



Московский государственный университет имени М.В. Ломоносова  
Факультет вычислительной математики и кибернетики  
Кафедра информационной безопасности

Фомин Сергей Александрович

## **Генерация рукописного текста на русском языке**

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

**Научный руководитель:**  
к.т.н.  
С.А.Ступников

Москва, 2023

## **Аннотация**

Генерация рукописного текста на русском языке

*Фомин Сергей Александрович*

Распознавание рукописного текста – это классическая задача информатики, для решения которой в последнее время успешно применяются нейронные сети. Однако, чтобы достичь высокой точности работы, современной нейронной сети требуются не только большие вычислительные ресурсы, но и огромное количество данных для обучения. Один из самых известных способов увеличить объем данных для обучения – аугментация – это генерация дополнительных данных из имеющихся данных. Существуют разные способы аугментировать графические данные – от самых простых, таких, как сдвиг изображения, до самых сложных, в которых данные генерируются с помощью генеративно-состязательных и рекурентных нейронных сетей. В работе рассматриваются различные подходы для генерации примеров рукописного текста с целью улучшения качества его распознавания. Генерация почерка – достаточно сложная процедура, включающая этапы генерации символов, соединения их в слова и финальной обработки. Существует множество родственных работ, описывающих эти этапы для иностранных языков, но не для русского языка. Цель данной работы – обобщить современные подходы в области генерации почерка, разработать подход к аугментации рукописного текста на русском языке и доказать применимость предложенного подхода для повышения качества распознавания почерка. В данной работе рассматриваются как простые алгоритмы генерации, основанные на эвристических подходах, так и сложные нейросетевые алгоритмы генерации почерка.

## **Abstract**

Generation of handwritten text in Russian

# Содержание

<b>1 Введение.....</b>	<b>4</b>
<b>2 Терминологические понятия.....</b>	<b>7</b>
<b>3 Постановка задачи.....</b>	<b>8</b>
<b>4 Обзор родственных работ.....</b>	<b>10</b>
<b>5 Основные этапы генерации почерка .....</b>	<b>14</b>
<b>6 Растрочный способ генерации .....</b>	<b>16</b>
6.1 Генерация глифов .....	16
6.1.1 Перенос стиля изображения .....	17
6.1.2 Архитектура UNet .....	18
6.1.3 Вариационный автокодировщик.....	19
6.1.4 Генеративно-состязательные нейронные сети .....	21
6.2 Генерация лигатур .....	23
6.3 Финальная обработка .....	29
6.4 Формализация подхода .....	32
<b>7 Векторный способ генерации .....</b>	<b>34</b>
7.1 Генерация глифов .....	34
7.2 Генерация лигатур .....	38
7.3 Финальная обработка .....	40
7.4 Формализация подхода .....	42
<b>8 Описание практической части.....</b>	<b>43</b>
<b>9 Оценка качества.....</b>	<b>46</b>
<b>10 Заключение .....</b>	<b>51</b>
<b>11 Список литературы .....</b>	<b>52</b>

# 1 Введение

Нейросетевые алгоритмы активно внедряются практически во всех сферах деятельности. Несмотря на сложность внутренней реализации нейронных сетей, их обучение и использование в последнее время значительно упростились. Разработано множество программных интерфейсов, позволяющих людям обучать большие нейросети на собственных данных.

Однако для качественного обучения и достижения высокой точности нейросетевых моделей необходимо собирать большие наборы данных. Чем больше набор данных, тем выше качество обученной на этом наборе модели. Важно помнить, что набор данных должен быть не только большим, но и разнообразным. Чем выше разнообразие данных в обучающей выборке, тем больше информации сможет «заучить» нейронная сеть. Чистота данных выборки тоже играет значимую роль. Нейросеть не сможет показывать хорошее качество после обучения, если данные, на которых она обучалась, будут содержать противоречивую информацию. Таким образом, основную роль в обучении нейронной сети занимает правильная подготовка обучающих данных.

В современном мире достаточно много областей, в которых сбор данных – дорогая и сложная задача. Поэтому применение нейронных сетей в них ограничено. Еще больше областей в которых данных очень много, но все они однообразные. Это тоже не позволяет обучать высококачественные нейронные сети. Бороться с этими проблемами можно посредством *аугментации* [1, 2, 3] – генерации дополнительных данных на основе имеющихся данных.

Аугментация данных должна удовлетворять некоторым условиям. Во-первых, сгенерированные данные должны быть близки к реальным данным и их отличия должны быть незаметны невооруженным взглядом. Это позволяет использовать сгенерированные данные в том же контексте, в котором использовались реальные данные. Во-вторых, данные должны отличаться между собой. Иначе генерация данных не принесет никакого положительного эффекта, а только напрасно увеличит размер обучающей выборки. Наконец, данные должны генерироваться на основе реальных данных. Если пренебречь последним пунктом, то может получиться так, что сгенерированные данные будут существенноискажаться с каждой новой итерацией генерации.

Одна из областей, где для достижения высокого качества нейросетевых моделей требуются внушительные объемы данных – распознавание человеческого почерка. При этом сбор рукописных данных в ручном режиме – трудоемкая и дорогая задача, а потому вопрос аугментации рукописных данных очень важен.

Генерация рукописного текста может принести пользу не только в области нейросетевых моделей распознавания почерка, но и в других областях. Так, для перевода кинематографических произведений могут быть использованы продвинутые механизмы перевода текста на изображениях видеоряда. При этом можно сохранять почерк и стиль написания оригинальных картин. Алгоритмы генерации почерка могут быть полезны для написания открыток или писем без участия автора почерка. Также в процессе генерации почерка можно изучать природу самого почерка и составлять портрет его автора.

Рукописный текст представляет собой графическое изображение, а задача аугментации изображений хорошо изучена: существует множество подходов к ее решению, начиная от простых искажений, и заканчивая сложными генеративно-состязательными и рекурентными нейронными сетями. Однако в задаче распознавания почерка классические подходы не приносят значимого улучшения качества, так как не изменяют начертание символов, а только сдвигают или поворачивают их. Поэтому аугментация почерка человека рассматривается в работе не с точки зрения генерации различных изображений с текстом, а с точки зрения последовательного написания символов почерком заданного человека.

Аугментация рукописного текста может рассматривать в виде двух задач. Первая задача решается в рамках доступности автора. Это значит, что автор целевого почерка, который пытаются генерировать, доступен для создания произвольного числа новых образцов любой сложности. Таким образом, при недостатке определенных символов в наборе данных можно запросить автора создать дополнительные эталонные образцы почерка, которые помогут решить возникшую проблему. Аугментация в этой задаче способствует лишь ускорению сбора данных о почерке автора. Вторая задача, напротив, решается в рамках отсутствия непосредственного доступа к автору целевого почерка. Это может быть задача аугментации исторических записей известных личностей. Таким образом, качественная аугментация может быть полезна по историческим причинам. Приведенные выше примеры с переводом кинематографических фрагментов почерка также относятся к этому типу задач.

Структура этой работы выглядит следующим образом. Раздел 2 содержит основные терминологические понятия в области генерирования почерка. В разделе 3 приведена формальная постановка задачи для этой работы. В разделе 4 приведен обзор родственных работ, на которые будет опираться теория из основного раздела. Раздел 5 посвящен рассмотрению основных этапов генерации почерка. Затем, в разделах 6 и 7 приведены обоснования и описаны предложенные нами подходы к генерации почерка. Также, в этих разделах приведены примеры генерации рукописного теста различными методами. Раздел 8 посвящен описанию практической части работы. В разделе 9 приводится оценка качества сгенерированного почерка. В заключении подведены итоги исследования и собраны результаты работы.

## 2 Терминологические понятия

Несмотря на то, что исследуемая область начала развиваться недавно, в ней существует устоявшаяся терминология [4]. Терминология пришла из других смежных сфер, таких, как лингвистика и письменность.

На Рис. 1 изображены обозначения, используемые в работе. Минимальная единица письменности называется *графема*. Графема используется в печатной речи как единица текста. Аналогом графемы в письменной речи является *глиф* (glyph). Глиф — это единица графики. Одной и той же графеме может соответствовать множество различных глифов (различное написание одной и той же буквы).

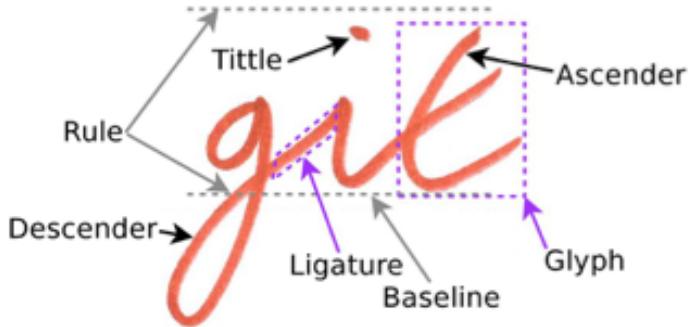


Рис. 1: Терминологические понятия в области генерации почерка. Изображение из [4].

Для обозначения строки, на которой находится текст при написании, используется понятие *линия шрифта* (baseline). В зависимости от расположения глифов по отношению к линии шрифта различают *восходящие* (ascender) и *подстрочные* (descender) глифы. К восходящим глифам в русском языке относятся такие буквы, как «б» и «в», а к подстрочным — «р», «у», «д». Также особое внимание уделяется *раздельным* глифам. Это глифы, соответствующие буквам, которые пишутся с отрывом. Например, в русском языке это буква «й».

Одну из самых важных ролей в генерации почерка занимает соединительный элемент между глифами — *лигатура* (ligature). Именно этот элемент делает сгенерированный почерк реалистичным. Лигатуры естественным образом соединяют глифы в единое письменное представление. Тем самым они позволяют имитировать безотрывное написание рукописного текста. Генерации лигатур посвящен отдельный подраздел работы.

### **3 Постановка задачи**

При рассмотрении ряда современных работ по генерации рукописного текста можно прийти к выводу, что все они посвящены тексту на английском языке, и подобные работы для русского языка отсутствуют. Цель данной работы – восполнить этот недостаток и предложить пути улучшения существующих методов. Таким образом в работе требуется разработать подход генерации рукописного текста на русском языке. А также показать применимость предложенного подхода в задаче аугментации набора данных для улучшения качества распознавания рукописного текста.

Достижение цели предполагает решение следующих задач:

- Исследование существующих подходов генерации рукописного текста на английском языке;
- Выделение основных этапов генерации рукописного текста;
- Сбор и обработка набора данных для генерации рукописного текста на русском языке;
- Разработка и программная реализация подходов генерации рукописного текста на русском языке;
- Оценка качества методов генерации рукописного текста на русском языке.

Предполагается, что на вход программной системе генерации подается короткий, аккуратно написанный рукописный образец почерка (или полный набор рукописных глифов данного почерка) в хорошем качестве и целевой текст для генерации. На выходе необходимо получить сгенерированный целевой текст входным почерком в виде изображения.

Для оценки качества генерации в таких задачах обычно используются методы опросов. При этом группе людей даются пары из реального почерка и сгенерированного образца. Респонденты должны ответить на несколько вопросов: насколько сгенерированный почерк отличается от настоящего, насколько сохранен стиль автора в сгенерированном почерке. В нашей же задаче аугментации рукописного теста, намного более правильным подходом измерения качества будет подход с оценкой качества задачи распознавания почерка. Так как конечной решаемой задачей этой работы является улучшение качества распознавания почерка за счет аугментации, то генерацию почерка будем оценивать по качеству целевой задачи. Для оценки качества алгоритмов распознавания будем использовать популярную метрику *WER (Word Error Rate)*,

подробное описание которой будет сделано в ходе работы. Ожидается, что генеративная аугментация набора данных повысит качество современных нейросетевых алгоритмов распознавания почерка.

## 4 Обзор родственных работ

В данном разделе подробно описываются существующие подходы решения задач, близких к поставленной. В работе рассматривается ряд современных трудов, посвященных генерации рукописного текста на английском языке. Каждая из рассматриваемых работ предлагает либо свой независимый метод генерации почерка, либо улучшение существующего метода генерации, либо преобразование сгенерированного изображения почерка в наиболее реалистичное с точки зрения человеческого восприятия.

Так, в работе [4] рассматривается алгоритм, основанный на извлечении рукописных символов из готового текста. На вход алгоритму подается набор рукописных текстов, на основе которых нужно научиться генерировать произвольный текст. Предполагается, что текст для генерации известен до того, как будет формироваться входной набор данных. Это дает большое преимущество при сборе данных. Можно ввести функцию подобия между текстом для генерации и входным текстом, и затем сосредоточиться на сборе близких образцов. Функция подобия основывается на двух простых эвристиках: а) подобие тем больше, чем меньше средняя разница количества уникальных букв в двух текстах; б) подобие тем больше, чем больше  $n$ -грамм целевого текста покрыто входным текстом (например, для  $n = 2$ ).

