МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий

(наименование института полностью)

Кафедра «Прикладная математика и информатика»

(наименование кафедры)

02.03.03 Математическое обеспечение и администрирование информационных систем

(код и наименование направления подготовки, специальности)

Технология программирования

(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему «Применение свёрточных нейронных сетей для интеллектуальной обработки изображений»

Студент	С.А. Корнилов	
	(И.О. Фамилия)	(личная подпись)
Руководитель	В.С. Климов	
	(И.О. Фамилия)	(личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент кафедры ПМИ, А.В. Очеповский

(ученая степень, звание, И.О. Фамилия)

_____

(личная подпись)

«_____» _____ 20___ г.

Тольятти 2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий

(наименование института полностью)

Кафедра «Прикладная математика и информатика»

(наименование кафедры)

УТВЕРЖДАЮ
Завкафедрой «Прикладная математика и информатика»
_____ _А.В. Очеповский__
(подпись) (И.О. Фамилия)
«____»_____ 20__ г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент Корнилов Сергей Алексеевич

1. Тема «Применение свёрточных нейронных сетей для интеллектуальной обработки изображений»

2. Срок сдачи студентом законченной выпускной квалификационной работы_____

3. Исходные данные к выпускной квалификационной работе: изображения для тестирования, IDE Intellij Idea, Java, алгоритм свёрточной нейронной сети VGG16

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов) Введение, Глава 1 Анализ состояния вопроса применения свёрточных нейронных сетей, Глава 2 Проектирование алгоритма переноса нейронного стиля, Глава 3 Тестирование разработанного программного решения, Заключение, Список используемой литературы, Приложения

5. Ориентировочный перечень графического и иллюстративного материала презентация, исходные и полученные изображения, фрагменты программного кода, графики

6. Консультанты по разделам

7. Дата выдачи задания «20» декабря 2018 г.

Руководитель выпускной квалификационной работы _____ В.С. Климов
(подпись) (И.О. Фамилия)

Задание принял к исполнению _____ С.А. Корнилов
(подпись) (И.О. Фамилия)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий

(наименование института полностью)

Кафедра «Прикладная математика и информатика»

(наименование кафедры)

УТВЕРЖДАЮ
Завкафедрой «Прикладная математика и информатика»
_____ _А.В. Очеповский_
(личная подпись) (И.О. Фамилия)
«____»_____ 20__ г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента Корнилова Сергея Алексеевича
по теме «Применение свёрточных нейронных сетей для интеллектуальной обработки изображений»

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Выбор и утверждение темы ВКР	20.12.2018	20.12.2018	Выполнено	
Написание введения	23.12.2018	31.12.2018	Выполнено	
Анализ состояния вопроса применения сверточных нейронных сетей	10.01.2019	15.01.2019	Выполнено	
Математический аппарат интеллектуальной обработки	15.02.2019	28.02.2019	Выполнено	

изображений				
Разработка программного обеспечения	22.03.2019	24.03.2019	Выполнено	
Тестирование разработанной программы	19.04.2019	22.04.2019	Выполнено	
Оформление пояснительной записки ВКР	01.05.2019	01.05.2019	Выполнено	
Разработка презентации к предзащите	25.05.2019	01.06.2019	Выполнено	
Предзащита	10.06.2019	10.06.2019	Выполнено	
Корректировка ВКР согласно сделанным замечаниям	17.06.2019	17.06.2019	Выполнено	
Проверка ВКР в Системе «Антиплагиат.ВУЗ»	19.06.2019	19.06.2019	Выполнено	
Оформление документов к защите	20.06.2019	20.06.2019	Выполнено	
Сдача пояснительной записки	25.06.2019	25.06.2019	Выполнено	
Защита ВКР	01.07.2019 – 04.07.2019	01.07.2019		

Руководитель выпускной квалификационной работы _____ (подпись) В.С. Климов (И.О. Фамилия)

Задание принял к исполнению _____ (подпись) С.А. Корнилов (И.О. Фамилия)

АННОТАЦИЯ

Название бакалаврской работы – «Применение свёрточных нейронных сетей для интеллектуальной обработки изображений».

Объект исследования – процесс функционирования свёрточных нейронных сетей.

Предмет исследования – алгоритм нейронного переноса стиля.

Цель исследования – разработка алгоритма нейронного переноса стиля с использованием свёрточной нейронной сети.

Для достижения цели решаются следующие задачи:

- проанализировать существующие способы применения свёрточных нейронных сетей;
- выявить малоизученную область применения свёрточных нейронных сетей и сформировать новую задачу, решаемую применением свёрточных нейронных сетей;
- разработать математический аппарат алгоритма сформированной задачи;
- разработать и протестировать систему, на основе построенного математического аппарата и выявленных требований.

В первой главе затрагиваются вопросы процесса работы сверточных нейронных сетей, способы и сферы их применения и на основании полученных данных формулируется новая задача и требования к её решению.

Во второй главе описывается архитектура выбранной для решения свёрточной нейронной сети и причины, по которым она была выбрана. Разрабатывается математический аппарат алгоритма решающего сформулированную задачу и его реализация.

В третьей главе предоставляются результаты тестирования разработанного алгоритма на основе построенного математического аппарата и выявленных требований. Производится корректировка системы и описываются возможности по её улучшению.

Эта работа показывает новые способы использования свёрточных нейронных сетей. Это позволит развивать технологию, открывать новые возможности по её использованию.

Бакалаврская работа выполнена на 42 страницах, состоит из введения, трёх глав, включающих 23 изображений, 8 листингов кода и 7 формул, заключения, списка используемых источников, включающего 18 источников, из них 8 на иностранном языке, и 1 приложения.

ABSTRACT

The title of the graduation thesis is «The use of convolutional neural networks for intelligent image processing».

The aim of the work is to show a new way of application the convolutional neural networks using neural style transfer algorithm.

The object of the thesis is a process of functioning of convolutional neural networks.

The subject of the senior thesis is the neural style transfer algorithm.

The first part of the graduation work describes the analysis of existing methods of using convolutional neural networks and formulation of a new method application of convolutional neural networks.

The second part of the thesis is devoted to the development and implementation of the neural style transfer algorithm.

The third part describes the testing of the developed the neural style transfer algorithm. We also report the results of experiments conducted.

This work shows a new way of using convolutional neural networks. Also, this work will allow to develop the technology and open up new opportunities for its subsequent application.

The graduation work consists of an explanatory note on 42 pages, introduction, including 23 figures, 8 code listing, the list of 19 references including 8 foreign sources and 1 appendix.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современных реалиях, когда идёт бурное развитие технологий, а автоматизация доходит до любого процесса жизни человека, не удивительно, что когда-то описываемое фантастами компьютерное зрение, развилось и дошло до той поры, когда это уже не будущее, а вполне реальное настоящее.

Бурное развитие теорий компьютерного зрения привело к появлению технологий машинного зрения. Словом, машинное зрение – это технологии, которые помогают оборудованию увидеть процесс производства чего-либо, проанализировать данные и принять информированное решение. И все это за доли секунды [17].

Еще в 2012 году технологии машинного зрения были не столь совершенными и могли распознать порядка 65-70% объектов, которые попадали в систему для анализа. Но всё изменилось с появлением свёрточных нейронных сетей. А именно в 2012 году, когда Алекс Крижевски благодаря им выиграл конкурс ImageNet, снизив рекорд ошибок классификации с 26% до 15%, что тогда стало прорывом [18]. Сегодня же существуют свёрточные нейронные сети с процентом классификации близким к 98.

Сверточная нейросеть – это такой особый вид нейросетей прямого распространения, и под прямым распространением понимается то, что переменные нейроны в этой сети разбиты на группы, называемые слоями. И когда такая слоистая нейронная сеть применяется к данным, то активация слоев – значение этих переменных – подсчитывается последовательно: сначала значение активации первого слоя, потом значение активации второго слоя, и так до последнего слоя. Результаты активации последнего слоя служат конечной точкой выхода результатов свёрточной нейронной сети.

Актуальность данной работы состоит в разработке нового алгоритма, решающего не классическую для свёрточных нейронных сетей задачу генерации изображения.

Новизна бакалаврской работы заключается в новом алгоритме и способе использования свёрточной нейронной сети.

Практическая ценность состоит в разработке нового алгоритма нейронного переноса стиля с использованием свёрточной нейронной сети.

Объект исследования – процесс функционирования свёрточных нейронных сетей.

Предмет исследования – алгоритм нейронного переноса стиля.

Бакалаврская работа состоит из введения, трёх глав, заключения, списка литературы и приложений.

В первой главе затрагиваются вопросы процесса работы сверточных нейронных сетей, способы и сферы их применения и на основании полученных данных формулируется новая задача и требования к её непосредственному решению.

Во второй главе описывается архитектура выбранной для решения свёрточной нейронной сети и причины, по которым она была выбрана. Разрабатывается математический аппарат алгоритма решающего сформулированную задачу и его реализация.

В третьей главе предоставляются результаты тестирования разработанного алгоритма на основе построенного математического аппарата и выявленных требований. Производится корректировка системы и описываются возможности по её улучшению.

В заключении подводятся итоги исследования, формируются окончательные выводы и результаты проделанной работы.

Глава 1 АНАЛИЗ СОСТОЯНИЯ ВОПРОСА ПРИМЕНЕНИЯ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

1.1 Описание работы свёрточных нейронных сетей

Нейронная сеть – это громадный распределенный параллельный процессор, состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и представляющих их для последующей обработки [15]. Иначе говоря нейронная сеть представляет из себя математическую модель схожую по своим качествам на мозг человека, таким как:

- возможностью обрабатывать поступающую в него информацию из окружающей среды, которая в последствии используется для обучения;
- накоплением знаний путем настройки синаптических весов на связях между нейроннами.

Пример простой схемы нейронной сети представлен на рисунке 1.1.

