МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА
Рецензент,
Ведущий разработчик
ООО «ВОРТЕКСКОД»
_____П.А. Михайлов
“___”_____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
д.ф.-м.н., профессор
_____Л.Б. Соколинский
“___”_____2019 г.

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ГЕНЕРАЦИИ ИЗОБРАЖЕНИЙ В ЗАДАННОЙ ХУДОЖЕСТВЕННОЙ СТИЛИСТИКЕ С ПРИМЕНЕНИЕМ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2019.308-164.ВКР

Научный руководитель,
ст. преподаватель кафедры СП,
_____К.Ю. Никольская

Автор работы,
студент группы КЭ-401
_____М.Г. Бардина

Ученый секретарь
(нормоконтролер)
_____О.Н. Иванова
“___”_____2019 г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
04.02.2019

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

студентке группы КЭ-401
Бардиной Марии Георгиевне,
обучающейся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. Тема работы (утверждена приказом ректора от 25.04.2019 № 899)
Разработка приложения для генерации изображений в заданной художественной стилистике с применением нейросетевых технологий.

2. Срок сдачи студентом законченной работы: 05.06.2019 г.

3. Исходные данные к работе

3.1. Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. [Электронный ресурс] URL: http://arxiv.org/pdf/1511.06434.

3.2. Jin Y., Zhang K., Li M., Tian Y., Zhu H., Fang Z. Towards the Automatic Anime Characters Creation with Generative Adversarial Networks. [Электронный ресурс] URL: http://arxiv.org/pdf/1708.05509.

4. Перечень подлежащих разработке вопросов

4.1. Провести обзор аналогов и научной литературы.

4.2. Подготовить обучающую и тестовую выборку.

4.3. Программно реализовать и обучить нейронную сеть.

4.4. Провести тестирование нейронной сети.

4.5. Разработать веб-приложение и провести его тестирование.

5. Дата выдачи задания: 09 февраля 2019 г.

Научный руководитель
ст. преподаватель кафедры СП К.Ю. Никольская
Задание принял к исполнению М.Г. Бардина

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Нейронные сети – математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма [21].

Генеративно-состязательная нейронная сеть (Generative Adversarial Networks) – алгоритм машинного обучения, работа которого строится на основе двух «соперничающих» нейронных сетей. Принцип алгоритма заключается в том, что одна из этих сетей, называемая генератором, пытается сгенерировать определенные образцы (например, изображения, видео или любые другие данные, на генерацию которых она запрограммирована), а другая сеть, называемая дискриминатором, старается решить, является ли представленный ей образец настоящим или сгенерированным. Задачей генератора является производство таких образцов, которые дискриминатор сочтет настоящими, в то время как задача дискриминатора противоположна – он должен отбраковать сгенерированные образцы [17].

АКТУАЛЬНОСТЬ ТЕМЫ ИССЛЕДОВАНИЙ

За последние годы возрос интерес к направлению в области исследования нейронных сетей, связанному с генерацией изображений. В первую очередь это связано с тем, что была представлена модель генеративно-состязательных нейронных сетей [4], с помощью которой удалось добиться значительных успехов в данной области.

Изображения, созданные подобными нейронными сетями, могут найти практическое применение в различных областях. Одним из вариантов использования является генерация изображений для обучающей выборки в исследованиях, сопряженных с медициной [14]. Помимо этого, подобные технологии могут быть использованы для улучшения качества существующих изображений [10]. Также они могут быть использованы для восстановления утраченных частей изображения [12]. Помимо этого, изображения,

полностью генерируемые искусственным интеллектом, имеют возможность стать перспективным направлением в современном искусстве.

Одной из главных задач в генерации художественных изображений является обучение нейронной сети способности творить в определенном стиле. Поскольку одним из залогов удачного обучения является наличие обширной обучающей выборки было выбрано направление обучения нейронной сети для генерации изображений в стилистике, характерной для японской анимации. Оно является перспективным за счет достаточно обширного количества материала для обучения, имеющего общие стилистические особенности.

ЦЕЛЬ И ЗАДАЧИ ИССЛЕДОВАНИЯ

Целью данной работы является разработка приложения, для генерации изображений в заданной художественной стилистике с применением нейросетевых технологий.

Для достижения поставленной цели необходимо выполнить следующие задачи.

1. Провести обзор существующих аналогов и научной литературы по заданной предметной области.
2. Подготовить обучающую и тестовую выборку изображений в стиле аниме.
3. Программно реализовать и обучить нейронную сеть.
4. Провести тестирование реализованной нейронной сети.
5. Разработать веб-приложение, генерирующее изображения и провести его тестирование.

СТРУКТУРА И ОБЪЕМ РАБОТЫ

Работа состоит из введения, 5 разделов, заключения, библиографии и приложения. Объем работы составляет 35 страниц, объем библиографии составляет 21 источник.

В первой главе приводится анализ предметной области, обзор аналогов, а также существующих решений поставленной задачи.

Во второй главе содержится описание архитектуры нейронной сети.

В третьей главе приведены функциональные и нефункциональные требования к системе, диаграмма вариантов, а также спецификация.

В четвертой главе содержится описание процесса разработки приложения.

В пятой главе приведены результаты тестирования системы.

В заключении описываются результаты проделанной работы, а также направления дальнейших исследований.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Анализ работ по генерации изображений

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [16]

В данной работе авторы предлагают новую архитектуру нейронной сети – глубокую сверточную генеративно-состязательныую сеть (Deep Convolutional Generative Adversarial Network). Основной идеей данной архитектуры является использование слоев свертки, как в генеративной, так и в дискриминативной модели. Архитектура генеративной модели данной сети представлена на рис. 1.

Данная модификация позволяет сделать обучение генеративно-состязательной сети более устойчивым.

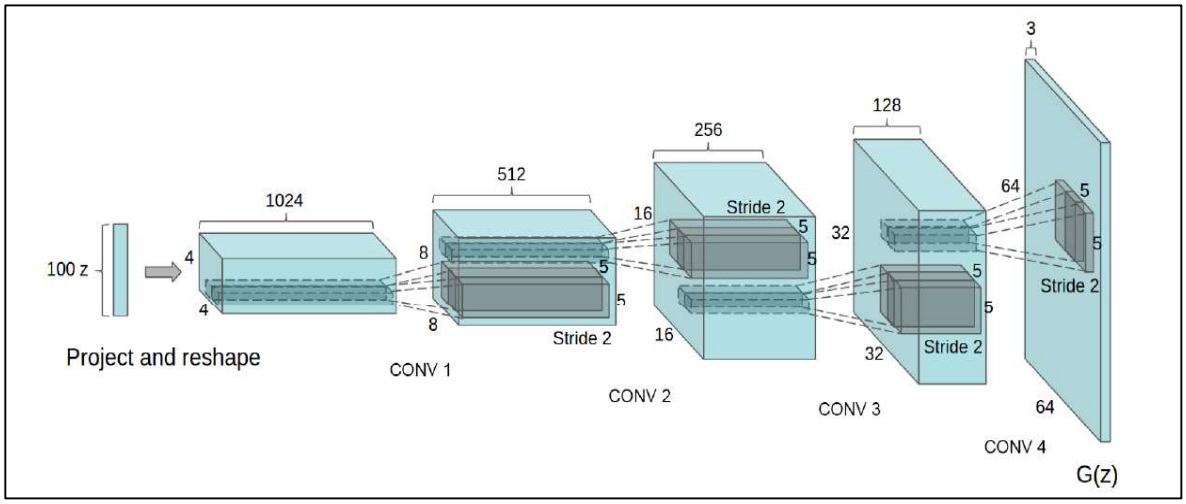

Рис. 1. Схема генеративной модели

В статье указываются основные требования, необходимые для построения новой архитектуры.

1. Заменить все слои подвыборки (pooling) на свертки с пробелами (strided convolutions) для дискриминативной модели, и на частичные свертки с пробелами (fractional-strided convolutions) для генеративной модели.

2. Использовать пакетную нормализацию (batch normalization) в дискриминативной и генеративной моделях.

3. Отказаться от использования полносвязных слоев для более глубоких архитектур.

4. Использовать как функцию активации усеченное линейное преобразование (Rectified Linear Units (ReLU)) для всех скрытых слоев генеративной модели, за исключением выходного, где необходимо воспользоваться функцией гиперболического тангенса (Tanh).