Стоит заметить, что такой подход применим только в случае наличия непосредственного контакта с человеком, почерк которого нужно генерировать. Это один из примеров генерации рукописного текста с доступом к автору почерка. Однако, авторы отмечают, если входной набор данных достаточно большой, то нет необходимости составлять его специальным способом.

Алгоритм генерации почерка в работе [4] состоит из нескольких этапов. Сначала производятся подготовительные работы. На этом этапе из входного рукописного почерка извлекаются отдельные глифы. Существуют автоматические системы распознавания почерка, именно они и используются для вырезания символов. Однако авторы предусмотрели возможную неточность работы такой системы и добавили в первый этап еще и ручное редактирование разметки. Следующий этап алгоритма состоит в непосредственной генерации почерка. Он разбит на следующие части: выбор нужных глифов из входного набора данных, вычисления расстояния между сгенерированными буквами, генерация соединительных элементов между буквами и текстурирование. Этот

этап генерации почерка описывается функцией потерь, которая штрафует алгоритм за неправильное выполнение одной из его частей, например, за невозможность соединения двух соседних глифов с помощью лигатуры или неправильный выбор буквы для генерации. Наконец, последний этап работы алгоритма состоит в наложении цвета. Наложение цвета также описывается функциональным законом для имитации неравномерного распределения чернил. Это позволяет сделать сгенерированный почерк более реалистичным.

В работе [5] рассматривается задача генерации почерка человека на основе набора существующих рукописных шрифтов. Широкое множество разнообразных шрифтов позволяет генерировать реалистичный почерк целевого человека. Алгоритм генерации почерка в работе [5] также разбит на несколько этапов. Первый этап состоит в том, чтобы найти генерируемую букву во всех рукописных шрифтах. Далее извлекаются и приводятся к одному размеру графы начертания каждой из этих букв. Затем происходит сравнение исходной буквы с графиками начертания каждой из подготовленных букв. Наиболее близкие по графикам буквы становятся кандидатами для генерации. На последнем этапе происходит ранжирование всех кандидатов по наиболее подходящей толщине шрифта. В итоге подходящая буква из готового рукописного шрифта попадает в сгенерированный текст. Генерация целевого текста на основе этого метода может быть произведена с помощью наиболее частых шрифтов, которые появлялись при выборе подходящей буквы. Такой подход имеет небольшую вариативность символов, поэтому чаще всего для одной и той же буквы исходного текста выбирается один и тот же шрифт. Этот недостаток хорошо заметен на итоговом изображении почерка. Однако алгоритм генерации лигатур и финальная обработка могут исправить недостатки этого подхода. Плюсом этого метода является то, что он очень легковесен и способен генерировать огромное количество данных за небольшой промежуток времени.

Работа [6] посвящена улучшению реалистичности готового изображения почерка. Этот этап очень важен для генерации почерка, так как качество решения в таких задачах оценивают люди. Повышение реалистичности сгенерированного изображения позволяет значительно увеличивать метрики качества решаемой задачи. Для увеличения реалистичности сгенерированного почерка авторы предлагают несколько техник искажения: пиксельные и геометрические. Общее решение по увеличению реалистичности почерка состоит из следующих этапов:

1. Упругое искажение сетки каждого символа;

2. Ухудшение качества чернил за счет случайного шума на темных участках изображения;
3. Добавление реалистичного фона бумаги;
4. Добавление случайного крупного шума на изображении;
5. Сплайновое искажение горизонтальной линии.

В результате применения этих техник искажения можно получить реалистичный почерк даже из рукописного шрифта. Подходы из работы [6] можно применять к любому способу генерирования рукописного текста. Их правильное применение способно значительно улучшить восприятие сгенерированного почерка.

В работе [7] рассматривается подход, основанный на глубоких нейронных сетях. Авторы предложили новую нейросетевую архитектуру *C-VRNN* (Conditional Variational Recurrent Neural Network), объединяющую популярную генеративную архитектуру (условный вариационный автокодировщик) и известную нейронную сеть для обработки последовательностей (рекуррентная нейронная сеть). Основная отличительная идея авторов состоит в том, чтобы разделить вектор признаков почерка на две составляющие: стиль почерка, который в виде вектора содержит в себе совокупную информацию о почерке автора, и содержание почерка, которое является непосредственным текстовым представлением фрагмента почерка. Таким образом, возможно перенести стиль почерка любого человека на произвольный текст, и задача генерации почерка сводится к тому, чтобы правильно выделить вектор стиля выбранного человека.

Работы [8, 9] также используют нейросетевые модели для генерации почерка. Однако, они основаны на генеративно-состязательных нейронных сетях (*GAN*). Эти сети хорошо зарекомендовали себя в задачах генерации изображений и были перенесены на задачу генерации почерка с некоторыми изменениями. Был добавлен новый модуль генерации изображения почерка из входного текста и шумового вектора. Также в стандартную архитектуру *GAN* был добавлен модуль распознавания почерка. Это позволяет не только контролировать обучение на уровне сходства с эталонными образцами, но и делать сгенерированный почерк понятным для систем распознавания. Однако, во всех работах, использующих различные модификации сетей *GAN*, нет возможности генерировать почерк конкретного человека. Также для *GAN*-архитектур остается нерешенной проблема лигатур. Соединять сгенерированные глифы в слова по-прежнему необходимо с использованием эвристических предположений. Еще одна проблема генеративных нейронных сетей – формирование входного набора данных.

Чтобы хорошо обучать такие сети, нужно обладать большим запасом разнообразных данных и огромными вычислительными ресурсами.

Довольно близкая к нашей задаче проблема решается в работе [10]. Авторами предлагается собственная архитектура *GAN*, которая способна генерировать рукописный текст с заданными настройками стиля. Это может быть толщина линий, сила нажатия, наклон или даже округлость почерка. Помимо этого, такие настройки можно извлекать из существующего почерка, что позволяет найти и скорректировать стиль любого человека. Эта работа также привлекательна тем, что авторы провели эксперимент с аугментацией тренировочного набора данных для распознавания почерка. Это позволило увеличить качество распознавания обученных на этих данных алгоритмов. Однако, проблема лигатур между соседними глифами все еще остается, хотя этому вопросу уделена существенная часть статьи и некоторые простые случаи решаются предложенной архитектурой *GAN*.

Одна из самых современных работ [11] по генерации рукописного текста также использует глубокие нейронные сети. В отличии от предыдущих работ авторы не используют генеративные модели. Идея подхода заключается в том, чтобы представить каждую графему некоторой квадратной матрицей, которая будет являться выходом обученной в конечном счете нейронной сети. Изображения глифов способны кодироваться и декодироваться в вектора за счет использования последовательных опорных точек глифа на плоскости (векторное представление). Таким образом, используя операции умножения матрицы на вектор и взятие обратной матрицы, можно генерировать заданные символы целевым почерком. Проблема генерации глифов в этой работе решается хорошо – небольшого набора данных достаточно для формирования вектора стиля почерка, однако задача генерации лигатур напрямую не рассматривается. Особенность функции декодирования вектора в глиф позволяет создавать простые лигатуры, но этого недостаточно для имитации полноценного почерка.

## 5 Основные этапы генерации почерка

Как и любая другая задача из области компьютерных наук, поставленная задача может быть декомпозирована. Разбиение задачи на разные этапы должно быть сделано так, чтобы полученные этапы были максимально независимы друг от друга. Таким образом, можно добиться параллельной реализации каждого этапа и, в случае необходимости, независимой замены одной реализации этапа на другую реализацию того же этапа.

В поставленной задаче довольно просто выделяются три этапа. Для генерации рукописного текста необходимо в первую очередь научиться генерировать глифы. Это основной и самый важный этап, так как он вносит существенную часть в визуальное восприятие итогового изображения. После того, как генерация глифов целевого текста окончена, необходимо научиться соединять глифы в полноценные слова. Наивный подход с расположением глифов рядом друг с другом в один ряд имеет право на существование. Однако для повышения качества итоговой генерации и улучшения визуального восприятия рукописных слов, нужно ввести этап генерации лигатур. Этот этап позволит соединять глифы между собой так, как будто они были написаны чернилами без отрыва руки от бумаги. Наконец, последний этап генерации почерка – финальная обработка. Под финальной обработкой понимаются все процедуры, связанные с расположением слов на изображении, в том числе многострочное расположение, наложением текстуры чернил и добавлением шума на изображение. В этот этап также входит заливка фона на изображении, вариация наклона текста, трансформация и искажение участков текста. Все эти методы обработки используются для того, чтобы придать почерку максимальную реалистичность и скрыть недостатки генерации предыдущих этапов.

Таким образом, поставленная в работе задача будет решаться в три независимых этапа:

1. Генерация глифов;
2. Генерация лигатур;
3. Финальная обработка.

Стоит отметить, что каждый из этих этапов сам по себе является сложной задачей в области обработки последовательностей и изображений. Если для решения задачи первого этапа существует множество родственных работ, речь о которых шла в обзоре, то для задачи второго этапа готового решения нет. Для выполнения задачи второго этапа часто

применяются эвристические подходы. Финальный этап обработки изображения может производиться известными методами компьютерной обработки, однако в работе [6] собрали основные методы, которые позволяют повысить реалистичность генерации рукописного текста.

В ряде работ [8, 9, 10, 11] с использованием генеративных нейронных сетей, авторам удавалось частично объединить первый и второй этапы генерации за счет векторного представления глифов. На выходе генеративной нейронной сети сразу получался почерк, который частично содержал лигатурные соединения глифов. Хотя это и не идеальное решение второго этапа, но оно порождает довольно естественные соединения и может быть использовано как начальный этап генерации лигатур.

Подходы, основанные на рассмотрении глифов как растровых и векторных изображений сильно отличаются друг от друга, как по способу генерации, так и по предварительной обработке данных. В связи с этим логично разделить их рассмотрение на разные главы. В одной из них подробно изучены различные методы генерации глифов в виде отдельных растровых изображений и эвристический подход для генерации лигатур, в другой рассмотрен нейросетевой подход на основе векторного представления глифов. Обе главы описывают полноценный алгоритм генерации рукописного текста и его финальную обработку.

## 6 Растрочный способ генерации

Генерацию почерка на основе растровых изображений глифов, как было сказано ранее, можно разбить на два этапа: генерация глифов и генерация лигатур. Затем, нужно произвести финальную обработку для формирования итогового результата. В этой части работы будут изложены способы решения всех трех задач, и приведены результаты их экспериментального применения для русского языка.

Все изображения в этой части работы рассматриваются как растровые матрицы пикселей. Это связано с тем, что большинство известных существующих подходов к обработке изображений используют их растровое представление и применяют методы машинного обучения, основанные на матричных свертках.

### 6.1 Генерация глифов

При генерации глифов важно учитывать, что глиф должен оставаться в пределах одного почерка, то есть не сильно искажаться при генерации. В данном подразделе проводятся эксперименты с различными способами генерации глифов, а также делаются выводы об их применимости для решения поставленной задачи.

Для рассматриваемых далее алгоритмов генерации глифов необходимо собрать набор данных для обучения соответствующих моделей генерации. В качестве набора данных были собраны и обработаны элементы почерка автора этой работы. В элементы почерка входят отдельные рукописные глифы русского алфавита, цифры и популярные знаки препинания. Характеристики этого набора данных приведены в Таблице 1.

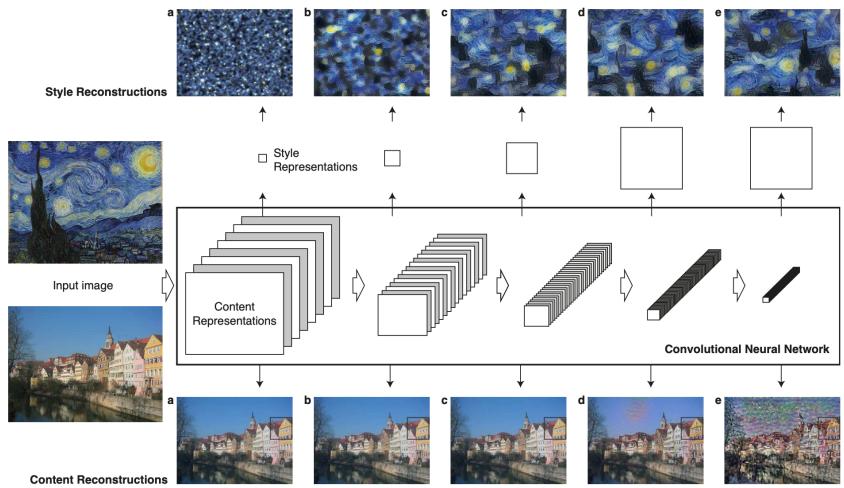
Характеристика набора	Значение	Примечание
Количество уникальных символов	89	Строчные и прописные буквы русского алфавита, цифры, 13 знаков препинания.
Количество повторений каждого символа	10	Повторения необходимы для повышения разнообразия набора данных.
Общий размер набора данных	890	Этого количества достаточно для обучения нейросетевых моделей.

