Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем
Направление подготовки / специальность 09.04.04 Программная инженерия
Направленность (профиль) / специализация Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____2024 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Генерация двухмерных персонажей с помощью нейронной сети

Исполнитель
студент группы 2105-ом _____ (подпись, дата) А.Р. Шаповалов

Руководитель
доцент, канд. техн. наук _____ (подпись, дата) С.Г. Самохвалова

Руководитель научного
содержания программы
магистратуры
профессор, доктор техн. наук _____ (подпись, дата) И.Е. Еремин

Нормоконтроль
доцент, канд. техн. наук _____ (подпись, дата) Т.А. Галаган

Рецензент
доцент, канд. техн. наук _____ (подпись, дата) И.С. Вирта

Благовещенск 2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____2024 г.

З А Д А Н И Е

К магистерской диссертации студента группы 2105-ом
Шаповалова Алексея Романович

1. Тема магистерской диссертации: Генерация двухмерных персонажей с помощью нейронной сети

(Утверждено приказом от 06.03.2024 № 632-уч)

2. Срок сдачи студентом законченной работы (проекта): 10.06.2024

3. Исходные данные к магистерской диссертации: учебная литература, интернет-ресурсы

4. Содержание магистерской диссертации (перечень подлежащих разработке вопросов): выбор архитектуры нейронной сети; разработка алгоритма взаимодействия пользователя с нейронной сетью; программная реализация решения задачи

5. Рецензент магистерской диссертации: И.С. Вирта

6. Дата выдачи задания 29.01.2024

Руководитель выпускной квалификационной работы: С.Г. Самохвалова, канд. техн. наук, доцент

(фамилия, имя, отчество, должность, уч.степень, уч.звание)

Заявление принял к исполнению _____

РЕФЕРАТ

Магистерская диссертация содержит 60 с., 28 рисунок, 51 источников

ГЕНЕРАЦИЯ ИЗОБРАЖЕНИЙ, ИГРОВОЙ СПРАЙТ, НЕЙРОННАЯ СЕТЬ, ПИКСЕЛЬНАЯ ГРАФИКА, ПРОТОТИП

В работе исследованы нейронные сети, компьютерная графика и генерация игровых спрайтов.

Цель выпускной квалификационной работы является разработка нейронной сети, генерирующей игровые спрайты в пиксельном стиле графики, а также разработать интерфейс для взаимодействия пользователя с нейронной сетью.

В список задач выпускной квалификационной работы входит:

- разработать алгоритм генерации игровых спрайтов;
- анализ существующих архитектур нейронных сетей;
- анализ существующих решений;
- обзор программного обеспечения;
- разработка и обучения нейронной сети;
- создание пользовательского интерфейса;
- тестирование программы.

По завершению всех задач была разработана и протестирована программа, использующая нейронную сеть для генерации и покраски игровых спрайтов. А также разработан интерфейс программы, для удобного использования программы пользователем.

СОДЕРЖАНИЕ

Введение 5

1 Общая характеристика предметной области и объекта исследования 7

1.1 Предметная область и объект проводимого исследования 7

1.2 Обзор существующих методов решений рассматриваемой задачи 23

2 Алгоритмическое и программное обеспечение решения задачи 26

2.1 Предлагаемый алгоритм компьютеризированного решения задачи 26

2.2 Обзор возможностей профильного программного обеспечения 33

2.3 Характеристика выбранного программно-технического обеспечения 35

3 Программная реализация предлагаемого алгоритма решения задачи 40

3.1 Основные этапы практической разработки программного продукта 40

3.2 Примеры фактического тестирования итоговой визуализации 44

Заключение 51

Библиографический список 53

ВВЕДЕНИЕ

Игровая индустрия с каждым годом развивается с огромной скоростью. С ней растёт и количество независимых разработчиков. Так же, для проверки идеи игр или поиска инвесторов, создаются прототипы и демоверсии. Из этого появляется необходимость в быстром создании спрайтов персонажей.

Для быстрого создания спрайтов рационально использовать нейронные сети. Это позволяет сгенерировать множество вариантов для отбора самых удачных. Но нейронные сети или затачиваются под определенные задачи или имеют универсальные решения с средним результатом.

Существуют различные стили графики, например, векторная графика, пиксельная графика, рисованная графика. Большинство нейронных сетей обучены на картинках с высоким разрешением, из-за чего генерация спрайтов, требующих низкое разрешение, происходит с проблемами.

Актуальность темы разработки генератора пиксельных спрайтов с использованием нейронных сетей заключается в том, что такие генераторы могут быть использованы в различных областях, таких как игры, анимация, визуализация, дизайн и т.д. Они могут создавать уникальные и красивые изображения, которые могут использоваться для создания реалистичных персонажей, фонов, логотипов и многого другого.

Кроме того, нейронные сети являются мощным инструментом для обработки изображений, и они могут быть обучены на большом количестве изображений, чтобы создавать более качественные спрайты. Это позволяет генерировать изображения с высокой степенью детализации и реалистичности, что делает их более привлекательными и интересными для использования в различных проектах.

Одно из главных сфер возможного применения генератора – прототипирование игр. Прототипирование – это процесс создания прототипа, который представляет собой прототип будущей игры. Это может быть прототип игрового мира, игрового персонажа, геймплея, интерфейса и другого.

Прототипы позволяют разработчикам игр тестировать различные идеи и концепции (рисунок 1), а также улучшать их перед созданием финальной версии игры.

Рисунок 1 – Прототип с использованием примитивов

Генератор позволяет ускорить процесс создания спрайтов для прототипов с необходимыми параметрами. Что позволяет быстрее реализовать необходимые механики, взаимодействующие со спрайтами или требующие спрайт определенной метрики.

Таким образом, разработка генератора пиксельных спрайтов с помощью нейронных сетей является актуальной темой, которая может быть использована в различных областях и предоставить новые возможности для создания уникальных и красивых изображений.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ИССЛЕДУЕМОЙ ЗАДАЧИ

1.1 Предметная область и объект проводимого исследования

Нейронная сеть – это математическая модель, вдохновлённая структурой и функционированием биологических нейронных сетей, таких как человеческий мозг (рисунок 2). Основная цель нейронной сети – обработка информации и решение задач, таких как классификация, регрессия, кластеризация и генерация данных, имитируя процесс обучения и адаптации. Он создает систему, способную адаптироваться в процессе обучения компьютера на тестовых данных, минимизирую ошибки прогнозов и ответов с каждой итерацией. Таким образом, программные нейронные сети решать сложные задачи автоматизации, такие как обработка документов, с получением результатов и отчетов, или распознавание лиц, с высокой точностью.

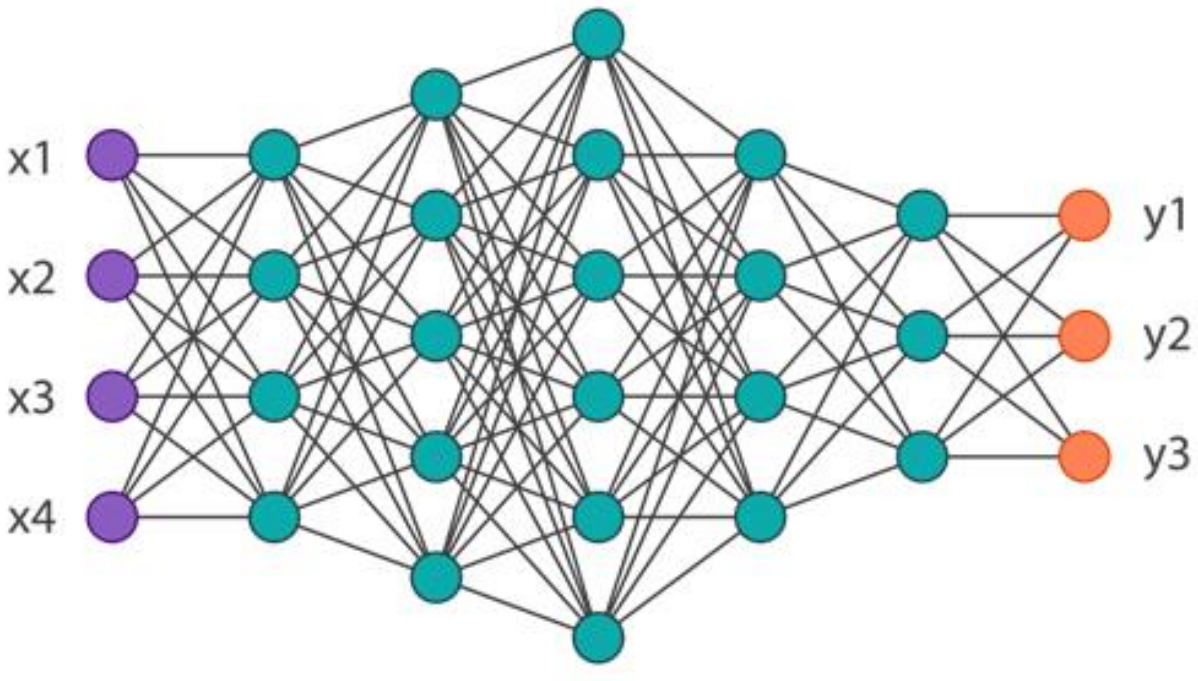

Рисунок 2 – Визуализация нейронной сети

Машинное зрение – это область искусственного интеллекта и компьютерных наук, которая фокусируется на разработке алгоритмов и систем, позволяющих компьютерам «видеть» и интерпретировать визуальную

информацию из окружающего мира. Эта технология применяется в различных сферах, таких как промышленная автоматизация, медицинская диагностика, автономные транспортные средства, системы безопасности и многие другие.

Основные компоненты машинного зрения включают:

- сбор данных: использование камер и датчиков для получения изображений или видео;
- предобработка изображений: коррекция и фильтрация изображений для улучшения качества данных, что может включать изменение яркости, контрастности, удаление шума и т.д.;
- извлечение признаков: выделение ключевых характеристик из изображений, таких как контуры, углы, текстуры и т.п.;
- распознавание объектов: идентификация и классификация объектов на изображениях с использованием различных алгоритмов машинного обучения и нейронных сетей;
- интерпретация и принятие решений: анализ распознанной информации для выполнения конкретных задач, например, обнаружение дефектов на производственной линии или распознавание лиц в системах безопасности.

Примеры применения машинного зрения:

- автономные транспортные средства: использование камер для распознавания дорожных знаков, пешеходов и других транспортных средств;
- промышленная автоматизация: контроль качества продукции, обнаружение дефектов и управление роботизированными системами;
- медицинская диагностика: анализ медицинских изображений (например, рентгеновских снимков или МРТ) для выявления патологий;
- безопасность и наблюдение: распознавание лиц и анализ видео для обеспечения безопасности на общественных и частных объектах;
- сельское хозяйство: мониторинг состояния посевов, сортировка продукции и управление техникой.

Современные системы машинного зрения (рисунок 3) часто используют методы глубокого обучения, в частности, нейронные сети, для повышения точности и эффективности анализа изображений. Это позволяет достигать высокой точности в сложных задачах распознавания и классификации, что делает машинное зрение важным инструментом в развитии технологий искусственного интеллекта. Машинное зрение значительно расширяет возможности автоматизации и анализа данных, делая системы более умными и адаптивными к изменениям окружающей среды.

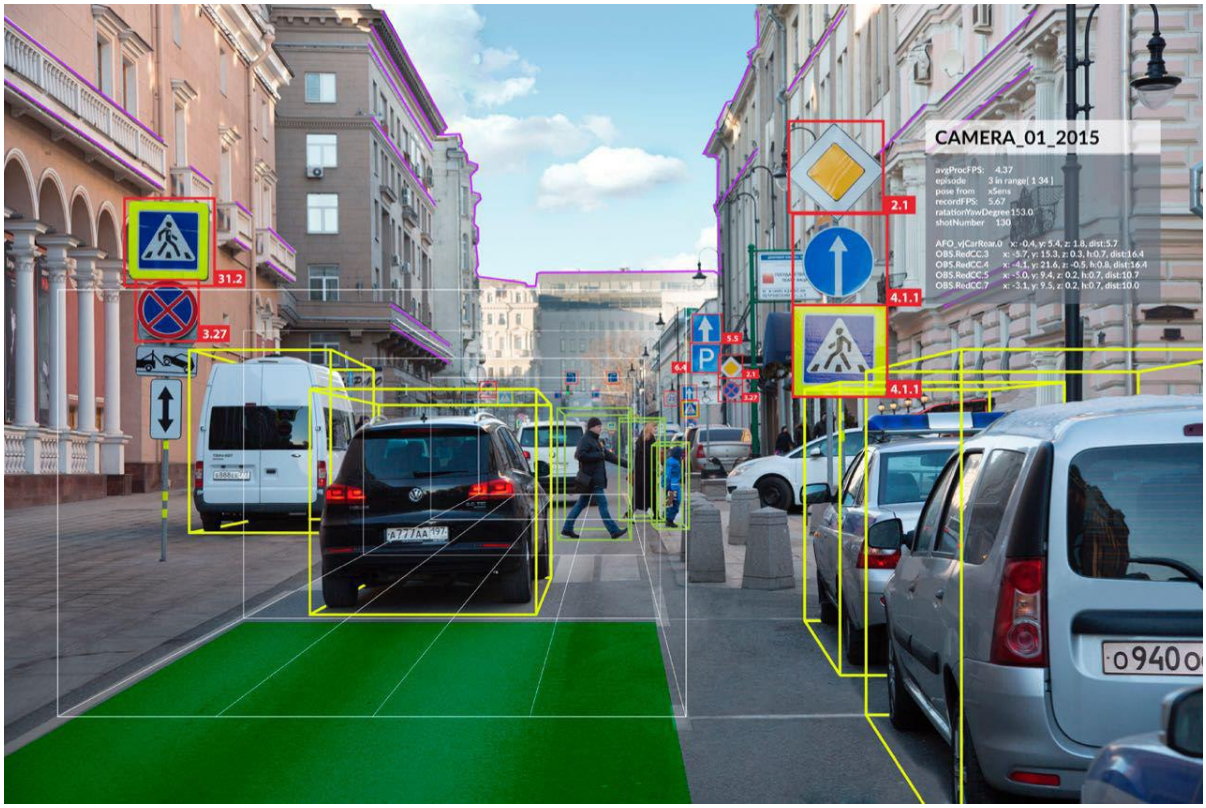

Рисунок 3 – Пример работы машинного зрения

Структура нейронных сетей частично повторяет структуру человеческого мозга. Клетки человеческого мозга называются нейронами, они образуют сложную сеть с большим количеством связей они посылают электрические сигналы друг другу, обрабатывая информацию, поступающую от других органов. И подобно человеческому мозгу, нейронная сеть состоит из нейронов, воссозданных программно, которые взаимодействуют между собой для решения

проблем Нейронные сети, которые часто используются в машинном обучении и искусственном интеллекте, имеют сложную многослойную структуру, состоящую из нескольких базовых компонентов.