5. Использовать как функцию активации усеченное линейное преобразование «с утечкой» (Leaky Rectified Linear Units (LeakyReLU)) в дискриминативной модели для всех слоев.

В работе авторы демонстрируют, что данный алгоритм хорошо показывает себя в задаче генерации изображений, в том числе – лиц людей (рис. 2).

Рис. 2. Лица, сгенерированные сетью

Auto-Encoding Variational Bayes [9]

Данная работа описывает тип нейронной сети, называемый вариационные автоэнкодеры. Схема данной сети представлена на рис. 3.

Автоэнкодеры отличаются тем, что количество нейронов на входе у них совпадает с количеством на выходе. Суть их работы заключается в сжатии входных данных и отображении их в скрытое пространство (latent-space), с последующим восстановлением, с целью получить данные, как наиболее близкие к входным. Сеть, сжимающую данные, именуют Кодером, а сеть, восстанавливающую их – Декодером. Автоэнкодеры, чаще всего, применяются для сглаживания шума и снижения размерности.

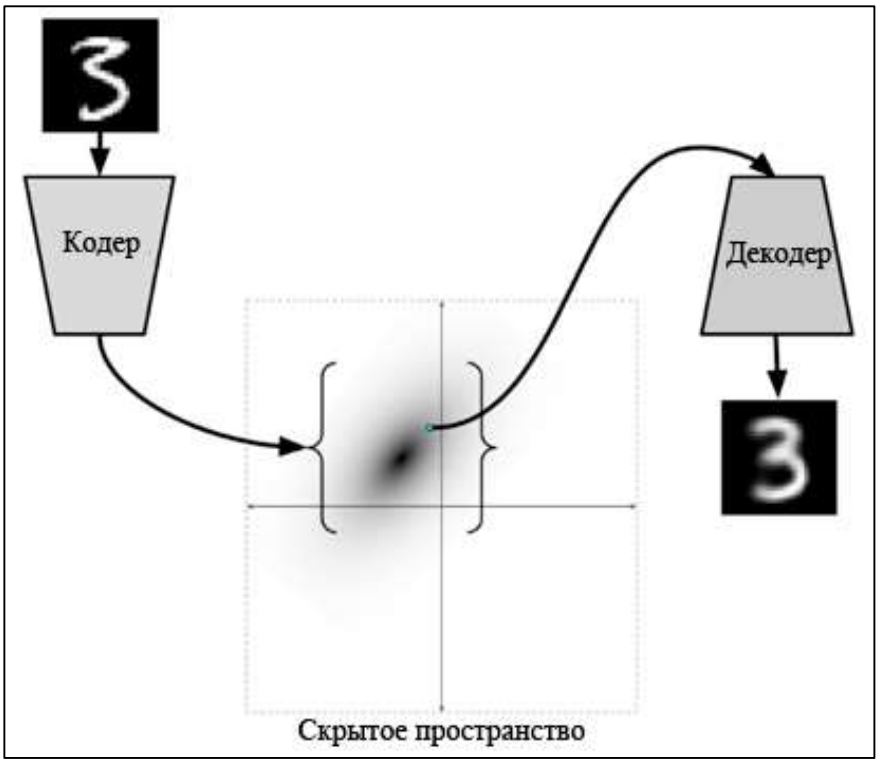

Рис. 3. Схема вариационного автоэнкодера

Вариационные автоэнкодеры, в отличие от авторэнкодеров, являются генеративными моделями. С их помощью выделяется информация о параметрах вероятностного распределения, характерного для набора данных. На основе этой информации, для генерации используется Декодер, генерирующий изображение, имеющие сходные характеристики с входными данными.

Недостатком данного подхода является недостаточное качество полученных результатов – изображения зачастую получаются размытыми. Пример изображений приведен на рис. 4.

Рис. 4. Пример изображений, сгенерированных вариационный автоэнкодером

DRAW: A Recurrent Neural Network For Image Generation [5]

В данной работе представлена архитектура нейронной сети Deep Recurrent Attentive Writer (DRAW). Предложенная авторами статьи нейронная сеть является разновидностью вариационного автоэнкодера. Главной особенностью архитектуры является то, что как для Кодера, так и для Декодера используются рекурентные нейронные сети. Пример генерации сетью цифр представлен на рис. 5.

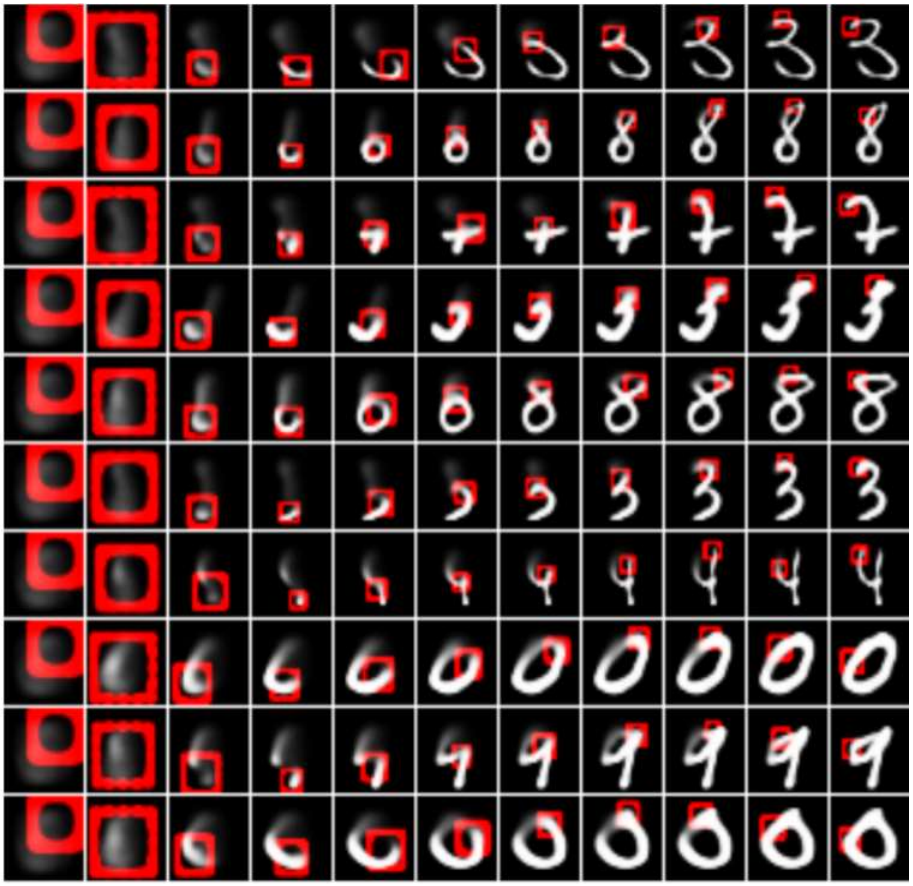

Рис. 5. Цифры, сгенерированные сетью DRAW

Отличительной особенностью является поэтапный процесс генерации изображения, схожий с процессом рисования в реальности.

1.2. Обзор аналогов и существующих решений

В процессе анализа предметной области был выделен один прямой, а также несколько частичных аналогов разрабатываемого проекта. Их краткое описание приведено ниже.

Towards the Automatic Anime Characters Creation with Generative Adversarial Networks [8]

Наибольших успехов удалось добиться в данной работе. Авторы используют Deep Regret Analytic Generative Adversarial Networks, архитектура которой приведена на рис. 6.

Разработчиками также был создан вебсайт [11], на котором пользователю предоставляется выбор из 4-х обученных моделей для генерации. Сервис позволяет создавать картинки размерностью 256x256, и также дает выбор из 12 параметров, предназначенных для персонализации изображения. Интерфейс сайта и пример сгенерированного изображения приведены на рис. 7.