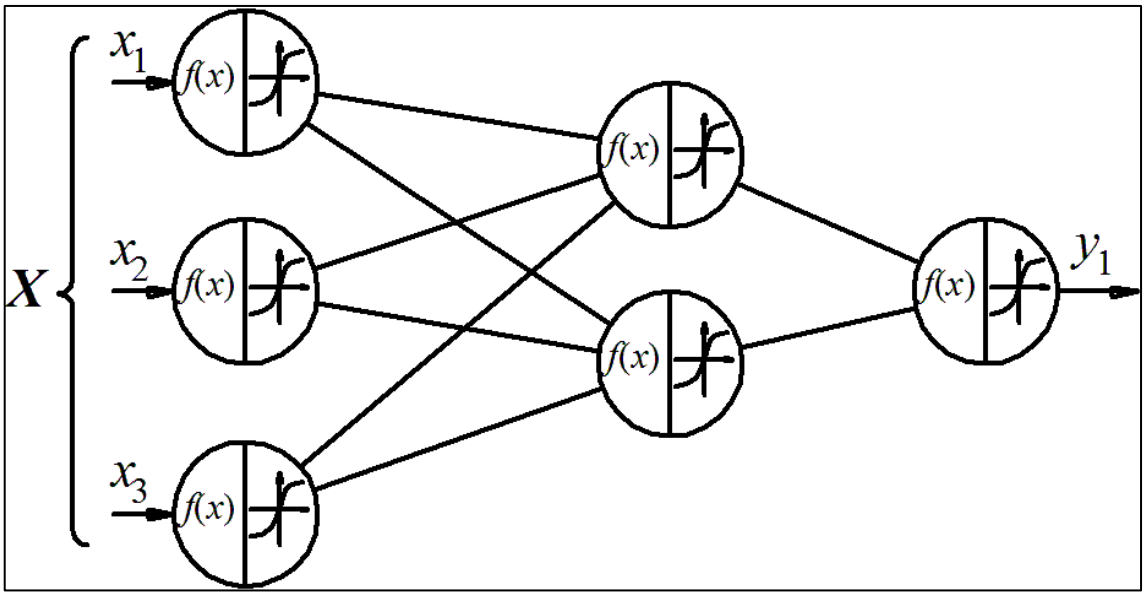

Рисунок 1.1 – Пример схемы простой полносвязной нейронной сети

Принято, что нейронные сети состоят из трёх значимых слоёв:

- слой входных нейронов, обозначенный на схеме зелёными нейронами;

- n-ое количество слоёв скрытых нейронов, обозначенный на схеме одни слоем голубых нейронов;
- выходные нейроны, обозначенные на схеме лишь одним жёлтым нейроном.

Процедура, используемая для процесса обучения, называется алгоритмом обучения (learning algorithm). Эта процедура выстраивает в определенном порядке синаптические веса нейронной сети для обеспечения необходимой структуры взаимосвязей нейронов [15].

Сверточные нейронные сети, СНС (англ. convolutional neural network, CNN) являются биологически вдохновленными вариантами многослойных персептронов (multilayer perceptron). Из ранних работ Хьюбел и Визела по зрительной коре кошки, мы знаем, что зрительная кора содержит сложное расположение клеток. Эти клетки чувствительны к небольшим подобластям поля зрения, называемым рецептивным полем. Субрегионы представлены сеткой, которая покрывает всё поле зрения. Данные ячейки действуют как локальные фильтры в пространстве ввода зрительной информации и хорошо подходят для использования сильной пространственной локальной корреляции, присутствующей в естественных изображениях [1].

Если говорить простым языком, то сверточные нейронные сети представляют из себя нейронную сеть, которая умеет определять контуры и цветовое представление входного изображения. Отличительной чертой подобных сетей является сам процесс свёртки изображения. В ходе которого мы “сворачиваем” матрицу признаков входного изображения и получаем еще одну матрицу признаков, но уже в уменьшенном варианте.

Данный процесс свёртки проходит благодаря ядру, состоящему из матрицы весов, и входному изображению, которое представляет из себя матрицу признаков, например яркость пикселя в диапазоне от 0 до 255. Пример процесса операции свёртки представлен на рисунке 1.2. Ядро проходя по

заданной ему области просчитывает выходную матрицу путем перемножения каждого параметра на вес с последующим их суммированием.

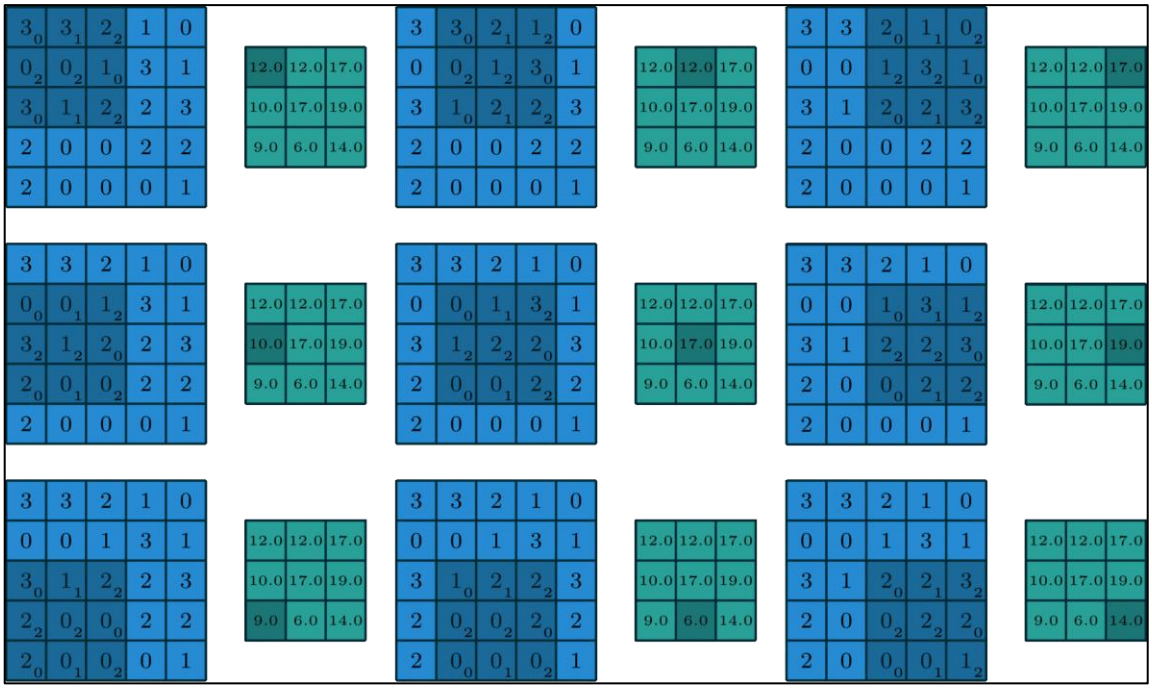

Рисунок 1.2 – Процесс операции свёртки

В примере, приведенном выше, мы имеем 5*5=25 признаков на входе и 3*3=9 признаков на выходе. Для стандартного слоя (standard fully connected layer) мы бы имели весовую матрицу 25*9 = 225 параметров, а каждый выходной признак являлся бы взвешенной суммой всех признаков на входе. Свертка позволяет произвести такую операцию с всего 9-ю параметрами, ведь каждый признак на выходе получается анализом не каждого признака на входе, а только одного входного, находящегося в “примерно том же месте” [11].

Приведенное выше объяснение касается только одноканальных изображений. На практике же мы обычно имеем дело с многоканальными изображениями, а в качестве примера для дальнейшего объяснения возьмем за основу трёхканальные, состоящие из красного, зеленого и синего каналов, известные как RGB изображения, так как они являются наиболее популярными.

Сам процесс свёртки любого многоканального изображения отличается не сильно и представлен на рисунке 1.3. Он заключается в том, что для каждого из каналов используется своё отдельное ядро со своими собственными весами нейронов. При этом каждое подобное ядро может иметь свой собственный вес,

изменяя значение которого, можно регулировать, какому из каналов стоит уделить больше внимания. То есть чем выше вес конкретного ядра, допустим красного канала, тем сильнее будет реакция на различия в образах красного. После того как пройдет процесс свертки каждого ядра, мы суммируем полученные значения, добавляем скалярное смещение и получаем один общий выходной канал. Данная совокупность ядер всех каналов и одного выходного канала называется фильтром.

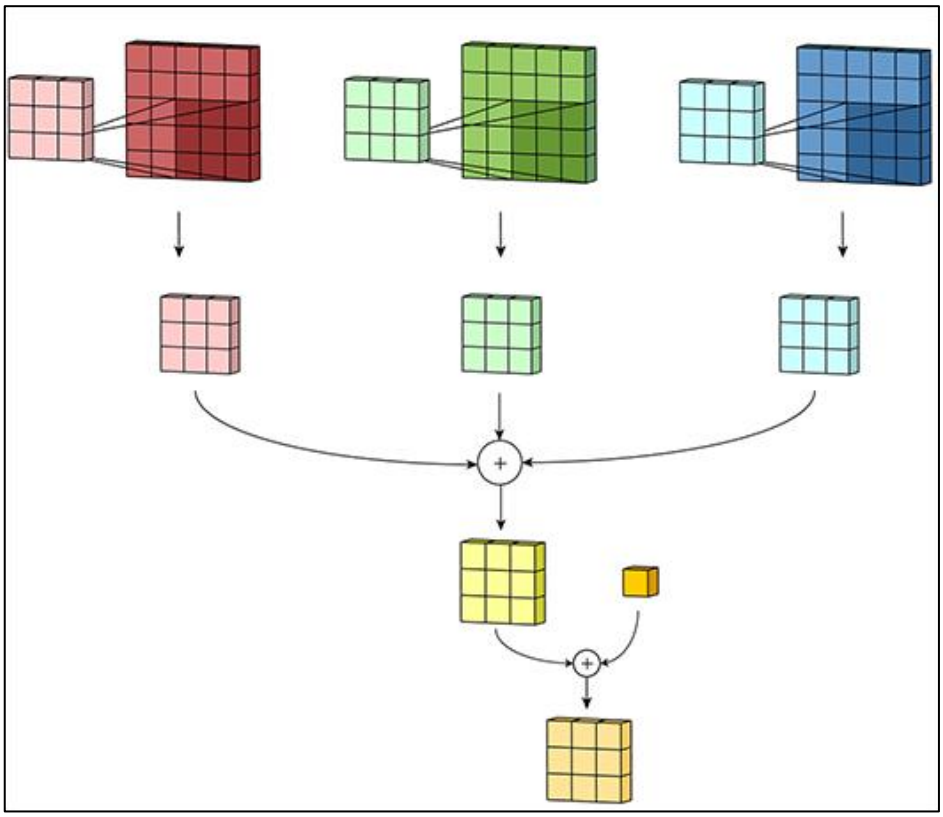

Рисунок 1.3 – Процесс свёртки одного фильтра

Результат для любого количества фильтров идентичен: каждый фильтр обрабатывает вход со своим отличающимся от других набором ядер и скалярным смещением по описанному выше процессу, создавая один выходной канал. Затем они объединяются вместе для получения общего выхода, причем количество выходных каналов равно числу фильтров. При этом обычно применяется нелинейность перед передачей входа другому слою свертки, который затем повторяет этот процесс [11].