Нейроны являются основными строительными блоками нейронных сетей. Каждый нейрон получает входные сигналы, которые умножаются на весовые коэффициенты, суммируются и проходят через активационную функцию для получения выходного сигнала. Основные части нейрона:

- входы (Input): сигналы, поступающие в нейрон;
- веса (Weights): коэффициенты, определяющие важность каждого входа;
- сумматор (Summation): компонент, который суммирует взвешенные входы;
- активационная функция (Activation Function): функция, которая определяет выход нейрона на основе суммы входов.
- Нейроны объединяются в слои, а слои, в свою очередь, образуют нейронную сеть. Слои делятся на три основные категории:
- входной слой (Input Layer): первый слой сети, который принимает исходные данные;
- скрытые слои (Hidden Layers): один или несколько промежуточных слоев, которые обрабатывают данные. Именно здесь происходит основное преобразование данных;
- выходной слой (Output Layer): последний слой, который выдает результат обработки.

Глубокие нейронные сети или сети глубокого обучения могут иметь множество скрытых слоев с миллионами, связанных между собой, нейронами. Вес связи – это число, указывающие на силу связи одного узла с другими. Вес связи может быть как положительным, тогда один нейрон возбуждает другой, так и отрицательным, тогда нейрон подавляет другой нейрон. Соответственно узел с высоким значением веса будет иметь большее влияние на другие узлы.

В зависимости от задач и архитектуры, нейронные сети могут включать различные типы слоев:

- полносвязные (Dense) слои: каждый нейрон одного слоя соединен со всеми нейронами следующего слоя;
- сверточные (Convolutional) слои: используются в сверточных нейронных сетях (CNN) для обработки данных, таких как изображения. Они применяют фильтры для обнаружения различных признаков в данных;
- подвыборки (Pooling) слои: уменьшают размерность данных, сохраняя важные признаки. Пример – max pooling;
- рекуррентные (Recurrent) слои: применяются в рекуррентных нейронных сетях (RNN) для обработки последовательностей данных, таких как текст или временные ряды;
- нормализационные (Normalization) слои: применяют нормализацию данных, что способствует стабильности и ускорению обучения сети.

Активационные функции играют ключевую роль в работе нейронов, определяя выходные значения на основе входных данных. Популярные активационные функции включают:

- ReLU (Rectified Linear Unit): используется в скрытых слоях. f(x) = max(0, x);
- Sigmoid: применяется в выходных слоях для задач бинарной классификации. f(x) = 1 / (1 + exp(-x));
- Tanh (Hyperbolic Tangent): используется для скрытых слоев. f(x) = (exp(x) - exp(-x)) / (exp(x) + exp(-x));
- Softmax: применяется в выходных слоях для задач много классовой классификации. Преобразует входные значения в вероятности.

Процесс обучения нейронной сети включает в себя следующие этапы:

- прямой проход (Forward Pass): входные данные проходят через сеть, и вычисляется выход;
- функция потерь (Loss Function): оценивает ошибку между предсказанным выходом и истинным значением;

- обратное распространение (Backpropagation): обновляет веса сети, минимизируя ошибку, вычисленную функцией потерь, путем обратного прохождения сигнала через сеть и применения градиентного спуска.

Для корректировки весов нейронной сети используются оптимизационные алгоритмы. Популярные алгоритмы включают:

- градиентный спуск (Gradient Descent): основной метод для минимизации функции потерь;
- Adam (Adaptive Moment Estimation): более продвинутый метод, который адаптирует скорости обучения для различных параметров.

Нейронные сети находят широкое применение в различных областях, включая компьютерное зрение, обработку естественного языка, прогнозирование временных рядов, игру в шахматы и другие задачи, требующие интеллектуальной обработки данных.

В теории глубокие нейронные сети (рисунок 4) могут сопоставить любой тип ввода с любым типом вывода. Однако стоит учесть, что для обучения требуется больше времени и ресурсов, что делает его более в сравнении с другими методам машинного обучения. Таким узлам нужны огромное количество примеров, данных для обучения, а не пару десятков или сотен, как в случае с простыми сетями.

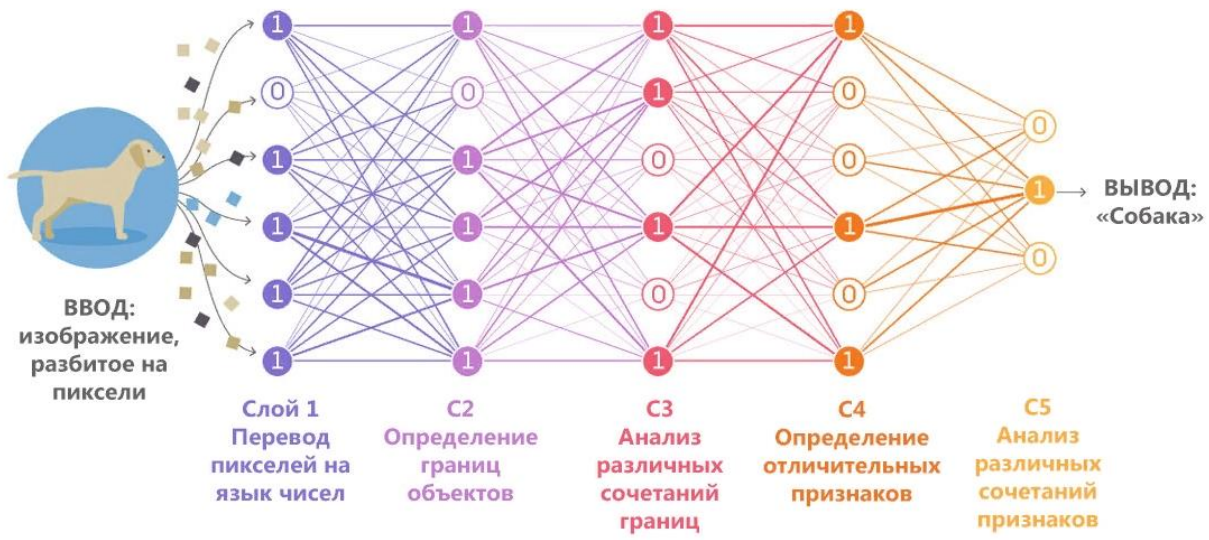

Рисунок 4 – Пример принципа работы глубокой нейронной сети

Алгоритмы ресайза изображений используются для изменения их размеров, сохраняя при этом качество и важные характеристики (рисунок 5). Существует несколько методов ресайза, каждый из которых имеет свои особенности и области применения. Алгоритмов ресайза изображения существует несколько.

Метод ближайших соседей – это самый простой метод ресайза. Он просто берет значение ближайшего пикселя из исходного изображения для каждого пикселя в новом изображении.

Преимущества данного метода в быстроте выполнения и простоте реализации. А недостаток заключается в пикселизации изображения, особенно при увеличении.

Билинейная интерполяция. Этот метод использует взвешенную среднюю величину четырех ближайших пикселей для определения значения нового пикселя.

Преимущества данного метода более гладкие результаты по сравнению с ближайшим соседом и улучшенное качество изображения при изменении размеров. Из недостатков можно выделить более высокую вычислительную сложность по сравнению с ближайшим соседом и метод может приводить к размытию изображения.

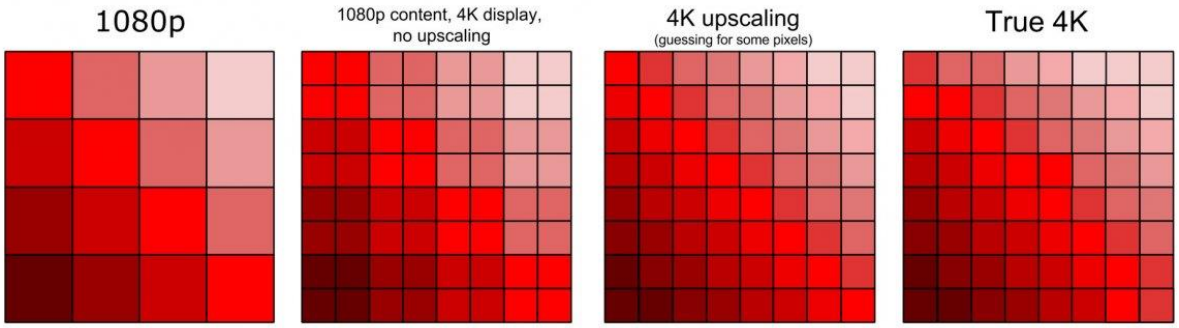

Рисунок 5 – Пример работы ресайза в сравнении с обычным увеличением

Бикубическая интерполяция – это метод, который использует взвешенную среднюю величину 16 ближайших пикселей (4x4 область) для вычисления нового значения пикселя.

Преимущества данного метода в высоком качестве изображения и меньшее количество артефактов и размытости по сравнению с билинейной интерполяцией. Недостаток лишь в более высокой вычислительной сложности.

Ланцош – это алгоритм, использующий синусоидальную функцию для взвешивания пикселей.

Высокое качество изображений и сохранение деталей и резкости – это главные преимущества. А вот высокая вычислительная сложность и возможны артефакты, рябь или ореолы весомые недостатки.

Super Resolution или суперразрешение. Методы суперразрешения используют машинное обучение и нейронные сети для повышения разрешения изображений (рисунок 6). Они обучаются на больших наборах данных и могут восстанавливать мелкие детали, отсутствующие в исходном изображении.

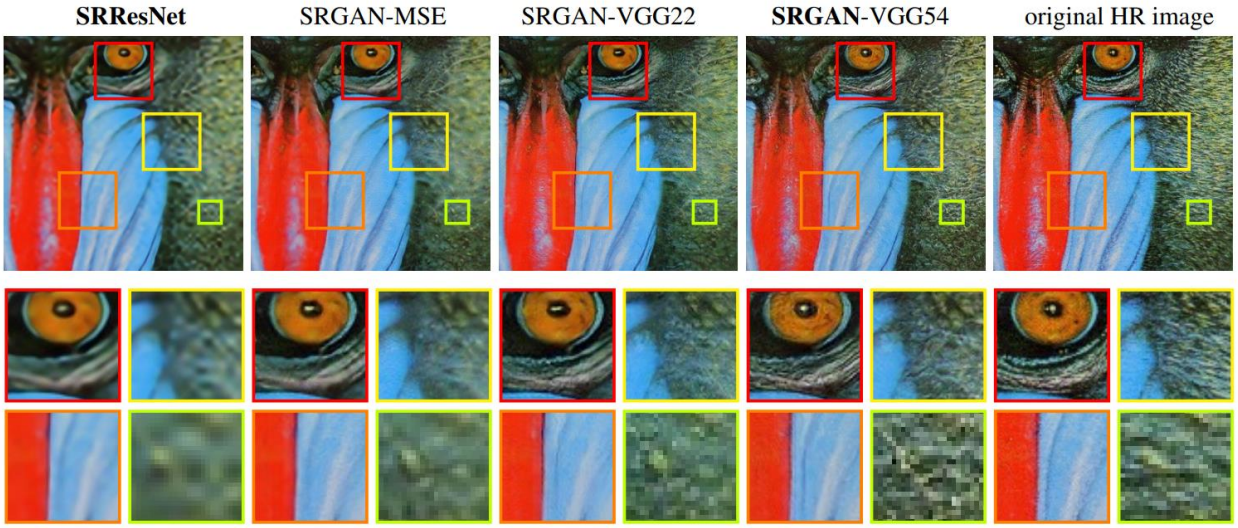

Figure 6: **SRResNet** (left: a,b), SRGAN-MSE (middle left: c,d), SRGAN-VGG2.2 (middle: e,f) and **SRGAN**-VGG54 (middle right: g,h) reconstruction results and corresponding reference HR image (right: i,j). [4× upscaling]

Рисунок 6 – Демонстрация различных методов суперразрешение

Преимущество в очень высоком качестве, получаемых изображений, и возможности восстановления мелких деталей и улучшения четкости

изображения. Но недостатки в требовании обучение модели на больших наборах данных и соответственно высокой сложности вычислений.

Преобразование методом наименьших квадратов – это метод, который использует математические модели для минимизации ошибки между исходным и новым изображением.

Преимущество в сохранении геометрических свойств изображений, а не достаток в том, что метод может быть сложен в реализации и требовать значительных вычислительных ресурсов.

Пирамидальные методы. Эти методы строят иерархию изображений с различными разрешениями и используют их для выполнения ресайза.

Преимущества заключается в эффективности для масштабирования изображений в реальном времени и хороших результатах в задачах понижения разрешения. Недостаток метод может требовать значительного объема памяти для хранения пирамидальных представлений.

