|  |  |
| --- | --- |
| Герб МГТУ | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
|  | |
| ФАКУЛЬТЕТ | «Робототехника и комплексная автоматизация» |
| КАФЕДРА | «Системы автоматизированного проектирования (РК-6)» |
|  | |
| **РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА** | |
| ***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ*** | |
| ***НА ТЕМУ:*** | |
| ***«Методы анализа распознавания изображения с использованием FPGA – платы»*** | |
|  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | РК6-82Б |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Караф С.М.** |
| (Группа) | (Подпись, дата) | (Фамилия И.О.) |
| Руководитель ВКР | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
| (Подпись, дата) | (Фамилия И.О.) |
| Нормоконтролёр | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Грошев С.В.** |
| (Подпись, дата) | (Фамилия И.О.) |

|  |
| --- |
|  |
| *2022 г.* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** | | | |
|  | УТВЕРЖДАЮ | | |
| Заведующий кафедрой | | | РК6 |
| (Индекс) |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | Карпенко А.П. | |
| (Фамилия И.О.) | |
| «15» февраля 2022 г. | | | |

|  |  |
| --- | --- |
| **ЗАДАНИЕ** | |
| **на выполнение выпускной квалификационной работы бакалавра** | |
| Студент группы | РК6-82Б |
| Караф Сармат Майк | |
| (фамилия, имя, отчество) | |
| Тема квалификационной работы: «Методы анализа распознавания изображения с использованием FPGA – платы» | |
| Источник тематики (НИР кафедры, заказ организаций и т.п.): кафедра. | |
| Тема квалификационной работы утверждена распоряжением по факультету РК № от « »  202 г. | |
| **Техническое задание** | |
| **Часть 1.** *Аналитическая часть* | |
| *Провести анализ различных датасетов, включая датасеты с учителем и без учителя, а также исследование сверточных нейронных сетей, используемых для распознавания изображений. Оценивается работа платы FPGA в контексте задачи распознавания изображений. Актуальность данной работы обусловлена растущим спросом на нейронные сети в компьютерном зрении и потенциалом FPGA в высокопроизводительных вычислениях.* | |
| **Часть 2.** *Практическая часть 1: Создание нейронной сети* | |
| * 1. *Создание датасета:*   *Необходимо разработать датасет для обучения нейронной сети на изображениях с распознаванием человека. Датасет должен включать изображения без разметки (для обучения без учителя) и с разметкой (для обучения с учителем). Учтите разнообразие условий, в которых может находиться человек на изображениях.*  *2.2* *Написание архитектуры нейронной сети:*  *Разработайте архитектуру сверточной нейронной сети, используя фреймворк PyTorch. Учтите задачу распознавания человека на изображениях и стремитесь достичь высокой точности.*  *2.3 Обучение нейронной сети:*  *Обучите нейронную сеть на созданном датасете до достижения высокой точности распознавания человека на изображениях. Для обучения используйте как обучение с учителем, так и без учителя с применением методов глубокого обучения, например, автокодировщиков или генеративных моделей.* | |
| **Часть 3.** *Практическая часть 2: Совместимость нейронной сети на плате FPGA*  *Исследуйте совместимость разработанной нейронной сети с платой FPGA, чтобы симулировать ее работу. Обеспечьте эффективное взаимодействие между нейронной сетью и аппаратным обеспечением FPGA для достижения оптимальной производительности и скорости обработки изображений.*  **Часть 4.** *Аналитическая часть:* | |
| *Анализ результатов работы нейронной сети на платформе FPGA и без нее. Оцените точность распознавания человека на изображениях, а также производительность и эффективность работы модели на плате FPGA по сравнению с работой на CPU или GPU. Сравните результаты существующих подходов и моделей в задаче распознавания изображений с применением нейронных сетей.* | |
| **Оформление выпускной квалификационной работы** | |
| Расчетно-пояснительная записка на 76 листах формата A4. | |
| Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.): | |
| |  | | --- | | *Работа содержит 7 графических листов формата A4.* | |  | | |
| Дата выдачи задания: «10» февраля 2023 г. | |
| В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до «16» июня 2023 г. | |

|  |  |  |
| --- | --- | --- |
| **Руководитель квалификационной работы** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | (Фамилия И.О.) |
| **Студент** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Караф С.М.** |
|  | (Подпись, дата) | (Фамилия И.О.) |