1.2 Обзор применения алгоритмов свёрточных нейронных сетей

К классическими задачам в ходе которых часто применяются CNN относятся [16]:

- определение границ;
- определение вектора нормали;
- определение объектов внимания ;
- семантическая сегментация;
- семантическое выделение границ;
- выделение частей тела человека;
- самая высокоуровневая задача – распознавание самих объектов.

Примеры выполнения данных задач приведены на рисунке 1.4.

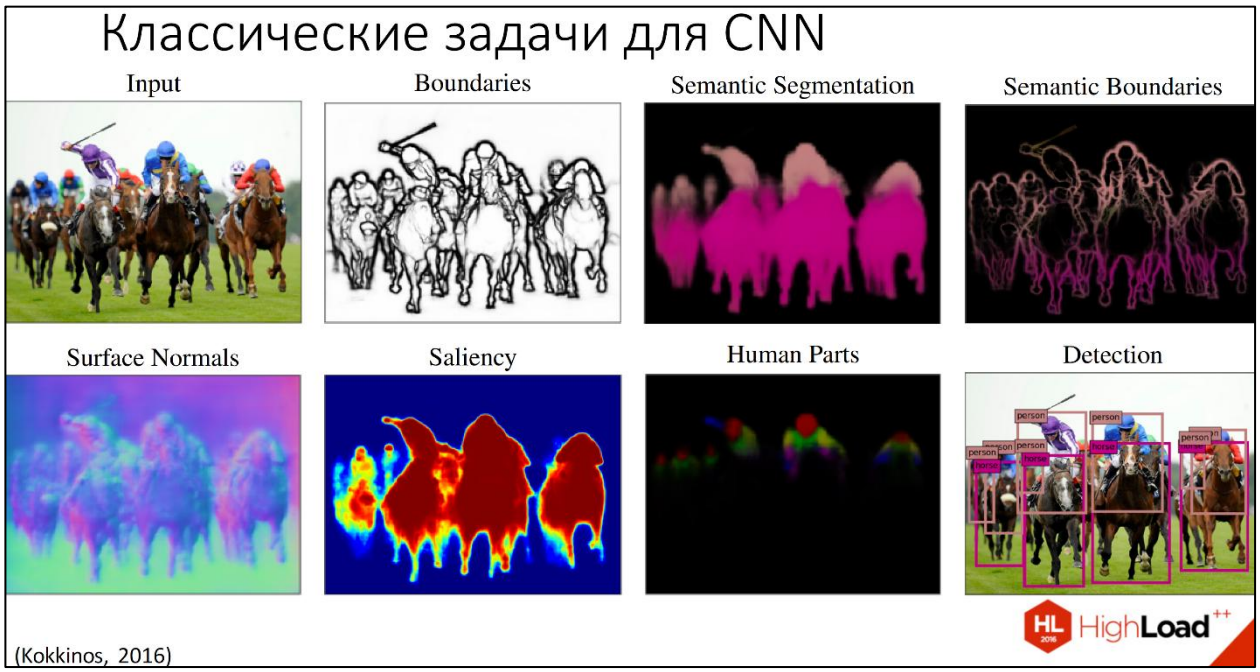

Рисунок 1.4 – Пример выполнения задач CNN [16]

Определение границ (Boundaries) – самая низкоуровневая задача среди перечисленных. Свёрточные нейронные сети, выполняющие данную задачу, находят себя во множестве разнообразных сфер применения. Начиная от дорожного движения, где их используют для определения автомобильных номеров с камер дорожного наблюдения [13], заканчивая распознаванием всем известных captcha.

Определение вектора нормали (Surface normals) – позволяет реконструировать трёхмерное изображение из двухмерного. Благодаря нейронным сетям, выполняющим эту задачу, получают множество снимков нормалей и реконструируют объемную 3d модель чего либо, будь то ваза, здание или реконструированное трехмерное компьютерное представление куска поверхности, созданное с помощью снимков со спутника [3].

Определение объектов внимания или задача локализации (Saliency) – это то, на что обратил бы внимание человек при рассмотрении этой картинки, то есть нейронная сеть занимается локализацией объекта внимания. Обученные на данную задачу свёрточные нейронные сети, способны отыскать как грибы в лесу [4], так и прогнозировать содержательность видео [7].

Семантическая сегментация (Semantic segmentation) и семантическое выделение границ (Semantic boundaries) позволяют разделить объекты на классы по их структуре, ничего не зная об этих объектах. Пожалуй одни из самых ключевых задач на сегодняшний день для нейронных сетей и компьютерного зрения в целом, так как именно решение этих задач позволяют компьютеру осознавать сцену, которую он видит, целиком. Сфера применения обширна, только на сегодняшний день данные сверточные нейронные сети используются в автопилотируемых транспортных средствах, при взаимодействии компьютера и человека или при использовании виртуальной реальности [2]. Как можно понять, были перечислены довольно сложные задачи, а это только малая часть из тех, в которых может пригодиться семантическая сегментация.

Выделение частей тела человека (Human parts) – подвид задачи семантической сегментации, по которой особое внимание уделяется выделению человека и определению его частей тела. Практическое применение находит в медицине.

Задача распознавания (Detection) – самая высокоуровневая задача из всех перечисленных. Название говорит само за себя. Свёрточные нейронные сети, выполняющие данную задачу, находят и классифицируют объекты на

изображении. С помощью данных свёрточных нейронных сетей можно определять виды и подвиды животных на изображении, распознавать человека по фотографии и находить его профиль в социальных сетях [16].

Из всего перечисленного можно сделать вывод, что свёрточные нейронные сети, которые выполняют ставшие классическими для неё самой задачи, используются в довольно обширном спектре разнообразных задач самой разной направленности, сложности и ответственности.

1.3 Постановка задачи

Отталкиваясь от обзора классических задач, для выполнения которых применяются свёрточные нейронные сети, можно сделать вывод, что применение CNN не ограничивается только ими, и можно найти ещё достаточное количество способов, как использовать данные нейронные сети в нестандартных для неё сферах применения.

Один из таких способов – это обработка и создание нового изображения. Как пример: создание нового изображения путем переноса стиля рисования с одного изображения на другое, которым будет заниматься свёрточная нейронная сеть. Английское название данного процесса звучит как Neural Style Transfer или NST, общепринятого перевода на русский язык еще нет, поэтому стоит называть данный процесс просто Нейронным Переносом Стиля изображения [12].

Нейронный перенос стиля – процесс при котором создаётся изображение беря основную часть изображения с изображения контента и стиль рисования с изображения стиля.

В качестве примера возьмем два изображения: одно случайно взятое из любого источника (контентное изображение) и художественную картину (изображение стиля), стиль которой нам понравился (рисунок 1.5).

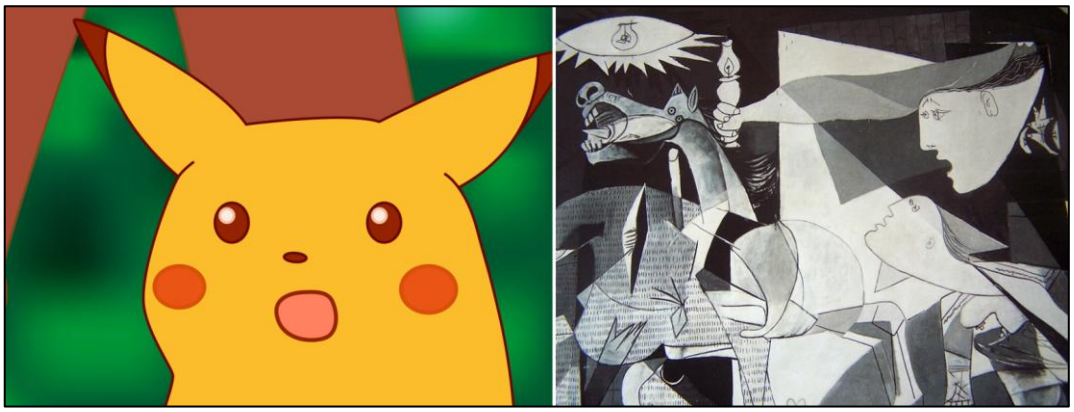

Рисунок 1.5 – Пример изображений: контентного и стиля

Как результатом будет сгенерированное изображение (рисунок 1.6):

Рисунок 1.6 – Результат переноса стиля с помощью CNN

Данное изображение и является результатом нейронного переноса стиля.

Исходя из этого необходимо реализовать программу, исполняющую алгоритм нейронного переноса стиля, которая должна:

- загружать свёрточную нейронную сеть с возможностью последующей работы с её архитектурой;
- получать на вход два изображения: изображение контента и изображение стиля;
- сохранять итоговый результат, которым должно служить изображение полученное в ходе генерации и имеющее признаки обоих входных изображений.

Таким образом, было описано понятие свёрточной нейронной сети и её отличие от классической нейронной сети.

Так же были проанализированы существующие классические задачи, выполняемые свёрточными нейронными сетями. Для каждой задачи описана сфера применения, для которой разрабатываются конкретные CNN.

Приведенное описание и анализ существующих задач показал, что найдены не все сферы применения свёрточных нейронных сетей.

Исходя из полученных данных была сформулирована задача, выполнение которой подразумевает разрабатываемое программное обеспечение.

