкурса. Название представляет собой аббревиатуру Competition on Recognition of Handwritten Mathematical Expressions - Соревнование по распознаванию рукописных математических выражений. Содержит 22 024 тренировочных изображений и 5 255 тестовых, столько же в датасете представлено аннотаций в формате InkML. InkML это формат файла основанный на XML содержит информацию об LaTeX аннотациях, трассеры движения пера для онлайн распознавания математических выражений. Также в датасете присутствуют файлы формата SymLG таком же количестве представляющие собой файлы описывающие каждый знак в формуле его положение и отношение между знаками.

**Первоначальная   
подготовка данных**

В качестве основы были взяты данные датасета CROHME за 2019 год поскольку эти данные стандартизированы по размеру, а именно 1010 на 1010 пикселей, данные по остальным годам необходимо дополнительно обрабатывать, обрезать по содержимому или приводить все к одному размеру. Данные за 2019 год представлены в количестве 10 979 тренировочных и 1 199 тестовых. Условно подготовку датасета CROHME можно разделить на следующие этапы:

1. Загрузка и препроцессинг изображений;
2. Парсинг аннотаций из файлов InkML и объединение загруженных данных в пары (изображение; аннотация);
3. Объединение данных в массивы NumPy и Масштабирование диапазона пикселей из [0; 255] до [0; 1];
4. Создание словаря токенайзера, а также настройка самого токенайзера и проверка корректности работы токенайзера методом обратной токенизации;
5. Сравнение оригинальных аннотированных формул и обратно токенезированных для проверки полноты словаря и оценка полноты датасета по словарю;

Подробно каждый из пунктов рассмотрен в тексте статьи.

**Загрузка и препроцессинг изображений**

Препроцессинг (Предобработка) изображений — это этап обработки данных, направленный на подготовку изображений к дальнейшему анализу и использованию в алгоритмах машинного обучения. Целью препроцессинга является улучшение качества изображений и приведение их к форме, которая наиболее подходит для последующего анализа и распознавания [1]. Может включать в себя следующие этапы:

1. Нормализация яркости и контрастности и бинаризация;
2. Удаление шумов и артефактов на изображениях;
3. Утолщение контуров изображения;
4. Изменение размера изображения;
5. Ротация и выравнивание;
6. Аугментация данных;

В случае обработки изображений датасета CROHME 2019 была задействована OpenCV — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. В качестве средств предобработки были задействованы следующие функции:

1. Размытие по Гауссу – является распространенным методом обработки изображений, который используется для уменьшения шума и деталей в изображении. Оно осуществляется путем применения гауссовского фильтра, который является сверткой изображения основанной на функции Гаусса:  
   G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}
   %2737fd9f-dc1c-4da2-b305-9c645d06e9ca  
   где:  
   ***x, y*** - координаты пикселей изображения;  
   𝝈 - стандартное отклонение;  
     
   Процесс гауссового размытия заключается в том, что каждому пикселю изображения присваивается взвешенное среднее значение его собственных значений и значений соседних пикселей. Веса задаются гауссовой функцией, которая придает больший вес пикселям, расположенным ближе к рассматриваемому пикселю, и меньший вес тем, что дальше. Это приводит к сглаживанию изображения и уменьшению шума, сохраняя при этом основные структуры и формы [2]. В библиотеке OpenCV гауссово размытие вызывается с помощью следующей команды: cv2.GaussianBlur
2. Также для обработки изображения была задействована функция адаптивной бинаризации - cv2. adaptiveThreshold. Эта функция позволяет преобразовать изображение в черно-белое, используя локальные области для вычисления пороговых значений, что делает ее полезной для изображений с неравномерным освещением или тенями. В качестве адаптивного метода было использовано cv2.ADAPTIVE\_ THRESH\_GAUSSIAN\_C: Взвешенное среднее значение соседних пикселей с использованием гауссова окна. Тип бинаризации cv2.THRESH\_ BINARY: Если значение пикселя больше порогового, присваивается maxValue, иначе 0.
3. Последним этапом будет изменение размера изображения с 1010 х 1010 пикселей до 256 х 256 пикселей, если изображения не уменьшить то на этапе создания массива NumPy будет ошибка “MemoryError” поскольку массив состоящий из 10 тысяч изображений 1010 на 1010 пикселей будет весить очень много и занимать большое место в оперативной памяти компьютера.