|  |
| --- |
| Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** | | | | | |
| ФАКУЛЬТЕТ | РК |  | УТВЕРЖДАЮ | | |
| КАФЕДРА | РК6 | Заведующий кафедрой | | | РК6 |
| (Индекс) |
| ГРУППА | РК6-82Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | Караф С.М. | |
| (Фамилия И.О.) | |
| «15» февраля 2022 г. | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **КАЛЕНДАРНЫЙ ПЛАН** | | | | | | | | |
| **выполнения выпускной квалификационной работы** | | | | | | | | |
| Студент группы | | | РК6-82Б | | | | | |
| Караф Сармат Майк | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | |
| Тема квалификационной работы: «Разработка метода повышения производительности рендеринга групп 3d-объектов на Unreal Engine 4» | | | | | | | | |
| **№ п/п** | **Наименование этапов выпускной квалификационной работы** | | **Сроки выполнения этапов** | | **Отметка о выполнении** | |
| **план** | **факт** | **Должность** | **ФИО, подпись** |
|  | Задание на выполнение работы. Формулирование проблемы, цели и задач работы | | 10.02.2023 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | 1 часть. Аналитическая часть | | 18.02.2023 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач | | 28.02.2023 | \_\_\_\_\_\_\_\_\_ | Заведующий кафедрой | Караф С.М. |
|  | 2 часть. Практическая часть 1 | | 21.04.2023 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | 3 часть. Практическая часть 2 | | 23.05.2023 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | 1-я редакция работы | | 28.05.2023 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | Подготовка доклада и презентации | | 04.06.2023 | \_\_\_\_\_\_\_\_\_ | Студент | Караф С.М. |
|  | Заключение руководителя | | 10.06.2023 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | Допуск работы к защите на ГЭК (нормоконтроль) | | 17.06.2023 | \_\_\_\_\_\_\_\_\_ | Нормоконтролер | Грошев С.В. |
|  | Внешняя рецензия | | 17.06.2023 | \_\_\_\_\_\_\_\_\_ |  |  |
|  | Защита работы на ГЭК | | 21.06.2023 | \_\_\_\_\_\_\_\_\_ |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Караф С,М. | Руководитель ВКР | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Витюков Ф.А. |
|  | (подпись, дата) | (Фамилия И.О.) |  | (подпись, дата) | (Фамилия И.О.) |

**АННОТАЦИЯ**

Данная работа посвящена разработке нейронной сети с использованием фреймворка PyTorch для эффективного распознавания изображений. Основная цель исследования заключается в сравнении разработанной модели с базовой моделью, предоставленной PyTorch, а также с аналогичной моделью, реализованной для работы на плате FPGA. В рамках работы проведен анализ датасетов с учителем и без учителя, а также рассмотрены сверточные нейронные сети, применяемые в компьютерном зрении.

Практическая часть включает создание собственного датасета, состоящего из изображений с разметкой. Далее разработана архитектура нейронной сети, оптимизированная для задачи распознавания изображений, и проведено ее обучение на созданном датасете с использованием PyTorch. Для сравнения производительности моделей, разработанная нейронная сеть протестирована на наборе тестовых изображений, а также выполнено сравнение с моделью, адаптированной для работы на плате FPGA. При сравнении учитывались такие показатели, как точность распознавания, время обработки.

Полученные результаты позволяют сделать выводы о преимуществах и недостатках разработанной нейронной сети в сравнении с базовой моделью PyTorch и моделью на плате FPGA. Оптимальная архитектура нейронной сети и использование специализированной платформы могут значительно повысить производительность и эффективность распознавания изображений.

Тип работы: выпускная квалификационная работа.

Тема работы: «Методы анализа распознавания изображения с использованием FPGA – платы».

Объекты исследований: нейронные сети, распознавание изображений, PyTorch, FPGA, датасет, сверточные нейронные сети, производительность, точность распознавания.

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

**Нейронные сети (НС):** компьютерные модели, инспирированные работой нервной системы, использующие параллельную обработку и обучение для решения задач.

**Распознавание изображений:** процесс автоматического определения и классификации содержимого изображений с использованием компьютерных алгоритмов и моделей.

**PyTorch**: фреймворк глубокого обучения с открытым исходным кодом, позволяющий создавать и обучать нейронные сети.

**Ошибка (loss):** мера расхождения между предсказаниями нейронной сети и ожидаемыми значениями, которая используется для оптимизации модели в процессе обучения.

**Эпоха (epoch):** один проход по всему обучающему набору данных в процессе обучения нейронной сети.

**Пакет (batch):** группа обучающих примеров, используемых для вычисления градиента и обновления параметров нейронной сети во время обучения.

**Функция активации:** математическая функция, применяемая к выходу нейрона, которая определяет активацию нейрона в ответ на входные данные.

**Обратное распространение** ошибки (backpropagation): алгоритм, используемый для вычисления градиента ошибки нейронной сети и обновления ее параметров в процессе обучения.

**Градиентный спуск (gradient descent):** метод оптимизации, используемый для настройки параметров нейронной сети путем минимизации функции ошибки.

**Функция потерь (loss** **function):** математическая функция, используемая для измерения ошибки нейронной сети на задаче обучения и оптимизации ее параметров.

**Архитектура нейронной сети:** структура и организация нейронной сети, включая количество слоев, количество нейронов в каждом слое и типы связей между нейронами.

**Переобучение (overfitting):** состояние, когда нейронная сеть слишком точно запоминает обучающие примеры и показывает плохую обобщающую способность на новых данных.

**Регуляризация (regularization):** методы, применяемые для уменьшения переобучения нейронной сети, путем добавления дополнительных ограничений на параметры модели.

**Перекрестная энтропия (cross-entropy):** функция потерь, используемая для задач классификации, которая измеряет расхождение между предсказаниями нейронной сети и истинными метками классов.

**Dropout:** метод регуляризации, при котором случайным образом отключаются некоторые нейроны в процессе обучения для уменьшения переобучения.

**Алгоритм обучения**: процедура, определяющая, как нейронная сеть обновляет свои параметры на основе обучающих данных и функции потерь.

**FPGA (Field-Programmable Gate Array):** программируемый матричный элемент, используемый для создания цифровых интегральных схем, специализированных для выполнения определенных вычислительных задач.

**ПЛИС (Programmable Logic Integrated Circuit):** тип интегральной схемы, содержащий логические блоки (CLB) и блоки ввода-вывода (IOB), которые могут быть программированы для выполнения определенных функций.