Глава 2 ПРОЕКТИРОВАНИЕ АЛГОРИТМА ПЕРЕНОСА НЕЙРОННОГО СТИЛЯ

2.1 Архитектура свёрточной нейронной сети VGG16

Основная задача, которую выполняет CNN, и на которую она была обучена, это классификация объектов на изображении, и чтобы она могла их классифицировать, нейронная сеть должна понимать само изображение. Это включает в себя использование необработанного изображения в качестве входных пикселей и построения внутреннего представления посредством преобразований, которые превращают необработанные пиксели изображения в сложные к пониманию функции, присутствующих в изображении [6]. То есть где-то между моментом когда, подаётся исходное изображение и выводится результат классификации, существует модель сложных признаков, называемые промежуточными слоями, используя которые мы можем описать содержание и стиль входного изображения.

Использовать будем свёрточную нейронную сеть VGG16 (Visual Geometry Group), модель которой представлена на рисунке 2.1.

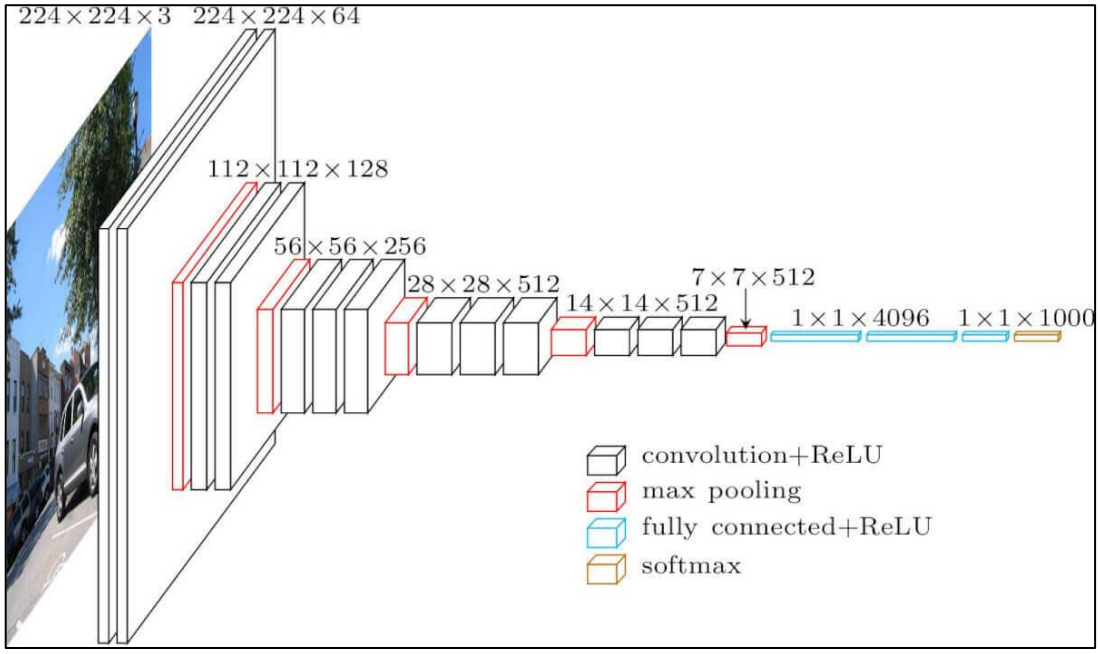

Рисунок 2.1 – Модель свёрточной нейронной сети VGG16

VGG16 – модель сверточной нейронной сети, предложенная K. Simonyan и A. Zisserman из Оксфордского университета в статье “Very Deep Convolutional Networks for Large-Scale Image Recognition”. Модель достигает точности 92.7% – топ-5, при тестировании на ImageNet в задаче распознавания объектов на изображении. Этот датасет состоит из более чем 14 миллионов изображений, принадлежащих к 1000 классам [9].

VGG16 – одна из самых знаменитых моделей, отправленных на соревнование ILSVRC-2014. Она является улучшенной версией AlexNet, в которой заменены большие фильтры (размера 11 и 5 в первом и втором сверточном слое, соответственно) на несколько фильтров размера 3x3, следующих один за другим. Сеть VGG16 обучалась на протяжении нескольких недель при использовании видеокарт NVIDIA TITAN BLACK [9].

Архитектура CNN VGG16 представлена на рисунке 2.2.

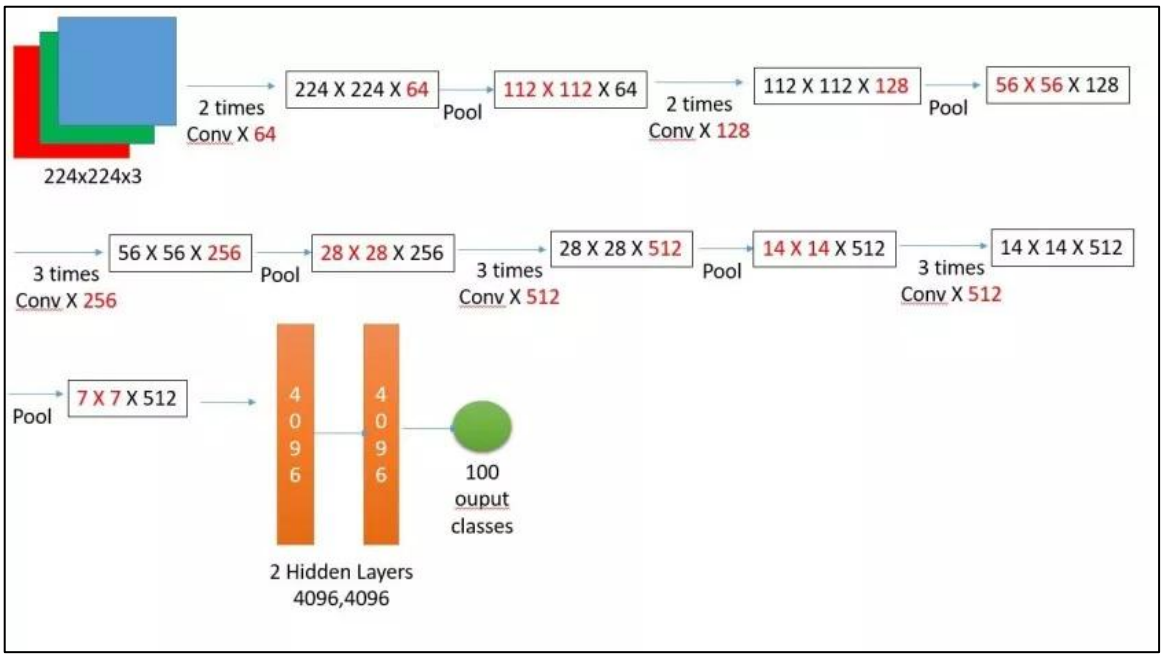

Рисунок 2.2 – Архитектура VGG16

Что же можно увидеть на данной архитектуре? Пойдём по порядку. На вход свёрточной нейронной сети подаётся RGB изображение размером 224x224 пикселя, которое как мы помним имеет 3 цветовых канала. Далее оно проходит через стек, свёрточных слоёв, которые реагируют на те или иные качества и характеристики изображения, с фильтрами, размер рецептивного поля которых

равен 3x3 – max-pooling слои, которые отвечают за процесс простой дискретизации изображения на основе выборки. Этот процесс состоит в том, чтобы выбрать максимальное число из подрегионов изображения, отсюда и название max. Пример данного процесса представлен на рисунке 2.2.

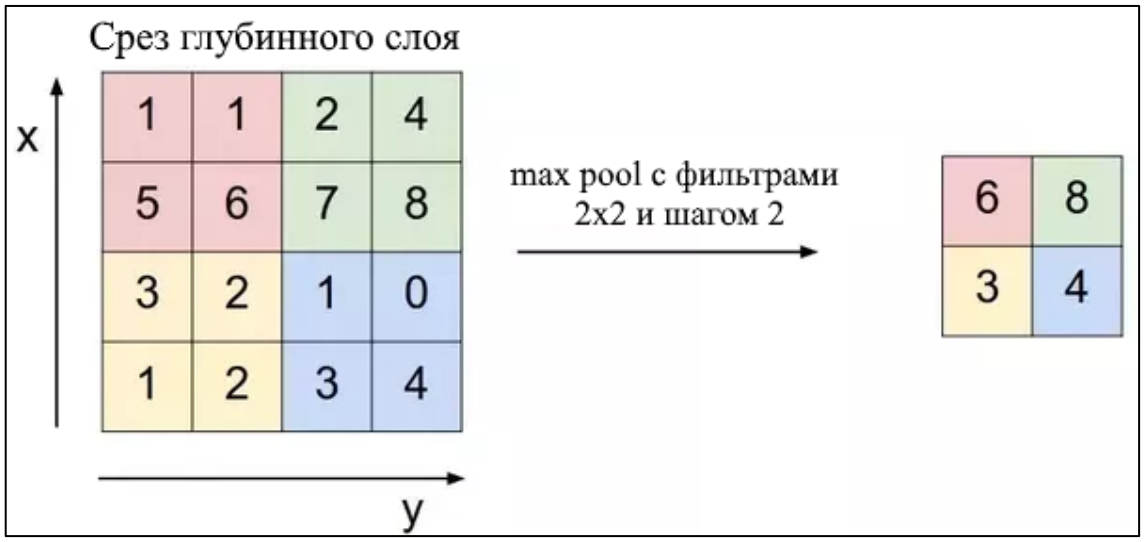

Рисунок 2.3 – Демонстрация работы max-pooling слоя

После стека сверточных слоёв идут 3 полносвязных слоя: первые два имеют по 4096 нейронов, третий – 1000 нейронов. Данные слои отвечают за классификацию, а третий (soft-max) слой как раз является результирующим слоем, нейроны которого и показывают отношение объекта на изображении к определенному классу. Стоит отметить, что данный слой зачастую применяется в машинном обучении, как раз для решения задач классификации, когда возможных классов больше двух.

Была выбрана сверточная нейронная сеть VGG16, потому она имеет следующие преимущества:

- Данная сеть уже обучена и реализована.
- Она имеет достаточно высокий показатель точности классификации объекта на изображении, что прямо говорит о её эффективности выделять содержательные и стилистические особенности изображения, не обращая внимания на шумы.
- Легка в реализации и использовании, то есть отлично подходит и для обучающих целей.

И к сожалению, сеть VGG имеет два серьёзных недостатка:

- Сама архитектура сети весит слишком много из-за глубины и количества полносвязных узлов.
- Очень медленная скорость обучения.