**Парсинг аннотаций LaTeX   
из файлов InkML**

InkML – называемое пространство имен стандарта XML файлов, имеет расширение файла .inkml содержит в себе информацию об аннотациях двух форматов компьютерной верстки, а именно LaTeX и MathML, а также координаты трассировки пера по планшету или другому захватывающему устройству.

| ***Пример содержимого InkML файла*** |
| --- |
| <ink xmlns="http://www.w3.org/2003/InkML">  <traceFormat>  <channel name="X" type="decimal"/>  <channel name="Y" type="decimal"/>  </traceFormat>  <annotation type="truth"> $y = Ax + A^2$</annotation>  <annotation type="UI">2012\_IVC\_CROHME\_F01\_E000</annotation>  <annotation type="copyright">LUNAM/IRCCyN</annotation>  <annotation type="writer">CROHME01</annotation>  <annotationXML type="truth" encoding="Content-MathML">  <math xmlns='http://www.w3.org/1998/Math/MathML'>  <mrow>  <mi xml:id="y\_1">y</mi>  <mrow>  <mo xml:id="=\_1">=</mo>  <mrow>  <mi xml:id="A\_1">A</mi>  </mrow>  </mrow>  </mrow>  </math>  </annotationXML>  <trace id="0">  1.0066 5.46535, 1.0066 5.4645,...  </trace>  <traceGroup xml:id="11">  <annotation type="truth"> Segmentation</annotation>  <traceGroup xml:id="12">  <annotation type="truth">y</annotation>  <traceView traceDataRef="0"/>  <annotationXML href="y\_1"/>  </traceGroup>  </traceGroup>  </ink> |

Файл содержит в себе следующие составляющие:

1. Открывающий тег c сылкой на xmlns (XML namespaces) <ink></ink>;
2. Тег <traceFormat></traceFormat> содержит в себе описание 2 каналов X и Y, type="decimal" говорит нам о том что координаты записаны в десятичных числах.
3. Первый тег аннотации с типом равным truth (истина) содержит в себе LaTeX аннотацию;
4. Тег <annotation type="UI">2012 \_IVC\_CROHME\_F01\_E000</annotation> содержит в себе информацию об уникальном идентификаторе отдельного InkML с указанием года создания, типом информации, конкурса и порядковые номера в каталоге;
5. Аннотация с типом copyright описывает правообладателя, в данном случае это университет Нанта во Франции;
6. Следующая аннотация описывает автора данного файла;
7. Тег <annotationXML> </annotationXML> описывает истинное значение InkML файла в формате компьютерной верстки MathML;
8. Тег <trace></trace> описывает координаты трассера электронного пера, например графического планшета через пробел записывается координата X и Y, а через запятую отдельные точки, с помощью id номера создается ссылка внутри файла на знак описываемый трассировкой в теге traceGroup внутри которого находится другой тег traceView с атрибутом traceDataRef=”0” с помощью этой конструкции мы объединяем описание конкретного знака компьютерной верстки с трассировкой, в нашей задаче трассировка напрямую не используется поскольку на вход нейронной сети подается изображение а не координаты точек, однако с помощью утилиты CROHMELib трассеры могут быть преобразованы в растровые изображения;
9. Тег <traceGroup></traceGroup> содержат в себе информацию об отдельных знаках которые составляют целую формулу, внутри тега есть информация в формате LaTeX и MathML с атрибутом truth, а также ссылку на трассер.

В нашей задаче для создания пар [изображение; аннотация] из InkML файла достаточно парсить LaTeX аннотации с атрибутом truth, затем чтобы объединить их создадим функцию на скриптовом языке программирования Python, в скрипте в качестве ключа будет использовано имя файла изображения и InkML поскольку имена совпадают, далее выведем первые 5 пар, пример пары показан на рисунке 1.

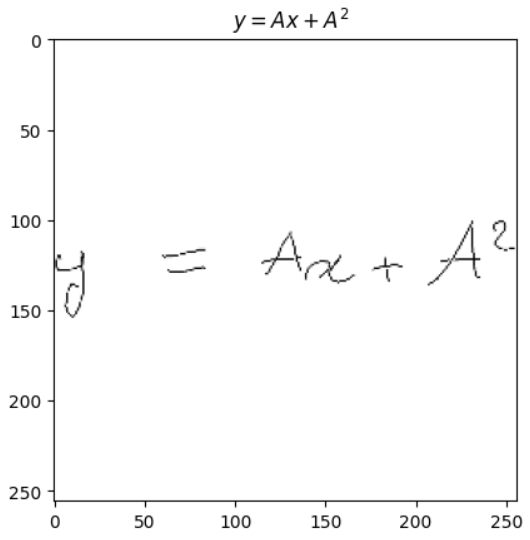


Рисунок 1. Пример готовой пары [изображение; аннотация]

На данном рисунке мы можем наблюдать сверху формулу полученную с помощью LaTeX и изображение, оси координат демонстрируют размер изображения в пикселях, в данном случае 256 на 256 пикселей, видна также работа функции Гауссовского размытия.