Таблица 1: Характеристики набора данных для растровых методов генерации глифов.

### 6.1.1 Перенос стиля изображения

Перенос стиля изображения [12] это популярная и хорошо решенная задача из области обработки изображений. Некоторые подходы [13] из этой области способны генерировать качественные картины в стиле произвольного художника. Для переноса стиля изображения обычно используется глубокая нейронная сеть, в нашем случае это *VGG16* [14]. Идея нашего подхода состоит в том, чтобы применить стиль исходного изображения лигатуры к целевому изображению графемы. То есть мы будем решать задачу наложения почерка автора через перенос стиля с изображений его глифов.

Нейронная сеть обучается таким образом, чтобы штрафовать выходное изображение за высокую метрику различия с целевым изображением, а также за сильное различие матриц Грамма изображения стиля и целевого изображения. Архитектура модели и общая идея подхода изображена на Рис. 2.



*Рис. 2: Процесс обучения нейронной сети по переносу стиля. Изображение из [12].*

Эксперименты проводились с печатной буквой в качестве целевого изображения и глифом почерка в качестве изображения стиля. Это не принесло положительных результатов, так как для качественного переноса стиля принято использовать изображения с однородной текстурой. Это позволяет нейронной сети одинаково воспринимать любую часть стилевого изображения. После этого производилось много экспериментов с однородными изображениями стиля в виде волн и геометрических примитивов, но это тоже не дало ожидаемого результата. Некоторые результаты экспериментов представлены на Рис. 3.

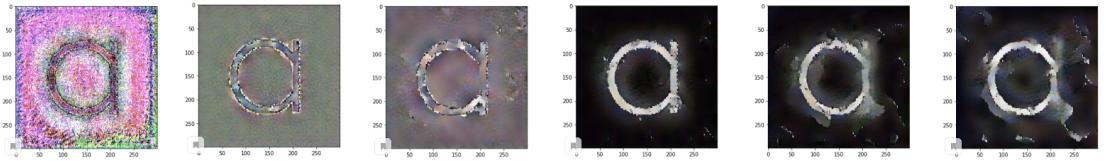


Рис. 3: Некоторые результаты применения подхода с переносом стиля изображения.

В ходе работы над этим подходом была изучена и реализована архитектура нейронной сети *VGG16*. На ее основе был реализован метод обучения, позволяющий переносить стиль изображения. Был проведен ряд экспериментов с несколькими стилевыми изображениями. После перебора гиперпараметров было обучено две модели переноса стиля. Однако приемлемого результата добиться не удалось. Было принято решение отказаться от этого подхода генерирования глифов почерка.

### 6.1.2 Архитектура UNet

Известная нейросетевая архитектура *UNet* [15] способна не только качественно решать задачу сегментации изображения, но и производить произвольные преобразования над матрицей изображения. Современная интерпретация [16] этой архитектуры позволяет повысить качество сегментации изображений. На Рис. 4 изображена классическая архитектура *UNet*.

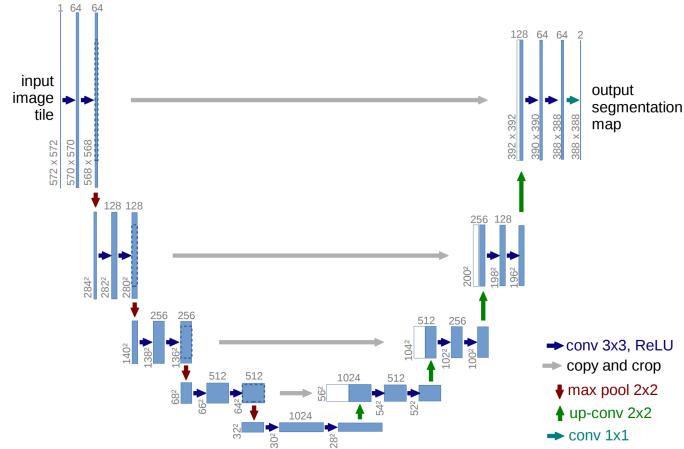
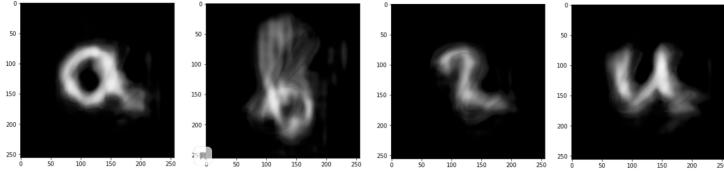


Рис. 4: Архитектура нейронной сети *UNet*. Изображение из [15].

Следующие эксперименты по генерации глифов производились с использованием этого похода. Его преимущество в том, что на вход *UNet* принимает изображение буквы, а на выходе получает преобразованное изображение той же буквы. Мы будем использовать архитектуру *UNet* не для сегментации, а для «умного» искажения изображения. Задача

состояла в том, чтобы научить *UNet* добавлять стиль почерка во входное изображение. На вход подавалось произвольное изображение буквы и на выходе ожидалось применение целевого почерка. Некоторые результаты экспериментов представлены на Рис. 5.



*Рис. 5: Некоторые результаты применения подхода на основе нейронной сети *UNet*.*

На рисунке прослеживается ожидаемый результат работы *UNet*: небольшое размытие сгенерированных символов. Это связано с тем, что набор данных имеет небольшие смещения в буквах почерка. Нейросеть обучилась так, чтобы показывать наилучший в среднем результат генерации, но общая картина генерации выглядит довольно размытой. Эту проблему может решить постобработка изображений с использованием операций свертки и фильтрации.

В ходе работы над этим методом генерации глифов была изучена и реализована архитектура нейронной сети *UNet*. Также был обработан набор данных под нужный формат входа модели. В результате обучения и подбора гиперпараметров не удалось добиться высоких результатов. Последовательное применение нейронной сети и методов постобработки не является перспективной идеей генерации глифов. Было решено также отказаться от этого подхода.

### 6.1.3 Вариационный автокодировщик

Класс генеративных нейросетей *VAE* имеет хорошее математическое обоснование [17] и позволяет генерировать реалистичные объекты. Его современная интерпретация [18] позволяет лучше понять природу латентного пространства и эффективно использовать *VAE* в самых актуальных задачах информатики. Автокодировщик *VAE* обучается исходя из двух факторов: входное изображение должно быть похожим на выходное и распределение латентного вектора (представление изображения внутри кодировщика) должно быть плотным (без пустых зон) и подчиняться нормальному закону. Схематичное описание работы вариационного автокодировщика представлено на Рис. 6.

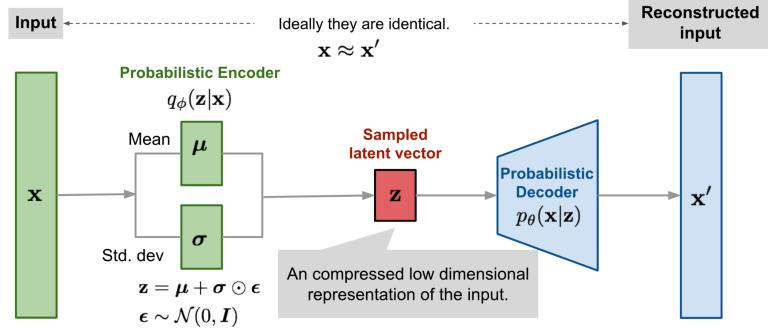


Рис. 6: Принцип работы вариационного автокодировщика.

В нашем случае для нейросетевого кодирования и декодирования была выбрана простая симметричная архитектура на основе *VGG16*. После обучения автокодировщика на человеческом почерке удалось реализовать плавные переходы от одной буквы к другой, что говорит о высоком уровне качества полученного латентного представления. На Рис. 7 представлен результат такой генерации.

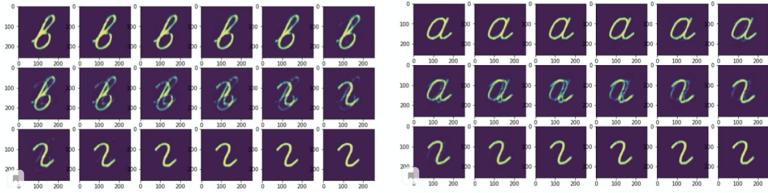


Рис. 7: Некоторые результаты применения подхода на основе вариационного автокодировщика. Переход от одной буквы к другой.

Далее, в результате экспериментов удалось добиться, чтобы автокодировщик генерировал реалистичные глифы одной буквы с небольшими изменениями. Результат представлен на Рис. 8: можно видеть результат генерации буквы «а» из малоразмерного представления этой буквы с добавлением случайного шума. Небольшая величина шума позволяет генерировать глифы различного внешнего восприятия с сохранением уникальности автора.

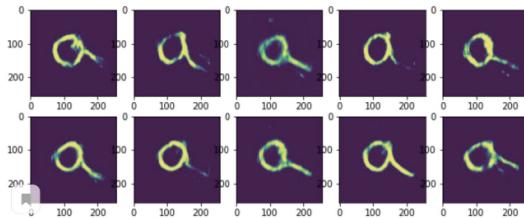


Рис. 8: Некоторые результаты применения подхода на основе вариационного автокодировщика. Генерация различных представлений одной буквы.

В ходе работы над этим способом генерации глифов была изучена и реализована архитектура нейронной сети *VAE*. Полученная модель была обучена на собранном ранее наборе данных. Был проведен успешный эксперимент, который показывает возможность преобразования одного глифа в другой путем перемещения в латентном пространстве. Также была произведена успешная генерация глифов целевого почерка. Этот подход способен достаточно качественно генерировать разнообразные глифы одной и той же графемы. Генерация глифов с помощью модели *VAE* является перспективным методом генерации растровых глифов.

#### 6.1.4 Генеративно-состязательные нейронные сети

Генеративно-состязательные нейронные сети (*GAN*) – это наиболее современные генеративные модели, которые способны создавать изображения любой сложности. Принцип работы таких нейронных сетей состоит в том, чтобы обучить две независимые модели, одна из которых генерирует изображение из латентного (внутреннего) представления (генеративная модель), а вторая отличает реальные изображения от изображений, сгенерированных первой моделью (дискриминативная модель). Схема работы генеративно-состязательных нейронных сетей показана на Рис. 9. Правильно подобранная функция ошибки, которая учитывает качество генерации и ошибку классификации, и умеренная скорость обучения позволяют обучать модели таких архитектур до высокого качества генерации.

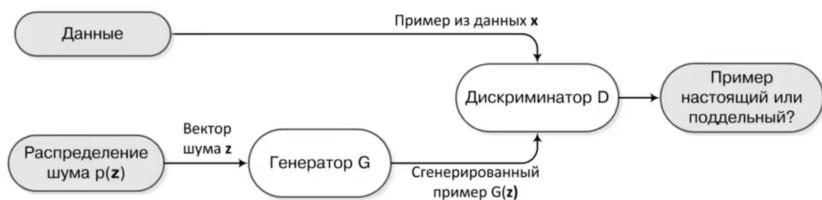
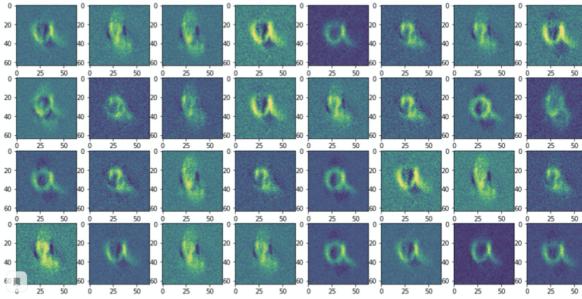


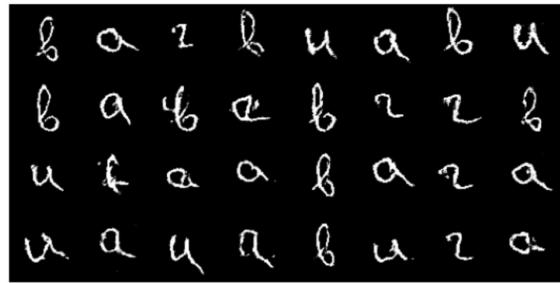
Рис. 9: Принцип работы генеративно-состязательной нейронной сети.