Хоть данные недостатки и существенные сами по себе, они не являются критериями, из-за которых можно было отказаться от использования этой сети в ходе выполнения данной бакалаврской работы. Так как использоваться будет уже обученная сеть, а вес не является значительным недостатком, который мог бы перекрыть плюсы данной сети.

2.2 Математический аппарат алгоритма нейронного переноса стиля

Процесс переноса нейронного стиля состоит из нескольких шагов, из которых в качестве основных можно выделить следующие:

- подготовка сверточной нейронной сети;
- подготовка исходных изображений и создание шумного;
- создание выходного изображения.

Далее опишем подробнее каждый из этих пунктов.

2.2.1 Подготовка сверточной нейронной сети

Подготовка сверточной нейронной сети подразумевает под собой обучение или загрузку этой самой сети, а именно всей её модели с весами каждого нейрона. С математической точки зрения сама нейронная сеть, а точнее её модель, это совокупность матриц каждого слоя этой нейронной сети. То есть имеется матрица слоёв свертки, матрица двух скрытых полносвязных слоёв и итоговый N-пространственный вектор.

Если с загрузкой нейронной сети все интуитивно понятно, мы берем модель и заносим необходимые значения весов, то, что же подразумевает обучение нейронной сети? Обучение нейронной сети – это ничто иное как решение многопараметрической задачи нелинейной оптимизации. Так же

обучающий процесс нейронной сети называют методом обратного распространения ошибки.

Метод обратного распространения ошибки можно разделить на 4 отдельных блока:

1. прямое распространение;
2. функцию потери;
3. обратное распространение;
4. обновление веса.

Во время прямого распространения, берётся тренировочное изображение — как помним, это матрица допустим 32x32x3 — и пропускается через всю сеть. В первом обучающем примере, так как все веса или значения фильтра были инициализированы случайным образом, выходным значением будет что-то вроде [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], то есть такое значение, которое не даст предпочтения какому-то определённому числу. Сеть с такими весами не может найти свойства базового уровня и не может обоснованно определить класс изображения. Это ведёт к функции потери. То, что мы используем – это обучающие данные. У таких данных есть и изображение и ярлык. Допустим, первое обучающее изображение – это цифра 3. Ярлыком изображения будет [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]. Функция потери может быть выражена по-разному, но часто используется СКО (среднеквадратическая ошибка) формула (2.1) [18].

$$E_{total} = \frac{1}{2}(target - output)^2 , \quad (2.1)$$

где target – реальность,

output – предсказание.

Теперь нам нужно выполнить обратное распространение через сеть, которое определяет, какие веса оказали большее влияние на потери, и найти способы, как их настроить, чтобы уменьшить потери. После того, как мы вычислим производную, перейдём к последнему этапу – обновлению весов. Возьмём все фильтровые веса и обновим их так, чтобы они менялись в направлении градиента уменьшения ошибки, по формуле (2.2) [18].

$$W = W_i - \eta \frac{dE_{total}}{dW}, \quad (2.2)$$

где W – вес нейрона,

W_i – изначальный вес,

η – размер шага обучения.

Фактически мы сводим к минимуму количество потерь, которые получаются в результате отработки сверточной нейронной сети. Со стороны математического анализа, мы минимизируем потери (ошибку), регулируя независимые переменные весов.

2.2.2 Подготовка исходных изображений и создание шумного

После того, как была подготовлена нейронная сеть методом её загрузки или обучения, необходимо загрузить в систему изображения. В нашем случае контентное и стиля. Как говорилось в предыдущей главе, представляющих из себя матрицы значений, которые в дальнейшем нормализуются, что поможет ускорить генерацию выходного изображения.

После этого мы создаём шумное изображение. Шумное изображение – комбинированное изображение из двух исходных, путём наложения изображения стиля на контентное с определенным процентом смешивания. Делается это в основном для ускорения получения хороших результатов. Конечно можно было бы использовать чисто шумное изображение, но как показывает практика, мы все равно придём к одному и тому же результату, но затратив на это на порядок больше времени. В последствии мы будем использовать именно данное шумное называемое комбинированным изображение.

2.2.3 Создание выходного изображения

После всех проделанных шагов остаётся просто создать новое изображение, которое при подаче в нейронную сеть в качестве входных данных генерирует более или менее те же значения активации, что и изображение

контента и стиля. Или еще проще, находить середину двух изображений редактируя комбинированное изображение.

Для нахождения этой середины необходимо высчитывать разницу между слоями активаций контентного изображения (или изображения стиля) и комбинированным. Для этого используются функции называемые функциями нахождения потерь изображений или функциями затрат (image cost functions).

В качестве слоя активации контентного изображения мы будем брать один из высокоуровневых свёрточных слоёв VGG16, например conv4_2. Связано это с желанием сохранить объекты с контентного изображения. А выполним это благодаря функции квадратичного различия контентного изображения от комбинированного, представленная на формуле (2.3), которая поможет нам перенести объекты захваченные выбранным нами слоем активации из контентного изображения в комбинированное.

$$L_{content}\ p, x, l\ = \frac{1}{2}\ \ _{i,j}(F_{ij}^{l} - P_{ij}^{l})^2\ , \tag{2.3}$$

где F_{ij}^{l} – значения активаций слоя комбинированного изображения,

P_{ij}^{l} – значения активаций слоя контентного изображения.

По сути, это просто Евклидово расстояние между двумя активациями в конкретном слое. И в идеале мы ждем, что эта разница будет равна нулю.

Далее необходимо вычислить разницу для изображения стиля. Концептуально оно не отличается от функции потери контентного изображения, но разница есть и заключается она в том, как обрабатывается и вычисляются значения активаций.

Стиль можно определить, как корреляцию между каждым из каналов в выбранном слое. Например, если имеется блок 24x24x16, то, если мы возьмем пятый канал, то все 24x24 = 576 значений пятого канала будут сопоставимы с каждым значением остальных пятнадцати каналов. В математике это ничто иное как матрица Грама и рассчитывается в данном случае как перемножение значений всех каналов по формуле (2.4).

$$G_{ij}^{l} =\ \ _{k}(F_{ik}^{l}F_{jk}^{l})\ , \tag{2.4}$$

где l – номер выбранного слоя,

k – индекс канала в слое, который не перебирается, так как мы сравниваем его с остальными каналами,

i и j – индексы выбранного блока.

Теперь когда у нас имеется матрица Грама для изображения стиля, необходимо провести ту же операцию что и с контентным изображением: посчитаем Евклидовое расстояние между матрицами Грама слоёв активации стиля и комбинированного изображений по формуле (2.5).

$$E_l = \frac{1}{4N_i^2 M_i^2} \quad _k(G_{ij}^l - A_{ij}^l)\,, \tag{2.5}$$

где l – номер выбранного слоя,

G_{ij}^l – значения активаций слоя комбинированного изображения,

A_{ij}^l – значения активаций слоя контентного изображения,

N – количество фильтров на слое l,

M – высота перемноженная с шириной слоя l.

Этого вполне достаточно для вычисления потерь изображения стиля, но на практике для получения лучшего результата, используется в расчетах не один слой активации, а несколько. Все что необходимо сделать, просто сложить их вычисленные потери. Пример представлен на формуле (2.6).

$$L_{style}\ a, x \ = \ \ _{l=0}^{L} w_l E_l, \tag{2.6}$$

где l – номер слоя,

E_l – результат формулы (2.5) для слоя l,

w_l – вес слоя, благодаря которому мы можем контролировать вклад данного слоя в конечный результат.

Последней функцией вычисления разницы изображений будет функция общих затрат (total cost function). Общая потеря измеряет, как исходное изображение отличается от содержимого изображения стиля. Чтобы иметь контроль над тем, как мы хотим, чтобы наше комбинированное изображение выглядело: как контент или как стиль, вводятся две переменные: α и β. Вычисляется по формуле (2.7).

$$L_{total}\ p, a, x\ = \alpha L_{content}\ p, x\ + \beta L_{style}\ a, x\ , \quad (2.7)$$

Наконец, имея на руках отличный способ измерить схожесть комбинированного изображения с исходными. Нам необходимо обновить его так, чтобы в следующий раз полученное изображение имело меньшую разницу, другими словами меньшее значение функции потерь. Получается, что шаг за шагом мы будем обновлять комбинированное изображение, чтобы оно становилось все больше похожим как на изображение контента так и на изображения стиля.

Обновляем его посредством вычета из функции потерь значение деривации (отклонения). Само значение деривации просто говорит нам, в какую сторону стоит изменить изображение. Данный процесс называется минимизацией затрат (minimize cost function). В результате мы имеем ничто иное как стохастический градиентный спуск, который стремиться к нулю. Для этого будем использовать Adam (Adaptive Moment Estimation) – метод, который вычисляет адаптивные скорости обучения для каждого параметра.

2.3 Разработка программного обеспечения

На текущий момент мы имеем полный математический аппарат разрабатываемого программного обеспечения. Теперь необходимо реализовать алгоритм, который будет решать поставленную перед ним задачу.

Для этих целей был выбран объектно-ориентированный язык Java. Чтобы не реализовывать низкоуровневые интерфейсы для работы с матрицами и другими математическими объектами было решено воспользоваться фреймворком deeplearning4j, с помощью которого так же можно загружать обученную архитектуру свёрточной нейронной сети VGG16.

Начнем с написания основных алгоритмов поиска разницы изображений. Код алгоритма реализации задачи нахождения потерь контентного изображения имеет вид, представленный в листинге 1.

Листинг 1. Код алгоритма нахождения потерь контентного изображения

```
    private double contentLoss(INDArray combActivations, INDArray
contentActivations) {
        return sumOfSquaredErrors(contentActivations, combActivations) / (4.0 *
(CHANNELS) * (WIDTH) * (HEIGHT));
    }
    /**
     * Квадратичная сумма ошибки.
     */
    private double sumOfSquaredErrors(INDArray a, INDArray b) {
        INDArray diff = a.sub(b); // разница
        INDArray squares = Transforms.pow(diff, 2); // поэлементное
возведение в квадрат
        return squares.sumNumber().doubleValue();
    }
```

Далее на листинге 2 представлен код реализующий нахождение потерь для изображения стиля.