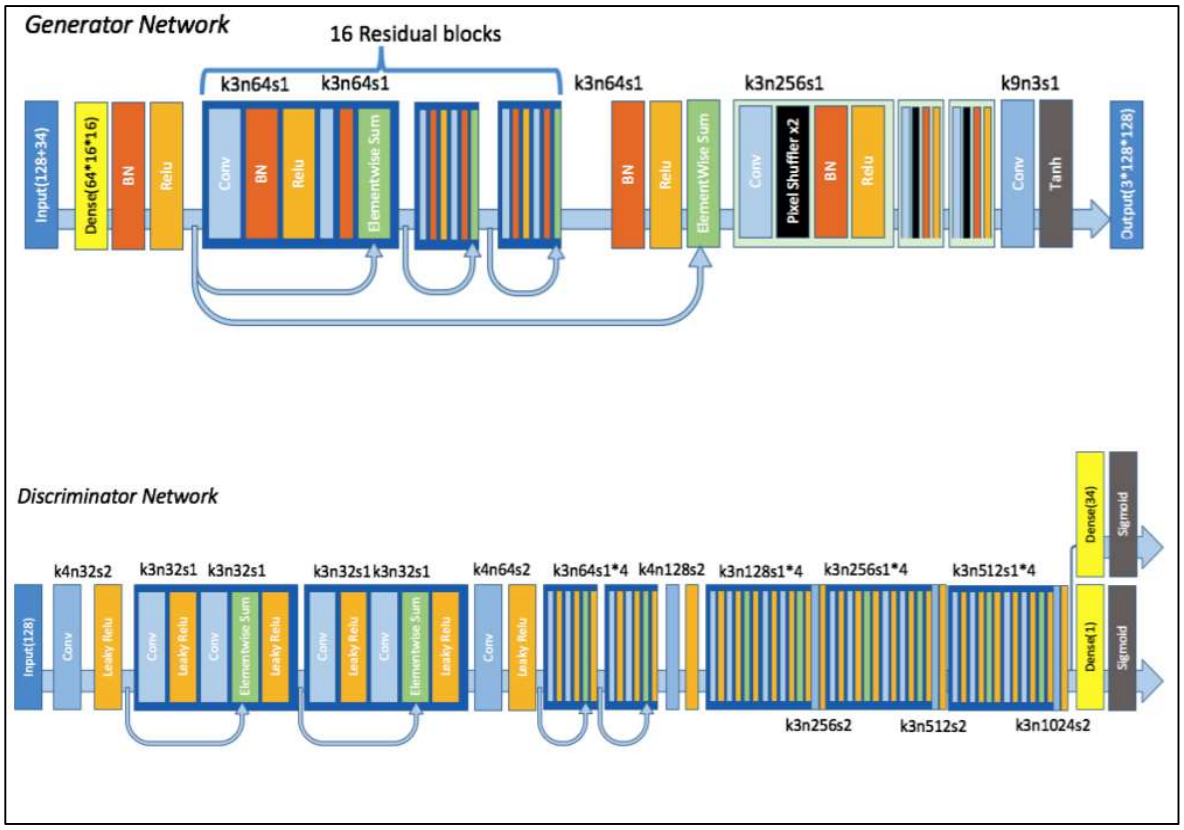

Рис. 6. Архитектура генеративной и дискриминативной моделей

AnimeGAN [2]

Данный аналог представляет собою проект на GitHub на тематику, схожую с поставленной задачей. Для генерации изображений используется глубокая сверточная генеративно-состязательная нейронная сеть [13].

Недостатком данного решения является отсутствие возможности протестировать систему пользователям, не имеющим навыков программирования.

Сгенерированные изображения представлены на рис. 8. Стоит отметить, что их размер и качество заметно уступают конкурентам.

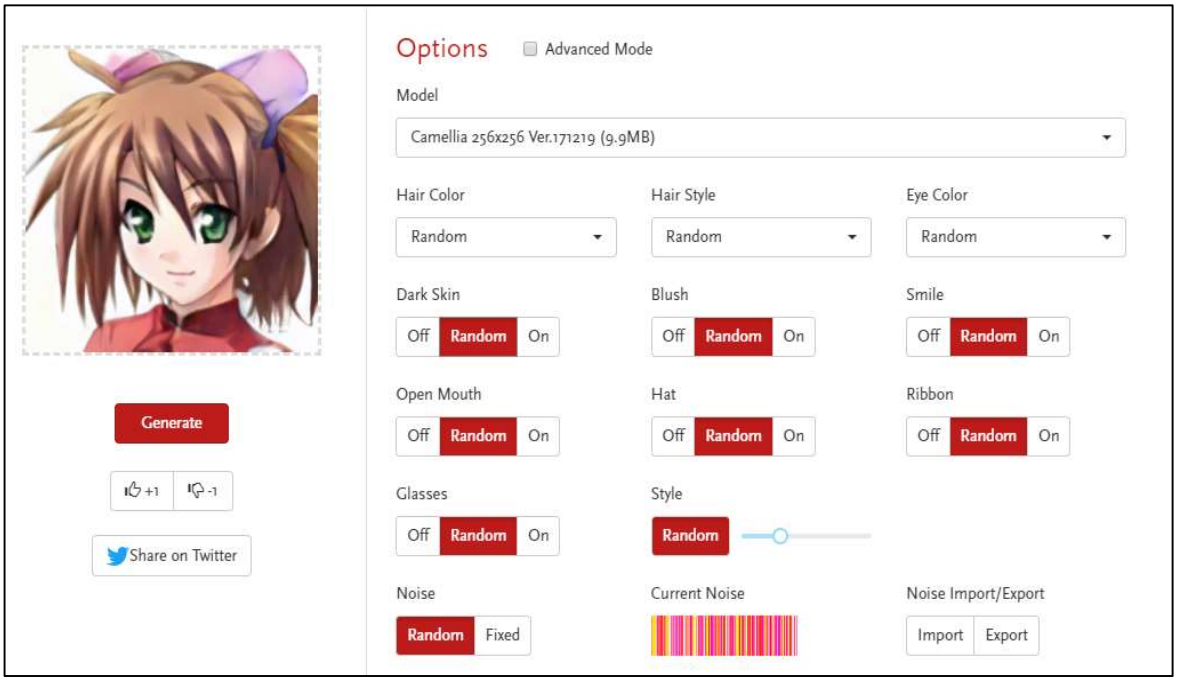

Рис. 7. Интерфейс создания изображения

Рис. 8. Результат работы AnimeGan

IllustrationGAN [3]

Функционал данного сервиса схож с предыдущим. Главными различиями являются изменения, внесенные в архитектуру нейронной сети, а также более тщательно подготовленная обучающая выборка.

Результат работы алгоритма представлен на рис. 9. Так же, как и у предыдущего сервиса, отсутствует реализация приложения, использующего результаты работы нейронной сети.

Рис. 9. Результат работы IllustrationGan

1.3. Обзор решений для создания нейронный сетей

Существует большое количество программных решений, связанных с машинным обучением, для языка Python. Далее будут рассмотрены основные их них.

TensorFlow [20]

Библиотека от Google, которая была разработана специально для обучения нейронных сетей. Для хранения данных в данной библиотеке используются многомерные массивы – тензоры, а для представления нейронной сети используются графы.

Keras [18]

Фрэймворк для создания нейронные сетей, являющийся надстройкой над Tensorflow, что позволяет использовать возможности этой библиотеки для проведения вычислений. Отличительными особенностями Keras являются простота в использовании, модульность и легкость масштабирования.

PyTorch [19]

Библиотека для машинного обучения от Facebook, обеспечивающая тензорные вычисления с GPU-ускорением. Характеризуется гибкостью в использовании за счет того, что в ней не используются статические расчет-

ные графы. Плюсами библиотеки являются удобная отладка, хорошая документация, а также большое количество готовых примеров в реализации генеративно-состязательных нейронных сетей.

В ходе обзора библиотек было решено использовать PyTorch, как наиболее гибкую, интуитивно понятную. Еще одним важным фактором было наличие реализаций схожих проектов с помощью данной библиотеки.

Выводы по первому разделу

Обзор существующих решений показал, что задача генерация изображения с помощью нейросетевых технологий, в том числе, в выбранной стилистике, является актуальной.

Анализ работ по генерации изображений показал, что из существующих решений наибольшего качества изображения можно добиться, используя генеративно-состязательные нейронные сети.

На данный момент существует несколько близких аналогов разрабатываемой системы. Часть этих аналогов, однако, реализует исключительно генератор картинок на основе нейронной сети, и не применяет получившиеся результаты для создания какого-либо программного решения.

Также стоит отметить существование широкого ряда библиотек для создания и обучения нейронных сетей. Использование подобных библиотек может значительно облегчить задачу реализации.

Принимая во внимание все описанные выше факторы, можно сделать вывод, что исследование в данной области является актуальным.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1. Сверточные нейронные сети

Архитектура сверточных нейронных сетей была разработана Яном Лекуном (Yan LeCun). За счет того, что подобные сети имеют возможность учитывать двумерную топологию изображений, они оказываются достаточно эффективными в задачах обработки и анализа изображений. Основной идеей архитектуры является чередование слоев свертки и слоев подвыборки. Схема работы сети представлена на рис. 10.