Сначала была рассмотрена классическая архитектура *GAN* [19] которая никак не учитывает специфику изображений. В качестве архитектуры использовались различные полносвязные сети. Наилучший результат, которого удалось достичь после экспериментов с набором данных, архитектурой сети и функцией ошибки представлен на Рис. 10.



*Рис. 10: Некоторые результаты применения подхода на основе генеративно-состязательной нейронной сети.*

Следующим этапом было обучение архитектуры *GAN* на основе сверточных нейронных сетей в качестве генеративной и дискриминативной моделей, как наиболее подходящих для работы с изображениями (подход *DCGAN* [21]). В результате экспериментов с архитектурами вида *VGG16* был получен результат, представленный на Рис. 11.

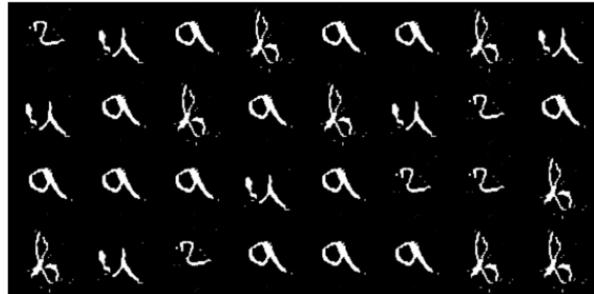


*Рис. 11: Некоторые результаты применения подхода на основе сверточной генеративно-состязательной нейронной сети.*

Во всех подходах, основанных на генеративных моделях, важно уметь генерировать не произвольное изображение, похожее на глиф целевого почерка, а изображение содержащее глиф конкретной графемы. С этой задачейправляются так называемые условные генеративные модели [20]. Они добавляют ряд дополнительных входных параметров, на основе которых делается выбор о генерации той или иной буквы.

Для улучшения последнего подхода была рассмотрена архитектура условной (*conditional*) *DCGAN*, которая может генерировать заданную заранее букву. Отличие условной *DCGAN* в том, что для каждой буквы в латентном пространстве вычисляется среднее значение, и буква всегда лежит в небольшой окрестности от своего среднего вектора в латентном пространстве. Функция ошибки в *GAN* гарантирует, что небольшое

отклонение от среднего не изменяет класса генерируемого объекта. Это позволяет генерировать заданные в тексте глифы с небольшими изменениями, но с сохранением стиля автора. После экспериментов с условной *DCGAN* удалось добиться результата, представленного на Рис. 12.



*Рис. 12: Некоторые результаты применения подхода на основе условной сверточной генеративно-состязательной нейронной сети.*

В ходе работы над генеративно-состязательными моделями генерации глифов были изучены архитектуры популярных моделей *GAN*, *DCGAN* и *Conditional DCGAN*. Каждая из этих моделей была реализована и обучена на собранным наборе данных. После перебора гиперпараметров было принято решение о слабой применимости модели *GAN*. Однако модели на основе *DCGAN* (обычные и условные) показали существенно более высокий результат.

Анализ результатов применения всех рассмотренных подходов генерации глифов показал, что наиболее удачная генерация глифов получается с применением вариационных автокодировщиков (*VAE*) и некоторых видов сверточных генеративно-состязательных нейронных сетей (*DCGAN*). Так как генеративно-состязательные нейронные сети являются более современной и гибкой архитектурой генерации, было принято решение использовать именно их в финальной генерации рукописного текста.

## 6.2 Генерация лигатур

Следующий этап этап генерации почерка, нацеленный на повышение качества генерации – добавление в почерк лигатур, существенно улучшающих его внешний вид. В этом разделе предполагается, что этап генерации глифов уже произведен и можно использовать его результаты для генерации лигатур. Так как разработка эффективного алгоритма должна производиться на качественных данных, в этой части работы будут

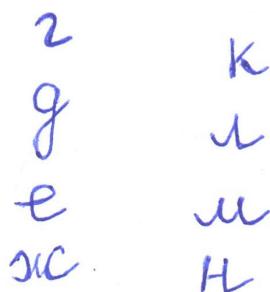
использоваться не сгенерированные глифы из предыдущего раздела, а идеальные собранные для этого этапа глифы аккуратного рукописного текста. Это позволит довести алгоритм генерации глифов до идеального решения, при этом у алгоритма не будет возможности «переучиться» на одном почерке с его собственными особенностями.

Предлагается использовать подход к разработке алгоритма генерации лигатур, который включает следующие этапы:

- Сбор примеров глифов аккуратного рукописного почерка;
- Определение границ глифов на изображении;
- Определение и нормализация высоты глифов;
- Подбор оптимального расстояния между глифами;
- Настройка лигатур соседних глифов.

Основные идеи и графические примеры каждого из упомянутых этапов рассмотрены ниже.

*Сбор примеров глифов аккуратного рукописного почерка.* Для экспериментов с генерацией лигатур можно использовать почерк, собранный самостоятельно. При этом достаточно одного экземпляра каждой буквы, так как разрабатываемый подход может быть применен к любому почерку, в том числе сгенерированному методами, рассмотренными в предыдущем разделе. Был собран, обработан и отфильтрован набор данных из отдельных строчных лигатур русского алфавита (но подход может быть применен и к прописным буквам). Примеры глифов из собранного набора данных приведены на Рис. 13.



*Rис. 13: Примеры глифов из собранного набора данных.*

Подробные характеристики собранного набора данных и примечания к каждому пункту приведены в Таблице 2.

Характеристика набора	Значение	Примечание
Количество уникальных символов	33	Строчные буквы русского алфавита.
Количество повторений каждого символа	1	В алгоритме генерации лигатур нет необходимости повторять символы.
Общий размер набора данных	33	—

Таблица 2: Характеристики набора данных для алгоритма генерации лигатур.

*Определение границ глифов на изображении.* Для правильной генерации лигатур и правильного расположения глифов на строке и между собой необходимо знать границы каждой буквы. Для определения границ была реализована программа обнаружения и вырезания букв на изображении. Это позволило автоматически находить левую, правую и нижнюю границы каждого глифа. Результат применения программы приведен на Рис. 14.

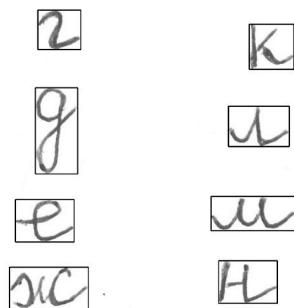


Рис. 14: Пример определения границ глифов на изображении.

*Определение и нормализация высоты глифов.* Вырезанные глифы необходимо научиться показывать в виде слов, то есть расставлять готовые буквы в одну строку. Это приводит к проблеме согласования восходящих и подстрочных глифов. Однако она легко решается, если найти и нормализовать среднюю высоту обычных глифов. Таким образом обычные и восходящие глифы можно располагать на строке по нижней границе буквы, а подстрочные – на уровне равном разнице между верхней границей буквы и ранее найденным средним значением высоты глифов. Пример расположения глифов представлен на Рис. 15. Также для правильного визуального восприятия все глифы нужно нормализовать к одному размеру. При этом подстрочные глифы обрабатываются отдельно от остальных, так как их нормализация проводится относительно верхней границы глифа.



Рис. 15: Пример расположения глифов на линии шрифта.

Подбор оптимального расстояния между глифами возможен после анализа результатов большого количества экспериментов. Так, на Рис. 16 показаны результаты наивной генерации почерка с различным расстоянием между буквами. Наивность генерации заключается в том, что в этом процессе используются заранее собранные и обработанные глифы. Суть такой генерации только в том, чтобы расположить эти глифы в нужном порядке на нужном расстоянии.

приветствуем участников и гостей соревнований  
приветствуем участников и гостей соревнований  
приветствуем участников и гостей соревнований  
приветствуем участников и гостей соревнований

Рис. 16: Результаты генерации почерка с различным расстоянием между глифами.

Настройка лигатур соседних глифов. На Рис. 16, даже в самом удачном варианте, хорошо заметны проблемы соединения некоторых глифов между собой при наивном расположении их рядом друг с другом. На Рис. 17 показаны основные проблемы соединений двух соседних глифов.

аз ы е ав

Рис. 17: Основные проблемы соединения глифов.

Для решения этих проблем был разработан эвристический подход, основанный на искажениях изображения. Основная идея состоит в том, чтобы вертикально сдвигать левую и правую части изображения таким образом, чтобы их соединения совпадали. Для этого нужно сделать разметку каждой буквы: способна ли она поддерживать правое или

левое соединение. Формальная последовательность действий предложенного эвристического подхода описана в Алгоритме 1. Для примера приводится функция сдвига только правой части изображения. Сдвиг левой части происходит аналогично.

```

1      def generate(l_img, r_img, l_char, r_char):
2          is_l_ligature = has_right_ligature(l_char)
3          is_r_ligature = has_left_ligature(r_char)
4
5          left_start = find_start_right_side(l_img)
6          right_start = find_start_left_side(r_img)
7
8          shift_distance = right_start - left_start
9
10         if is_l_ligature and is_r_ligature:
11             shift_right_side(l_img, shift_distance / 2)
12             shift_left_side(r_img, shift_distance / 2)
13         else if is_l_ligature:
14             shift_right_side(l_img, shift_distance)
15         else:
16             shift_left_side(r_img, shift_distance)
17
18         return concatenate(l_img, r_img)
19
20     def shift_right_side(img, distance):
21         alpha = calculate_alpha(distance)
22         if distance >= 0:
23             quadratic_shift_up(img, alpha)
24         else:
25             quadratic_shift_down(img, alpha)

```

*Алгоритм 1: Генерация лигатур.*

Представленный алгоритм работы сдвига части изображения глифа состоит из следующих этапов:

- Для каждой буквы с той стороны, с которой она способна поддерживать соединение (`has_right_ligature` и `has_left_ligature`), возьмем 15% пикселей от ее ширины и будем их вертикально двигать согласно квадратичному закону  $\frac{x^2}{\alpha}$  вверх (`quadratic_shift_up`) или вниз (`quadratic_shift_down`);
- Сдвиг изображения происходит вверх, если часть буквы, с которой происходит соединение, находится выше текущей лигатуры (`distance >= 0`). Иначе движение происходит вниз;
- Расстояние перемещения определяется половиной (`shift_distance / 2`) вертикального расстояния между концами существующих лигатур (`right_start - left_start`);

- Параметр  $\alpha$  в законе сдвига рассчитывается исходя из расстояния перемещения (`calculate_alpha`).

Пример применения этого алгоритма представлен на Рис. 18. Заметим, что соединение может быть настроено достаточно произвольным образом.



*Рис. 18: Генерация различных лигатур для одного глифа.*

Расстояние для сдвига считается исходя из уровня первого непрозрачного пикселя в крайних столбцах матрицы изображения. Таким образом для каждой соседней пары глифов считается расстояние, на которое нужно сдвинуть их соединения. Если оба глифа поддерживают соединения – сдвигать необходимо оба изображения на половину этого расстояния. Если только у одно глифа есть поддержка соединения – необходимо сдвигать это соединение на полное расстояние.

Генерация лигатур с использованием приведенного алгоритма способна улучшить качество генерации рукописного текста и исправить недостатки, возникающие на предыдущем этапе. На Рис. 19 представлен пример генерации лигатур.



*Рис. 19: Генерация лигатур между различными глифами.*

В результате применения предлагаемого подхода к генерации лигатур удалось добиться хорошего результата: пример наивной генерации рукописного текста приведен на Рис. 20. Соединения букв внешне выглядят достаточно естественно. Однако, все еще можно видеть некоторые артефакты генерации, например, в соединении букв «ы» и «е» в слове «данные». В качестве дальнейшего улучшения алгоритма генерации лигатур можно рассмотреть подход с динамической длиной соединения или векторное представление лигатуры. Это позволит выровнять расстояния между буквами и сгладить длинные соединения.

магистратура  
большие данные  
виртуальная интеграция

Рис. 20: Генерация почерка на основе алгоритма лигатур.

Заметим, что алгоритм генерации лигатур работает без этапа обучения и может быть применен к любому почерку. Алгоритм может работать как с верхними, так и с нижними соединениями глифов. Вне зависимости от того, какие глифы сгенерированы и поданы на вход алгоритму, он сможет создать соответствующую лигатуру.

### 6.3 Финальная обработка

На данном этапе работы разработан качественный алгоритм генерации глифов в виде растровых изображений. Также разработан эвристический алгоритм генерации лигатур. Исходя их описанного выше плана, остался последний этап, который объединит результаты предыдущих и придаст финальной генерации рукописного текста реалистичный вид.