**Объединение данных в массив NumPy и масштабирование диапазона пикселей**

NumPy – это библиотека Python для работы с большими и многомерными массивами, большинство методов и функций библиотеки написаны на Fortran и Си что позволяет большой ресурсной эффективностью и скоростью обрабатывать большие данные.

Объединение данных таким образом является классическим для машинного обучения. Напоминаем что в датасете присутствует около 11 тысяч тренировочных и почти 1200 тестовых изображений, столько же InkML аннотаций что по умолчанию подразумевает использование NumPy массивов. Иначе создать такой массив методами языка Python будет очень ресурсозатратно и медленно.

В свою очередь масштабирование диапазона пикселей это процесс обработки данных изображения со стандартного диапазона [0; 255] где 0 это черный цвет, а 255 белый, до [0; 1] соответственно. Данный шаг способствует стабилизации работы алгоритмов машинного обучения и относится к методам оптимизации, подобный шаг положительно сказывается на скорости и стабильности процесса обучения, нормализация данных подобным образом положительно сказывается на проблеме “исчезающих” и “взрывающихся” градиентов, уменьшает разброс данных и гармонизирует их что хорошо для алгоритма градиентного спуска.

Можно выделить 3 метода масштабирования диапазона float16, float32 и float64. Где float говорит нам о том что это числа с плавающей точкой 16, 32, 64 это разрядность в битах.

В нашей работе задействован метод float32 этот выбор был сделан руководствуясь принципом “золотой середины”.

Поскольку float16 хоть и очень быстро работает в современных вычислениях он применяется редко из-за потерь большого количества информации, а float64 в разы увеличивает требования по ресурсам памяти компьютера.

**Создание словаря токенайзера и настройка токенайзера**

Процесс создания словаря токенайзера и его токенизация, это важнейшая часть настройки датасета для обучения модели, поскольку задача связана с преобразованием двумерного изображения с математической формулой в одномерную LaTeX последовательность, именно для последнего и необходимо создание токенайзера. Создание токенайзера условно можно поделить на 2 этапа, первый этап представляет собой создание массива со всеми элементами синтаксиса LaTeX кода. В него входят:

1. Цифры от 0 до 9;
2. Латинский алфавит в нижнем регистре;
3. Латинский алфавит в верхнем регистре;
4. Греческий алфавит в нижнем регистре;
5. Уникальные буквы греческого алфавита в верхнем регистре (поскольку часть из букв присутствует в латинском алфавите например, альфа и бета в верхнем регистре эквивалентна от А и В в латинском);
6. Уникальные знаки синтаксиса LaTeX например знаки доллара, прямой слэш, обратный слэш, квадратные и фигурные скобки, операторы сравнения и логические операторы;
7. Операторы представляющие собой целое слово синтаксиса LaTeX;

Второй этап представляет собой создание словаря состоящего из двух компонент [ключ (в качестве ключа выступают элементы массива созданного в предыдущем шаге); числовое значение] Итоговый вид словаря будет таким:

| ***Python код*** |
| --- |
| latex\_tokens = {  “\\frac”: 1,  "\\parallel": 2,  "\\alpha": 3,  "\\pi": 4,  "\\beta": 5,  "=",  …  "\\leq": 283,  } |

В качестве алгоритма токенизации был выбран алгоритм из библиотеки Tensorflow Keras и настроен таким образом чтобы работать с готовым словарем, поскольку по умолчанию этот токенайзер работает только с текстом, для настройки токенайзера на словарь LaTeX выражения используется функция “re.findall” из библиотеки Regex (сокр. от Regular Expressions) которая работает с регулярными выражениями описываемыми в словаре LaTeX синтаксиса. Функция проверяет аннотации из файла InkML со значениями в словаре и разбивает формулу на отдельные составляющие и каждому из них присваивает номер, например формула:

$y = Ax + A^2$

Токенезируется в:

['$', 'y', '=', 'A', 'x', '+', 'A', '^', '2', '$']

Затем кодируется:

[40, 101, 5, 103, 100, 29, 103, 21, 69, 40]

Последние два этапа настройки токенайзера это One-Hot Encoding и Приведение последовательностей к одинаковой длине, первое используется для выравнивание категорий чтобы избежать случаи создания так называемых “ложных приоритетов” когда модель часто встречает объекты определенного класса, One-Hot Encoding выравнивает процесс обучения. Привидение последовательностей к одинаковой длине необходимо для повышения эффективности обучения поскольку например RNN нейросети лучше работают с последовательностями одинаковой (предсказуемой) длины, в нашей работе это достигается путем выбора максимальной длины исходя из самого длинной формулы плюс два знака. Далее если длина остальных формул не достигает максимальной длины, остальная часть вектора заполняется нулями.