Алгоритмы ресайза изображений находят применение в различных областях, таких как:

- компьютерное зрение: подготовка данных для нейронных сетей;
- графический дизайн: изменение размеров изображений для веб-дизайна и печати;
- мультимедиа: изменение размеров видео и изображений для различных устройств и экранов;
- медицина: анализ и обработка медицинских изображений.

Выбор подходящего алгоритма зависит от конкретных требований задачи, таких как скорость, качество и вычислительные ресурсы.

Аффинные преобразования – это класс линейных преобразований, которые включают такие операции, как сдвиг, масштабирование, вращение и отражение (рисунок 7). Эти преобразования сохраняют параллельность прямых и отношения расстояний вдоль прямых, но не обязательно сохраняют углы и длины.

Рисунок 7 – Пример аффинных преобразований

Принцип действия заключается в том, что для каждой точки конечного изображения берется фиксированный набор точек исходного и интерполируется в соответствии с их взаимным положением и выбранным фильтром. Количество точек тоже зависит от фильтра. Для билинейной интерполяции берется 2x2 исходных пикселя, для бикубической 4x4. Такой метод дает гладкое изображение при увеличении, но при уменьшении результат очень похож на ближайшего соседа. Смотрите сами: теоретически, при бикубическом фильтре и уменьшении в 3 раза отношение обработанных пикселей к исходным равно 4^2 / 3^2 = 1,78. На практике результат значительно хуже т.к. в существующих реализациях окно фильтра и функция интерполяции не масштабируются в соответствии с масштабом изображения, и пиксели ближе к краю окна берутся с отрицательными коэффициентами (в соответствии с функцией), т.е. не вносят полезный вклад в конечное изображение. В результате изображение, уменьшенное с бикубическим фильтром, отличается от изображения,

уменьшенного с билинейным, только тем, что оно еще более четкое. Ну а для билинейного фильтра и уменьшения в три раза отношение обработанных пикселей к исходным равно 2^2 / 3^2 = 0.44, что принципиально не отличается от ближайшего соседа. Фактически, аффинные преобразования нельзя использовать для уменьшения более чем в 2 раза. И даже при уменьшении до двух раз они дают заметные эффекты лесенки для линий.

Теоретически, должны быть реализации именно аффинных преобразований, масштабирующие окно фильтра и сам фильтр в соответствии с заданными искажениями, но в популярных библиотеках с открытым исходным кодом они почти не встречаются.

Аффинные преобразования широко применяются в различных областях:

- компьютерная графика: для преобразования изображений и объектов на экране;
- распознавание образов: для нормализации изображений перед их анализом;
- геометрическая обработка изображений: для выравнивания изображений и устранения искажений;
- реальное время: в системах дополненной реальности и компьютерного зрения для корректировки изображений в реальном времени.

Аффинные преобразования являются основой многих алгоритмов обработки изображений и графики, обеспечивая гибкость и мощность для манипулирования визуальными данными.

Кроме того, именно этот метод используется видеокартами для отрисовки трехмерных сцен. Но разница в том, что видеокарты для каждой текстуры заранее подготавливают набор уменьшенных версий (mip-уровней), и для окончательной отрисовки выбирается уровень с таким разрешением, чтобы уменьшение текстуры было не более двух раз. Кроме этого, для устранения резкого скачка при смене mip-уровня (когда текстурированный объект приближается или отдаляется), используется линейная интерполяция между соседними mip-уровнями (это уже трилинейная фильтрация). Таким образом для

отрисовки каждого пикселя трехмерного объекта нужно интерполировать между 2^3 пикселями. Это дает приемлемый для быстро движущейся картинки результат за время, линейное относительно конечного разрешения.

Суперсемплинг (super sampling) – это метод улучшения качества изображения, применяемый в графике и компьютерном зрении, который заключается в рендеринге изображения с более высоким разрешением, чем требуется, и последующем уменьшении его до нужного размера. Этот процесс помогает сгладить края объектов и уменьшить артефакты, такие как алиасинг (рисунок 8)

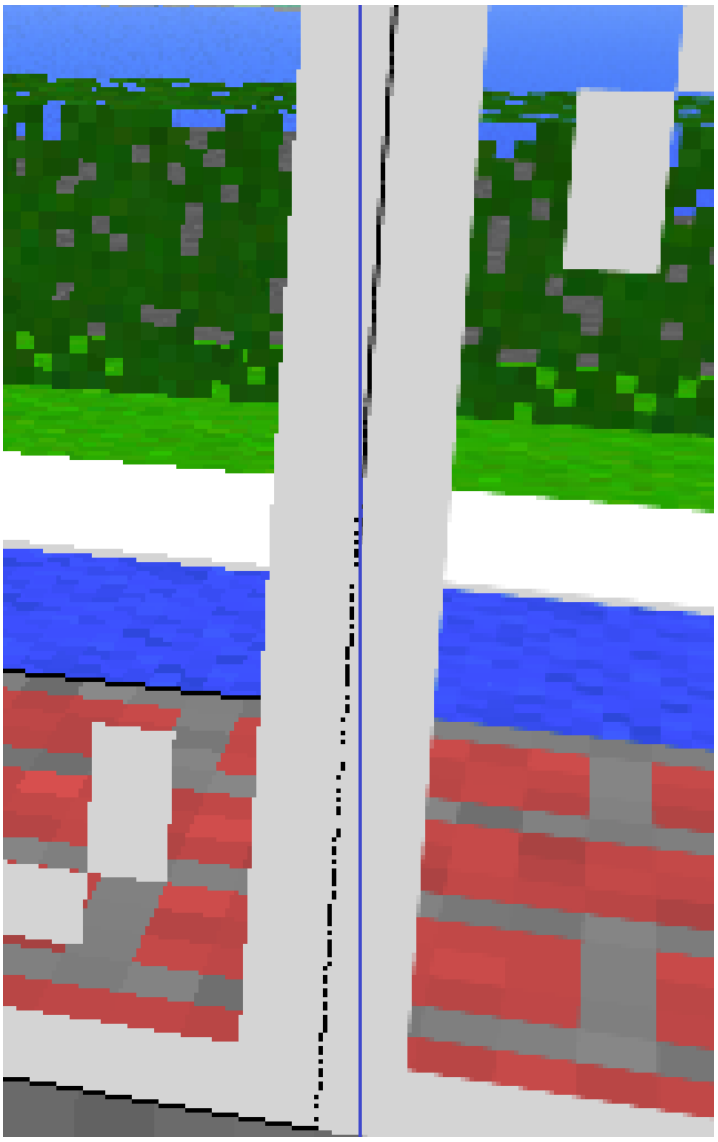

Рисунок 8 – Сравнение изображений без сглаживания и с ним

Основные этапы суперсемплинга:

- рендеринг с высоким разрешением. Изображение рендерится с разрешением, значительно превышающим целевое. Например, если конечное изображение должно быть 1920x1080 пикселей, его можно сначала отрендерить с разрешением 3840x2160 пикселей (в 4 раза большее разрешение);
- фильтрация. Применяются различные фильтры для устранения шумов и артефактов в изображении с высоким разрешением;
- уменьшение до целевого разрешения. Отрендеренное изображение уменьшается до целевого разрешения с использованием методов интерполяции, таких как билинейная или бикубическая интерполяция. При этом мелкие детали усредняются, что позволяет сгладить края и уменьшить эффект алиасинга.

Преимущества суперсемплинга:

- улучшенное качество изображения: сглаживание краев объектов и устранение артефактов;
- реалистичность: повышенная четкость и реалистичность изображений, особенно при рендеринге сцен с мелкими деталями;
- гибкость: метод может быть применен к любому изображению и легко интегрируется в существующие рендеринговые пайплайны.

Недостатки суперсемплинга:

- высокие вычислительные затраты: рендеринг изображения с высоким разрешением требует больше вычислительных ресурсов и времени;
- память: увеличение разрешения изображений требует больше памяти для их хранения и обработки.

Существует несколько вариаций суперсемплинга, направленных на оптимизацию и улучшение процесса:

- SSAA (Super-Sampling Anti-Aliasing). Классический метод суперсемплинга, где изображение рендерится с высоким разрешением, затем фильтруется и уменьшается до нужного размера;

- MSAA (Multi-Sample Anti-Aliasing). Классический метод суперсемплинга, где изображение рендерится с высоким разрешением, затем фильтруется и уменьшается до нужного размера;
- FXAA (Fast Approximate Anti-Aliasing). Классический метод суперсемплинга, где изображение рендерится с высоким разрешением, затем фильтруется и уменьшается до нужного размера;
- TAA (Temporal Anti-Aliasing). Классический метод суперсемплинга, где изображение рендерится с высоким разрешением, затем фильтруется и уменьшается до нужного размера.

Применение суперсемплинга;

- компьютерные игры: для улучшения графики и уменьшения артефактов на краях объектов;
- фотореалистичная визуализация: в архитектурной визуализации, анимации и спецэффектах для создания высококачественных изображений;
- компьютерное зрение: для повышения точности анализа и распознавания объектов на изображениях.

Суперсемплинг – мощный инструмент в арсенале разработчиков графики и компьютерного зрения, который позволяет значительно улучшить качество изображений за счет увеличения разрешения и последующего уменьшения.

Пиксельный стиль графики – это художественный подход, основанный на использовании пикселей в качестве основных строительных блоков изображения. В отличие от гладких и векторных графических стилей, пиксельный стиль представляет изображение как сетку пикселей, где каждый пиксель имеет определенный цвет и является независимым от соседних пикселей (рисунок 9).

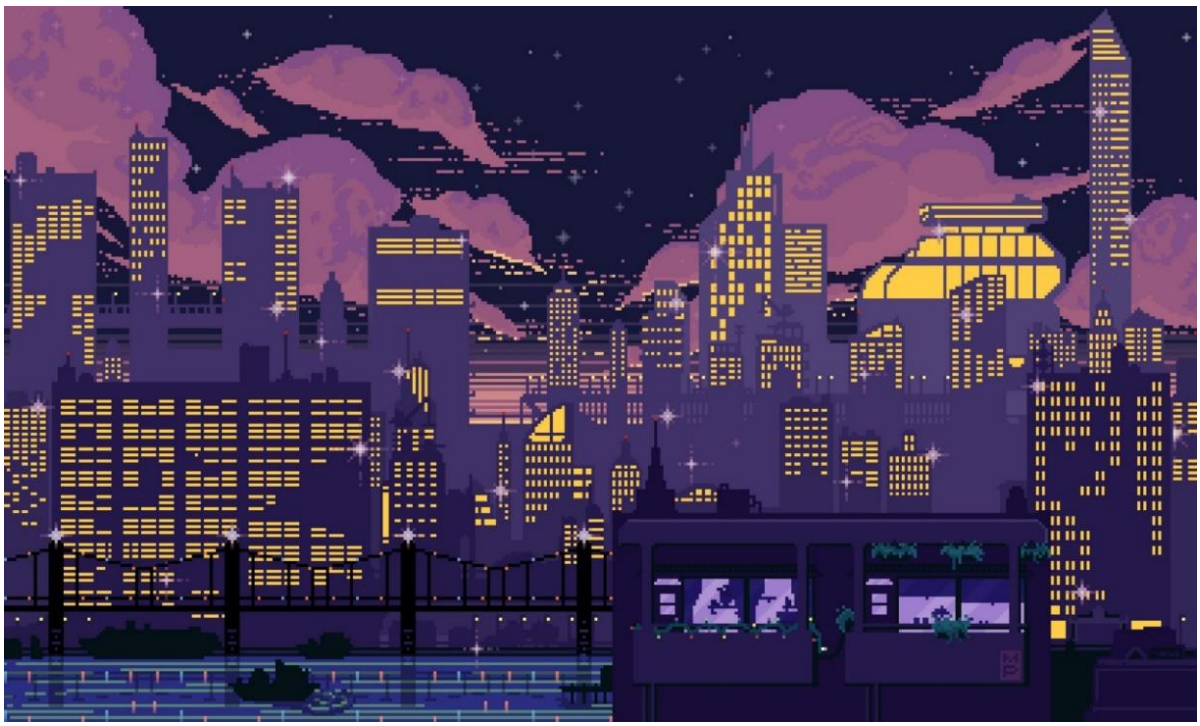

Рисунок 9 – Изображение выполненное в пиксельном стиле

Исторически пиксельный стиль был широко распространен в компьютерных играх и ранних компьютерных графических приложениях, где графика была ограничена разрешением экрана и способностями графических процессоров. Этот стиль превратился в культурное явление и стал иконой ретро-гейминга.

Пиксельный стиль также часто используется в современных видеоиграх, анимации, искусстве пикселей и других цифровых медиа. Он может быть применен для создания абстрактных или реалистичных изображений, и часто включает в себя ограниченную палитру цветов, острые края и детализацию на уровне отдельных пикселей.

Создание пиксельной графики может осуществляться с помощью специализированных редакторов пикселей, таких как Aseprite (рисунок 10) или GraphicsGale, или с использованием графических программ, которые поддерживают инструменты и фильтры для работы с пикселями.