Для генерации полноценного почерка предложенным подходом необходимо сгенерировать отдельно каждый глиф целевого текста. Генерация каждого глифа должна производиться путем запуска разработанного ранее алгоритма генерации глифов. Причем алгоритм разработан таким образом, чтобы генерируемые глифы одной и той же графемы получались разными. Это позволит сделать рукописный текст наиболее разнообразным и тем самым приблизит его к почерку реального автора. Для генерации глифов был выбран подход на основе *условной сверточной генеративно-состязательной* нейронной сети (*Conditional DCGAN*). Она показала самый высокий уровень качества и разнообразия выходных глифов среди всех экспериментов по генерации глифов.

Следующим этапом генерации рукописного текста является применение эвристического подхода генерации лигатур, который описан в предыдущем пункте. После его применения можно получить отдельные рукописные слова. Если рассматривать (Рис. 21) эти слова по отдельности, то можно видеть, что благодаря разнообразию генерации

глифов и наличию лигатур, они выглядят так, как будто их написание велось без отрыва руки.

*Генерация изображения рукописного текста.* На этапе финальной обработки прежде всего надо научиться располагать сгенерированные слова в виде предложений. Причем стоит иметь в виду, что предложение может располагаться на нескольких строках. Учитывая это, получается несложный алгоритм, который способен принимать на вход последовательность сгенерированных рукописных слов и давать на выходе изображение рукописного текста с расставленными пробелами и переносами строк. Результат применения такого алгоритма представлен на Рис. 21.

распознавание рукописного текста это классическая  
задача информатики для решения которой в  
последнее время успешно применяются нейронные  
сети

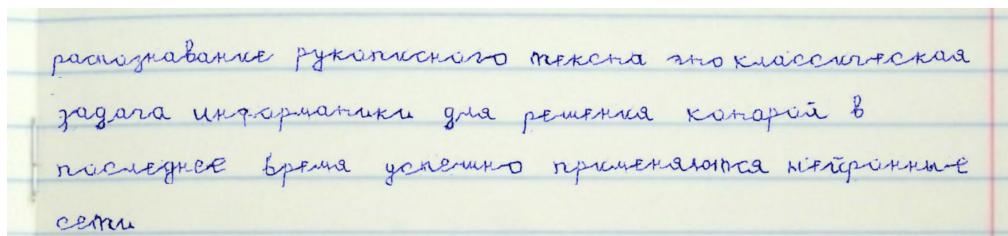
*Рис. 21: Генерация почерка предложенным подходом.*

*Имитация чернил.* Заметим, что сгенерированное изображение уже довольно сильно похоже на настоящий рукописный текст. Однако, этап финальной обработки предполагает намного более глубокую и качественную проработку изображения. Для начала проведем базовую обработку изображения классическими подходами: увеличим контрастность и четкость изображения. Затем приступим к непосредственной имитации чернил. Этот этап включает в себя наложение цвета, которое приблизит почерк к реальному, и неравномерность затемнения участков текста, которая будет имитировать различное усилие при нажатии на бумагу. Если наложение цвета, довольно простая операция, то неравномерность затемнения раскроем подробнее. Для затемнения только некоторых участков текста хорошо подходит шум Перлина. Наложение этого шума на изображение позволяет с элементом случайности затемнить участки текста. На Рис. 22 показан результат применения описанных этапов.

распознавание рукописного текста это классическая  
задача информатики для решения которой в  
последнее время успешно применяются нейронные  
сети

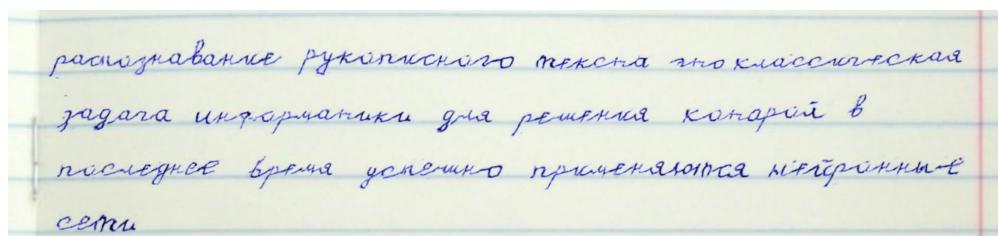
*Рис. 22: Наложение цвета и неравномерность затемнения.*

*Добавление фона.* Следующий этап финальной обработки – добавление фонового изображения. Как и наложение цвета, это тоже хорошо изученная область обработки изображений. Однако в нашей задаче она осложняется тем, что нужно соблюдать согласованность между текстом и фоном, чтобы избежать ситуаций, когда текст не соответствует масштабу или расположению фона. Решением этой проблемы может быть нанесение разметки на фоновое изображение, которая обозначит наклон, масштаб и межстрочный интервал текста, который поддерживает данное изображение. Для примера было размечено фоновое изображение тетрадного листа и на него нанесен сгенерированный рукописный текст (Рис. 23).



*Рис. 23: Добавление фонового изображения.*

*Наклон и деформация* – последний этап обработки сгенерированного рукописного текста. В данном контексте под деформацией понимаются продольные волнообразные преобразования вдоль линии шрифта. Это добавит эффект усталости и неточности к генерируемому почерку. Наклон текста необходим для придания ему естественного рукописного формата, которым обладает любой почерк. Результат применения этих преобразований представлен на Рис. 24.



*Рис. 24: Деформация и наклон текста.*

Таким образом, после объединения всех этапов генерации рукописного текста и финальной обработки, получили естественное изображение почерка заданного автора. Можно заметить, что рукописный текст получился достаточно качественным. В нем отсутствуют повторения один и тех же глифов, а также соединения в словах естественным образом формируют целостность почерка. Однако, при должном погружении и после

тщательного анализа можно научиться отличать такой почерк от авторского. Это происходит по причине небольшого разнообразия данных, на которых обучалась генеративная модель глифов и несовершенного алгоритма построения лигатур. В разделе 7 для решения этой проблемы и дальнейшего совершенствования подходов генерации рукописного текста будут применяться векторные способы представления глифов.

## 6.4 Формализация подхода

В этой части работы будет приведена формализация подхода генерации рукописного текста на основе растровых изображений глифов. В Алгоритме 2 описаны все рассмотренные выше этапы генерации рукописного текста в виде одной последовательности действий.

```
1     def generate(text):
2         glyphs = []
3         for char in text:
4             glyph = generate_glyph(char)
5             glyphs.append(glyph)
6
7         words = []
8         word = None
9         for i in range(len(text)):
10            if text[i] == " ":
11                words.append(word)
12                word = None
13                word = generate_ligature(
14                    word,
15                    glyphs[i],
16                    text[i-1],
17                    text[i],
18                )
19                words.append(word)
20
21            img = generate_handwriting(words)
22            img = process_ink(img)
23            img = process_background(img)
24            img = process_tilt(img)
25            img = process_deformations(img)
26
27        return img
```

Алгоритм 2: Генерация рукописного текста растровым способом.

Для генерации отдельных изображений глифов в Алгоритме 2 используется функция `generate_glyph`. После того как все глифы сгенерированы, для каждой пары соседних глифов вызывается функция `generate_ligature`, которая добавляет на

изображение из первого аргумента соответствующую лигатуру и следующий глиф. Таким образом получается массив изображений, каждый элемент которого содержит рукописное слово. Этот массив проходит через все этапы финальной обработки: генерация изображения рукописного текста (`generate_handwriting`), имитация чернил (`process_ink`), добавление фона (`process_background`), наклон и деформация (`process_tilt` и `process_deformations`).

## 7 Векторный способ генерации

До этого момента представление глифа рассматривалось нами как растровое изображение. Это был вполне обоснованный подход, так как сверточные нейронные сети и методы обработки изображений хорошо работают с изображениями в виде матриц. В этом разделе предлагается рассматривать глиф или целое рукописное слово в виде набора векторов, которые формируются по ходу движения руки во время написания текста. Таким образом, задача генерации будет заключаться не в том, чтобы сгенерировать готовое изображение, а в том, чтобы сгенерировать набор векторов, который на этапе финальной обработки будет преобразован в рукописный текст.

Однако, стоит понимать, что для работы с таким подходом, данные, собираемые о почерке, должны предоставляться в специальном виде. Разметка векторного представления на готовом почерке – довольно трудоемкая задача, а иногда вовсе неразрешимая, когда нет информации о последовательности движения ручкой по бумаге. Если такая последовательность не имеет значения, то приложив некоторое количество усилий, можно собрать информацию для произвольного почерка. Наилучшим образом собирать данные о почерке для такого подхода можно при помощи специальной программы, которая фиксирует перемещение ручки по бумаге. Таким образом, предлагаемый подход будет эффективно решать поставленную задачу при непосредственном доступе к автору почерка.

### 7.1 Генерация глифов

Для генерации глифов в предлагаемом методе предлагается использовать нейросетевой подход, основанный на подходе из работы [11]. Однако, отличие нашего метода, от того, который рассматривается в статье, будет заключаться в том, что а) наш подход будет применяться для генерации рукописного текста на русском языке (соответственно, переобучаться на другом наборе данных) и б) мы будем применять алгоритмы генерации лигатур в полученном почерке, а также проведем финальную обработку изображения. Таким образом, основываясь на опыте родственных работ, будет решена поставленная в этой работе задача: генерация рукописного текста для улучшения качества его распознавания.

Как было сказано выше, для предложенного подхода генерации глифов необходимо собрать набор данных векторного представления целых слов или словосочетаний. Для сбора такого набора данных была реализована специальная программа, подробное описание которой будет в практической части этой работы. В этот набор данных входят отдельные слова и словосочетания на русском языке. В качестве почерка используется почерк автора этой работы. Подробные характеристики собранного набора данных приведены в Таблице 3.

<b>Характеристика набора</b>	<b>Значение</b>	<b>Примечание</b>
Количество слов и словосочетаний	100	Слова и словосочетания брались из текста этой работы.
Количество повторений каждого элемента набора	1	Для предложенного метода нет необходимости повторять данные.
Общий размер набора данных	100	Именно такой размер набора данных используется в оригинальной работе [11]
Количество уникальных символов	89	Тексты для набора данных подбирались так, чтобы в них вошли все необходимые символы.

Таблица 3: Характеристика набора данных для алгоритма генерации лигатур.

Метод генерации глифов заключается в том, чтобы обучить глубокую нейронную сеть представлять графему в виде квадратной матрицы чисел. Добиться такого представления можно путем фиксации входного алфавита. Нами был зафиксирован входной алфавит из всех строчных и прописных букв русского алфавита, всех цифр и популярных знаков препинания. Каждому символу алфавита был присвоен порядковый номер. Это позволяет подавать на вход нейронной сети произвольный символ из алфавита в виде *One-Hot Encoding* вектора. На выходе нейронной сети получается квадратная матрица чисел, которая характеризует некоторое внутреннее представление каждого символа. Чтобы использовать эту матрицу для генерации глифа, нужно умножить ее на некоторый вектор, который представляет из себя вектор почерка автора. Способ получения такого вектора будет описан ниже. После умножения матрицы на вектор получается так называемый вектор глифа, который содержит в себе необходимые вектора для представления глифа на плоскости. Путем его декодирования можно получить итоговое представление сгенерированного глифа.

Для того, чтобы получить вектор почерка автора, необходимо собрать образцы почерка этого автора. Чем больше будет таких образцов, тем больше информации получится извлечь из почерка и сохранить в цифровом виде. Каждый глиф целевого почерка подвергается кодированию. Это обратная описанной выше процедуре декодирования. С помощью кодирования можно из векторного представления глифа получить внутренний вектор фиксированной длины. Заметим, что сейчас этот вектор содержит в себе информацию как о самом символе, который был изображен автором, так и информацию о непосредственном стиле автора. Из повествования выше, становится понятно, что такой вектор мог быть получен не только операцией кодирования глифа, но и операцией генерации: умножения матрицы символа на вектор стиля автора. Следовательно, чтобы извлечь вектор стиля из вектора глифа, достаточно решить несложное матричное уравнение или умножить вектор глифа на матрицу, обратную к матрице рассматриваемого символа. Так как вектор стиля, извлеченный из одного глифа может быть неточным, нужно произвести эту операцию на множестве всех глифов данного автора. Финальный вектор стиля автора формируется усреднением векторов стилей от всех глифов. Полный алгоритм генерации глифов предложенным методом представлен на Рис. 25.