**CLB (Configurable Logic Block):** программируемый логический блок в ПЛИС, состоящий из комбинационных элементов и регистров для выполнения логических операций.

**IOB (Input/Output Block):** блок ввода-вывода в ПЛИС, предназначенный для обмена данными с внешними устройствами.

**VHDL (Very High-Speed Integrated Circuit Hardware Description Language):** язык описания аппаратуры, используемый для моделирования и синтеза цифровых систем.

**Verilog:** язык описания аппаратуры, используемый для моделирования и синтеза цифровых систем.

**ONNX (Open Neural Network Exchange):** формат обмена нейронными сетями с открытым исходным кодом, который обеспечивает переносимость моделей между различными фреймворками глубокого обучения.

**Virgilio:** библиотека машинного обучения на основе Python, предоставляющая набор инструментов и моделей для разработки и обучения нейронных сетей.

**Оптимизация:** процесс улучшения производительности или эффективности нейронной сети путем изменения ее структуры или параметров.

**Компиляция**: процесс преобразования высокоуровневого кода нейронной сети в низкоуровневое представление, которое может быть выполнено на аппаратных устройствах или специализированных платформах.

**Virgilio ONNX:** инструмент, позволяющий конвертировать модели, разработанные с использованием библиотеки Virgilio, в формат ONNX.

**Quartus:** интегрированная среда разработки (IDE) для проектирования и программирования ПЛИС (Programmable Logic Integrated Circuit).

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 10](#_Toc137665926)

[1.Теоретическая часть 13](#_Toc137665927)

[1.1. Сверточная нейронная сеть 13](#_Toc137665928)

[1.1.1. Сверточный слой (Convolution Layer) 17](#_Toc137665929)

[1.1.2. Слой пулинга 20](#_Toc137665930)

[1.1.4. Классификация. Полносвязный слой (FC Layer) 22](#_Toc137665931)

[1.4. Запекание текстур 23](#_Toc137665932)

[1.5. Создание текстур 24](#_Toc137665933)

[1.6. Создание LOD моделей 25](#_Toc137665934)

[1.7. Экспорт ассетов 26](#_Toc137665935)

[1.8. Импорт ассетов в Unreal Engine 4 27](#_Toc137665936)

[2. Разработка внутриигровых систем 28](#_Toc137665937)

[2.1. Разработка системы «поле боя» 29](#_Toc137665938)

[2.1.1. Основные принципы реализации 30](#_Toc137665939)

[2.1.2. Программная реализация 31](#_Toc137665940)

[2.2 Разработка системы «руки» игрока 32](#_Toc137665941)

[2.2.1. Анализ возможных подходов 33](#_Toc137665942)

[3. Повышение производительности при рендеринге групп 3d-объектов 34](#_Toc137665943)

[3.1.1 Механизм инстансинга геометрии 35](#_Toc137665944)

[3.2. Разработка системы пулов отрисовки (Render Pools) 36](#_Toc137665945)

[3.2.1. Принципы реализации 37](#_Toc137665946)

[3.2.2. Анализ производительности 38](#_Toc137665947)

[ЗАКЛЮЧЕНИЕ 39](#_Toc137665948)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 40](#_Toc137665949)

ВВЕДЕНИЕ

Распознавание изображений с использованием нейронных сетей представляет собой активно развивающуюся область в области компьютерного зрения. Возможность компьютерных систем эффективно классифицировать объекты на изображениях на уровне или даже превосходящем человеческие возможности имеет широкий спектр практических применений, таких как автоматическая обработка изображений, медицинская диагностика, робототехника, автономные транспортные средства и другие.

Актуальность исследования обусловлена значимостью точного и эффективного распознавания изображений во множестве приложений. Однако сложность этой задачи требует разработки и оптимизации мощных моделей машинного обучения. В этом контексте фреймворк PyTorch заслуживает особого внимания, поскольку он обеспечивает гибкость и высокую производительность при разработке и обучении нейронных сетей. PyTorch предоставляет широкий набор инструментов и функций, таких как автоматическое дифференцирование, оптимизаторы и предварительно обученные модели, что упрощает и ускоряет процесс создания и обучения нейронных сетей для распознавания изображений.

Однако, несмотря на преимущества PyTorch, существуют и другие подходы к выполнению задач распознавания изображений. Одним из таких подходов является использование платы FPGA (Field-Programmable Gate Array). FPGA представляет собой программируемое логическое устройство, которое может быть настроено для выполнения специализированных вычислительных задач. Использование FPGA для выполнения задач распознавания изображений может привести к улучшению скорости обработки и оптимизации энергопотребления.

Поэтому данная работа нацелена на разработку нейронной сети для распознавания изображений с использованием фреймворка PyTorch и ее сравнение с моделью, специально разработанной для платы FPGA. В ходе исследования будет создан и обучен датасет, разработана архитектура нейронной сети, а затем сравнены результаты работы модели на PyTorch и FPGA. Это позволит оценить эффективность и сравнительные преимущества обоих подходов в контексте задачи распознавания изображений.

Таким образом, данное исследование имеет значимость для развития области компьютерного зрения и может привести к улучшению производительности и эффективности систем распознавания изображений с использованием нейронных сетей и плат FPGA.

Исследование посвящено разработке нейронной сети для распознавания изображений с использованием фреймворка PyTorch и сравнению ее с моделью, разработанной для платы FPGA. В ходе исследования была проведена аналитическая часть, в рамках которой были изучены различные датасеты с учителем и без учителя, сверточные нейронные сети и возможности платы FPGA.