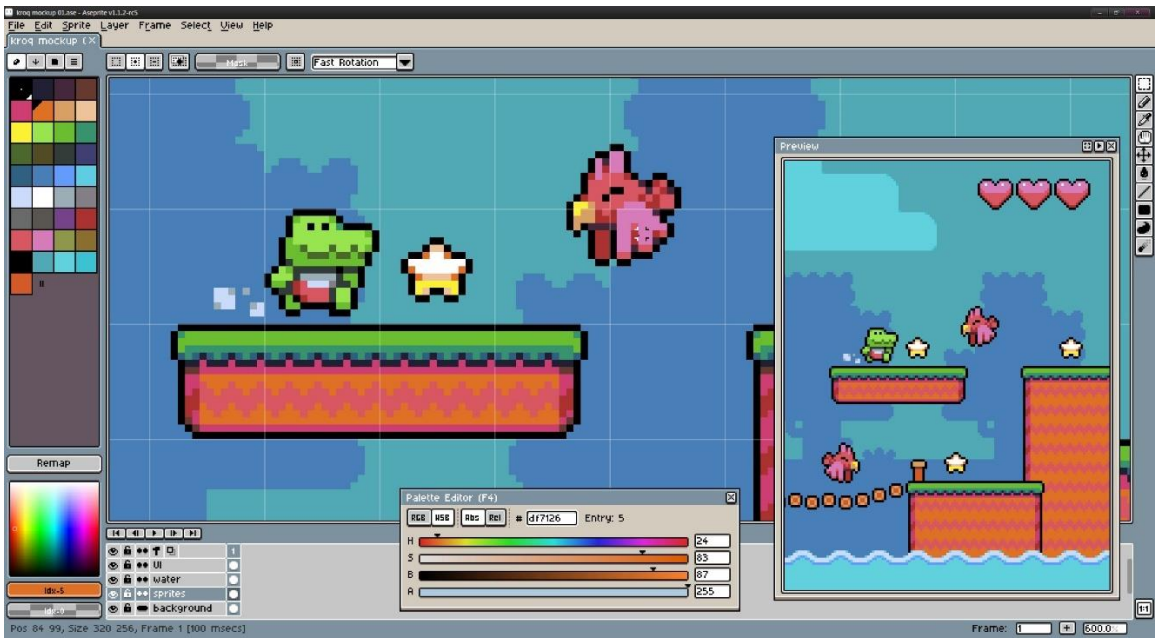

Рисунок 10 – Aseprite и созданные в нем спрайты

Пиксельный стиль графики часто ассоциируется с ностальгией, ретро-стилем и визуальной эстетикой, и его применение может придавать работам уникальный и характерный вид.

Спрайты в 2D играх представляют собой графические объекты или изображения, которые используются для представления персонажей, объектов, фонов и других элементов игрового мира. Они состоят из пикселей и могут иметь различные форматы файлов, такие как PNG или GIF.

Создание спрайтов включает несколько шагов:

- разработка концепции: сначала важно определить, какие объекты и персонажи вам понадобятся для вашей игры. Создание скетчей или концептуальных рисунков поможет вам определить общий внешний вид и стиль игрового мира;

- рисование: после определения концепции можно начать создавать спрайты. Один из самых распространенных способов создания спрайтов – это использование графических редакторов, таких как Adobe Photoshop или GIMP. Вы можете нарисовать спрайты вручную, используя инструменты редактора, или использовать комбинацию рисования и редактирования существующих изображений или текстур;

- анимация: если вам нужны анимированные спрайты, то каждый спрайт должен иметь несколько кадров анимации. Вы можете создать последовательность изображений, отображающих разные позиции или состояния объекта во времени (рисунок 11). Эти кадры затем переключаются в игре, чтобы создать впечатление движения;
- оптимизация: после создания спрайтов важно оптимизировать их для использования в игре. Это может включать уменьшение размера файлов спрайтов, сжатие изображений без значительной потери качества и установку оптимальных параметров цвета и прозрачности;
- импорт в игровой движок: после завершения создания спрайтов они должны быть импортированы в ваш игровой движок или среду разработки игр. Это обычно делается путем загрузки спрайтов в библиотеку ресурсов и использования соответствующих функций или скриптов для отображения и анимации спрайтов в игре.

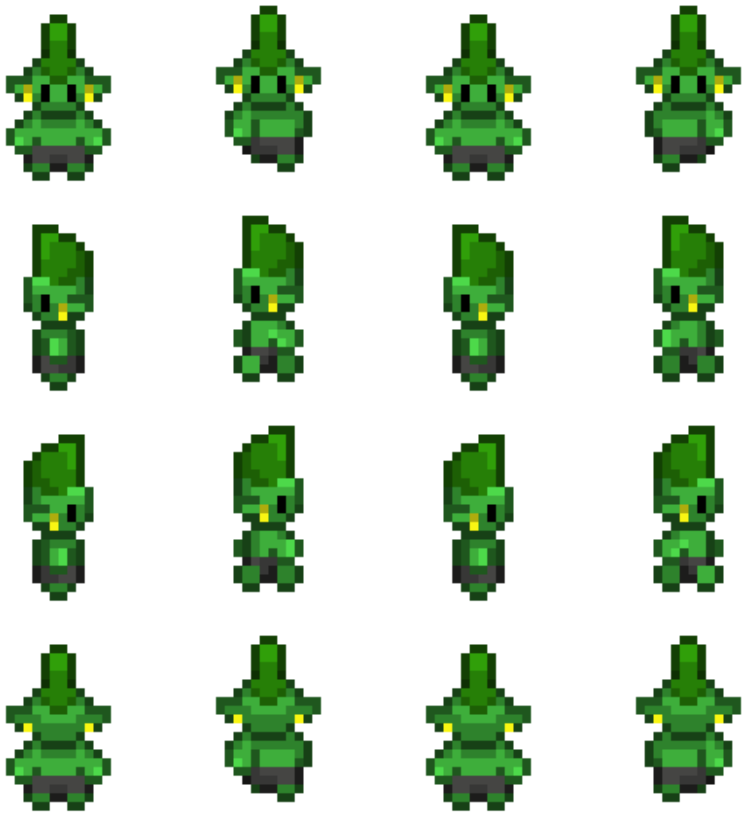

Рисунок 11 – Пример игрового спрайта с кадрами анимации

Процесс создания спрайтов может варьироваться в зависимости от предпочтений и инструментов разработчика, а также от требований игры. Он может включать более сложные шаги, такие как риггинг (создание скелетной анимации) или использование спрайтовых атласов (объединение нескольких спрайтов в один изображение для повышения производительности). Однако, описанные выше шаги являются общим руководством по созданию спрайтов для 2D игр.

1.2 Обзор существующих методов решения аналогичных задач

Существует множество решений по созданию игровых спрайтов, которые могут использоваться для создания персонажей и объектов в играх. Некоторые из них:

- Spriter – это бесплатный редактор спрайтов для игр, который позволяет создавать и редактировать персонажей, объекты и фоны. Он имеет простой интерфейс и поддерживает различные форматы файлов, такие как PNG, GIF и SVG;
- Piskel – это онлайн редактор спрайтов, который позволяет рисовать персонажей и объекты в пиксельном стиле. Он имеет множество инструментов и настроек, которые позволяют создавать уникальные спрайты;
- Twine Game Maker – это программа для создания игр на основе текстовых файлов. Она может использоваться для генерации спрайтов с помощью скриптов;
- GameMaker Studio – это профессиональная программа для создания 2D-игр. Она имеет встроенный редактор спрайтов, который может создавать персонажей и объекты;
- Tiled – это бесплатное программное обеспечение для создания карт для игр. Оно может использоваться для создания спрайтов персонажей и объектов.

Однако, все эти редактор имеют свои недостатки. Некоторые из них могут быть сложными в использовании, другие могут иметь ограниченную функциональность или не поддерживать определенные форматы файлов. Кроме

того, многие из них требуют знания программирования или использования специализированных инструментов. Но основной их недостаток в сравнении с предлагаемым генератором, необходимость создавать все спрайты вручную, отрисовкой или скриптом.

Также существуют генераторы изображений, основанных на нейронных сетях. Вот несколько из них:

- Deep Dream: разработанный компанией Google, этот генератор использует сверточные нейронные сети для создания психоделических и уникальных изображений, перерабатывая входные данные и добавляя художественные элементы;
- StyleGAN: этот генератор известен своей способностью создавать фотореалистичные изображения лиц людей, а также высококачественные изображения различных объектов, используя методы генеративно-состязательных сетей (GAN);
- DALL-E: разработанный OpenAI, DALL·E представляет собой модель, способную генерировать уникальные изображения на основе текстового описания, что позволяет пользователям создавать изображения на основе словесных описаний (рисунок 12);
- BigGAN: этот генератор обеспечивает возможность генерации высокоразрешенных и реалистичных изображений различных классов объектов, что делает его популярным инструментом в области компьютерного зрения и генерации изображений.

Но основой недостаток представленных генераторов, заключается в том, что материал обучения не является игровыми спрайтами. Из-за чего попытка сгенерировать игровое изображении создаёт много графических артефактов и лишние детали фона.

Рисунок 12 – Изображение созданное DALL-E

Проблема сгенерированного изображение заключается в несоблюдении единой пиксельной сетки, из-за чего появляются субпиксели.

2 АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ

2.1 Предлагаемый алгоритм компьютеризированного решения задачи

Авторское решение заключается в создании нейронной сети, обученной на отобранном массиве изображений персонажей в пиксельном стиле. Генерация спрайта происходит в два этапа: генерация начального силуэта на основе шума, покраска силуэта, полученного не предыдущем шаге (рисунок 13).

Рисунок 13 – Представление генерации в два этапа

Генерация изображения в два этапа подразумевает создание изображения сначала в виде набора параметров или признаков, а затем преобразование этих параметров в конкретное изображение. Этот подход обладает несколькими преимуществами:

- контроль над содержанием и стилем: позволяет изменять содержание изображения (например, объекты на изображении) отдельно от его стиля (например, цветовой палитры или текстуры);

- эффективное управление: позволяет более точно контролировать процесс генерации изображения, поскольку можно работать с набором параметров или признаков, а не непосредственно с пикселями изображения;
- улучшенная модификация: облегчает модификацию изображений, так как изменение параметров или признаков первоначального представления может привести к существенным изменениям в конечном изображении.

Этот метод широко используется в различных алгоритмах генерации и модификации изображений, таких как StyleGAN, где сперва генерируются параметры изображения, а затем они трансформируются в окончательное изображение (рисунок 14).

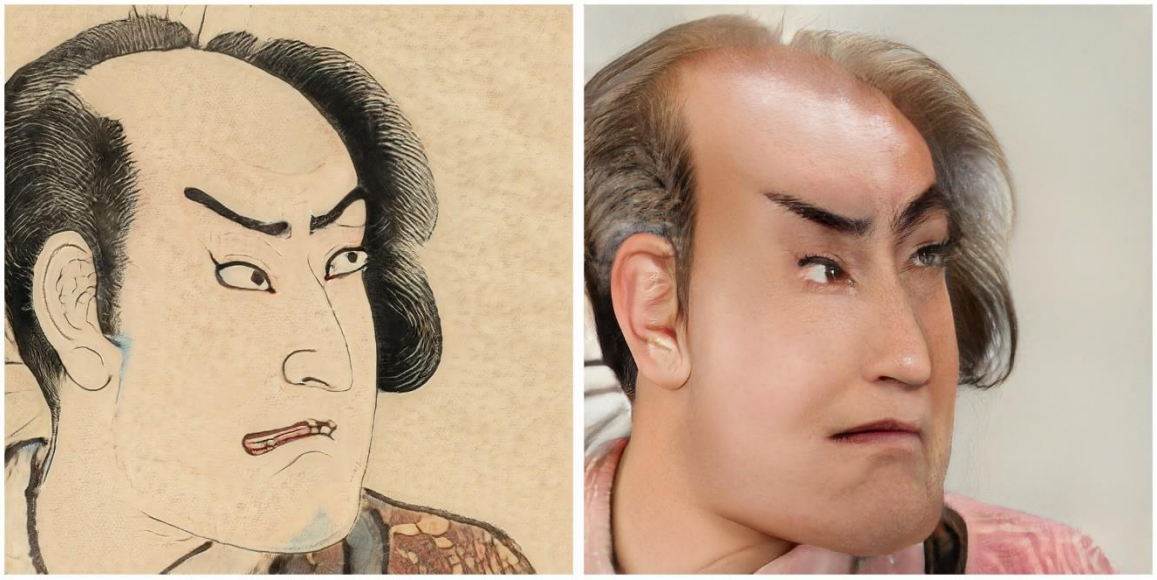

Рисунок 14 – Пример работы StyleGAN

Возможность загрузки пользователем основы изображения предоставляет несколько преимуществ и возможностей:

- персонализация: пользователь может загрузить собственное изображение в качестве основы, что позволяет персонализировать процесс обработки или генерации изображений под свои потребности;
- улучшенный контроль: при загрузке основы изображения пользователь может более точно контролировать результат, так как базовое изображение служит отправной точкой для любых операций или модификаций;

- творческие возможности: это открывает новые творческие возможности для пользователей, позволяя им создавать уникальные композиции и обрабатывать изображения в соответствии с собственным видением;
- интеграция с другими инструментами: загруженное пользователем изображение может быть использовано в сочетании с другими инструментами или алгоритмами для достижения конкретных целей, таких как редактирование, улучшение качества или анализ изображения.

Таким образом, возможность загрузки пользователем основы изображения значительно расширяет функциональные возможности приложений или инструментов для работы с изображениями, делая их более гибкими и удобными для конечных пользователей.

Генерация изображений на основе шума – это процесс создания изображений, используя случайный шум как основу. Этот подход часто применяется в компьютерной графике и генеративных моделях для создания текстур, фонов, абстрактных паттернов и других визуальных эффектов.

Одним из наиболее распространенных методов генерации изображений на основе шума является использование шумовых функций, таких как шум Перлина или шум Симплекса. Эти функции генерируют случайные значения для каждой точки изображения на основе их координат (рисунок 15). Результатом является черно-белое или цветное изображение, где каждый пиксель представляет собой случайное значение.