К примеру у нас есть три вектор строки разной длины:

[1, 2, 4]

[2, 11, 15, 8, 72, 67]

[10, 24, 31, 17]

В данном случае максимальная размерность вектор строки будет 6 + 2 если обратить внимание на вторую вектор строку, приведя к одинаковой длине получим:

[1, 2, 4, 0, 0, 0, 0, 0]

[2, 11, 15, 8, 72, 67, 0, 0]

[10, 24, 31, 17, 0, 0, 0, 0]

Теперь размерность каждой вектор строки равна 8.

Для проверки корректности работы токенайзера необходимо произвести обратную токенизацию, это метод при котором закодированные последовательности чисел в каждой вектор строке с помощью словаря преобразуются обратно в формулы, полученные от обратной токенизации данные можно использовать для оценки полноты словаря и датасета.

**Проверка полноты словаря и датасета**

Для проверки полноты словаря был создан скрипт который построчно проверяет оригинальные аннотированные формулы и формулы которые были полученные методом обратной токенизации, они должны совпадать, если словарь не полный то в обратно токенезированной формуле будет отсутствовать необходимый оператор, в таком случае оригинал формулы и обратно токенезированная формула записываются в текстовый файл для сравнения. Затем проанализировав можно дополнить словарь отсутствующим оператором.

Так же необходимо произвести проверку тех меток что будут поданы на вход нейронной сети, если какая либо метка есть в словаре но отсутствует в аннотациях датасета она игнорируется, так проверив полноту датасета CROHME 2019 было выяснено что из 283 меток которые есть в словаре 154 из них отсутствуют в датасете, и только 129 присутствуют. Большая часть LaTeX операторов в датасете не представлены.

**Заключение**

Исходя из проверки полноты датасета можно сделать вывод что обучение нейронной сети по распознаванию рукописных математических выражений только на данных за 2019 год нецелесообразно, поскольку если обучить нейросеть на неполных данных то можно столкнуться с ситуацией когда нейросеть будет некорректно распознавать отдельные математические выражения.

Рекомендуется дополнить данные за 2019 год дополнить датасетами CROHME за другие года, доработать скрипт препроцессинга изображений поскольку данные предыдущих годов имеют существенные различия в размерах изображений, например в данных за 2013 год все изображения имеют разный размер и разрешение.

Также следует отойти от концепции заранее разделенных данных на тестовые и тренировочные, смешать их и с помощью скрипта случайным образом разделить в соотношении 80% тренировочные данные и 20% тестовые, и рассмотреть возможность аугментации данных.

В таком случае объем данных датасета составит 27 279 экземпляров изображений и столько же аннотаций в формате inkml.

**Список литературы**

1. Гонсалес, Р. Ц., Вудс, Р. Е. Цифровая обработка изображений / Р. Ц. Гонсалес, Р. Е. Вудс. — 3-е изд. — М.: Техносфера, 2012. — 1072 с.
2. Рашка, С., Миржалили, В. Python и машинное обучение / С. Рашка, В. Миржалили. — 3-е изд. — Бирмингем: Packt Publishing, 2019. — 770 с.
3. Сзелиски, Р. Компьютерное зрение: алгоритмы и приложения / Р. Сзелиски. — М.: Вильямс, 2011. — 1104 с.
4. Гудфеллоу, И., Бенжио, Й., Курвилль, А. Глубокое обучение / И. Гудфеллоу, Й. Бенжио, А. Курвилль; пер. с англ. — М.: ДМК Пресс, 2018. — 732 с.
5. Бишоп, К. М. Распознавание образов и машинное обучение / К. М. Бишоп; пер. с англ. — М.: ИД «Вильямс», 2007. — 736 с.
6. Сонка, М., Главач, В., Бойл, Р. Обработка изображений, анализ и машинное зрение / М. Сонка, В. Главач, Р. Бойл; пер. с англ. — М.: ИД «Вильямс», 2004. — 770 с.
7. Черемисин Д.Г., Мкртчян В.Р. Использование нейронных сетей в задачах распознавания математических выражений // Международный научный журнал “СИМВОЛ НАУКИ” №12-2 2022 год 34-36 с.
8. Карахтанов А.А. Использование нейронных сетей для распознавания математических выражений // Математическое и компьютерное моделирование в экономике, страховании и управлении рисками, №5 2020 год 86-89 с.
9. Шолле, Ф. Глубокое обучение с использованием Python / Ф. Шолле. — 2-е изд. — Шелтер-Айленд: Manning Publications, 2021. — 544 с.
10. Сегалович, И., Шувалов, И. Машинное обучение. Нейронные сети и глубокое обучение / И. Сегалович, И. Шувалов. — М.: Russian Edition, 2018. — 512 с.
11. Документация по Tensorflow Keras URL: <https://www.tensorflow.org/> (дата обращения: 11.06.2024)