Практическая часть работы включала создание датасета для обучения нейронной сети в задаче распознавания людей на изображениях. Затем была разработана архитектура нейронной сети с использованием сверточных слоев, пулинга и полносвязных слоев. Обучение нейронной сети проводилось на созданном датасете с использованием алгоритма обратного распространения ошибки и метода градиентного спуска. Далее была обеспечена совместимость разработанной нейронной сети с платой FPGA с помощью инструмента FiftyOne.

В ходе исследования была проведена оптимизация работы модели нейронной сети на плате FPGA с целью повышения ее эффективности. Были применены методы оптимизации алгоритмов обработки изображений, использованы специфические функции и возможности FPGA, а также проведена настройка параметров модели.

Ожидаемыми результатами исследования являются выводы о преимуществах и недостатках разработанной нейронной сети и модели на плате FPGA. Ожидается, что модель на плате FPGA обладает высокой скоростью обработки изображений и эффективным использованием ресурсов. В то же время, разработанная нейронная сеть на основе фреймворка PyTorch может обеспечить высокую точность в распознавании изображений. Таким образом, данное исследование имеет научную и практическую значимость, поскольку способствует развитию области компьютерного зрения и оптимизации систем распознавания изображений с использованием нейронных сетей и плат FPGA.

Цели работы:

1. Анализ сверточных сетей для понимания их принципов и применения в задачах распознавания изображений.
2. Исследование плат FPGA для понимания их возможностей и применимости в области обработки изображений.
3. Создание собственного датасета для распознавания людей на изображениях.
4. Разработка, обучение и тестирование сверточной нейронной сети на основе созданного датасета.
5. Проверка совместимости модели с платой FPGA.
6. Оценка эффективности работы нейронной сети на плате FPGA и выявление ее преимуществ и ограничений в задаче распознавания изображений.

Для выполнения работы были использованы средства следующих программ:

* PyTorch - фреймворк для глубокого обучения.
* COCO Dataset - набор данных с размеченными изображениями.
* FiftyOne - инструмент для аннотирования и визуализации данных компьютерного зрения.
* ONNX - формат представления моделей и нейронных сетей.
* Quartus Prime - среда разработки для плат FPGA.
* VHDL - язык описания аппаратуры для плат FPGA.

1.Теоретическая часть

1.1. Сверточная нейронная сеть

Сверточные нейронные сети (СНС) это специализированный тип нейронных сетей, которые используют свертку вместо общего матричного умножения по крайней мере в одном из своих слоев. или, другими словами, Сверточная нейронная сеть (CNN) - это тип искусственной нейронной сети, используемой в распознавании и обработке изображений, которая специально предназначена для обработки пиксельных данных.

В обычном перцептроне, который представляет собой полносвязную нейронную сеть, каждый нейрон связан со всеми нейронами предыдущего слоя, причём каждая связь имеет свой персональный весовой коэффициент. В свёрточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале — непосредственно по входному изображению), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют ядром свёртки. Её интерпретируют как графическое кодирование какого-либо признака, например, наличие наклонной линии под определённым углом. Тогда следующий слой, получившийся в результате операции свёртки такой матрицей весов, показывает наличие данного признака в обрабатываемом слое и её координаты, формируя так называемую карту. Естественно, в свёрточной нейронной сети набор весов не один, а целая гамма, кодирующая элементы изображения (например линии и дуги под разными углами). При этом такие ядра свёртки не закладываются исследователем заранее, а формируются самостоятельно путём обучения сети классическим методом обратного распространения ошибки. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многоканальной (много независимых карт признаков на одном слое). Также следует отметить, что при переборе слоя матрицей весов её передвигают обычно не на полный шаг (размер этой матрицы), а на небольшое расстояние. Так, например, при размерности матрицы весов 5×5 её сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не «перешагнуть» искомый признак.

Операция субдискретизации, выполняет уменьшение размерности сформированных карт признаков. В данной архитектуре сети считается, что информация о факте наличия искомого признака важнее точного знания его координат, поэтому из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон уплотнённой карты признаков меньшей размерности. За счёт данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения.

Рис. 11. Типовая архитектура свёрточной нейронной сети

Для компьютера изображение в формате RGB представляет собой суммарную информацию трех разных матриц. Для каждого пикселя изображения определяется его цвет. Красная компонента описывается в первой матрице, зеленая компонента - во второй, а синяя - в последней. Таким образом, для изображения размером 3 на 3 пикселя мы получаем три различные матрицы размером 3х3.

Для обработки изображения каждый пиксель подается на вход нейронной сети. Таким образом, для изображения размером 200х200х3 (т.е. 200 пикселей по ширине, 200 пикселей по высоте и 3 цветовых канала: красный, зеленый и синий) нам нужно предоставить 200 \* 200 \* 3 = 120 000 входных нейронов. Затем каждая матрица имеет размер 200 на 200 пикселей, то есть в общей сложности 200 \* 200 записей. Эта матрица затем существует в трех экземплярах, по одному для красного, синего и зеленого цветов. Проблема возникает в первом скрытом слое, потому что каждый нейрон в этом слое должен иметь 120 000 весов от входного слоя. Это означает, что количество параметров будет очень быстро Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описаниеувеличиваться с увеличением числа нейронов в скрытом слое.

Рис. 2. Изображение 3x3x3 RGB.