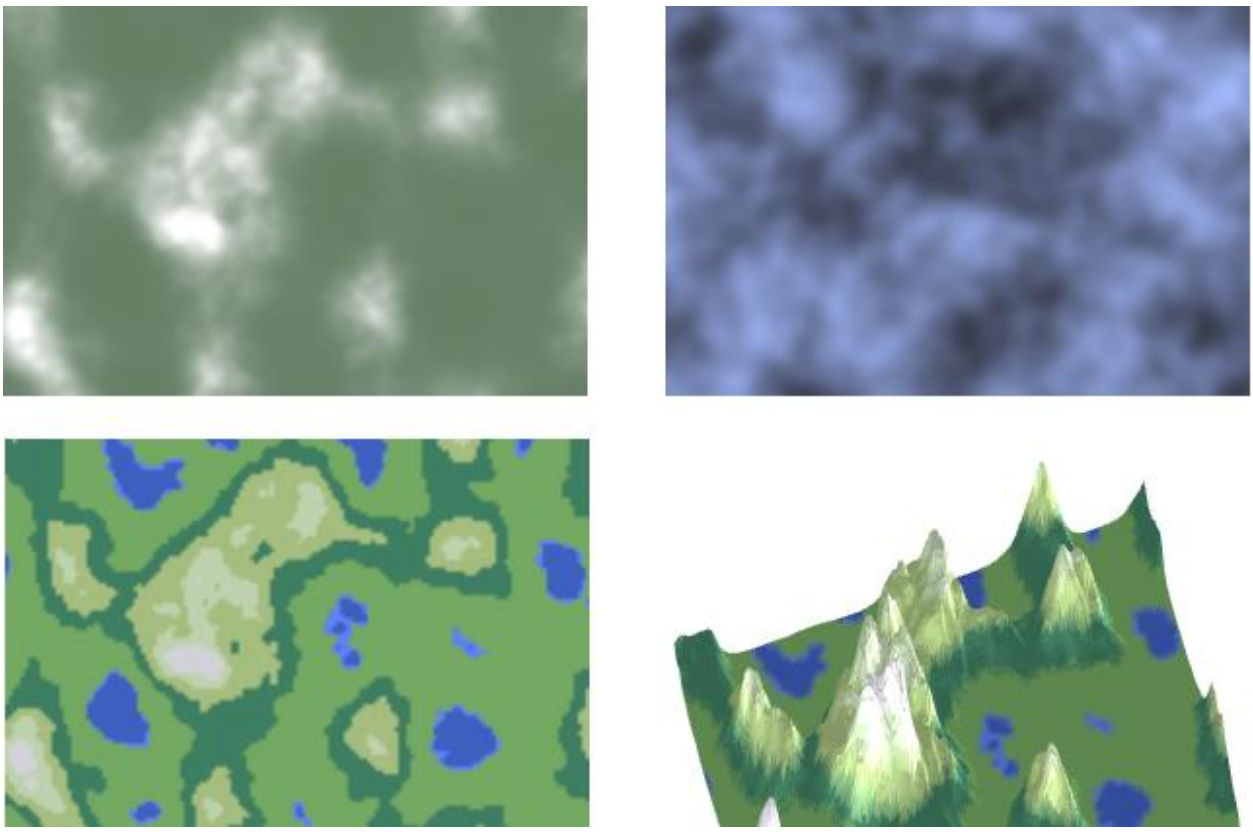

Рисунок 15 – Генерация ландшафта на основе шума

Однако, для получения более интересных и разнообразных изображений, можно применять дополнительные операции и преобразования:

- фильтрация: использование различных фильтров и ядер свертки для изменения характеристик шума. Например, можно применить размытие, усиление резкости или эффекты искажения, чтобы изменить текстуру и структуру шума;
- манипуляция параметрами: изменение параметров шумовой функции, таких как масштаб, амплитуда или частота, для создания разных вариаций шума и текстур. Это позволяет получить разнообразные результаты от одной и той же шумовой функции;
- преобразование цвета: применение различных цветовых схем и градиентов к шуму, чтобы получить цветные изображения. Например, можно сопоставить значения шума с определенными цветами или использовать градиенты для плавного перехода между цветами;

- комбинирование шумов: сочетание нескольких слоев или разных типов шумов для создания более сложных и интересных текстур. Например, можно объединить шум Перлина с шумом Симплекса или добавить дополнительные слои шума с различными параметрами;

- использование генеративных моделей: применение глубоких нейронных сетей и генеративных моделей, таких как генеративные состязательные сети (GAN) или автокодировщики, для генерации изображений на основе шума. Эти модели обучаются на большом наборе данных и способны генерировать реалистичные изображения с высоким уровнем детализации.

Генерация изображений на основе шума предоставляет мощный инструмент для создания разнообразных и уникальных визуальных эффектов в компьютерной графике и игровой разработке.

Сверточные нейронные сети (Convolutional Neural Networks, CNNs) являются типом глубоких нейронных сетей, которые широко используются в обработке изображений и анализе видео. Они получили свое название из-за использования операции свертки в своей основе (рисунок 16).

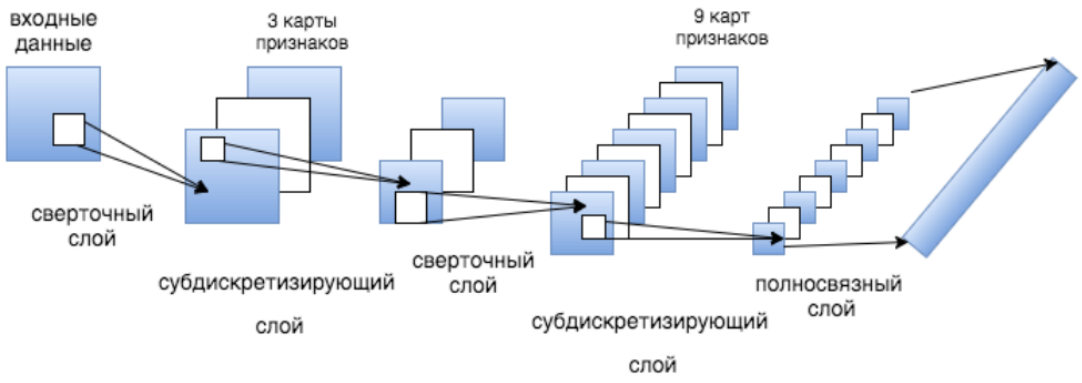

Рисунок 16 – Визуализация сверточной нейронной сети

Основная идея сверточных нейронных сетей заключается в том, чтобы использовать фильтры, называемые ядрами свертки или фильтрами свертки, для извлечения локальных признаков из входных данных. Эти фильтры применяются к различным областям входного изображения, чтобы найти

различные шаблоны и структуры. Затем результаты свертки объединяются с помощью операции подвыборки, чтобы уменьшить размерность данных и уловить более абстрактные признаки.

Сверточные нейроны состоят из нескольких слоев. Типичная архитектура сверточной нейронной сети включает в себя один или несколько сверточных слоев, слои подвыборки для уменьшения размерности, полносвязные слои для классификации и, возможно, слои активации, такие как ReLU (Rectified Linear Unit), для добавления нелинейности (рисунок 17).

Сверточные нейронные сети обладают рядом преимуществ, которые делают их эффективными для обработки изображений. Они автоматически извлекают важные признаки изображения, что позволяет им быть устойчивыми к небольшим изменениям входных данных, таким как сдвиг, масштабирование или искажение. Кроме того, сверточные нейронные сети могут автоматически изучать характеристики на разных уровнях абстракции, начиная с простых геометрических форм и текстур и заканчивая более сложными объектами и концепциями.

В целом, сверточные нейронные сети являются мощным инструментом для анализа и обработки изображений и широко применяются в таких задачах, как классификация изображений, обнаружение объектов, семантическая сегментация и многие другие.

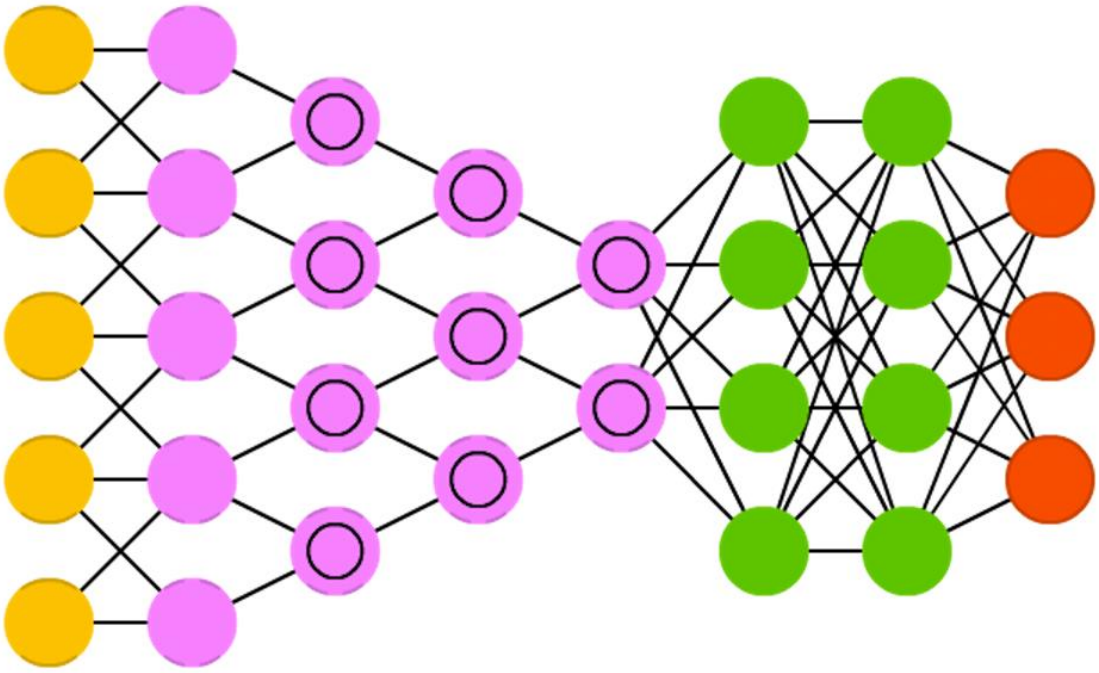

Рисунок 17 – Пример сверточной нейронной сети

Обучение нейронных сетей на наборе изображений, таком как набор картинок, является распространенным применением сверточных нейронных сетей. Вот общий процесс обучения нейронной сети на наборе картинок:

- подготовка данных: изображения из набора картинок должны быть предварительно обработаны и приведены к одному формату. Это может включать изменение размера изображений, нормализацию значений пикселей или применение других методов предварительной обработки;
- создание архитектуры нейронной сети: необходимо определить архитектуру сверточной нейронной сети. Это включает определение числа сверточных слоев, слоев подвыборки, полносвязных слоев и функций активации. Архитектура может быть настроена и оптимизирована для конкретной задачи классификации, обнаружения объектов и т.д.;
- определение функции потерь: функция потерь определяет, насколько хорошо модель предсказывает правильные метки для изображений. В задаче классификации обычно используется категориальная кросс-энтропия в качестве функции потерь;

- оптимизация параметров: для обучения нейронной сети необходимо определить оптимальные значения параметров модели. Это делается с помощью оптимизационных методов, таких как стохастический градиентный спуск (SGD) или его вариации. Оптимизационный метод минимизирует функцию потерь, обновляя веса и смещения нейронной сети на каждой итерации;

- разделение данных на обучающую, проверочную и тестовую выборки: набор картинок обычно разделяется на обучающую, проверочную и тестовую выборки. Обучающая выборка используется для обучения модели, проверочная выборка используется для настройки гиперпараметров и оценки производительности модели, а тестовая выборка используется для окончательной оценки производительности модели;

- обучение и оценка модели: нейронная сеть обучается на обучающей выборке путем прямого распространения (forward propagation) и обратного распространения ошибки (backpropagation). После каждой эпохи обучения производится оценка модели на проверочной выборке для настройки гиперпараметров и предотвращения переобучения. Когда модель достигает достаточно хорошей производительности, она может быть протестирована на тестовой выборке для окончательной оценки ее способности к обобщению;

- оптимизация и настройка: процесс обучения может потребовать множество экспериментов с различными гиперпараметрами, архитектурами и методами оптимизации для достижения наилучшей производительности модели.

Это общий процесс обучения нейронных сетей на наборе картинок. Он может быть дополнен другими техниками, такими как аугментация данных, регуляризация и т.д., чтобы улучшить производительность модели.

2.2 Обзор возможностей профильного программного обеспечения

Для начала необходимо выбрать язык программирования. Для написания нейронных сетей часто используют следующие языки программирования и фреймворки:

- Python: язык Python широко используется в машинном обучении благодаря богатым библиотекам, таким как TensorFlow, Keras, PyTorch и другим;

- R: R также популярен в области статистики и машинного обучения. Он имеет множество библиотек, специализированных на анализе данных и построении моделей;
- C++ и Java: некоторые большие вычислительные системы, особенно в области обработки изображений и видео, могут использовать C++ или Java для реализации быстрых вычислений нейронных сетей;
- MATLAB: MATLAB предоставляет множество инструментов для работы с нейронными сетями и обработки сигналов, что делает его популярным выбором для исследований в этой области;
- Julia: Julia – относительно новый язык программирования, который становится все более популярным в машинном обучении благодаря своей производительности и удобству использования;
- JavaScript: для разработки веб-приложений с использованием нейронных сетей можно использовать фреймворки, такие как TensorFlow.js, позволяющие выполнять обучение и прогнозирование непосредственно в браузере.

Также необходимо выбрать среду разработки. Для разработки нейронных сетей часто используют различные среды разработки (IDE) и инструменты:

- Jupyter Notebook: интерактивная среда, позволяющая вам создавать и выполнять код Python (включая код для нейронных сетей) в удобном интерфейсе;
- PyCharm: мощная интегрированная среда разработки для Python, которая поддерживает создание и отладку кода для нейронных сетей;
- Google Colab: онлайн-сервис, предоставляемый Google, который позволяет писать и выполнять код Python, включая обучение нейронных сетей, прямо в облаке;
- Visual Studio Code: легкий и гибкий редактор, расширяемый для работы с множеством языков программирования, включая Python и JavaScript для нейронных сетей;

- MATLAB: для тех, кто предпочитает использовать MATLAB для работы с нейронными сетями, сама среда MATLAB предоставляет мощные инструменты для этой цели.