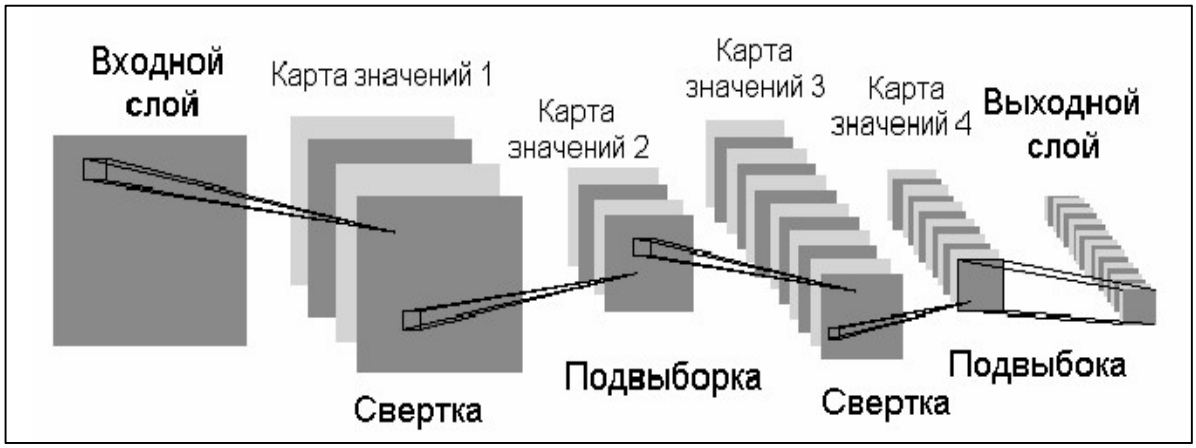

Рис. 10. Модель сверточной нейронной сети

Каждый слой свертки представляет из себя набор из одной или нескольких карт признаков, у каждой из которых есть сканирующее ядро. Ядро представляет из себя матрицу весов. С заданным шагом, окно, совпадающее по размерам с ядром, проходит по всему изображению. На каждом шаге содержимое окна и ядра поэлементно умножаются, после чего полученные произведения суммируются, и результат записывается в результирующую матрицу. Один из этапов процесса свертки представлен на рис. 11.

После осуществления свертки, зачастую, идет операция подвыборки. Целью данной операции является уменьшение размерности карт признаков, что позволяет выделить наиболее важные признаки, а также увеличить скорость обучения. Для этого вся сверточная матрица делится на непересекающиеся окна, внутри которых ко всем элементам применяется определенная функция. Из результатов применения данных функций формируется новая

матрица, меньшей размерности. Зачастую, для подвыборки используется функция максимума. Пример приведен на рис. 12.

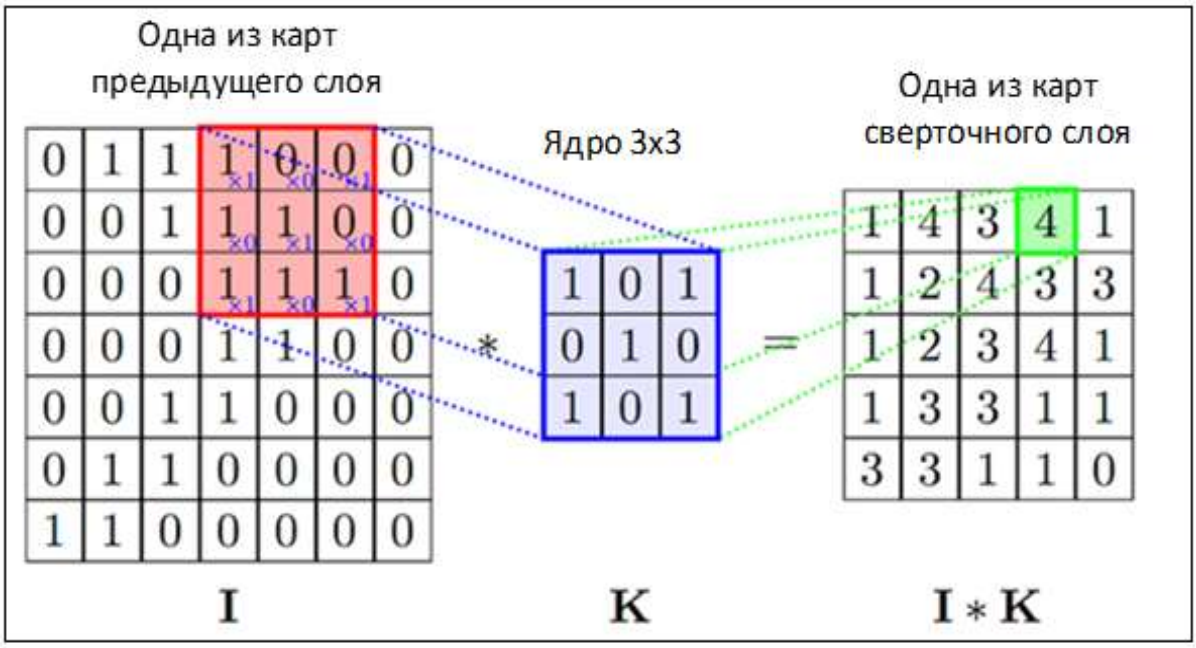

Рис. 11. Пример работы слоя свертки

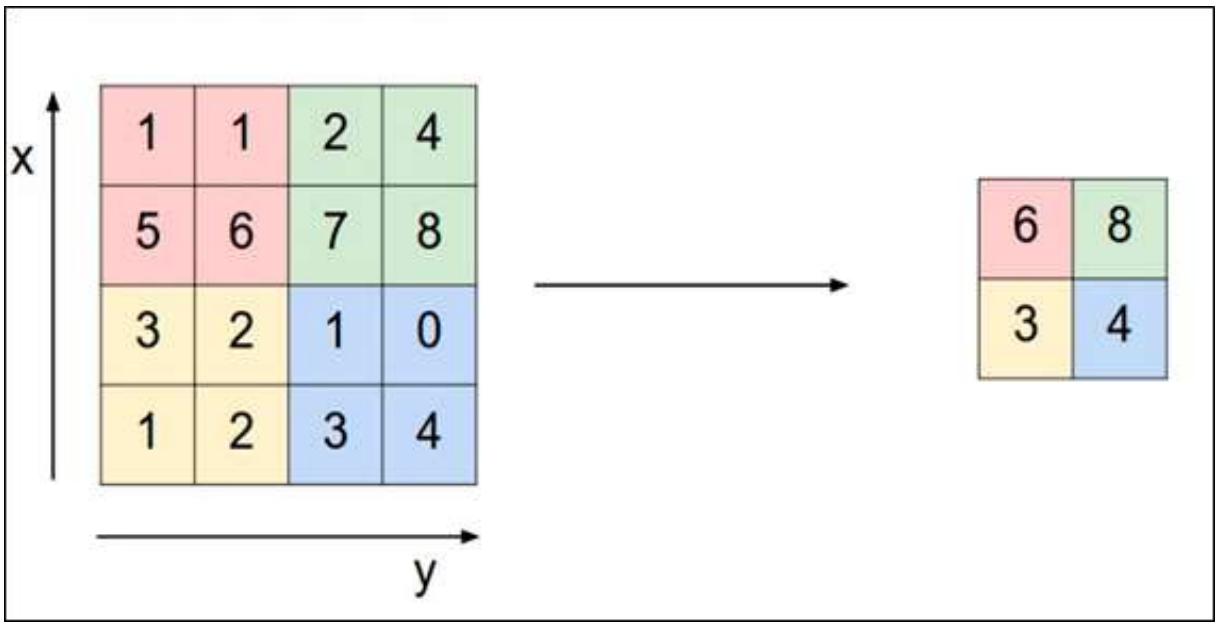

Рис. 12. Пример работы слоя подвыборки

В последнее время, при проектировании сверточных нейронных сетей, для задачи уменьшения размерности, отказываются от слоя подвыборки в пользу сверточных слоев с использованием увеличенного шага скольжения ядра.