Эта проблема усугубляется, когда мы хотим обрабатывать более крупные изображения с большим количеством пикселей и цветовых каналов. Такая сеть с огромным количеством параметров, скорее всего, столкнется с переобучением. Это означает, что модель будет давать хорошие предсказания для обучающего набора данных, но не будет хорошо обобщаться на новые случаи, с которыми она еще не сталкивалась. Кроме того, из-за большого количества параметров сеть, скорее всего, перестанет обращать внимание на отдельные детали изображения, так как они будут потеряны в общей массе. Однако, если мы хотим классифицировать изображение, например, определить, есть ли на нем собака или нет, эти детали, такие как нос или уши, могут быть решающим фактором для правильного результата.

Поэтому сверточная нейронная сеть идет по-другому пути, имитируя то, как мы воспринимаем окружающую среду с помощью глаз. Когда мы видим изображение, мы автоматически разделяем его на множество маленьких подизображений и анализируем их поочередно. Собирая эти подизображения, мы обрабатываем и интерпретируем изображение. Как можно реализовать этот принцип в сверточной нейронной сети?

Работа происходит в так называемом сверточном слое. Для этого мы определяем фильтр, который определяет размер частичных изображений, которые мы рассматриваем, и шаг, который определяет, на сколько пикселей мы продолжаем между вычислениями, т.е. насколько близко друг к другу располагаются частичные изображения. Благодаря этому шагу мы значительно сокращаем размерность изображения.

Следующий шаг - слой пулинга. С точки зрения вычислений здесь происходит то же самое, что и в сверточном слое, с той разницей, что мы берем только среднее или максимальное значение из результата в зависимости от задачи. Это сохраняет маленькие особенности в нескольких пикселях, которые являются ключевыми для решения задачи.

Наконец, есть полносвязный слой, как мы уже знаем из обычных нейронных сетей. Теперь, когда мы значительно сократили размеры изображения, мы можем использовать плотно связанные слои. Здесь отдельные подизображения связываются снова, чтобы распознать связи и выполнить классификацию.

* + 1. Сверточный слой (Convolution Layer)

Сверточный слой (Convolution Layer) - это ключевой компонент сверточных нейронных сетей, предназначенный для извлечения признаков и шаблонов из входных изображений. Он применяет операцию свертки между фильтрами (ядрами) и входным изображением, позволяя выделять локальные особенности и структурные характеристики изображения.

В сверточном слое фильтры сканируют пиксели входного изображения, перемещаясь по нему с определенным шагом. Каждое значение в выходной матрице сверточного слоя представляет результат операции свертки фильтра с соответствующей подматрицей изображения. Это позволяет обнаруживать различные признаки, такие как контуры, углы, текстуры и другие локальные особенности, независимо от их расположения в изображении.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описаниеСверточный слой играет важную роль в анализе и обработке изображений, позволяя нейронным сетям автоматически выделять значимые признаки, которые могут быть использованы для классификации, распознавания объектов, сегментации изображений и других задач компьютерного зрения. Он помогает сети учиться находить важные шаблоны и паттерны, что делает его основным инструментом в области обработки визуальной информации.

Рис. 2. Сверточный слой.