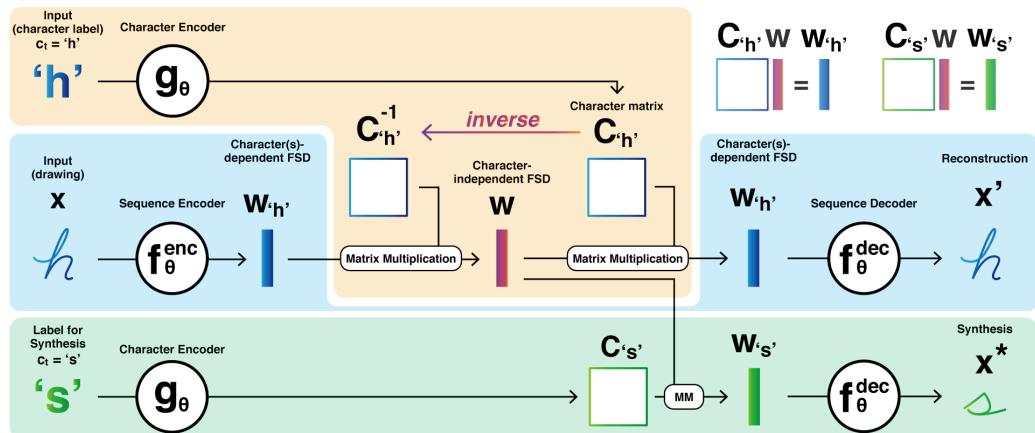


Рис. 25: Идея вектороного подхода генерации глифов. Изображение из [11].

В данном подходе на самом деле принимает участие две нейронные сети. Если про первую из них, которая преобразует номер символа в его матрицу, было сказано достаточно подробно, то о второй пока речи не шло. Вторая нейросеть – это основной компонент предложенного подхода. Она выполняет функции кодировщика-декодировщика рукописного текста [22]. Задача кодировщика в том, чтобы преобразовать векторное представление глифа произвольной длины во внутреннее представление

фиксированной длины. Причем сделать это так, чтобы внутреннее представление сохранило как можно больше информации о рукописном глифе. Более того, из внутреннего представления нужно уметь восстанавливать рукописное представление глифа в виде набора векторов на плоскости. Этим занимается часть нейросети под названием *декодировщик*. Обучается такая нейросеть на произвольном почерке. При обучении ее задача состоит в том, что преобразовать рукописный глиф во внутренне представление, а потом из этого представления восстановить первоначальный рукописный глиф. Функция потерь сравнивает входной и выходной глифы и штрафует за наличие отклонений. После обучения на большом количестве данных можно получить качественный кодировщик-декодировщик.

Обучение таких нейронных сетей с нуля – достаточно трудоемкая задача. Поэтому для решения поставленной задачи были взяты предобученные модели из родственных работ. Для реализации нашего подхода, был собран набор данных из размеченных рукописных текстов, который включает в себя с повторениями все символы описанного ранее алфавита. На этом наборе данных были дообучены обе нейронные сети. На Рис. 26 можно видеть пример генерации одной и той же буквы русского алфавита. Заметим, что все примеры генерации этой буквы различаются между собой и даже похожие генерации имеют отличительные черты. Это положительное свойство предложенного подхода, которое повышает качество генерации рукописного текста.



*Рис. 26: Разнообразие генерации глифов векторным способом.*

Генерация полноценных слов этим методом производится с учетом контекста предыдущих символов. Это делается благодаря тому, что модель, которая занимается генерацией матриц символов, представляет из себя рекуррентную нейронную сеть (*RNN* [23]). Благодаря этому, при генерации следующего символа, модель получает информацию обо всех предыдущих символах. На Рис. 27 видно, что при генерации полноценных слов некоторые соседние глифы естественным образом соединяются между собой. Такие автоматические лигатуры появляются благодаря наличию рекуррентности в обученной модели. Среди всех соединений, которые появляются автоматически на выходе нейронной сети, наиболее частыми являются соединения в сочетании букв «ем», «ов»,

«ев» или «ни». Это связано с тем, что при обучении, нейронная сеть «видела» в обучающей выборке такие сочетания букв с готовыми соединениями наиболее часто.

приветствуем участников и гостей соревнований

*Рис. 27: Генерация почерка векторным способом.*

Из результатов генерации видно, что этап генерации глифов работает достаточно качественно. В почерке соблюдается авторская составляющая и отсутствуют повторяющиеся элементы наивной генерации. Однако, видна проблема редких лигатур, которая делает такую генерацию неполноценной.

В ходе работы над векторным способом генерирования глифов был изучен и реализован подход из работы [11]. Был собран и преобразован в нужный формат набор данных для обучения этого подхода. На обработанном наборе данных были дообучены модели, составляющие каркас рассматриваемого метода генерации. Эксперименты показали высокое качество генерации глифов данным способом.

## 7.2 Генерация лигатур

В предыдущем подразделе было показано, что обученная нейронная сеть способна самостоятельно генерировать лигатуры. Экспериментальная генерация показала, что лигатуры генерируются достаточно редко. Генерация некоторых слов происходит вовсе без лигатур и это достаточно сильно искажает общий вид рукописного текста. Для решения этой проблемы нужно дополнительно генерировать лигатуры там, где их не сгенерировала нейронная сеть. В одном из предыдущих разделов был описан эвристический подход генерации лигатур. Его применение с незначительными изменениями должно решить описанную проблему.

Стоит отметить, что разрабатываемый ранее подход генерации лигатур рассматривал генерацию соединений в предположении, что глифы сгенерированы в виде растровых изображений. Однако, последний рассматриваемый подход генерации глифов отличается тем, что глифы представляются набором векторов на плоскости. Для переиспользования уже реализованного алгоритма генерации лигатур, можно производить

конвертацию векторных изображений глифов в растровые изображения. При соблюдении аккуратности с масштабом это позволит без изменений использовать готовый подход генерации лигатур.

На выходе нейронной сети получаются глифы не всегда пригодные для применения на них алгоритма генерации лигатур. Это сильно затрудняет задачу. Однако, анализ показал, что количество таких глифов крайне мало (около 11%). Поэтому перед применением алгоритма лигатур достаточно проверить некоторую эвристическую, которая установит, способен ли алгоритм генерации лигатур создать качественное соединение. Помимо разметки о возможности правого и левого соединения каждого глифа, которая осталась после разработки лигатурного подхода, нужно разметить тип этого соединения. Тип соединения включает в себя наклон и высоту соединения. Таким образом, эвристический подход, который проверяет пригодность алгоритма лигатур, будет на основе разметки проверять 15% крайних пикселей изображения глифа на наличие ожидаемого соединения. В случае отсутствия такого соединения будем пропускать этап генерации лигатуры для этой пары глифов. Это может положительно сказаться на общей генерации рукописного текста, так как при написании слова, человек может отрывать руку на середине слова.

На основе описанного выше метода была проведена экспериментальная генерация рукописного текста с применением эвристического подхода генерации лигатур. На Рис. 28 представлен результат такой генерации.

The image shows a single line of handwritten text in cursive script. The word "манистратура" is written in a fluid, flowing style. The letters are connected by various ligatures, such as "м" and "а", "и" and "с", and "т" and "р", which are generated by the algorithm. The strokes vary in thickness and orientation, giving it a natural, handwritten appearance.

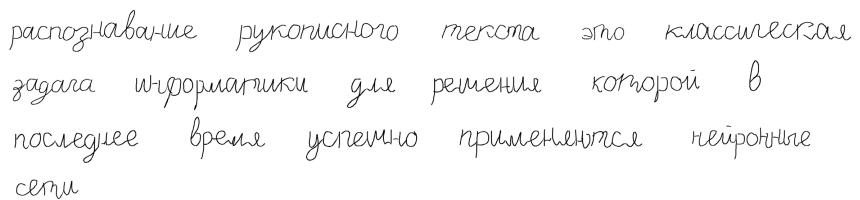
*Рис. 28: Векторная генерация почерка с применением алгоритма лигатур.*

Результат генерации получился довольно качественным. Глифы имеют разнообразное представления даже для одинаковых графем. Получилось исправить проблемы, которые оставались после этапа генерации глифов: отсутствие большого количества лигатур. Теперь почерк выглядит намного лучше, но все еще недостаточно натурально. Этап финальной обработки способен довести сгенерированный рукописный текст до идеального состояния.

### 7.3 Финальная обработка

При рассмотрении метода генерации рукописного текста основанного на отдельной генерации каждого глифа удалось выделить последовательность действий необходимую для качественной финальной обработки. В этом подразделе работы будем переиспользовать опыт предыдущей обработки с незначительными изменениями. Финальная обработка должна существенным образом изменить сгенерированный почерк и придать ему естественный вид.

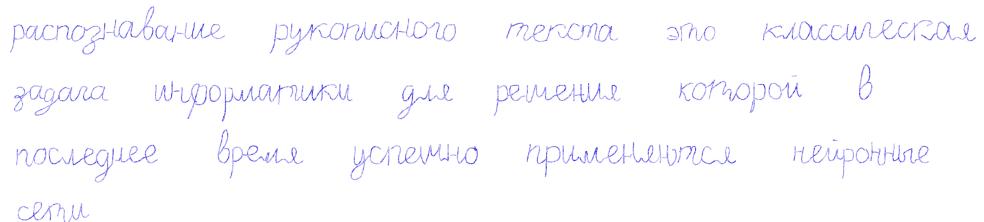
*Генерация изображения рукописного текста.* Как и ранее, на первом этапе финальной обработки сгенерируем отдельные рукописные слова и правильным образом поместим их на изображение. Тут важно учитывать пробельные символы и переносы строк. В результате этого этапа получится «сырой» рукописный текст (Рис. 29).



распознавание рукописного текста это классическая  
задача информатики для решения которой в  
последнее время успешно применяются нейронные  
сети

Рис. 29: Генерация почерка предложенным подходом.

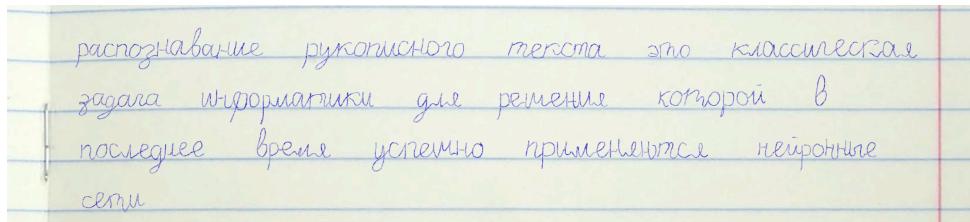
*Имитация чернил.* На следующем этапе повысим четкость и яркость изображения. Также на этом этапе производится текстурирование рукописного текста: наложение цвета и неравномерность затемнения участков текста. Напомним, что наложение цвета делается стандартными методами обработки изображений, а неравномерность затемнения путем наложения шума Перлина. Результат обработки рукописного почерка после этого этапа представлен на Рис. 30.



распознавание рукописного текста это классическая  
задача информатики для решения которой в  
последнее время успешно применяются нейронные  
сети

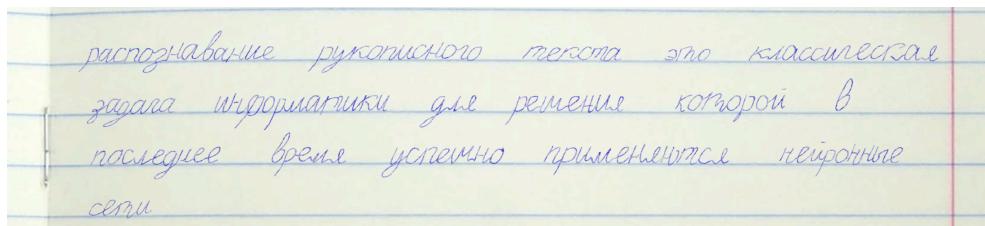
Рис. 30: Наложение цвета и неравномерность затемнения.

*Добавление фона.* Одним из самых важных этапов финальной обработки является добавление фонового изображения. Оно приносит максимальный эффект реалистичности генерируемого рукописного текста. С учетов всех описанных ранее нюансов наложения фона этот этап производится достаточно трудоемко. На Рис. 31 представлен результат добавления фонового изображения на сгенерированный ранее почерк.



*Рис. 31: Добавление фонового изображения.*

*Наклон и деформация.* Последний этап финальной обработки – естественная деформация текста вдоль линии шрифта и добавление наклона. На Рис. 32 представлен финальный результат генерации рукописного почерка после всех этапов финальной обработки.



*Рис. 32: Деформация и наклон текста.*

В результате применения предложенного подхода генерации рукописного текста получилось добиться эффекта подлинности почерка заданного автора. С точки зрения субъективного восприятия, сгенерированный почерк получился неотличимым от настоящего авторского. Однако в работе ставилась задача не только добиться натуральности генерируемого почерка, но и с помощью генерации рукописного почерка повысить качество алгоритмов распознавания почерка за счет обогащения обучающей выборки. В разделе 9 будут проведены эксперименты, которые доказывают применимость данного подхода в реальных задачах.