2.2. Генеративно-состязательные нейронные сети

Данный тип архитектуры нейронных сетей был предложен в 2014 году Йеном Гудфеллоу (Ian Goodfellow) [4]. Модель сети представлена на рис. 13. Основной идеей представленного подхода является соперничество двух нейронных сетей, именуемых Генератором и Дискриминатором. Первая сеть, получает на вход случайный шум, после чего пытается сгенерировать на его основе объект. Далее, данный объект получает на вход Дискриминатор, который пытается определить, является ли полученный объект реальным, или же сгенерированным. Задачей Генеративной модели будет являться генерация таких изображений, которые будут неотличимы от настоящих для Дискриминативной модели. Предполагается, что, соревнуясь, обе модели будут обучаться.

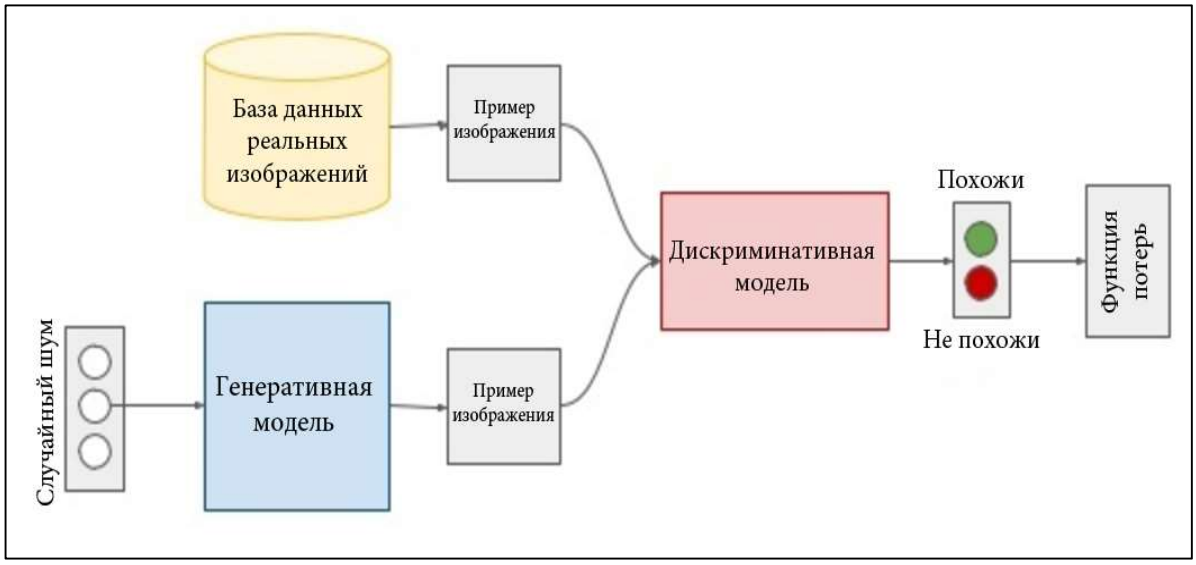

Рис. 13. Модель генеративно-состязательной сети

Визуализация подхода представлена на рис. 14.

Линяя из черных точек – это распределение данных, которое мы хотим приблизить нейросетью.

Зеленая линия – это распределение генератора, в процессе обучения оно приближается к распределению данных (обучение происходит последовательно).

Синяя линия – это разделяющая поверхность дискриминатора, который в конце обучения не может отличить примеры из реального набора данных от подделки, созданной генератором.

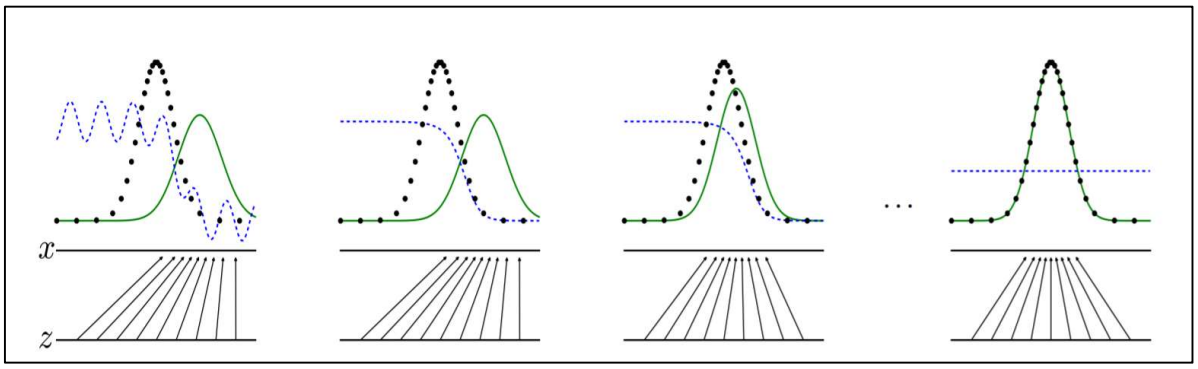

Рис. 14. Визуализация подхода

Стрелками показано отображение, которое ставит в соответствие каждому значению из распределения данных точку из априорного распределения.

Выводы по второму разделу

В данном разделе была рассмотрены теоретические принципы работы сверточных нейронных сетей, а также генеративно-состязательных нейронных сетей.

3. ПРОЕКТИРОВАНИЕ

3.1. Требования к системе

Проектируемая система генерации изображений будет представлять из себя веб-сервис. Пользователь, при взаимодействии с сервисом, сможет сгенерировать изображение и сохранить результат.

Функциональные требования

Функциональные требования – условия, накладывающиеся на поведение системы, а также действия, которые система имеет возможность выполнять. В ходе проектирования приложения были определены следующие функциональные требования.

1. Система должна предоставлять пользователю возможность генерировать изображения в аниме-стилистике на основе обученной нейронной сети.
2. Система должна быть реализована в качестве веб-приложения.
3. Система должна предоставить пользователю выбрать параметры для генерации.

Нефункциональные требования

Нефункциональные требования представляют собой условия, не связанные с поведением системы и характеризующие условия окружающей среды, или же качества, которыми должна обладать система.

Для проектируемой системы были определены следующие нефункциональные требования.

1. Система должна быть реализована на языке Python.
2. Система должна иметь простой и понятный интерфейс.

3.2. Варианты использования системы

Для проектирования приложения был использован язык графического описания для объектно-ориентированного программирования UML. Была построена модель взаимодействия внешнего актера с программной систе-

мой в виде диаграммы вариантов использования (use-case diagram). Полученная диаграмма изображена на рис. 15.

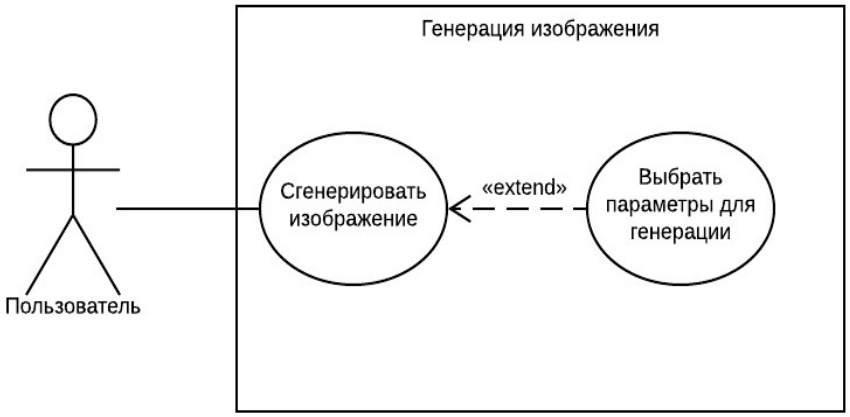

Рис. 15. Диаграмма вариантов использования

В Табл. 1. Представлена спецификация варианта использование «Сгенерировать изображение».

Табл. 1. Спецификация варианта использования «Сгенерировать изображение»

UseCase: Сгенерировать изображение
ID: Generate
Аннотация: Генерация стилизованного изображения
Главные актеры: Пользователь
Предусловия: Отсутствуют
Основной поток: 1. Вариант использования начинает свою работу, когда Пользователь нажимает кнопку «Сгенерировать» 2. Система считывает значения параметров. 3. Система генерирует изображение на основании параметров.
Постусловия: Сгенерированное изображение отображается в приложении
Альтернативные потоки: Отсутствуют
Точки расширения: На 1 шаге основного потока Пользователь может инициировать вариант использования «Выбрать параметры для генерации»

Основные актеры, взаимодействующие с системой

Пользователь – человек, взаимодействующий с системой.