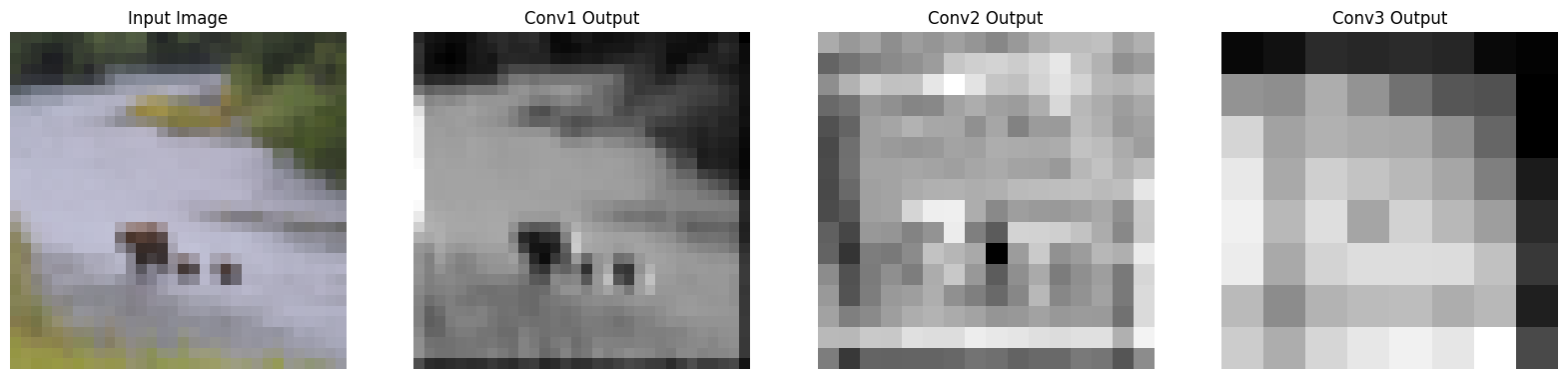
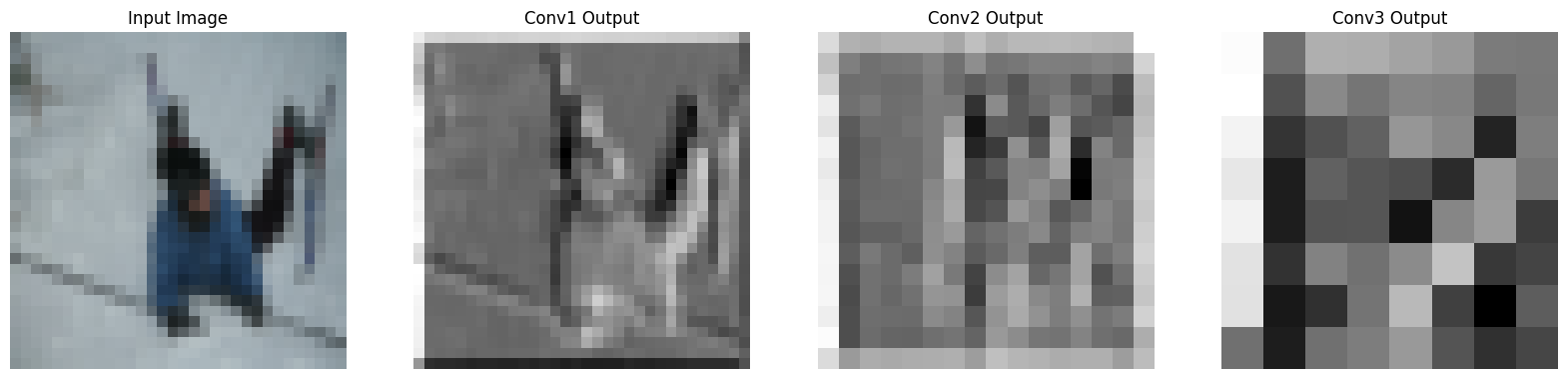
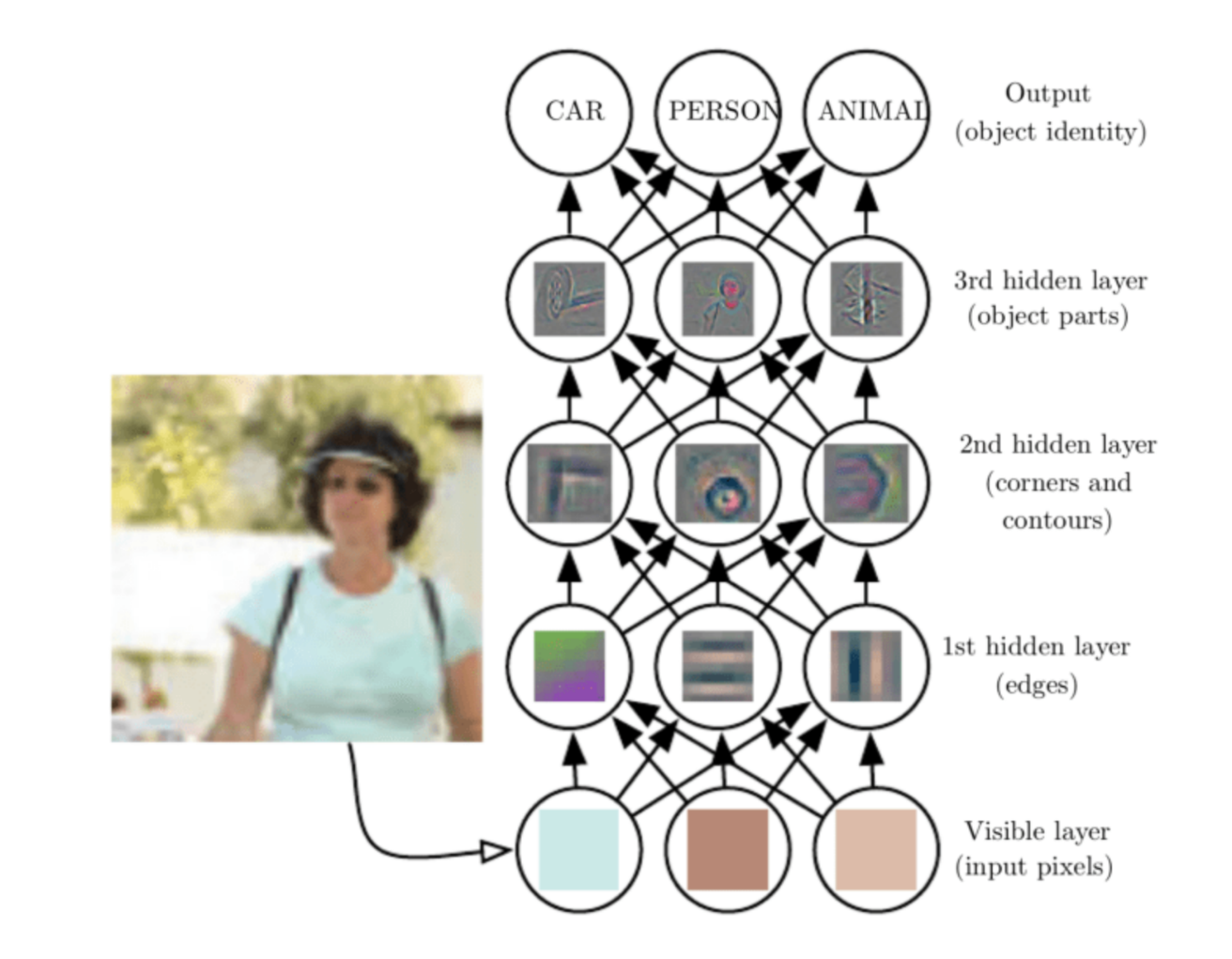
В первом шаге происходит снижение размерности изображения 4x4x3. Для этой цели определяется фильтр размером 2x2 для каждого цвета. Шаг длиной 1 позволяет перемещать фильтр вперед на один пиксель после каждого вычислительного шага. Это приводит к получению матрицы размером 3x3 в сверточном слое.