- Некоторые из популярных библиотек и фреймворков для написания нейронных сетей:

- TensorFlow: разработанный Google, TensorFlow предоставляет гибкие инструменты для построения и обучения различных видов нейронных сетей;

- Keras: высокоуровневый API, работающий поверх TensorFlow или Theano. Keras упрощает создание нейронных сетей и является популярным выбором для начинающих и опытных разработчиков;

- PyTorch: открытый исходный код, разработанный командой Facebook, PyTorch обеспечивает гибкость и производительность при создании нейронных сетей, а также часто используется для исследований в области глубокого обучения;

- Caffe: фреймворк, специализирующийся на скорости и моделях сверточных нейронных сетей, широко применяется в обработке изображений.

2.3 Характеристика выбранного программно-технического обеспечения

Язык программирования Python популярен в области разработки нейронных сетей по нескольким причинам:

- простота использования: Python известен своей лаконичностью и читаемостью, что делает его привлекательным для разработчиков всех уровней опыта. Это особенно важно при работе с сложными алгоритмами машинного обучения, такими как нейронные сети;

- богатые библиотеки: Python имеет обширную экосистему библиотек для машинного обучения и глубокого обучения, такие как TensorFlow, Keras, PyTorch, scikit-learn и многие другие, которые значительно упрощают создание и обучение нейронных сетей;

- гибкость: Python является очень гибким языком, который поддерживает различные стили программирования. Это позволяет разработчикам легко экспериментировать с различными моделями и их конфигурациями;
- сообщество и ресурсы: Python имеет огромное активное сообщество разработчиков, готовых делиться знаниями, обсуждать проблемы и предоставлять полезные библиотеки и инструменты;
- интеграция: Python хорошо интегрируется с другими технологиями, такими как базы данных, веб-фреймворки и инструменты для визуализации данных, что делает его привлекательным выбором для полного цикла разработки приложений на основе нейронных сетей.

Эти преимущества делают Python наиболее приоритетным при выборе для разработки нейронных сетей.

Jupyter Notebook предоставляет несколько преимуществ для разработки нейронных сетей:

- интерактивная среда: Jupyter Notebook позволяет вам выполнять код поэтапно, что особенно полезно при обучении и отладке нейронных сетей, поскольку вы можете наблюдать результаты каждого небольшого участка кода;
- визуализация данных: вы можете легко встраивать графику и другие формы визуализации прямо в блокнот, что помогает визуализировать результаты обучения нейронных сетей и изучать поведение моделей;
- удобство документирования: Jupyter Notebook позволяет вам вставлять текст, изображения и математические формулы вместе с кодом, что полезно для создания наглядной и понятной документации к вашим нейронным сетям, исследованиям и экспериментам;
- широкая поддержка языков: несмотря на то, что изначально разработан для языка Python, Jupyter Notebook поддерживает множество других языков программирования, так что вы можете использовать его для работы с различными типами нейронных сетей и алгоритмов машинного обучения;

- общий доступ к знаниям: Jupyter Notebook позволяет делиться своим кодом, данными и выводом с коллегами и сообществом, что способствует коллективной работе и обмену идеями в области разработки нейронных сетей.

TensorFlow – это популярная открытая библиотека машинного обучения, разработанная компанией Google. Она предоставляет инструменты и ресурсы для создания и обучения различных моделей машинного обучения, включая нейронные сети. TensorFlow широко используется как исследователями, так и разработчиками в различных областях, включая компьютерное зрение, естественный язык, рекомендательные системы, обработку звука и другие.

Основные особенности TensorFlow:

- графовое представление: TensorFlow использует графовое представление для описания вычислительных операций. Граф представляет собой набор узлов (операций) и ребер (поток данных между операциями). Это позволяет TensorFlow оптимизировать выполнение операций и автоматически распараллеливать их на доступных вычислительных ресурсах;
- многоуровневые абстракции: TensorFlow предоставляет набор высокоуровневых абстракций для упрощения разработки моделей машинного обучения. Это включает в себя высокоуровневые API, такие как Keras, который облегчает создание и обучение нейронных сетей с помощью простого и интуитивного интерфейса;
- гибкость: TensorFlow предлагает гибкость в выборе способа создания и обучения моделей. Он поддерживает различные типы моделей, включая сверточные нейронные сети (CNN), рекуррентные нейронные сети (RNN), генеративные модели (GAN) и другие. Кроме того, TensorFlow может быть использован на разных уровнях, от разработки моделей на низком уровне с использованием операций TensorFlow до использования предобученных моделей и API высокого уровня;
- высокая производительность: TensorFlow обладает оптимизациями для эффективного использования вычислительных ресурсов, включая

поддержку распределенного вычисления на нескольких устройствах и многопоточность для параллельного выполнения операций;

- кроссплатформенность: TensorFlow поддерживает различные платформы и устройства, включая CPU, GPU и TPU (Tensor Processing Unit). Он также предоставляет возможность экспортировать модели в форматах, которые могут быть использованы на мобильных устройствах и в веб-приложениях.

TensorFlow имеет обширную документацию, богатое сообщество пользователей и множество доступных ресурсов и инструментов, делая его одной из наиболее популярных библиотек машинного обучения.

Pillow – это популярная библиотека Python, которая предоставляет удобные инструменты для обработки и манипулирования изображениями. Она является форком библиотеки Python Imaging Library (PIL) и предлагает улучшенный интерфейс и дополнительные возможности.

Основные возможности библиотеки Pillow включают:

- открытие и сохранение изображений: Pillow поддерживает множество форматов файлов изображений, таких как JPEG, PNG, GIF, BMP и другие. Она позволяет открывать изображения из файловых источников, а также сохранять измененные изображения в файлы;

- манипуляции с изображениями: Pillow предоставляет различные операции для изменения размера, обрезки, поворота, изменения цветового пространства и других манипуляций с изображениями. Вы можете изменять яркость, контраст, насыщенность, применять фильтры и эффекты, а также выполнять сложные преобразования изображений;

- работа с пикселями: Pillow позволяет получить доступ к пикселям изображения и производить операции с ними. Вы можете изменять значения пикселей, применять фильтры, устанавливать и получать значения цветовых каналов и выполнять другие операции на уровне пикселей;

- работа с изображениями в различных режимах цветности: Pillow поддерживает работу с изображениями в различных режимах цветности, включая одноцветные (черно-белые), градационные, палитровые и

полноцветные изображения. Она предоставляет инструменты для конвертации изображений между различными режимами цветности;

- создание изображений и рисование: Pillow позволяет создавать новые изображения и рисовать на них с помощью различных инструментов, таких как линии, прямоугольники, круги, текст и другие графические элементы. Вы можете управлять цветами, размерами, положением и другими свойствами элементов рисунка.

Pillow имеет простой и интуитивно понятный интерфейс, хорошую производительность и широкий набор функций для работы с изображениями в Python.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

3.1 Основные этапы практической разработки программного продукта

Создание нейронной сети для генерации изображений – это многоступенчатый процесс, который требует глубоких знаний в области машинного обучения, обработки изображений и архитектур нейронных сетей.

Первый этап, представляет собой сбор данных (рисунок 18). Необходимо собрать массив данных, состоящий из изображений, которые будут служить "примерами" для нейронной сети. Качество и разнообразие данных напрямую влияют на качество генерируемых изображений. Так же нужно убедиться, что изображения соответствуют желаемой области применения сети.

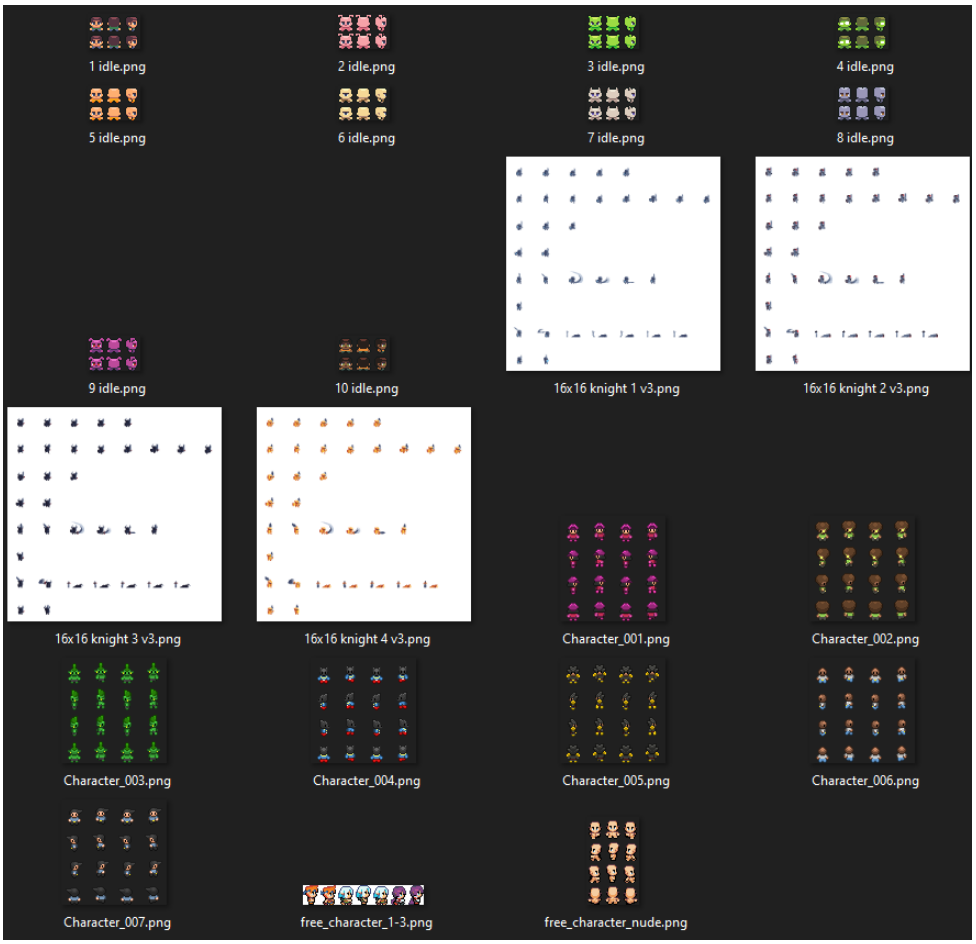

Рисунок 18 – Собранные данные для обучения

Второй этап – подготовка данных или же препроцессинг. Цель препроцессинга – привести данные в определённый формат, нормализовать их и обработать для улучшения производительности и сходимости модели. Препроцессинг может включать в себя масштабирование, нормализацию, центрирование, аугментацию данных и удаление выбросов. В данном случае все изображение были приведены к размеру 26x26 пикселей, а также убраны фон и лишние кадры (рисунок 19).

Рисунок 19 – Обработанные данные для обучения

Третий этап – это выбор архитектуры нейронной сети. Существует множество архитектур нейронных сетей, предназначенных для генерации изображений.

Популярные варианты:

- Generative Adversarial Networks (GANs). Используют две конкурирующие сети: генератор и дискриминатор. Генератор создает изображения, а дискриминатор пытается отличить их от реальных. Состязательный процесс приводит к тому, что генератор учится создавать все более реалистичные изображения (рисунок 20);

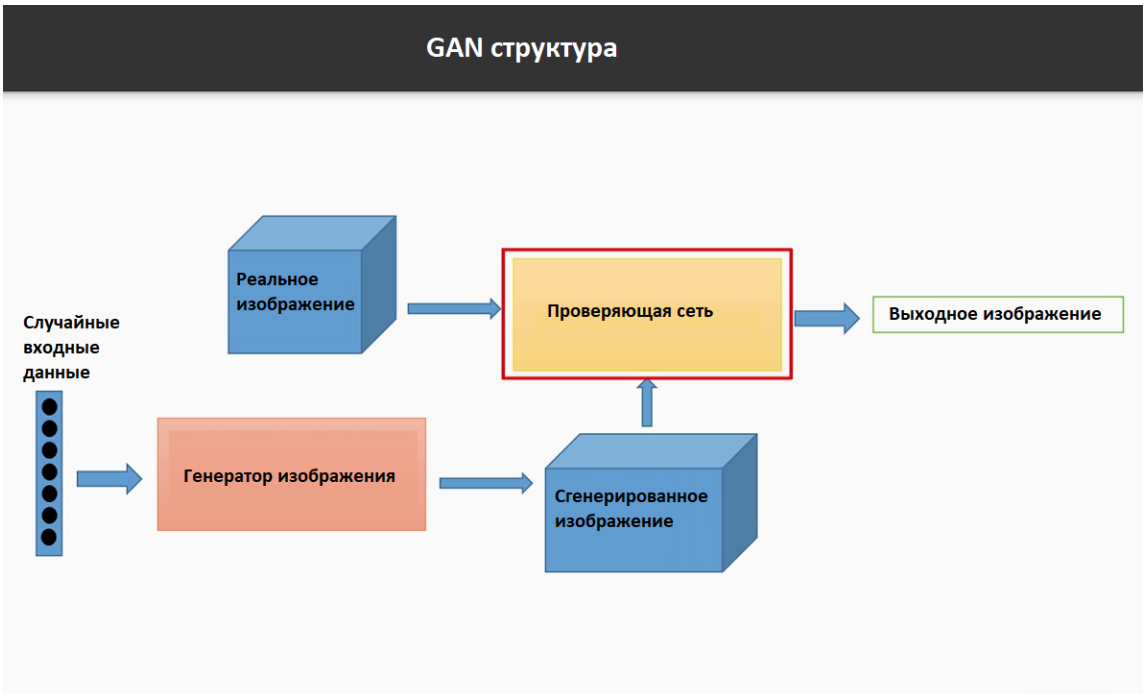

Рисунок 20 – Визуализация работы GAN архитектуры

- Variational Autoencoders (VAEs). Представляют собой нейронную сеть, которая обучается сжимать изображения в латентное представление, а затем воссоздавать их из этого представления. Позволяют генерировать новые изображения, интерполируя между существующими данными в латентном пространстве.