Краткое описание вариантов использования:

Пользователю доступны следующие функции:

1. Сгенерировать изображение – пользователь нажимает на кнопку «сгенерировать», после чего начинается процесс генерации изображения.
2. Выбрать параметры для генерации – пользователь выбирает из списка значения для параметров генерации.

3.3. Компоненты, входящие в систему

Основными компонентами системы являются: MainProgram, Generator, GUI, NeuralNetwork. Диаграмма компонентов системы изображена на рис. 16.

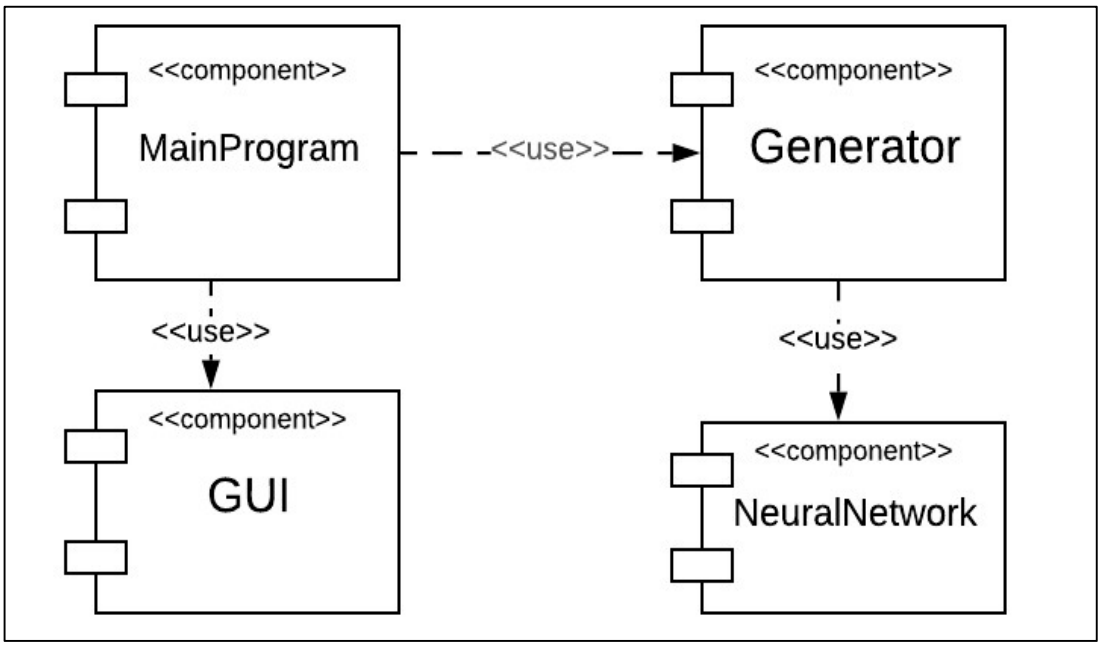

Рис. 16. Диаграмма компонентов системы

MainProgram – главный модуль, инициирующий работу системы и обеспечивающий связь между всеми остальными компонентами.

GUI – графический интерфейс программы. Данный компонент обеспечивает взаимодействие пользователя с системой.

Generator – компонент системы, обеспечивающий связь с моделью нейронной сети и генерацию изображений.

NeuralNetwork – компонент системы, представляющий собою обученную модель нейронной сети.

3.4. Диаграмма деятельности системы

Диаграмма деятельности системы, описывающая процесс работы приложения, представлена на рис. 17.

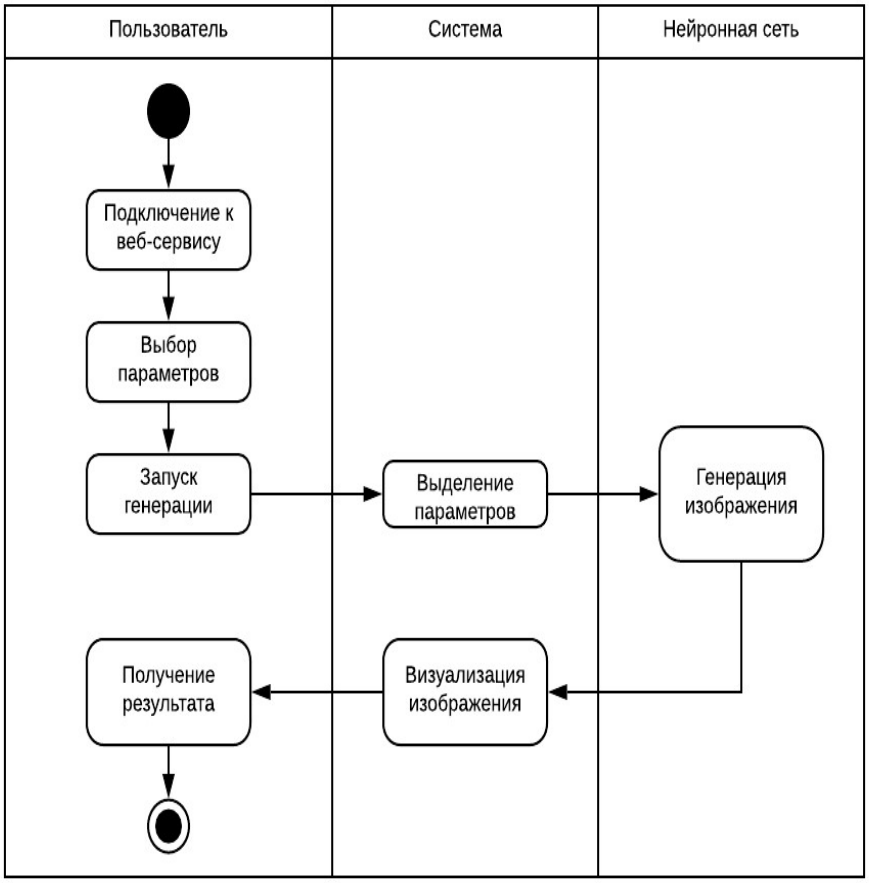

Рис. 17. Диаграмма деятельности системы

3.5. Проектирование графического интерфейса

Интерфейс должен обеспечивать необходимую функциональность, и при этом не быть перегруженным. Интерфейс веб-сервиса состоит из трех основных частей – шапки, основной секции и подвала. В основной секции приложения находится поле для отображения сгенерированного изображения, под которым располагается поле с параметрами для выбора, а также –

кнопка, по нажатию которой будет происходить генерация.

Макет графического интерфейса проектируемого приложения изображен на рис. 18.

Рис. 18. Макет графического интерфейса приложения

Выводы по третьему разделу

Были определены функциональные и нефункциональные требования к системе. Также были составлены диаграммы вариантов использования и деятельности. Была определена топология нейронной сети, а также разработан макет графического интерфейса системы.

4. РЕАЛИЗАЦИЯ

4.1. Средства разработки

Для разработки программной части был использован высоко-уровневый язык программирования Python версии 3.6.4. Обучение нейронной сети проводилась на ОС Ubuntu 18.04 с характеристиками: Nvidia RTX 2080 Ti GPU, 12 CPU cores(Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz) и 32 GB RAM, а разработка веб-сервиса ОС Windows 10. Разработка велась в среде разработки PyCharm Community Edition 2018.2.3.

Для реализации использовались следующие технологии.

1. Для языка программирования Python были использована библиотека машинного обучения TensorFlow [20].
2. Для реализации веб-приложения были использованы технологии HTML5, фреймворк Bootstrap, фреймворк Flask.

4.2. Формирование обучающей выборки

Для создания обучающей выборки были взят датасет от Kaggle [1], состоящий из 33430 изображений. После, просмотрев вручную, из выборки были удалены картинки, не подходящими для обучения (ложно определенные при составлении архива как лица). Количество подобных изображений составило не более 2% от всей выборки. Для успешного обучения сети было необходимо, чтобы изображения обучающей выборки были поделены на категории. С помощью Illustration2Vec [6], предобученной сверточной нейронной сети, предназначенной для присвоения иллюстрациям тэгов. С помощью данной сети, изображениям были присвоенны тэги, далее записанные в файл tags_clean.scv. Пример изображений из одной группы приведен на рис. 19.