## 7.4 Формализация подхода

Как и в предыдущем подходе, в этом разделе будет описан полноценный алгоритм генерации рукописного текста на основе векторного способа генерации глифов. В Алгоритме 3 описана соответствующая последовательность действий.

```
1      def generate(text):
2          words = []
3          for word in text.split(" "):
4              glyphs = generate_glyphs(word)
5              words.append(glyphs)
6
7          words_with_l = []
8          for word in words:
9              word_with_l = None
10             for glyph in word:
11                 if use_ligatures(word_with_l, glyph):
12                     word_with_l = gen_ligature(
13                         word_with_l, glyph
14                     )
15             else:
16                 word_with_l = concatenate(
17                     word_with_l, glyph
18                 )
19             words_with_l.append(word_with_l)
20
21             img = generate_handwriting(words_with_l)
22             img = process_ink(img)
23             img = process_background(img)
24             img = process_tilt(img)
25             img = process_deformations(img)
26
27         return img
```

Алгоритм 3: Генерация рукописного текста векторным способом.

Для генерации глифов необходимо подавать на вход функции `generate_glyphs` целое генерируемое слово (для правильной работы рекуррентности). Затем между глифами добавляются лигатуры (`gen_ligature`). Однако, как было сказано в описании метода, лигатура генерируются только если для них работает эвристика, описанная в функции `use_ligatures`. После выхода из цикла генерации лигатур получается массив сгенерированных слов. Он проходит через все этапы финальной обработки: генерация изображения рукописного текста (`generate_handwriting`), имитация чернил (`process_ink`), добавление фона (`process_background`), наклон и деформация (`process_tilt` и `process_deformations`).

## 8 Описание практической части

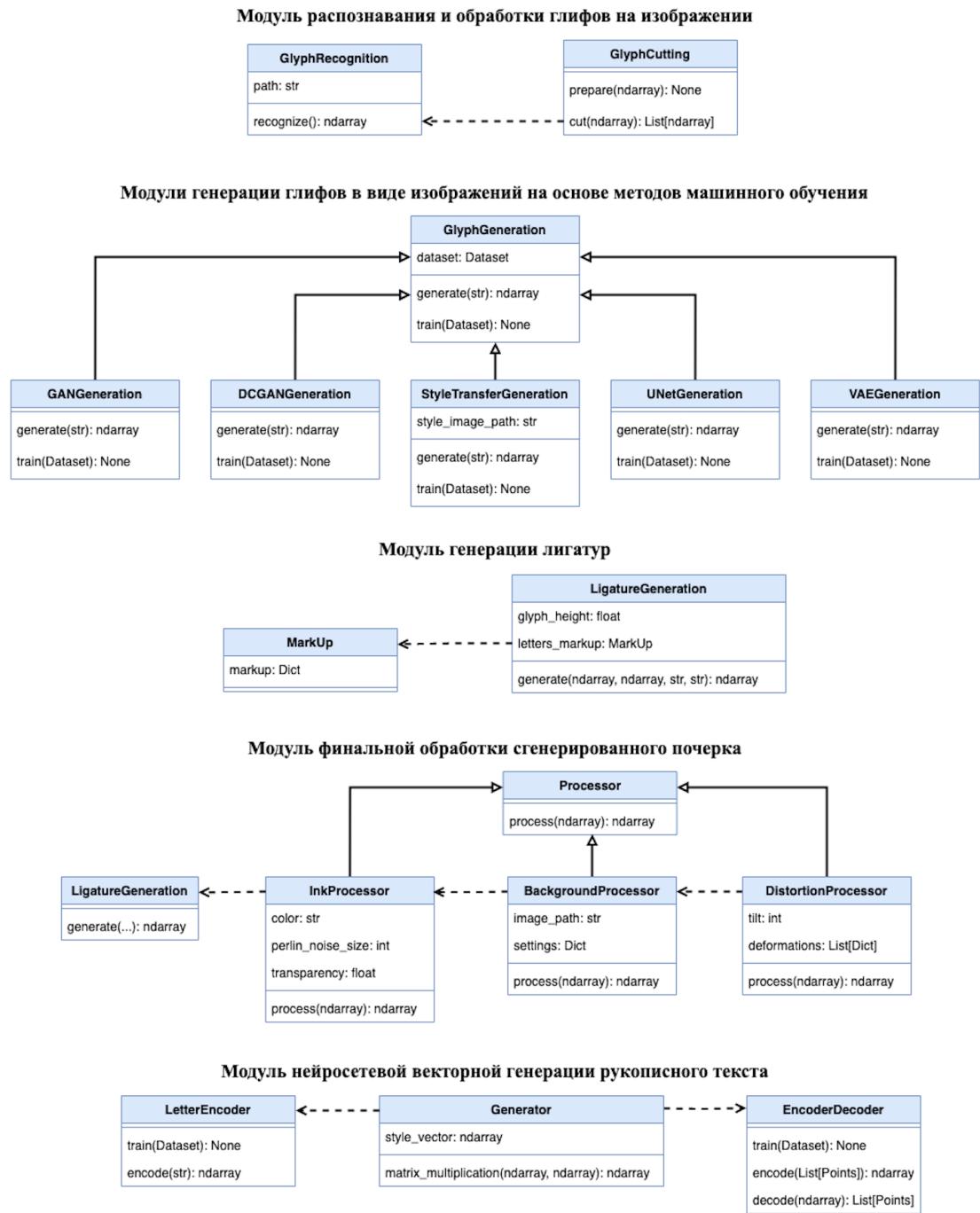
Для реализации всех компонентов и программ в этой работе использовался язык программирования *Python*. Он достаточно прост в освоении и предоставляет все необходимые библиотеки для работы с файлами, изображениями и нейронными сетями. Также у этого языка достаточно богатая стандартная библиотека, что позволяет иметь минимальное количество внешних зависимостей. В ходе работы было разработано множество модулей, в которых использовались следующий внешние библиотеки:

- *Pillow, OpenCV, scikit-image* – библиотеки для обработки изображений;
- *PyTorch, torchvision, scikit-learn, transformers* – библиотеки машинного обучения для реализации моделей;
- *Pandas, numpy* – библиотеки для удобной работы с наборами данных;
- *PyQt5* – библиотека для реализации графического интерфейса.

Все эксперименты в работе производились в среде разработки *jupyter-notebook*. После успешных попыток тестовая реализация подвергалась рефакторингу и переносилась в отдельные модули языка *Python*. Таким образом на выходе получались независимые модули, способные решать свою задачу. В ходе работы были разработаны следующие модули:

- Модуль распознавания и обработки глифов на изображении.
- Модули генерации глифов в виде изображений на основе методов машинного обучения.
- Модуль генерации лигатур.
- Модуль финальной обработки сгенерированного почерка.
- Модуль нейросетевой векторной генерации рукописного текста.

Для лучшего понимания структуры выполненной работы и ее программной реализации на Рис. 33 представлена диаграмма классов перечисленных модулей. Каждый модуль имеет набор соответствующих классов, которые связаны между собой отношением наследования или зависимости. Дополнительные функции, которые помогают решать поставленные задачи, не отражены на диаграмме с целью повышения наглядности реализации.

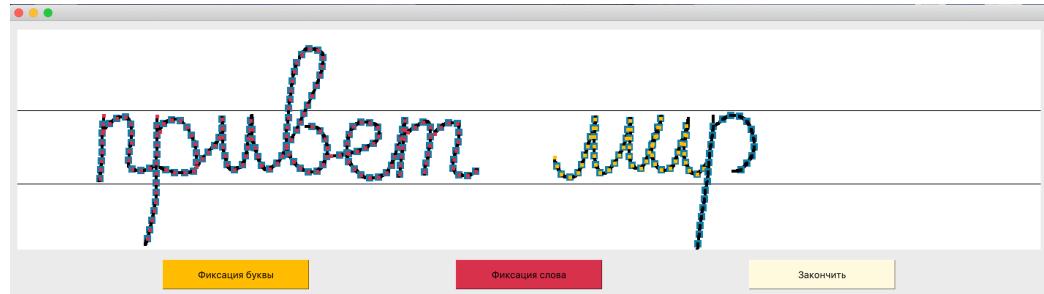


*Рис. 33: Диаграмма классов реализованных в работе модулей.*

Помимо прочего, в процессе изучения предметной области было изучено, переработано и запущено большое количество реализаций (с открытым исходным кодом) алгоритмов из родственных работ. В ходе изучения и переработки существующих подходов было обнаружено и исправлено несколько ошибок. Также код из других работ подвергался оптимизациям перед его использованием на больших наборах данных.

В ходе работы также была разработана программа с графическим интерфейсом для облегчения сбора данных для векторного подхода генерации рукописного текста. Она

способна собирать данные о движении ручки во время рукописного ввода текста. Для управления вводом в интерфейсе предусмотрены соответствующие кнопки. Также она сразу преобразовывает введенные данные в нужный для нейронной сети формат. На Рис. 34 представлен пример работы данной программы.



*Рис. 34: Программа сбора и обработки данных для векторного подхода.*

Исходный код реализации всех модулей и программ опубликован в репозитории на платформе *github.com*: <https://github.com/TotalChest/MastersThesis>.

## 9 Оценка качества

На данном этапе работы уже готовы полноценные алгоритмы генерации рукописного текста. Теперь вернемся к постановке задачи. В работе требовалось разработать подход к генерации рукописного текста, который не просто выглядит естественно и напоминает оригинальный авторский почерк, но и позволяет качественно аугментировать обучающую выборку для нейросетевых моделей распознавания почерка. Таким образом, в этой части работы будет проведен ряд экспериментов, которые докажут применимость предложенного подхода генерации рукописного текста для улучшения качества его распознавания.

В работе рассматривается генерация почерка для русского языка. Поэтому для обучения модели распознавания, будем использовать набор данных также на русском языке. Одним из самых известных общедоступных наборов данных рукописных текстов на русском языке является набор данных *HKR* [24]. В данных этого набора содержатся изображения рукописных текстов с расшифровками на двух языках (руссом – 95% и казахском – 5%). В наборе данных содержится почерк около 200 различных авторов. Для проведения экспериментов будут выбраны три русскоязычных автора с различными почерками. Для чистоты эксперимента, будут выбраны авторы с различными степенями сложности почерков.

В результате анализа для распознавания почерка была выбрана современная модель *AttentionHTR* [25]. Выбору этой модели способствовало несколько факторов: современность подхода, качество распознавания, наличие исходного кода и предобученных ресурсов. *AttentionHTR* имеет архитектуру типа *seq2seq* [26], содержащую *BiLSTM* [27] модуль в качестве кодировщика и модуль внимания [28] в качестве декодировщика. В оригинальной работе эта модель обучалась для английского языка. Для решения этой проблемы нами была дообучена *AttentionHTR* для распознавания русского рукописного текста при помощи готовых весов обученной англоязычной модели и русскоязычной части набора данных *HKR*.

Для проведения экспериментов было выбрано три рукописных почерка из набора данных *HKR*. Внешний вид этих почерков представлен на Рис. 35. Выбирать почерки старались таким образом, чтобы в экспериментах участвовали как аккуратные и понятные рукописи, так и сложные с точки зрения распознавания почерка.

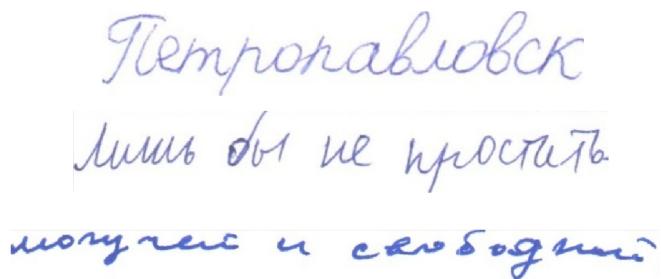


Рис. 35: Примеры почерков для распознавания.

Проведем следующий эксперимент: для каждого выбранного почерка дообучим модели на наборе данных из 1000 дополнительных изображений рукописного текста. Такое увеличение позволит в 2–5 раз расширить набор данных. В первом случае дополнительные изображения будут браться из обучающего набора данных с применением к ним базовых техник аугментации (сдвиг, поворот, масштабирование). Во втором случае дополнительный набор данных будет генерироваться предложенным методом с растровой генерацией глифов (*DCGAN*, генерация лигатур, финальная обработка). В третьем случае дополнительный набор данных будет генерироваться предложенным методом на основе векторного представления (векторное генерирование глифов, генерация лигатур, финальная обработка). Заметим, что для второго и третьего случаев необходимо проводить этап разметки, обработки и обучения генерирующих алгоритмов. В силу низкой требовательности предложенных подходов обучение генерирующих алгоритмов делается достаточно просто.