Рис. 3. Пример сверточного слоя.

Сверточный слой позволяет выделять особенности и пространственные шаблоны в изображении, сохраняя информацию о его структуре. Фильтр сканирует пиксели изображения и вычисляет свертку с подматрицей изображения, выявляя определенные признаки.

Каждое значение в выходной матрице сверточного слоя является результатом операции свертки и объединяет информацию из определенной области входного изображения. Это позволяет извлекать локальные особенности, такие как контуры, углы и текстуры, независимо от их положения в изображении.

Для выполнения свертки в сверточном слое, фильтр перемещается по изображению и вычисляется скалярное произведение с соответствующей подматрицей изображения. После применения свертки ко всем подматрицам изображения получается выходная матрица размером 3x3.

Рис. 4. Процесс переходов от сверточного слоя до итого результата.

Визуализация сверточного слоя позволяет лучше понять, какие признаки и особенности извлекаются на каждом шаге. Входное изображение проходит через фильтр, и каждое значение в выходной матрице связано с определенной характеристикой или признаком, выявленным фильтром.

Сверточные слои являются ключевым элементом сверточных нейронных сетей и позволяют эффективно обрабатывать изображения. Они позволяют сетям распознавать и идентифицировать объекты, а также выделять важные признаки, необходимые для решения конкретных задач классификации или обработки изображений.

* + 1. Слой пулинга



Подобно сверточному слою, слой пулинга отвечает за уменьшение пространственного размера сверточной характеристики. Это делается для уменьшения вычислительной мощности, необходимой для обработки данных путем снижения размерности. Кроме того, это полезно для извлечения доминирующих признаков, которые инвариантны к повороту и позиции, таким образом, поддерживая эффективный процесс обучения модели.

Существуют два типа пулинга: максимальный и средний. Максимальный пулинг возвращает максимальное значение из области изображения, покрытой ядром. С другой стороны, средний пулинг возвращает среднее значение всех значений из области изображения, покрытой ядром.

Максимальный пулинг также действует как средство снижения шума. Шумные активации полностью отбрасываются, а также выполняется устранение шума вместе с уменьшением размерности. С другой стороны, средний пулинг просто осуществляет снижение размерности в качестве механизма снижения шума. Таким образом, можно сказать, что максимальный пулинг работает намного лучше, чем средний пулинг.Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рис. 4. Слой пулинга.

Сверточный слой и слой пулинга вместе формируют i-й слой сверточной нейронной сети. В зависимости от сложности изображений количество таких слоев может быть увеличено для захвата более низкоуровневых деталей, но за счет большей вычислительной мощности.

Пройдя через вышеуказанный процесс, модель успешно осознает особенности. Затем окончательный вывод преобразуется в одномерный массив и передается обычной нейронной сети для целей классификации.

1.1.4. Классификация. Полносвязный слой (FC Layer)

Добавление полносвязного слоя является (обычно) дешевым способом изучения нелинейных комбинаций высокоуровневых признаков, представленных выходом сверточного слоя. В полносвязном слое изучается возможно нелинейная функция в этом пространстве.

Теперь, когда входное изображение преобразовано в подходящую форму для многоуровневого перцептрона, изображение сглаживается в виде столбцового вектора. Сглаженный вывод передается прямой нейронной сети, и для каждой итерации обучения применяется обратное распространение ошибки. В течение нескольких эпох модель способна различать доминирующие и определенные низкоуровневые особенности в изображениях и классифицировать их с использованием метода мягкой классификации Softmax.

Существуют различные архитектуры сверточных нейронных сетей, которые были ключевыми при разработке алгоритмов, обеспечивающих и будущему развитию искусственного интеллекта в целом. Ниже перечислены некоторые из них:

1.4. Запекание текстур

Для того, чтобы перенести детализацию поверхности высокополигональной модели на низкополигональную, существует процесс, называемый *запеканием текстур*. В результате его выполнения можно получить набор текстур, содержащих информацию о поверхности high-poly модели.

1.5. Создание текстур

С помощью запекания текстур нам удалось перенести детализацию поверхности высокополигональной модели на низкополигональную. Однако, множество свойств поверхности все еще не задано. Для решения данной проблемы необходимо создать набор текстур. С их помощью можно задать цветовые свойства, а также такие параметры, как жесткость, металличность, и т.д.

1.6. Создание LOD моделей

Сделав low-poly версию модели, мы уже получили значительный прирост производительности. Однако, зная, что модель будет находиться на различных дистанциях от камеры, либо занимать крайне малую часть экрана, можно ввести дополнительную оптимизацию. Тратить время на отрисовку даже ранее рассмотренной low-poly версии модели бесполезно, если объект настолько визуально мал, что сложно различить силуэт. Следовательно, можно создать дополнительные версии модели, которые будут содержать гораздо меньшее количество информации, но всё ещё достаточное для корректного восприятия на планируемом расстоянии.