Листинг 2. Код алгоритма нахождения потерь для изображения стиля

```
    private double styleLoss(INDArray style, INDArray combination) {
        INDArray styleGram = gramMatrix(style);
        INDArray combinationGram = gramMatrix(combination);
        long[] shape = style.shape();
        long N = shape[0]; // количество фильтров на слое
        long M = shape[1] * shape[2]; // высота перемноженная с шириной
слоя
        return sumOfSquaredErrors(styleGram, combinationGram) / (4.0 * (N *
N) * (M * M));
    }
```

На листинге 3 представлен код функции, который реализует нахождение потерь для всех выбранных нами слоёв

Листинг 3. Код алгоритма нахождения потерь для всех выбранных слоёв стиля

```
        // Выбранные слои стиля
        private static final String[] STYLE_LAYERS = new String[]{
                "block1_conv1,0.5",
                "block2_conv1,1.0",
                "block3_conv1,1.5",
                "block4_conv2,3.0",
                "block5_conv1,4.0"
        };


        private Double allStyleLayersLoss(Map<String, INDArray>
activationsStyleMap, Map<String, INDArray> activationsCombMap) {
          Double styles = 0.0;
          for (String styleLayers : STYLE_LAYERS) {
            String[] split = styleLayers.split(",");
            String styleLayerName = split[0];
            double weight = Double.parseDouble(split[1]);
            styles += styleLoss(activationsStyleMap.get(styleLayerName).dup(),
        activationsCombMap.get(styleLayerName).dup()) * weight;
          }
          return styles;
        }
```

Из функций потерь остаётся только написание функции поиска общих затрат изображения контента и изображения стиля, код которой представлен на листинге 4.

Листинг 4. Код алгоритма нахождения общих затрат изображений контента и стиля.

```
    private double totalLoss(Map<String, INDArray> activationsStyleMap,
Map<String, INDArray>activationsCombMap, Map<String, INDArray>
activationsContentMap) {
        Double stylesLoss = allStyleLayersLoss(activationsStyleMap,
activationsCombMap);
         return ALPHA *
contentLoss(activationsCombMap.get(CONTENT_LAYER).dup(),
activationsContentMap.get(CONTENT_LAYER).dup()) + BETA * stylesLoss;
    }
```

Регулируя ALPHA и BETA в коде приложения, можно добиться необходимых результатов по соотношению найденных черт объекта контентного с необходимым стилем.

После того, как были реализованы основные функции нахождения затрат, необходимо загрузить саму свёрточную нейронную сеть. Делается это просто благодаря выбранному ранее фреймворку, Реализация функции загрузки обученной сверточной нейронной сети VGG16 представлена на листинге 5.

Листинг 5. Код алгоритма загрузки обученной свёрточной нейронной сети VGG16.

```
    private ComputationGraph loadModel() throws IOException {
      ZooModel zooModel = VGG16.builder().build();
      ComputationGraph vgg16 =
zooModel.initPretrained(PretrainedType.IMAGENET);
      vgg16.initGradientsView();
      log.info(vgg16.summary());
      return vgg16;
    }
```

Теперь всё готово для реализации алгоритма реализующего генерацию изображения. Код алгоритма приведен на листинге 6.

Листинг 6. Код алгоритма реализующего перенос стиля с изображения стиля на изображение контента.

```
    private void transferStyle() throws IOException {
        ComputationGraph vgg16FineTune = loadModel();
        INDArray content = loadImage(IMAGE_CONTENT);
        INDArray style = loadImage(IMAGE_STYLE);
        INDArray combination = createCombinationImage();
        Map<String, INDArray> activationsContentMap = vgg16FineTune.feedForward(content, true);
        Map<String, INDArray> activationsStyleMap = vgg16FineTune.feedForward(style, true);
        HashMap<String, INDArray> activationsStyleGramMap = buildStyleGramValues(activationsStyleMap);
        AdamUpdater adamUpdater = createADAMUpdater();

        for (int iteration = 0; iteration < ITERATIONS; iteration++) {
          log.info("iteration  " + iteration);

          INDArray[] input = new INDArray[]{combination};
          Map<String, INDArray> activationsCombMap = vgg16FineTune.feedForward(input, true, false);

          INDArray styleBackProb = backPropagateStyles(vgg16FineTune, activationsStyleGramMap, activationsCombMap);
          INDArray backPropContent = backPropagateContent(vgg16FineTune, activationsContentMap, activationsCombMap);
          INDArray backPropAllValues = backPropContent.muli(ALPHA).addi(styleBackProb.muli(BETA));
```

```
                adamUpdater.applyUpdater(backPropAllValues, iteration, 0);
                combination.subi(backPropAllValues);

                log.info("Total Loss: " + totalLoss(activationsStyleMap,
activationsCombMap, activationsContentMap));
                if (iteration % SAVE_IMAGE_CHECKPOINT == 0) {
                    saveImage(combination.dup(), iteration);
                }
            }
        }
```

Данный алгоритм выполняет шаги, описанные в предыдущем параграфе при создании математического аппарат алгоритма, а именно:

1) загружает и подготавливает CNN VGG16;
2) загружает изображения в виде матриц;
3) создаёт комбинированное изображение;
4) получает матрицы слоёв активации контента, стиля и матрицу стиля Грама;
5) создаёт комбинированное изображение необходимое количество раз, уменьшая разницу с каждой итерацией.

Таким образом было описана сверточная нейронная сеть VGG16, её архитектура и ход работы.

Было проведено проектирование математического аппарата алгоритма нейронного переноса стиля и его последующая реализация. Так как работа происходит в основном с матрицами сложность алгоритма – квадратичная.

Данная программа позволяет создать новое изображение с помощью классифицирующей свёрточной нейронной сети VGG16 с желаемым содержимым, взятым с изображения контента, и понравившимся стилем рисования, аналогично взятым с изображения стиля. Минусом данного алгоритма является его долгая обрабатывание на слабых процессорах и низкое разрешение выходного изображения (224 x 224 пикселей).

ГЛАВА 3 ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО РЕШЕНИЯ

-

3.1 Проведение экспериментов генерации изображения

Ключевыми параметрами алгоритма нейронного переноса стиля являются: скорость и качество генерации выходного изображения.

Под скоростью понимается количество итерации до достижения приемлемого результата, так как скорость выполнения оных будет зависеть от мощности ЦП компьютера. Но чтобы иметь общее представление о количестве затрачиваемого времени на генерацию одного изображение будем фиксировать время исполнения каждой итерации. В свою очередь качество – признак не определенный, с одной стороны зависит от мнения оценивающего конечный результат человека (эксперта), но с другой ранее было сказано, что алгоритм с каждой итерацией идет к равному соотношению контентного изображения и изображения стиля по параметром, которые для него задаются, то есть качество изображения, для алгоритма, измеряется в коэффициенте разницы, поэтому, чтобы не отталкиваться от абстрактных величин, для оценивания качества, будем брать именно этот параметр. Для принятия решения о результате разработанного алгоритма необходимо провести ряд тестов, которые позволят проверить работоспособность приложения в реальных условиях.

Чтобы оценить работу алгоритма по необходимым критериям, было проведено 3 теста с разными парами изображений контента и стиля. Во время каждого теста было зафиксировано время выполнения (Time spent), общая разница (Total Loss) и сохранено сгенерированное изображение на каждой 50-ой итерации.

Первый тест проводился с изображениями представленными на рисунке 3.1. Были выбраны контентное изображение и изображение стиля с одинаковым размером разрешения, а именно данные изображению имеют размеры 800 x 600 точек.

Рисунок 3.1 – Используемые изображения при первом тесте.

За время тестирования было пройдено 1000 итераций. И сгенерировано 21 изображение. Некоторые из сгенерированных изображений, а именно с итераций: 100, 250, 500 и 1000, в качестве результата представлены на рисунке 3.2.

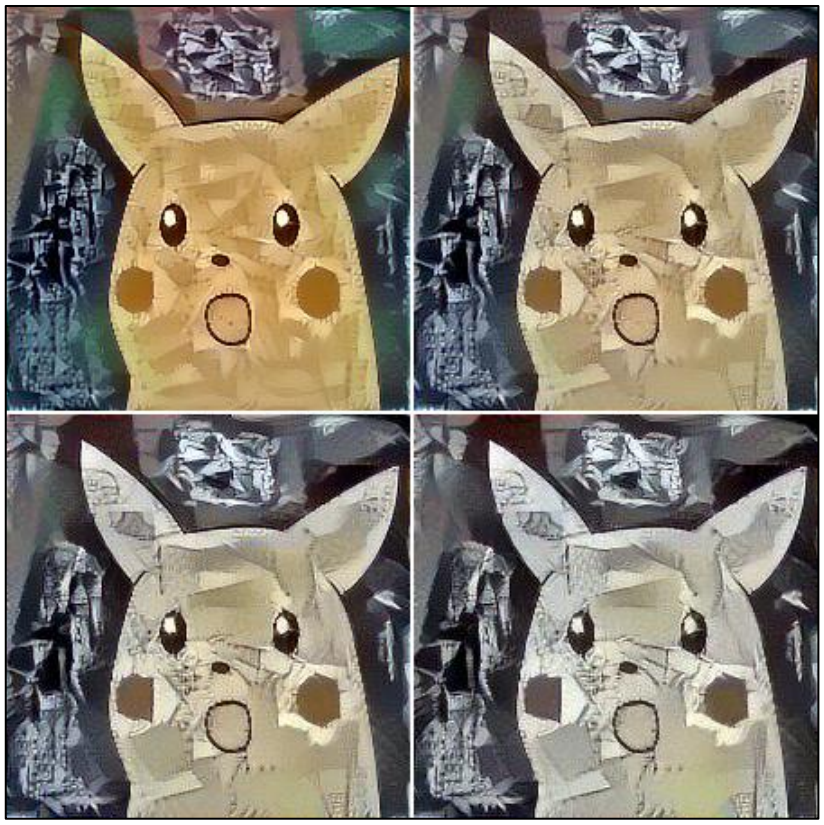

Рисунок 3.2 –Сгенерированные изображения во время первого теста на 100, 250, 500 и 1000 итераций соответственно.

На данном рисунке можно наблюдать процесс генерации изображения. Визуально уже видно, что за первые 250 итераций была выполнена самая массивная работа. Результаты в числовом виде представлены на рисунке 3.3.