Рис. 19. Пример изображений, которым был присвоен тэг «черные волосы»

Для разделения на категории в качестве параметра были выбраны цвет глаз и волос, а также наличие румянца. Всего получилось 25 категорий. Поскольку не все изображения содержали одновременно метки для цвета глаз и волос, при формировании обучающей выборки он был расширен за чет дублирования подходящих изображений, с внесением изменений: поворота, отражения.

4.3. Топология нейронной сети

Для решения представленной задачи решено было использовать генеративно-состязательную нейронную сеть.

Данный подход был выбран поскольку генеративно-состязательные нейронные сети показывают наиболее высокие результаты в задаче генерации образов, в том числе – в стилистике аниме. За основу была взята модель, предложенная в статье [7], а также реализация, представленная в [15].

На рис. 20 представлена структура генеративной модели данной сети. На вход подается изображение. Блок Dense – полносвязный слой. Для сверточных слоев сети на рисунке представлены характеристики: k(kernel) – ядро свертки, n – количество карт признаков, s(stride) – величина шага свертки.

Elementwise Sum – поэлементная сумма.

PixelShuffler x2 – субпиксельные сверточные слои.

Для всех слоев, кроме последнего, в качестве функции активации используется усеченное линейное преобразование (ReLU). На выходе же используется функция гиперболического тангенса (Tanh), что повышает скорость обучения [13], для ускорения обучения используется метод Batch Normalization [8]. Этот метод предполагает добавление дополнительного слоя, который позволяет снизить искажение данных путем их нормализации.

Дискриминативная модель, структура которой отображена на рис. 21, состоит из 20 сверточных слоев с ядром 3x3, а также 6 слоев с ядром 4x4. В

качестве активационной функции сверточных слоев используется LeakyReLu.

На выходе сеть имеет два полносвязных слоя, с последующими функциями сигмоида (Sigmoid). Первый слой, с 25 выходами, используется для классификации лейблов, а второй, с 1, выдает информацию о том, реально или сгенерировано изображение, полученное на вход.

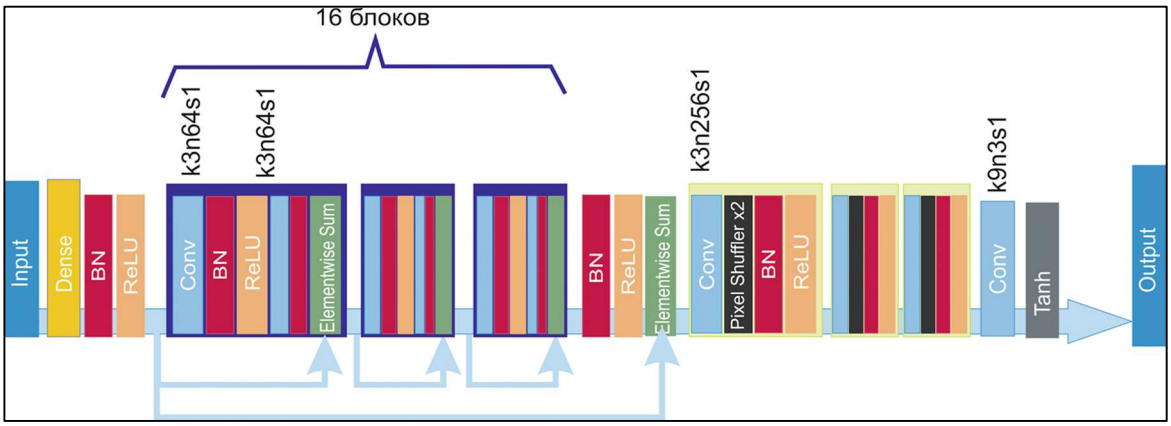

Рис. 20. Структура генеративной модели нейронной сети

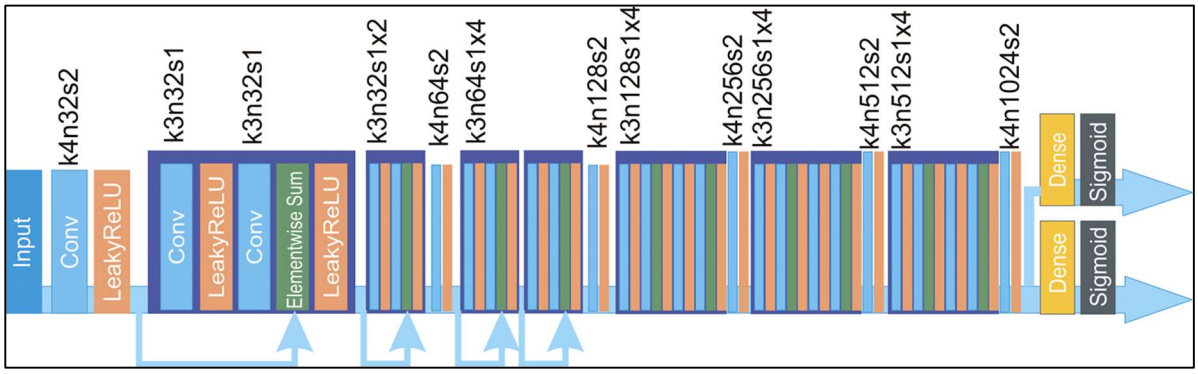

Рис. 21. Структура дискриминативной модели нейронной сети

4.4. Программная реализация нейронной сети, обучение

За основу была взята реализация, представленная в [15].

В рамках разработки системы в нее были внесены изменения. Во-первых, было расширено количество признаков, влияющих на генерацию изображения. Соответствующие изменения были внесены в модуль обработки лейблов, модуль подготовки лейблов для последующей генерации изобра-

жения (см. листинг 1), а также архитектуру нейросети, а именно – в полносвязный слой дискриминативной модели, отвечающий за классификацию по признакам. Во-вторых, чтобы соответствовать решаемой задаче, была изменена функция генерации изображения.

Листинг 1. Функция подготовки лейблов для генерации изображения

```
def load_test(test_path, hair_map, eye_map, blush_map):

    test = []
    with open(test_path, 'r') as f:

        for line in f.readlines():
            hair = 0
            eye = 0
            blush = 0
            if line == '\n':
                break
            line = line.strip().split(',')[1]
            p = line.split(' ')

            p1 = ' '.join(p[:2]).strip()
            p2 = ' '.join(p[-2:]).strip()
            p3 = ' '.join(p[2:3]).strip()

            if p1 in hair_map:
                hair = hair_map[p1]
            elif p2 in hair map:
                hair = hair_map[p2]
            if p1 == 'random hair' or p2 == 'random hair':
                hair = random.randint(0, len(hair_map) - 1)
            if p1 in eye_map:
                eye = eye_map[p1]
            elif p2 in eye_map:
                eye = eye_map[p2]
            if p1 == 'random eyes' or p2 == 'random eyes':
                eye = random.randint(0, len(eye_map) - 1)
            if p3 == 'noblush':
                p3= 'no blush'
            if p3 in blush_map:
                blush = blush_map[p3]
            test.append(make_one_hot(hair, eye, blush))

    return test
```

При обучении используется оптимизатор Adam (Aadaptive Moment Estimation). Результаты, достигнутые при обучении представлены на рис. 22.

Рис. 22. Результат обучения

Во время обучения было пройдено 190500 итераций, коэффициент скорости обучения составлял 0,0002. Стоит отметить, что в большинстве сгенерированных изображений угадываются как черты лица, так и характерный для аниме стиль.

4.5. Веб-интерфейс

Пользовательский интерфейс системы решено было реализовать в виде веб-сервиса. Итоговый вариант приложения представлен на рис. 23. При разработке клиентской части сервиса использовались такие технологии, как HTML 5, а также фреймворк Bootstrap. При разработке серверной части использовался фреймворк Flask на языке программирования Python. Интерфейс приложение состоит из заголовка, поля для отображения результата, а также кнопки «Сгенерировать».