Выбор архитектуры зависит от конкретных целей и задач, доступных вычислительных ресурсов и желаемого качества изображения. В данном случае конкурирующий вариант будет рациональным, так как размер генерируемого изображения не превышает 676 пикселей. Что позволяет использовать небольшое количество ресурсов для генерации, а также добиться более разнообразных вариантов в отличии от интерполяции.

Четвертый этап. Обучение нейронной сети. Требует значительных вычислительных мощностей, особенно для больших наборов данных и сложных архитектур (рисунок 21).

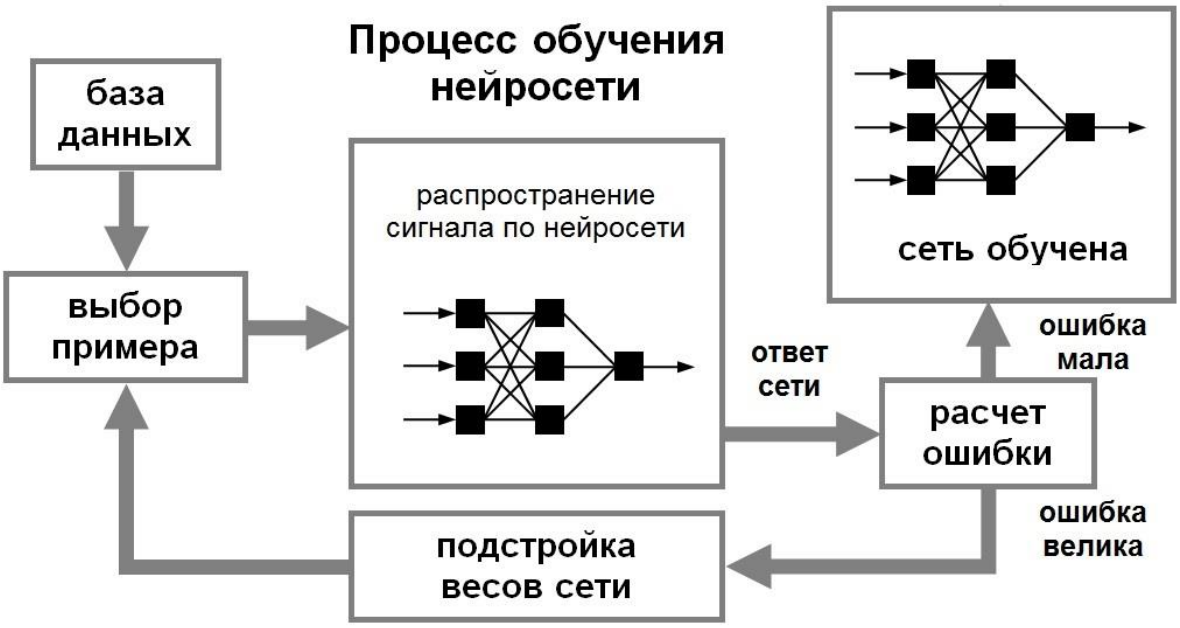

Рисунок 21 – Схема процесса обучения

Процесс обучения включает в себя:

- оптимизацию параметров сети с использованием алгоритмов обратного распространения ошибки;
- мониторинг процесса обучения и внесение корректировок в случае необходимости;
- может потребоваться многократная попытка с различными гиперпараметрами и архитектурами, прежде чем будет достигнута желаемая производительность.

Оценка и доработка являются следующим шагом, к которой можно повторять многократно для совершенствования нейронной сети, расширение её функционала и уменьшения ошибок. Необходимо провести анализ генерируемых изображений на предмет артефактов, ошибок и соответствия желаемому качеству.

Возможные доработка сети:

- настройка гиперпараметров;
- изменение архитектуры;
- дополнение набора данных;
- расширение функционала.

Пятый этап - развертывание и использование. Интеграция обученной сети в приложение или веб-сервис для генерации изображений по запросу (рисунок 22). Необходимо обеспечить доступность вычислительных ресурсов для работы сети. Этого можно достичь запуском программы на удалённом сервере или заранее прописать системные требования необходимые для работы программы на компьютере пользователя. Мониторинг производительности и внесение изменений по мере необходимости, что позволит оптимизировать работу программы и повысить производительность.

Рисунок 22 – Интерфейс программы

3.2 Примеры фактического тестирования программного продукта

Результатами тестирования нейронной сети, генерирующей изображения, включают в себя:

- качество сгенерированных изображений. После прохождения алгоритмом 200–250 итераций можно уже представить, каким по качеству будет

конечный результат. Для генерации хорошего варианта изображения нужно около 1000–1250 итераций;

- время, затраченное на одну итерацию. Среднее время, которое было затрачено на одну итерацию во время тестирований, равняется 12 секундам;
- пути улучшения разработанного решения. Например, повышением количества возможных разрешений входных и выходных изображений. Увеличение массива данных, позволит разнообразить выходные варианты изображений, а также расширить палитру цветов.
- В результате долго тестирования были получены как неудачные результаты, так и хорошие готовые спрайты.

Ошибка в неудачных тестах:

- повторение изображение из массива данных из обучения с добавлением не примыкающих пикселей;
- генерацией пикселей, примыкающих к основному телу лишь по диагонали;
- генерация двух отдельных частей персонажа;
- проблема с покраской персонажа.

Большая часть проблем решалась увеличением числа итераций. А другая часть решалась изменением гиперпараметров.

Далее будет рассмотрено и разобрано несколько примеров генераций, полученных в ходе обучения нейронной сети. А также разобраны ошибки и метод их исправления.

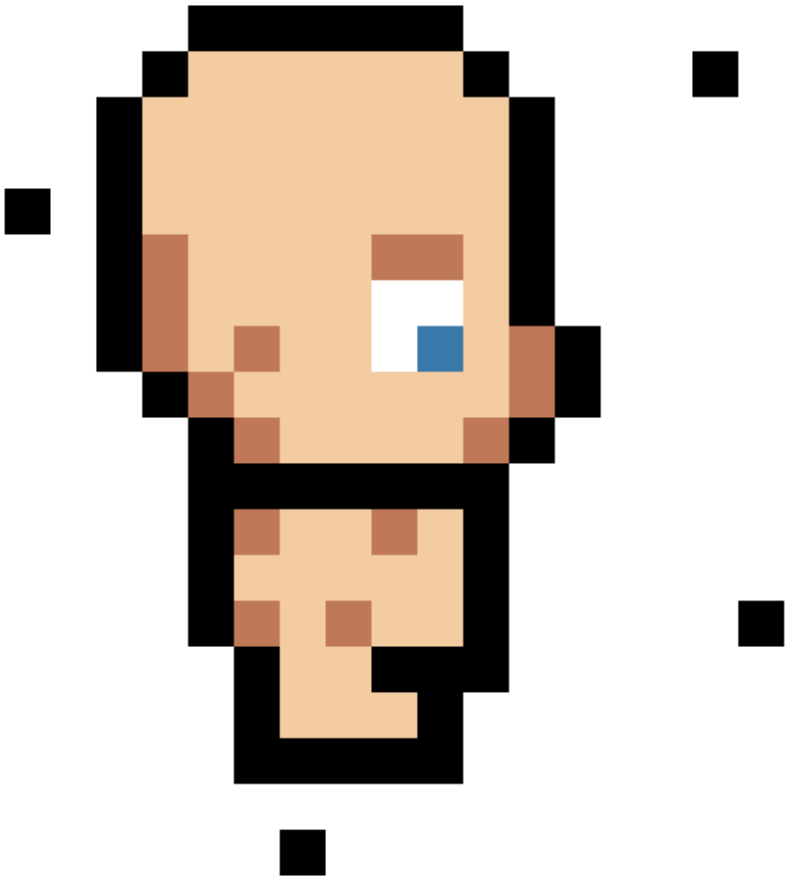

Рисунок 23 – Спрайт, повторяющий входные изображения с не примыкающими пикселями

Проблема заключается в недостаточном обучении (рисунок 23). Нейронная сеть пытается дополнить изображение новыми пикселями, не меняя основное изображение. Также необходимо скорректировать некоторые гиперпараметры. Чтобы нейронная сеть смогла менять цвета, изменять форму основного спрайта, как добавляя новые пиксели, так и убирая старые.

Висящие пиксели в воздухе не являются значащими пикселями, так как не являются частью основного тела. Но даже висящие пиксели сохраняют черный цвет. Или нейронная сеть воспринимает их как контур или обводку, так как это крайние пиксели. Или нейронная сеть не может менять цвета у новых пикселей не присутствующих на обучающем изображении.

Рисунок 24 – Спрайт с диагональными пикселями и проблемой в покраске

Диагональные пиксели не имеют прямого соединения с основной фигурой, что влияет на целостность восприятия спрайта (рисунок 24). Иногда это может использоваться для выделение отдельных частей, например элементов одежды или у гуманоидных существ усиков или антенн на голове. Так же какие-то вещи в руках или, например хвост.

Так как в входных данных все изображения не имели диагональных пикселей или выступающих элементов, значит ошибка заключается в недостаточном обучении нейронной сети.

Рисунок 25 – Спрайт с двумя частями

В данном случае результат нейронной сети близок к удовлетворительному, так как изображение отличается от изображения, на котором обучалась (рисунок 25). Но целостность тела была нарушена, и хоть эти пиксели и находятся в воздухе, это уже не полноценные элементы спрайта, а не просто висящие в воздухе пиксели.

Для исправления данной ошибки необходимо скорректировать гиперпараметры, чтобы на изображении присутствовал лишь одно основное тело. Также необходимо продолжить дальнейшее обучение нейронной сети для получение лучших результатов, как покраски, так и изменения формы спрайтов, на основе которых она обучается.

Рисунок 26 – Спрайт проблемой в покраске

В данном изображение (рисунок 26) присутствуют проблемы покраски, но форма уже близка к удовлетворительному. Для решения этой проблемы необходимо продолжить обучение.

Рисунок 27 – Удачный спрайт

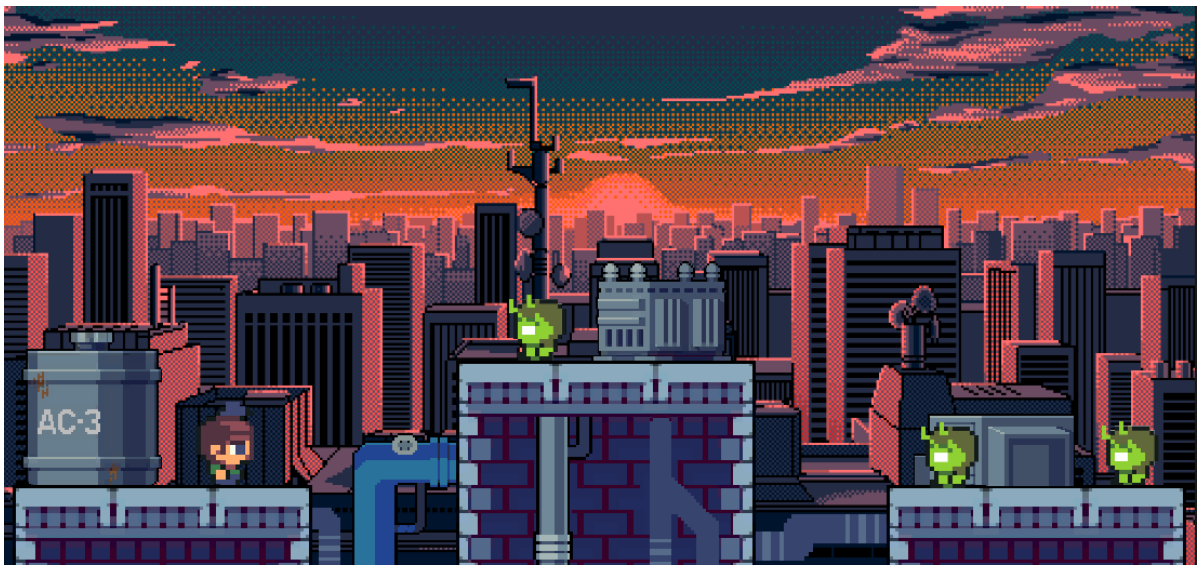

Рисунок 28 – Использование моделей в прототипе игры

Также для, подтверждение возможности использования спрайтов в прототипах, был создан проект на Unity. Он представляет из себя прототип игры в жанре платформер. В прототипе используется несколько спрайтов (рисунок 27), сгенерированных нейронной сетью. Остальное оформление взято из открытых источников (рисунок 28).

ЗАКЛЮЧЕНИЕ

В результате разработки была достигнута поставленная цель, а также выполнены все задачи. Была получена программа, использующая нейронную сеть. Нейронная сеть генерирует изображение в два этапа. В первом этапе она генерирует силуэт игрового спрайта. Во втором, производится покраска текущего силуэта.

Для удобного взаимодействия пользователя с нейронной сетью был разработан пользовательский интерфейс. Были реализованы функции сохранения спрайта и силуэта, возможность загрузки своего силуэта, генерация силуэта и покраска текущего силуэта.

Интерфейс простой и не перегруженный, чтобы не путать пользователя во время работы с программой. Возможна дальнейшая доработка интерфейса, например, замена шрифта или панель, адаптированная под мобильные устройства.

Также учтена ошибка запуска покраски без силуэта. В этом случае принудительно будет запущена генерация, а после запуститься покраска, сгенерированного силуэта. И предусмотрена проверка загружаемого изображения на размер и формат загружаемого изображения.

В дальнейшем нейронную сеть можно адаптировать под веб-приложение или бота в социальных сетях и мессенджерах. Сами спрайты могут применяться при прототипировании и разработки игр, а также анимации, создаваемой на основе пиксельной графики.