№ Итерации	Общая разница	Время (сек.)
1	1,57E+14	12
51	1,32E+13	11
101	3,84E+12	11
151	1,91E+12	11
201	1,19E+12	11
251	8,40E+11	11
301	6,37E+11	11
351	5,10E+11	11
401	4,23E+11	11
451	3,61E+11	12
501	3,16E+11	12
551	2,80E+11	11
601	2,51E+11	11
651	2,28E+11	11
701	2,09E+11	11
751	1,93E+11	11
801	1,79E+11	11
851	1,68E+11	11
901	1,57E+11	11
951	1,48E+11	11
1001	1,40E+11	11

Рисунок 3.3 – Результаты первого тестирования.

На рисунке с результатами первого тестирования видно, что время затраченное на каждую итерацию – линейно и в основном составляет 11 секунд на каждую итерацию. Это позволяет посчитать примерное суммарное время генерации изображения, а это примерно 3 часа.

Так же из результатов можно видеть, что самое быстрое приближение общей разницы к нулю произошло за первую четверть проделанных итераций. Вероятно связано с тем, что за первую сотню итераций выстраивается контур, взятый со стилевого изображения. Так же это хорошо видно на графике, представленном на рисунке 3.4.

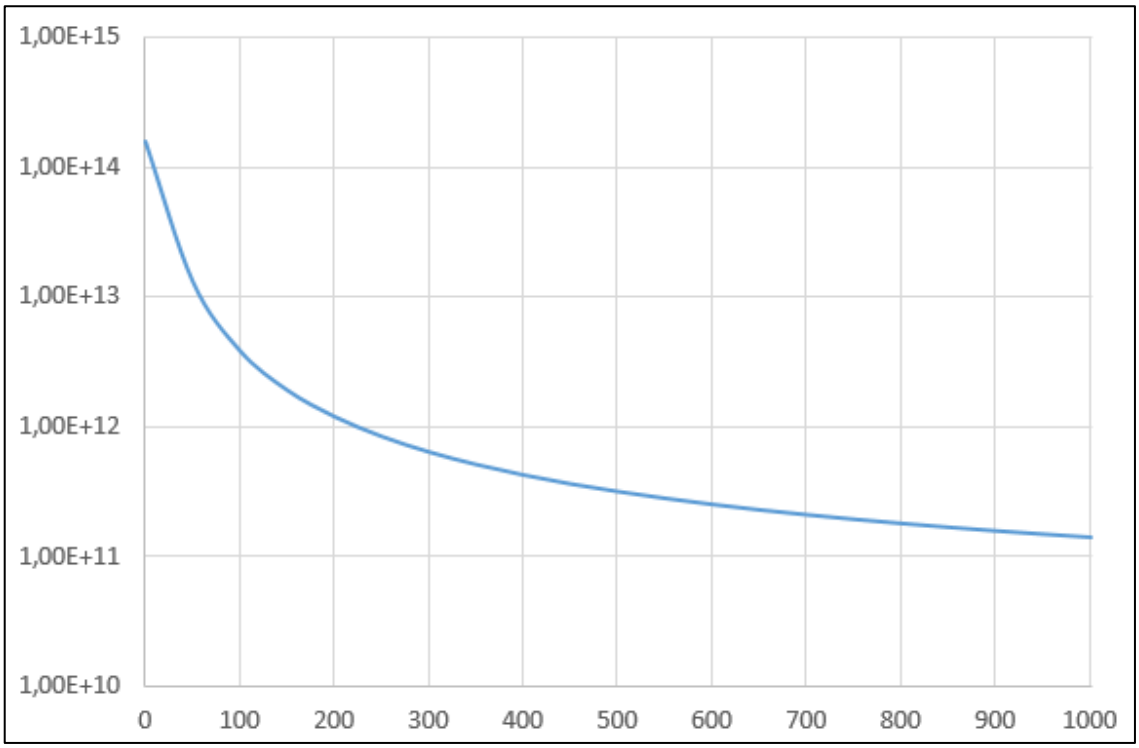

Рисунок 3.4 – График общей разницы по итерациям первого теста.

После просмотра на данный график появляется естественное желание найти функцию, чтобы в дальнейшем экстраполировать её и узнать, за какое количество итераций мы сможем получить «идеальное» изображение, то есть когда общая разница будет максимально приближена к 0. Но не стоит торопится, лучше продолжить тестирование алгоритма.

Второй тест был проведен с изображениями представленными на рисунке 3.5. Размеры которых отличаются. Контентное имеет 450 x 300 точек, а изображение стиля 900 x 600. Это было сделано с целью проверки влияния размеров изображений на скорость выполнения итераций.

Рисунок 3.5 – Используемые изображения при втором тесте.

Аналогично предыдущему во время данного тестирования было выполнено 1000 итераций. Результаты сгенерированных изображений с 100, 250, 500 и 1000 итераций приведены на рисунке 3.6.

Рисунок 3.6 – Сгенерированные изображения во время второго теста на 100, 250, 500 и 1000 итераций соответственно.

Аналогично с предыдущим тестом можно наблюдать ситуацию, в которой за первую четверть прохождений была выполнена основная работа по созданию изображения. За остальное время алгоритм вносил минимальные правки в контуры, а больше времени уделил правке цветов конечного изображения, но уже в текущем тесте исключением является последний кадр, на котором можно наблюдать уклон в более глубокую проработку контуров в правых вернем и нижнем углах.

Это так же хорошо видно в результатах тестирования представленных на рисунке 3.7.

№ Итерации	Общая разница	Время (сек.)
1	4,20E+13	12
51	1,18E+12	13
101	4,92E+11	12
151	2,94E+11	12
201	2,08E+11	12
251	1,60E+11	11
301	1,30E+11	11
351	1,10E+11	12
401	9,49E+10	12
451	8,37E+10	12
501	7,50E+10	11
551	6,79E+10	12
601	6,20E+10	12
651	5,71E+10	12
701	5,29E+10	13
751	5,13E+10	12
801	4,78E+10	12
851	4,42E+10	11
901	4,69E+10	11
951	3,96E+10	11
1001	5,30E+10	11

Рисунок 3.7 – Результаты второго тестирования.

Как можно заметить на рисунке с результатами второго тестирования скорость выполнения каждой итерации работы алгоритма в среднем выше на 1 секунду, по сравнению с предыдущим тестированием.

Второе на что стоит обратить внимание – резкие скачки общей разницы при выполнении 900 – 1000 итераций, которые хорошо видно на графике представленному на рисунке 3.8.

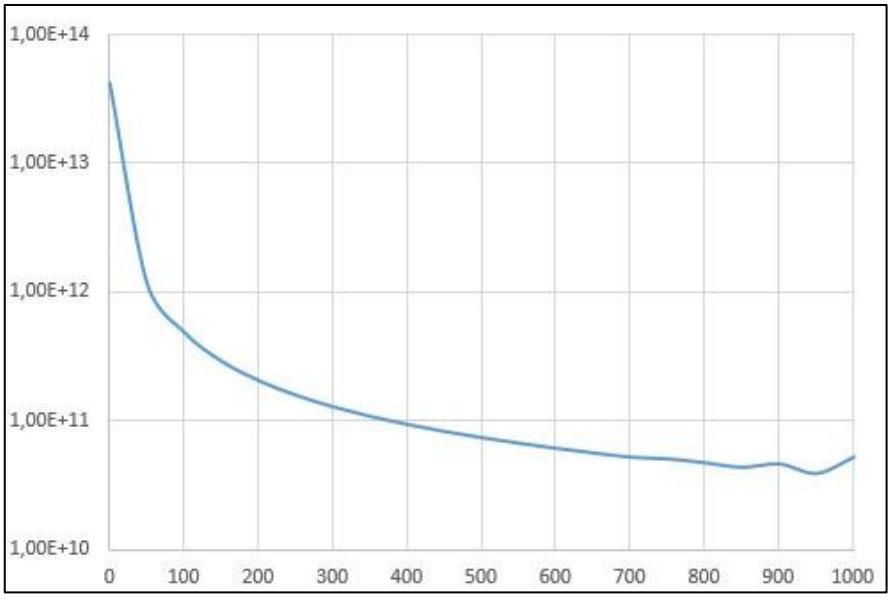

Рисунок 3.8 – График общей разницы по итерациям второго теста.

Проведем третье и конечное тестирование алгоритма нейронного переноса стиля. Если до этого выбирались нарисованное изображение и фотография техники, то для этого теста было решено в качестве контентного изображения выбрать фотографию человека, а в качестве изображения стиля популярная картина «Мона Лиза», художником которой является Леонардо да Винчи. Их размеры 650 x 650 и 800 x 1000 точек соответственно. Выбранные изображения представлены на рисунке 3.9.

Рисунок 3.9 – Используемые изображения при втором тесте.

За время третьего тестирования было пройдено 5000 итераций. И сгенерировано 101 изображение. Время продолжительности выполнения одной итерации аналогично составляет 12 секунд. Некоторые из сгенерированных изображений представлены на рисунке 3.10, а именно с итераций: 250, 1000, 2100, 2600, 3150 и 5000.