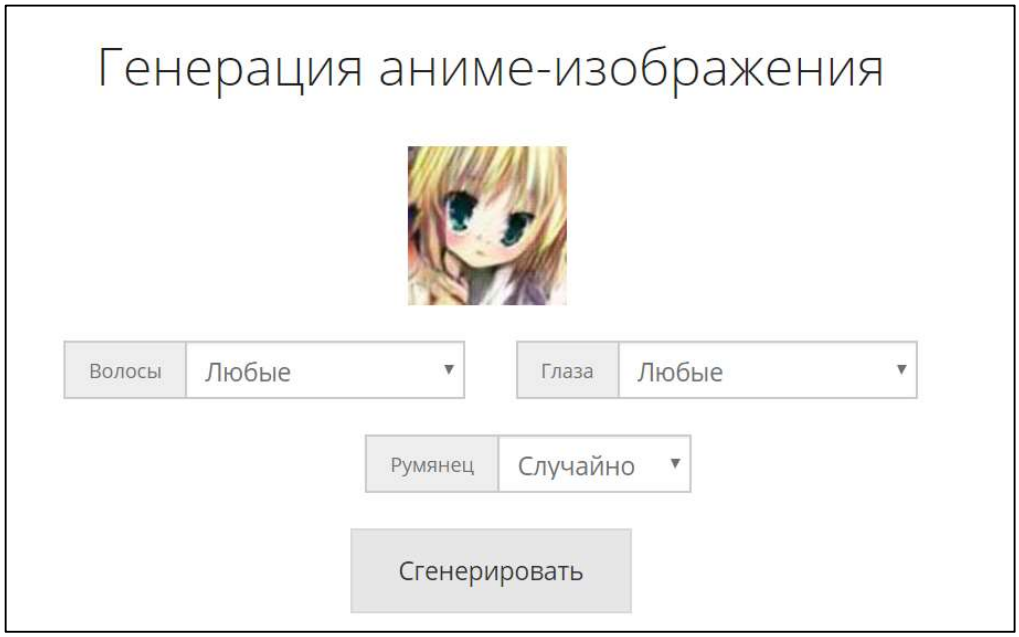

Рис. 23. Приложение по генерации изображений

Принцип работы сервиса заключается в следующем:

1. Пользователь выбирает параметры для генерации из двух выпадающих списков.
2. Пользователь нажимает кнопку «Сгенерировать».

3. На основе обученной нейронной сети (генеративной части) формируется изображение.

4. Изображений отображается в веб-приложении.

5. Повторное нажатие кнопки «Сгенерировать» приведет к генерации нового изображения.

Выводы по четвертому разделу

В ходе выполнения работы была подготовлена обучающая выборка, реализована и обучена генеративно-состязательная нейронная сеть, а также создано веб-приложение, позволяющее пользователям генерировать изображения на основе полученной генеративной нейронной сети.

5. ТЕСТИРОВАНИЕ

Тестирование нейронной сети

Обучение нейронной сети производилось на 190500 итерациях, с коэффициентом скорости обучения 0,0002. При обучении функция потери генератора была равна 36.95585, а функция потери дискриминатора – 1.6073.

В качестве функции подсчета потерь использовалась бинарная кросс-энтропия (Binary (Sigmoid) Cross-Entropy), а для оптимизации – метод адаптивной инерции (Adam Optimizer).

Функциональное тестирование

В соответствии с функциональными требованиями, представленными в разделе 3.1. было выполнено функциональное тестирование. Результаты тестирования приведены ниже.

Тест №1. Цель: Проверить процесс генерации изображения

Действие: Пользователь нажимает кнопку «Сгенерировать»

Ожидаемый результат: Приложение отображает сгенерированное изображение.

Тест пройден? Да.

Тест №2. Цель: проверить возможность повторной генерации изображения.

Действие: Пользователь повторно нажимает кнопку «Сгенерировать».

Ожидаемый результат: Отображается обновленное сгенерированное изображение.

Тест пройден? Да.

Тест №3. Цель: проверить возможность задания параметров для генерации.

Действие: Пользователь выбирает параметры из выпадающих списков, после чего нажимает кнопку «Сгенерировать».

Ожидаемый результат: Сгенерированное изображение соответствует выбранным параметрам.

Тест пройден? Да.

Выводы по пятому разделу

В ходе тестирования была обучена нейронная сеть. Также было проведено тестирование приложения, в ходе которого было выявлено, что система отвечает всем поставленным функциональным требованиям.

ЗАКЛЮЧЕНИЕ

В рамках данной дипломной работы было разработано приложение, для генерации изображений в заданной художественной стилистике с применением нейросетевых технологий.

В ходе выполнения работы были решены следующие задачи.

1. Проведен обзор существующих аналогов и научной литературы по заданной предметной области.
2. Подготовлена обучающая и тестовая выборка изображений в стиле аниме.
3. Программно реализована и обучена нейронную сеть.
4. Проведено тестирование реализованной нейронной сети.
5. Разработано веб-приложение, генерирующее изображения.
6. Проведено его тестирование.

Направление дальнейших исследований является улучшение качества и правдоподобности генерируемых изображений.

ЛИТЕРАТУРА

1. AnimeFace dataset. [Электронный ресурс] URL: http://kaggle.com/daxiangpandaa/animeface (дата обращения: 08.02.2019).

2. GitHub репозиторий проекта AnimeGAN. [Электронный ресурс] URL: https://github.com/jayleicn/animeGAN/ (дата обращения: 20.03.2019).

3. GitHub репозиторий проекта IllustrationGAN. [Электронный ресурс] URL: https://github.com/tdrussell/IllustrationGAN/ (дата обращения: 23.03.2019).

4. Goodfellow I. Generative Adversarial Nets / I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al. // Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, (December 8–13 2014, Montreal, Quebec, Canada). – 2014. – P. 2672–2680.

5. Gregor K. DRAW: A Recurrent Neural Network For Image Generation // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, (Lille, France, 6–11 July). – 2015. – P. 1462–1471.

6. Illustration2Vec. [Электронный ресурс] URL: http://github.com/rezoo/illustration2vec (дата обращения: 08.02.2019).

7. Ioffe S. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift / S. Ioffe, C. Szegedy // Proceedings of the 32nd International Conference on International Conference on Machine Learning. – 2015. – Vol. 37. – P. 448–456.

8. Jin Y. Towards the Automatic Anime Characters Creation with Generative Adversarial Networks / Y. Jin, K. Zhang, M. Li, et al. // ArXiv preprints, 2017. [Электронный ресурс] URL: http://arxiv.org/pdf/1708.05509 (дата обращения: 13.03.2019).

9. Kingma D.P. Auto-Encoding Variational Bayes / D.P. Kingma, M. Welling // 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings.

10. Ledig C. Photo-realistic single image super-resolution using a generative adversarial network / C. Ledig, L. Theis, F. Huszar, et al. //

Conference on Computer Vision and Pattern Recognition, C VPR 2017 (Honolulu, HI, USA, July 21–26, 2017). – P. 105–114.

11. MakeGirlsMoe. [Электронный ресурс] URL: https://make.girls.moe/ (дата обращения: 20.04.2019).

12. Pathak E.D. Context Encoders: Feature Learning by Inpainting / E.D. Pathak, P. Krahenbuhl, J. Donahue, et al. // CVPR 2016. (Las Vegas, NV, USA, June 27-30, 2016) – P. 2536–2544.

13. Radford A. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks / A. Radford, M. Metz, S. Chintala // 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.

14. Schlegl T. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery / T. Schlegl, P. Seeböck, S.M. Waldstein, et al. // Niethammer M. et al. (eds) Information Processing in Medical Imaging. IPMI 2017. Lecture Notes in Computer Science, Vol 10265. Springer, Cham. – P. 146–157.

15. TensorFlow ACGAN Anime Generation [Электронный ресурс] URL: https://github.com/ctwxdd/Tensorflow-ACGAN-Anime-Generation (дата обращения: 08.02.2019).

16. Yousefi F. Unsupervised Learning with Imbalanced Data via Structure Consolidation Latent Variable Model / F. Yousefi, Zh. Dai, C.H. Ek, et al. // CoRR. abs/1607.00067 (2016).

17. Айрапетов А.Э. Исследование генеративно-состязательной сети / А.Э. Айрапетов, А.А. Коваленко // Политехнический молодежный журнал. – 2018. – № 10.

18. Документация Keras. [Электронный ресурс] URL: https://keras.io/ (дата обращения: 13.03.2019).

19. Документация PyTorch. [Электронный ресурс] URL: https://pytorch.org/docs/stable/index.html (дата обращения: 13.03.2019).

20. Документация TensorFlow. [Электронный ресурс] URL: https://www.tensorflow.org/api_docs/python/ (дата обращения: 23.03.2019).

21. Романов В.П. Интеллектуальные информационные системы в экономике: Учебное пособие / В.П. Романов; под ред. д.э.н., проф. Н.П. Тихомирова. – М.: Экзамен, 2003. – 496 с.