Для проверки удобства использование нейронной сети в прототипировании игр, был создан небольшой проект. Для создания прототипа был использован игровой движок Unity. После создания сцены, были добавлены платформы, задний фон, а также элементы окружения. Далее создан персонаж с контроллером управления, позволяющий ходить прыгать и взаимодействовать с противниками. После созданы противники, которые двигаются по заданной траектории.

Спрайты персонажа и противников были сгенерированы с помощью нейронной сети. В спрайт противника были внесены небольшие изменения. Спрайт главного героя перенесён без изменений.

По итогу использование нейронной сети позволило сильно ускорить создание прототипа. Периодически возникают небольшие артефакты, которые можно быстро поправить, или же запустить генерацию повторно.

В дальнейшем, планируется добавить возможность редактировать изображения внутри программы. Также добавить выбор различных размеров сеток для выходного спрайта. Возможность добавлять свои данные для переобучения нейронной под свой стиль. Самое крупное и сложное нововведение, возможность генерировать объекты по текстовому запрос. Для этого необходимо научить нейронную сеть самообучаться на примерах из интернета, а для этого надо научить отбирать нужные примеры и по возможности вырезать нужные изображения, из карты персонажа.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Аксёнова, Н. А. Компьютерная графика: учебно-методическое пособие / Н. А. Аксёнова, А. В. Воруев, О. М. Демиденко. – Гомель: ГГУ имени Ф. Скорины, 2023. – 130 с. – ISBN 978-985-577-917-0. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/329723. – 28.05.2024.

2 Алексейчук, А. С. Введение в нейронные сети: модели, методы и программные средства: учебное пособие / А. С. Алексейчук. – М.: МАИ, 2023. – 105 с. – ISBN 978-5-6049766-0-9. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/383072. – 03.02.2024.

3 Басараб, М. А. Интеллектуальные технологии на основе искусственных нейронных сетей: учебное пособие / М. А. Басараб, Н. С. Коннова. – М.: МГТУ им. Н.Э. Баумана, 2017. – 56 с. – ISBN 978-5-7038-4716-9. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/103496. – 15.03.2024.

4 Барский, А. Б. Введение в нейронные сети: учебное пособие / А. Б. Барский. – 2-е изд. – М.: ИНТУИТ, 2016. – 358 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/100684. – 07.04.2024.

5 Барский, А. Б. Искусственный интеллект и логические нейронные сети: учебное пособие / А. Б. Барский. – СПб.: Интермедия, 2019. – 360 с. – ISBN 978-5-4383-0155-4. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/161343. – 28.02.2024.

6 Барский, А. Б. Логические нейронные сети: учебное пособие / А. Б. Барский. – 2-е изд. – М.: ИНТУИТ, 2016. – 492 с. – ISBN 978-5-94774-646-4. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/100630. – 27.04.2024.

7 Баюк, Д. А. Практическое применение методов кластеризации, классификации и аппроксимации на основе нейронных сетей: монография / Д. А. Баюк, О. А. Баюк, Д. В. Берзин. – М.: Прометей, 2020. – 448 с. – ISBN 978-5-00172-079-9. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/166775. –30.01.2024.

8 Белозерова, Г. И. Нечеткая логика и нейронные сети: учебное пособие: в 2 частях / Г. И. Белозерова, Д. М. Скуднев, З. А. Кононова. – Липецк: Липецкий ГПУ, [б. г.]. – Часть 1 – 2017. – 64 с. – ISBN 978-5-88526-875-2. – Текст: электронный // Лань: электронно-библиотечная система.). – Режим доступа: https://e.lanbook.com/book/111969. – 30.03.2024.

9 Борисова, А. Ю. Компьютерная графика: учебно-методическое пособие / А. Ю. Борисова, М. В. Царева, И. М. Гусакова, О. В. Крылова. – М.: МИСИ – МГСУ, 2020. – 76 с. – ISBN 978-5-7264-2347-0. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/165179. – 28.05.2024.

10 Булаев, М. П. Нейронные сети для адаптивной обработки данных: учебное пособие / М. П. Булаев, А. Н. Кабанов, И. С. Маркова. – Рязань: РГРТУ, 2012. – 64 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/168099. – 06.04.2024.

11 Бусыгина, Н. А. Компьютерная графика: учебно-методическое пособие / Н. А. Бусыгина. – Екатеринбург: УГЛТУ, 2022. – 72 с. – ISBN 978-5-94984-859-3. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/329849. –28.05.2024.

12 Вакуленко, С. А. Практический курс по нейронным сетям: учебное пособие / С. А. Вакуленко, А. А. Жихарева. – СПб.: НИУ ИТМО, 2018. – 71 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/136500. – 30.04.2024.

13 Галушкин, А. И. Нейронные сети: основы теории / А. И. Галушкин. – М.: Горячая линия-Телеком, 2017. – 496 с. – ISBN 978-5-9912-0082-0. – Текст:

электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/111043. – 14.02.2024.

14 Графический редактор. [Электронный ресурс] / Приборостроительный колледж ТОГБПОУ. – Режим доступа: https://psk68.ru/files/metod/uchebnik_Informatika/graf.html. – 06.02.2024.

15 Гудфеллоу, Я. Глубокое обучение [Текст] / Я. Гудфеллоу, И. Бенджио, А. Курвилль. – под ред. Мовчан Д. А. – М.: ДМК Пресс, 2018. – 652 с.

16 Данилов, В. В. Нейронные сети : учебное пособие / В. В. Данилов. – Донецк: ДонНУ, 2020. – 158 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/179953. – 25.03.2024.

17 Данилов, В. В. Проектирование искусственных нейронных сетей: методические указания / В. В. Данилов. – Донецк: ДонНУ, 2020. – 133 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/179954. – 25.03.2024.

18 Доррер, М. Г. Моделирование нейронный сетей на языке Python: Лабораторный практикум для студентов бакалавриата по направлениям подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» всех форм обучения: учебное пособие / М. Г. Доррер, Г. Ш. Шкаберина, А. В. Коробко. – Красноярск: СибГУ им. академика М. Ф. Решетнёва, 2022. – 76 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/330107. – 09.02.2024.

19 Дружинин, А. И. Компьютерная графика: учебное пособие / А. И. Дружинин, В. В. Вихман, Г. В. Трошина. – Новосибирск: НГТУ, 2022. – 76 с. – ISBN 978-5-7782-4706-2. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/306155. – 28.05.2024.

20 Костылев, В. И. Обработка и анализ изображений с помощью обучения нейронных сетей: учебное пособие / В. И. Костылев, Ю. С. Левицкая. – Воронеж: ВГУ, 2019. – 89 с. – Текст: электронный // Лань: электронно-

библиотечная система. – Режим доступа: https://e.lanbook.com/book/405953. – 28.05.2024.

21 Колодников, А.И. Ранние формы компьютерного дизайна: пиксельная графика и растровая система [Текст] / А.И. Колодников // Terra artis. Искусство и дизайн. 2022. № 3. С. 36–41.

22 Кузнецов, В. П. Нейронные сети: практический курс: учебное пособие / В. П. Кузнецов. – Рязань: РГРТУ, 2014. – 72 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/168060. – 12.03.2024.

23 Ламмерс, К. Шейдеры и эффекты в Unity. Книга рецептов [Текст] / К. Ламмерс. – под ред. Симонова В. В. – пер. с англ. Шапочкин Е. А. – М.: ДМК Пресс, 2017. – 306 с.

24 Лекун, Я. Как учится машина: Революция в области нейронных сетей и глубокого обучения / Я. Лекун. – М.: Альпина Паблишер, 2021. – 351 с. – ISBN 978-5-907470-52-5. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/213980. – 27.02.2024.

25 Маркина, Н. В. Основы искусственного интеллекта: практические работы по созданию и обучению искусственных нейронных сетей на языке Python: учебно-методическое пособие / Н. В. Маркина, Э. И. Беленкова, Г. А. Диденко [и др.]. – Челябинск: ЮУГМУ, 2023. – 72 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/379403. – 30.01.2024.

26 Масленкова, Н. А. «Читатель+зритель=?». К вопросу о новых практиках восприятия текста [Текст] / Н. А. Масленкова // Международный журнал исследований культуры. – 2018. – №3, С. 74–80.

27 Мэннинг Д. Unity для разработчика. Мобильные мультиплатформенные игры [Текст]. – Д. Мэннинг, П. Батфилд-Эддисон. – СПб.: Питер, 2018. – 304 с.

28 Назаров, А. В. Компьютерная графика. Практикум: учебное пособие для вузов / А. В. Назаров, О. В. Назарова. – СПб.: Лань, 2024. – 72 с. – ISBN 978-

5-507-48595-6. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/385967. – 28.05.2024.

29 Ньюстром, Р. Шаблоны Игрового Программирования [Текст] / Р. Ньюстром. – СПб.: Питер, 2015. – 72 с.

30 Овчинников, П. Е. Применение искусственных нейронных сетей для обработки сигналов: учебно-методическое пособие / П. Е. Овчинников. – Нижний Новгород: ННГУ им. Н. И. Лобачевского, 2012. – 32 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/153253. – 30.01.2024.

31 Разработка 2D игры на Unity [Электронный ресурс] // Pikabu. – Режим доступа: https://pikabu.ru/story/razrabotka_2d_igryi_na_unity_25_podgotovka_stsenyi_3860968. – 06.02.2024.

32 Ростовцев, В. С. Искусственные нейронные сети / В. С. Ростовцев. – 4-е изд., стер. – СПб.: Лань, 2024. – 216 с. – ISBN 978-5-507-47362-5. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/364517. – 24.02.2024.

33 Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И. Д. Рудинского: учебное пособие / Д. Рутковская, М. Пилиньский, Л. Рутковский. – 2-е изд. – М.: Горячая линия-Телеком, 2013. – 384 с. – ISBN 978-5-9912-0320-3. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/11843. – 05.02.2024

34 Сальникова, В. В. Компьютерная графика: учебное пособие / В. В. Сальникова, Д. В. Третьяков. – СПб.: ПГУПС, 2023. – 67 с. – ISBN 978-5-7641-1810-9. – Текст: электронный // Лань: электронно-библиотечная систем. – Режим доступа: https://e.lanbook.com/book/355091. –28.05.2024.

35 Семериков, А. В. Классификация объектов на основе нейронной сети и методами дерева решения и ближайших соседей: учебное пособие / А. В. Семериков, М. А. Глазырин. – Ухта: УГТУ, 2022. – 68 с. – Текст: электронный //

Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/267857. – 28.03.2024.

36 Сириченко, А. В. Искусственные нейронные сети. Практикум: учебное пособие / А. В. Сириченко. – М.: МИСИС, 2022. – 26 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/305447. – 28.04.2024.

37 Соробин, А. Б. Сверточные нейронные сети: примеры реализаций: учебно-методическое пособие / А. Б. Соробин. – М.: РТУ МИРЭА, 2020. – 159 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/163853. – 28.01.2024.

38 Темкин, И. О. Искусственные нейронные сети в АСУ ТП: учебник / И. О. Темкин, В. Б. Трофимов. – М.: МИСИС, 2023. – 352 с. – ISBN 978-5-907560-95-6. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/395666. – 14.03.2024.

39 Федотов, Г. В. Компьютерная геометрия и графика / Г. В. Федотов. – СПб.: Лань, 2024. – 84 с. – ISBN 978-5-507-48165-1. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/367397. – 28.05.2024.

40 Филиппов, Ф. В. Моделирование нейронных сетей глубокого обучения: учебное пособие / Ф. В. Филиппов. – СПб.: СПбГУТ им. М.А. Бонч-Бруевича, 2019. – 79 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/180053. – 11.03.2024.

41 Филиппов, Ф. В. Моделирование нейронных сетей на R: учебное пособие / Ф. В. Филиппов. – СПб.: СПбГУТ им. М.А. Бонч-Бруевича, 2016. – 83 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/180047. – 21.03.2024.

42 Хабр [Электронный ресурс]: офиц. сайт. – Режим доступа: https://habr.com/ru/companies/skillfactory/articles/581794/. – 21.03.2024.

43 Шматов, Г. П. Нейронные сети и генетический алгоритм: учебное пособие / Г. П. Шматов. – Тверь: ТвГТУ, 2019. – 200 с. – ISBN 978-5-7995-1007-

7. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/171312. – 28.02.2024.

44 Шолле, Ф. Глубокое обучение на Python [Текст] / Ф. Шолле. – СПб.: Питер, 2023. – 576 с.

45 Шульдова, С. Г. Компьютерная графика: учебное пособие / С. Г. Шульдова. – Минск: РИПО, 2020. – 299 с. – ISBN 978-985-503-987-8. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/154207. – 28.05.2024.

46 Ярышев, С. Н. Технологии глубокого обучения и нейронных сетей в задачах видеоанализа: учебное пособие / С. Н. Ярышев, В. А. Рыжова. – СПб.: НИУ ИТМО, 2022. – 82 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/283967. – 28.01.2024.

47 Project Jupiter [Электронный ресурс]: офиц. сайт. – Режим доступа: https://pythonru.com/baza-znanij/jupyter-notebook-dlja-nachinajushhih. – 05.03.2024.

48 Python [Электронный ресурс]: офиц. сайт. – Режим доступа: https://www.python.org/. – 06.03.2024.

49 PythonRU [Электронный ресурс]: офиц. сайт. – Режим доступа: https://www.tensorflow.org/?hl=ru. – 06.03.2024.

50 Timeweb Комьюнити [Электронный ресурс]: офиц. сайт. – Режим доступа: https://timeweb.com/ru/community/articles/obrabotka-i-generaciya-izobrazheniy-v-python-biblioteka-pillow. – 06.02.2024.

51 Visual Studio [Электронный ресурс]: офиц. сайт. – Режим доступа: https://visualstudio.microsoft.com/ru/vs/. – 04.03.2024.