1.7. Экспорт ассетов

1.8. Импорт ассетов в Unreal Engine 4

На предыдущем этапе были получены все необходимые ассеты. Теперь их необходимо импортировать в движок. Для этого воспользуемся встроенной в Unreal Engine 4 утилитой. Вызвать ее можно через меню Add/Import в Content Br

2. Разработка внутриигровых систем

ККИ (CCG, Collectible Card Game, коллекционная карточная игра) – разновидность настольных и компьютерных игр. Как правило, ККИ являются стратегическими играми с элементом случайности. Каждая карта представляет собой элемент игры: существо, заклинание, ресурс, и т.д.

* кладбище (Graveyard) – набор использованных карт.

В данной работе рассматривается разработка визуальной части двух систем: поле боя и «рука»

.

2.1. Разработка системы «поле боя»

Поле боя – одна из основных зон компьютерных ККИ. Представляет собой часть 3d-сцены, на которую происходит выкладка карт игроков. При создании системы поля боя необходимо учесть следующие требования:

2.1.1. Основные принципы реализации

Каждый игрок имеет собственную область на поле боя, в которую происходит выкладка карт. Выкладка происходит по рядам, поддерживающим группировку карт.

2.1.2. Программная реализация

Для реализации объекта «поле боя» были разработаны следующие классы:

2.2 Разработка системы «руки» игрока

1. Shadowverse.

2.2.1. Анализ возможных подходов

Magic: The Gathering Arena

space виджете в левом верхнем углу.

3. Повышение производительности при рендеринге групп 3d-объектов

При создании 3d-приложений с рендерингом в реальном времени используется множество различных способов оптимизации на разных уровнях.

Например, при создании 3d-моделей для дальнейшего использования,

3.1.1 Механизм инстансинга геометрии

Инстансинг геометрии – один из способов оптимизации рендеринга. Он используется в тех случаях, когда необходимо отрисовывать большое количество однообразных объектов [3][8]. Основная его цель – понижение количества запросов на отрисовку (draw calls).

Формированием запросов на отрисовку занимается центральный

1. (ISMC);
2. UHierarchicalInstancedStaticMeshComponent (HISMC).

HISMC является дополненной версией ISMC. Он имеет поддержку Level of Detail (LOD).

3.2. Разработка системы пулов отрисовки (Render Pools)

В рамках движка Unreal Engine 4 с помощью существующих компонентов инстансинга (ISMC/HISMC) можно добиться значительного прироста

пулов отрисовки

3.2.1. Принципы реализации

Разработка проводилась с помощью средств движка Unreal Engine 4 на языке программирования C++.

В процессе разработки были созданы два класса:

1. URenderPoolComponent;
2. ARenderPoolManager.
3. данные о меше и материалах.

3.2.2. Анализ производительности

В рамках работы был проведен анализ производительности с инстансингом и без. Для сравнения было измерено:

1. количество кадров в секунду (FPS);
2. количество отрисованных примитивов сцены (треугольников);

ЗАКЛЮЧЕНИЕ

В результате работы был разработан метод повышения производительности при рендеринге групп 3d-объектов. Также был изучен полный цикл создания статических 3d-моделей. Изучены средства разработки на языке C++, предоставляемые движком Unreal Engine 4. Приобретены знания об организации проектов.

Также в рамках данной работы были разработаны элементы компьютерной коллекционной карточной игры. Созданы, оптимизированы и подготовлены для использования в движке 3d-модели карт. Проведен анализ реализаций внутриигровых систем в существующих продуктах. Разработаны следующие внутриигровые системы: система выкладки карт на поле боя, система «руки» игрока.

Проведена оптимизация отрисовки большого количества однообразных 3d-объектов. Для достижения этой цели разработана система пулов отрисовки (Render Pool System), которая обобщает и упрощает работу с инстансингом геометрии.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Unreal Engine 4 Documentation // Unreal Engine Documentation URL: https://docs.unrealengine.com/. Дата обращения: 07.04.2022;
2. Display aspect ratio // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Display\_aspect\_ratio/. Дата обращения: 13.04.2022;
3. Geometry instancing // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Geometry\_instancing. Дата обращения: 21.04.2022;
4. Modeling – Blender Manual // Blender Manual URL: https://docs.blender.org/manual/en/latest/modeling/index.html. Дата обращения: 18.02.2022;
5. Mipmap // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Mipmap. Дата обращения 05.03.2022;
6. Level of Detail (computer graphics) // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Level\_of\_detail\_(computer\_graphics). Дата обращения: 05.04.2022;
7. Creating and Using LODs // Unreal Engine Documentation URL: https://docs.unrealengine.com/4.27/en-US/WorkingWithContent/Types/StaticMeshes /HowTo/LODs/. Дата обращения: 05.04.2022;
8. Инстансинг [Электронный ресурс] // Habr: интернет-портал. URL: <https://habr.com/ru/post/352962/>. Дата обращения: 03.04.2022;
9. UE4 Optimization: Instancing // YouTube: видео хостинг. URL: <https://www.youtube.com/watch?v=oMIbV2rQO4k>. Дата обращения: 03.04.2022;
10. Божко А.Н., Жук Д.М., Маничев В.Б. Компьютерная графика. [Электронный ресурс] // Учебное пособие для вузов. − М.: Изд-во МГТУ им. Н. Э. Баумана, 2007. - 389 с., - ISBN 978-5-7038-3015-4, Режим доступа: http://ebooks.bmstu.ru/catalog/55/book1141.html. Дата обращения: 10.02.2022;
11. Programming Quick Start // Unreal Engine Documentation URL: <https://docs.unrealengine.com/5.0/en-US/unreal-engine-cpp-quick-start/>. Дата обращения: 29.12.2021.