В результате дообучения модели *AttentionHTR* на дополнительном наборе данных получилось по три модели распознавания для каждого рассматриваемого почерка. Проведем замеры качества распознавания рукописного теста на отложенной выборке каждого из рассматриваемых почерков. Отложенная выборка данных не участвовала в обучении модели. Для оценки качества будем использовать одну из самых популярных метрик распознавания почерка *WER* (*Word Error Rate*). Эта метрика считается как расстояние Левенштейна [29] между оригинальным текстом и текстом, который распознал алгоритм. Минимальной единицей замены в такой метрике является слово. Для подсчета значений этой метрики можно использовать следующую формулу:

$$WER = 100 * \frac{insertions + substitutions + deletions}{len(original)}$$

Где *original* – оригинальная текстовая расшифровка рукописного фрагмента; *insertions* – количество вставленных слов, не присутствующих в оригинальном тексте; *substitutions* – количество замен одного слова на другое; *deletions* – количество слов, пропущенных в распознанном тексте.

В результате экспериментов получились значения метрики *WER*, представленные в Таблице 4. Первое, что видно из таблицы – три выбранных почерка имеют разное качество распознавания. Как и предполагалось, удалось подобрать такие почерки, которые имеют разную степень сложности. Можно заметить, что во всех трех почерках удалось добиться повышения качества распознавания. Если рассматривать аккуратный и понятный почерк, то видно, что существенного прироста качества не получилось, так как модель уже довольно хорошо его распознает. Однако для двух других почерков прирост качества распознавания получился более значимым.

	Почерк 1	Почерк 2	Почерк 3
Базовая аугментация	16.54	18.23	27.98
Растровый способ генерации	16.47	18.03	<b>26.96</b>
Векторный способ генерации	<b>16.44</b>	<b>17.91</b>	27.60

Таблица 4: Оценка качества распознавания почерка по метрике *WER* после аугментации.

Важно заметить, что в двух случаях из трех более предпочтительным алгоритмом аугментации оказался подход на основе векторного способа генерации глифов. Это связано с тем, что векторный подход способен запоминать и генерировать более качественный рукописный текст, в том числе с автоматическим генерированием лигатур. Даже визуальное восприятие нейросетевой генерации получалось лучше, чем генерация на основе растровых глифов. Однако, при рассмотрении результатов распознавания почерка в третьем случае можно заметить, что подход с растровым генерированием глифов оказался лучшим методом аугментации. Это объясняется тем, что третий почерк достаточно неаккуратный и тем самым добавляет сложности в его распознавание. Подход на основе отдельного генерирования глифов смог обучиться этой неаккуратности и воспроизвести ее в дополнительном наборе данных. Тогда как подход на основе нейросетевой генерации смог обучиться только повторять вектора движения ручки по листу бумаги, без возможности воспроизвести неровность чернил. Это оказалось важной характеристикой последнего почерка.

В целом, результаты, полученные от аугментации, согласуются с результатами, полученными в работе [10]. В ней авторы также проводили замеры качества распознавания почерка до и после аугментации. Улучшение в 1–2% сопоставимо с нашими результатами.

Проведем еще один эксперимент, который покажет зависимость качества распознавания от количества дополнительных аугментированных данных. В качестве почерка будем использовать второй почерк из предыдущего эксперимента, который показал значительное увеличение качества распознавания при добавлении в него новых данных. У нас уже есть два обученных метода, которые способны генерировать заданный почерк. Для проведения эксперимента дообучим модель *AttentionHTR* на распознавание этого почерка без дополнительных аугментированных данных. Затем с шагом в 100 новых сгенерированных изображений будем дообучать модель и измерять ее качество на каждой такой итерации. В результате получим множество значений метрики *WER*, которые показывают, как изменялось качество распознавания рукописного текста в процессе дообучения модели. Результат эксперимента представлен на Рис. 36 в виде графика.

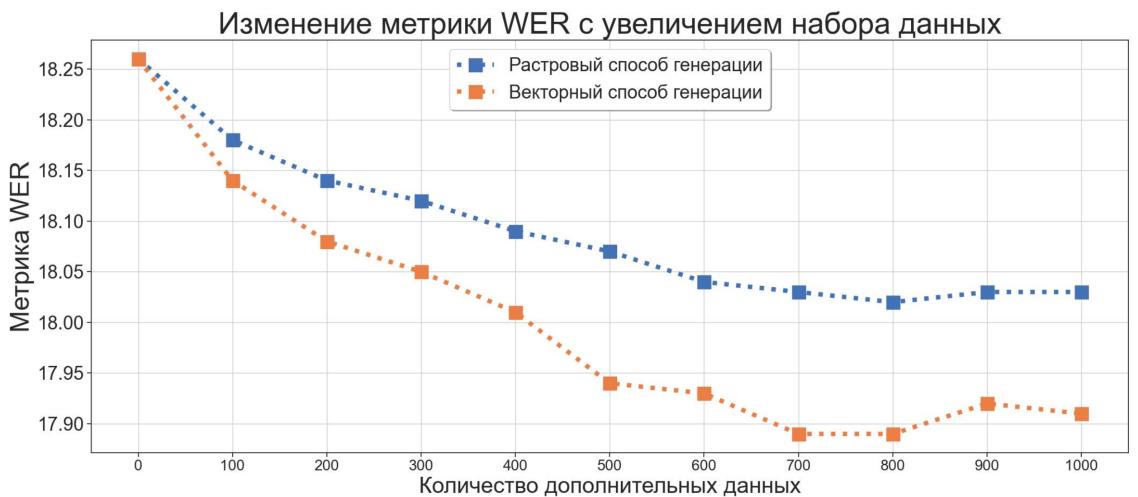


Рис. 36: Изменения качества модели распознавания с увеличением обучающего набора.

Из графика видно, что увеличение набора данных для обучения положительно сказывается на качестве модели распознавания. Причем первые итерации увеличения объема данных имеют больший вклад в повышении качества. В какой-то момент даже хорошая аугментация перестает увеличивать метрики качества. Это связано с тем, что модель распознавания ошибается не по причине плохого обучения, а по причине наличия в тестовых данных странных изображений, на которых даже человек не способен распознать оригинальный текст. Также можно заметить, что график ближе к концу

обучения теряет монотонность. Это связано с тем, что новые данные, которые генерируются предложенными подходами, могут путать модель распознавания на пике ее обучения.

## 10 Заключение

В ходе работы был проведен анализ ряда исследований, посвященных генерации почерка на английском языке. Выделены основные этапы генерации почерка: генерация глифов, генерация лигатур и финальная обработка. Было собрано и обработано несколько наборов данных разных форматов. Проведены экспериментальные исследования современных нейросетевых моделей различных архитектур, подходящих для генерации глифовых изображений на русском языке; отобраны наиболее качественные и перспективные модели. Исследованы современные подходы векторной нейросетевой генерации полноценного рукописного текста и на их основе реализован метод генерации рукописного текста на русском языке с почерком заданного автора. Разработан и программно реализован подход к генерации лигатур. С его помощью можно соединять отдельные глифы в рукописные слова, что позволяет не только генерировать полноценный рукописный текст, но и улучшать существующие подходы. Также реализован многоэтапный подход финальной обработки рукописного текста.

В результате получено два полноценных подхода к генерации рукописного текста на русском языке. Помимо визуального восприятия, предложенные подходы продемонстрировали применимость в задаче аугментации данных. Эксперименты показали, что оба подхода имеют право на существование. Они по-разному решают задачу аугментации рукописного почерка и тем самым имеют свои преимущества и недостатки.

С помощью ряда экспериментов и метрики качества *WER* было доказано, что предложенные подходы генерации рукописного текста способны повысить качество современных нейросетевых алгоритмов распознавания почерка. Также была показана зависимость качества распознавания рукописного текста от количества сгенерированных подходами данных.

В качестве дальнейшего исследования в области генерации рукописного текста можно рассмотреть векторный алгоритм генерации лигатур, который может потенциально повысить качество векторного способа генерации. Также можно подробнее рассмотреть так называемый вектор почерка и изучить влияние его размера и отдельных его составляющих на генерируемый почерк.

## 11 Список литературы

- [1] Yang S. et al. Image data augmentation for deep learning: A survey //arXiv preprint arXiv:2204.08610. – 2022.
- [2] Perez L., Wang J. The effectiveness of data augmentation in image classification using deep learning //arXiv preprint arXiv:1712.04621. – 2017.
- [3] Kumar T. et al. Advanced Data Augmentation Approaches: A Comprehensive Survey and Future directions //arXiv preprint arXiv:2301.02830. – 2023.
- [4] Haines T. S. F., Mac Aodha O., Brostow G. J. My text in your handwriting //ACM Transactions on Graphics (TOG). – 2016. – Т. 35. – №. 3. – С. 1-18.
- [5] Balreira D. G., Walter M. Handwriting synthesis from public fonts //2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). – IEEE, 2017. – С. 246-253.
- [6] Blanes A. R. Synthetic handwritten text generation. – Tech. Rep, 2018.
- [7] Aksan E., Pece F., Hilliges O. Deepwriting: Making digital ink editable via deep generative modeling //Proceedings of the 2018 CHI conference on human factors in computing systems. – 2018. – С. 1-14.
- [8] Alonso E., Moysset B., Messina R. Adversarial generation of handwritten text images conditioned on sequences //2019 international conference on document analysis and recognition (ICDAR). – IEEE, 2019. – С. 481-486.
- [9] Ji B., Chen T. Generative adversarial network for handwritten text //arXiv preprint arXiv:1907.11845. – 2019.
- [10] Fogel S. et al. ScrabbleGAN: Semi-supervised varying length handwritten text generation //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2020. – С. 4324-4333.
- [11] Kotani A., Tellex S., Tompkin J. Generating handwriting via decoupled style descriptors //Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16. – Springer International Publishing, 2020. – С. 764-780.
- [12] Gatys L. A., Ecker A. S., Bethge M. Image style transfer using convolutional neural networks //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – С. 2414-2423.
- [13] Gatys L. A., Ecker A. S., Bethge M. A neural algorithm of artistic style //arXiv preprint arXiv:1508.06576. – 2015.
- [14] Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition //arXiv preprint arXiv:1409.1556. – 2014.

- [15] Ronneberger O., Fischer P., Brox T. U-net: Convolutional networks for biomedical image segmentation //Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. – Springer International Publishing, 2015. – C. 234-241.
- [16] Guo J. et al. UNet-2022: Exploring Dynamics in Non-isomorphic Architecture //arXiv preprint arXiv:2210.15566. – 2022.
- [17] Kingma D. P., Welling M. Auto-encoding variational bayes //arXiv preprint arXiv:1312.6114. – 2013.
- [18] Pastrana R. Disentangling Variational Autoencoders //arXiv preprint arXiv:2211.07700. – 2022.
- [19] Goodfellow I. et al. Generative adversarial networks //Communications of the ACM. – 2020. – T. 63. – №. 11. – C. 139-144.
- [20] Mirza M., Osindero S. Conditional generative adversarial nets //arXiv preprint arXiv:1411.1784. – 2014.
- [21] Curtó J. D. et al. High-resolution deep convolutional generative adversarial networks //arXiv preprint arXiv:1711.06491. – 2017.
- [22] Cho K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation //arXiv preprint arXiv:1406.1078. – 2014.
- [23] Sherstinsky A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network //Physica D: Nonlinear Phenomena. – 2020. – T. 404. – C. 132306.
- [24] Nurseitov D. et al. Handwritten Kazakh and Russian (HKR) database for text recognition //Multimedia Tools and Applications. – 2021. – T. 80. – C. 33075-33097.
- [25] Kass D., Vats E. AttentionHTR: Handwritten text recognition based on attention encoder-decoder networks //Document Analysis Systems: 15th IAPR International Workshop, DAS 2022, La Rochelle, France, May 22–25, 2022, Proceedings. – Cham : Springer International Publishing, 2022. – C. 507-522.
- [26] Sutskever I., Vinyals O., Le Q. V. Sequence to sequence learning with neural networks //Advances in neural information processing systems. – 2014. – T. 27.
- [27] Hochreiter S., Schmidhuber J. Long short-term memory //Neural computation. – 1997. – T. 9. – №. 8. – C. 1735-1780.
- [28] Bahdanau D., Cho K., Bengio Y. Neural machine translation by jointly learning to align and translate //arXiv preprint arXiv:1409.0473. – 2014.
- [29] Levenshtein V. I. et al. Binary codes capable of correcting deletions, insertions, and reversals //Soviet physics doklady. – 1966. – T. 10. – №. 8. – C. 707-710.