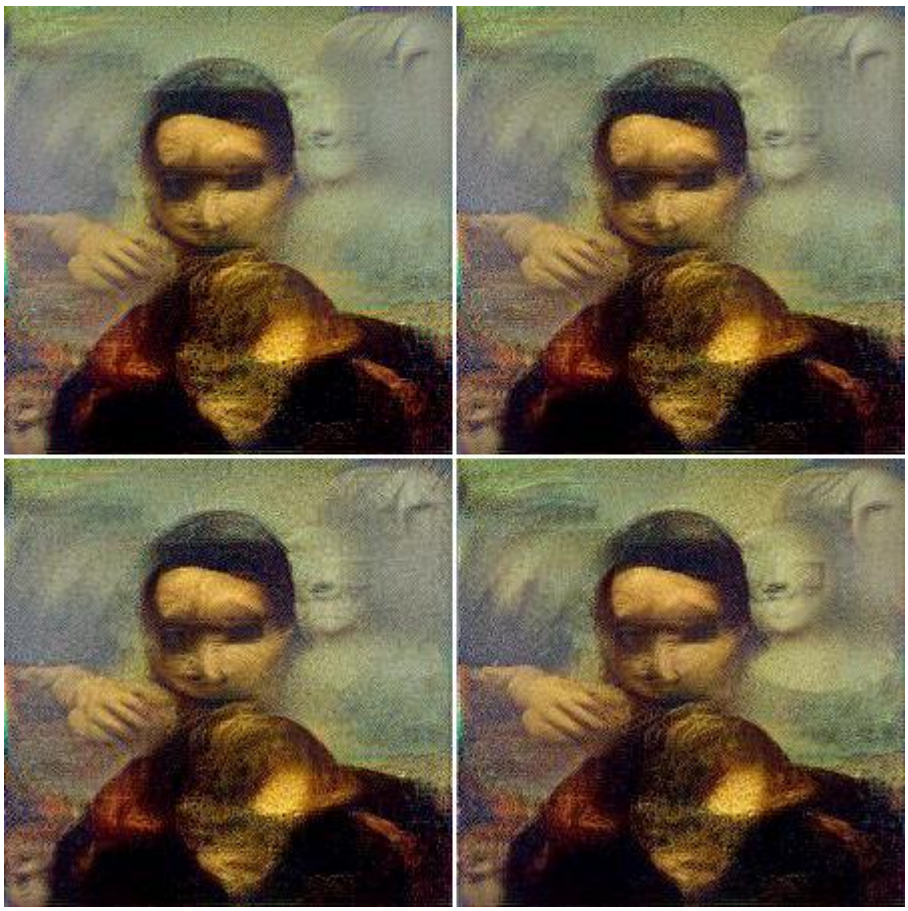

Рисунок 3.10 – Сгенерированные изображения во время третьего теста на 250, 1000, 2100, 2600, 3150 и 5000 итерациях соответственно.

Прежде чем делать какие-либо выводы по работе алгоритма во время третьего тестирования, стоит составить график результатов общей разницы на каждой итерации. Таблица с результатами, по которой был составлен график, представлена в приложении 1. Сам график представлен на рисунке 3.11.

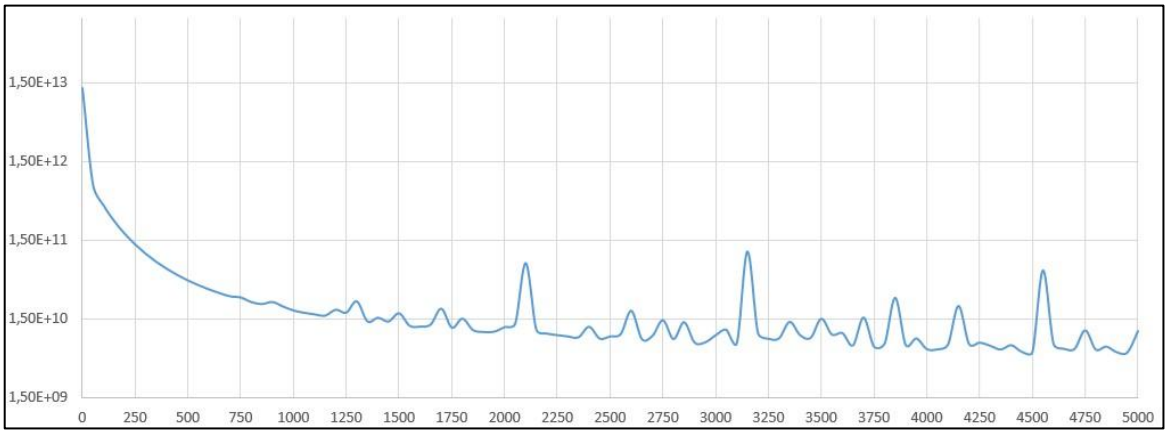

Рисунок 3.11 – График общей разницы по итерациям третьего теста.

Какие выводы можно сделать, посмотрев на сгенерированные изображения и график?

Во-первых, если алгоритм нейронного переноса стиля более-менее неплохо справлялся с переносом стиля с абстрактных картин, то с переносом стиля с портретов он справляется отнюдь плохо, именно поэтому признак качества был ранее назван «не определенным».

Во-вторых, при приближении к планке в 1000 итераций, алгоритм находит свой предел и начинает колебаться, и если в некоторых случаях это от части полезно, как было например во время второго тестирования, то в некоторых приводит еще к более худшему результату, что видно на результатах третьего тестирования.

Подводя итоги всех тестирований можно выявить некоторые закономерности, а именно:

- 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;
- из-за специфики архитектуры свёрточной сети VGG16, связанной с сжатием входных изображений, время затрачиваемое на одну итерацию остаётся константным с небольшими отклонениями, с высокой вероятностью возникающими из-за фоновой параллельной загрузки ЦП компьютера;
- среднее время, которое было затрачено на одну итерацию во время тестирований, равняется 12 секундам, что в совокупности с предыдущими пунктами, в сумме даёт 50 минут на получение более менее приемлемого варианта или 3 часа для хорошего;
- алгоритм лучше работает с входными изображениями меньшего разрешения, так как затрачивает меньшее количество итераций до достижения приемлемого результата.

3.2 Корректировка разработанной системы для улучшения результатов

Исходя из итогов тестирования, можно выделить следующие пути улучшения разработанного решения, а именно:

1. Повышение качества выходного изображения за счет увеличения его разрешения.
2. Уменьшение количества итераций для достижения приемлемого и хорошего по качеству вариантов конечного изображения.
3. Уменьшение затрачиваемого времени на одну итерацию.
4. А так же исправление небольшого минуса связанного с размером сверточной нейронной сети.

Повышение качества за счет увеличения разрешения можно добиться путем выбора другой сверточной нейронной сети, например VGG19, что в свою очередь приведет к тому, что при большем размере выходного изображения, входные изображения будут большего размера, а следовательно CNN сможет более корректно выделить уникальные черты как контентного изображения, так и изображения стиля. Но за этим последует и большее количество необходимых просчетов, что скажутся на скорости выполнения одной итерации. При увеличении разрешения обрабатываемых изображений в 2 раза, то количество времени затрачиваемого на одну итерацию увеличиться в 4 раза. Исходя из тестов будет затрачено не 12 секунд, а 48.

Уменьшения количества итерации для достижения приемлемого результата так же можно достичь выбором другой сверточной нейронной сети, архитектура которой это позволяет, что в свою очередь может привести к более худшему результату по оценке эксперта, так как сверточная сеть VGG16 является одной из наиболее точных сверточных сетей. Данный способ так же позволит решить проблему с весом конечного решения.

Уменьшение затрачиваемого времени на одну итерацию можно добиться путем выполнения части работы параллельно.

Проанализировав математический аппарат алгоритма и разработанное решение были выявлен участок кода, который можно выполнять параллельно, представленный на листинге 7.

Листинг 7. Выявленный участок кода, который можно распараллелить.

```
INDArray styleBackProb = backPropagateStyles(vgg16FineTune, activationsStyleGramMap, activationsCombMap);
INDArray backPropContent = backPropagateContent(vgg16FineTune, activationsContentMap, activationsCombMap);
```

- Данный участок отвечает за подсчет значений разницы со слоёв активации изображений.
- Измененный участок кода представлен на листинге 8.

Листинг 8. Распараллеленные операции с помощью Java Concurrent.

```
ExecutorService exec = Executors.newFixedThreadPool(4);

FutureTask<INDArray> styleBackProbTask = new FutureTask<>(() -> backPropagateStyles(vgg16FineTune, activationsStyleGramMap, activationsCombMap));
FutureTask<INDArray> backPropContentTask = new FutureTask<>(() -> backPropagateContent(vgg16FineTune, activationsContentMap, activationsCombMap));

exec.execute(styleBackProbTask);
exec.execute(backPropContentTask);
```

Проведем повторное тестирование с изображениями первого теста с целью замера времени выполнения хотя бы первых 50 итераций. Этого должно хватить для подведения итогов, так как ранее уже было выявлено, что время выполнения линейно и держится в пределах одних значений.

Результаты тестирования представлены на рисунке 3.12.

№ Итерации	Общая разница	Время (сек.)
1	7,85E+12	9
6	4,84E+12	10
11	3,50E+12	9
16	2,71E+12	9
21	2,15E+12	9
26	1,74E+12	10
31	1,42E+12	9
36	1,16E+12	9
41	9,48E+11	9
46	7,86E+11	9
51	6,58E+11	10

Рисунок 3.12 – Результаты повторного тестирования.

Как можно наблюдать из результатов время выполнения уменьшилось вплоть до 9 секунд, что на 15% меньше по сравнению с соответствующим тестированием.

Еще вариантом для уменьшения затрачиваемого времени на одну итерацию является переводом разработанного математического аппарата алгоритма на решение с использованием графического адаптера, например с помощью технологии CUDA. В данном случае скорость вычисления может повыситься в 7 и более раз[10].

Таким образом было проведено три тестирования разработанного решения. Представлены результаты выполнения системы и созданные изображения. Выявлены его недостатки, проведена корректировка алгоритма путем распараллеливания части операций и предложены пути решения оставшихся проблем.

ЗАКЛЮЧЕНИЕ

Выполненная работа посвящена исследованию свёрточных нейронных сетей и разработке нового способа их использования. Цель работы определила её основное направление – разработка алгоритма нейронного переноса стиля с использованием свёрточной нейронной сети для создания изображений имея на входе пару необходимых изображений контента и стиля.

Для этого были рассмотрены и проанализированы способы использования свёрточных нейронных сетей в современных реалиях. В ходе анализа была выявлена область, где свёрточные сети малоприменимы, и была сформулирована новая задача и пути её реализации.

В процессе разработки были реализованы:

- математический аппарат алгоритма нейронного переноса стиля;
- алгоритм по нейронному переносу стиля изображения.

К тому же были выявлены недостатки разработанного алгоритма, предложены способы по их устранению с описанием возможных последствий.

Данная система позволяет создать изображение, имея на входе одно исходное изображение и второе изображения стиля, беря черты первого и стиль рисования второго. Система достаточно требовательна к вычислительной мощности компьютера, так как подразумевает выполнение многочисленных математических операции, и в меньшей мере к оперативной памяти компьютера для загрузки в неё архитектуры свёрточной нейронной сети.

Полученный алгоритм был протестирован, были продемонстрированы результаты трёх генераций разного уровня сложности и выявлены недостатки.

В результате работы были продемонстрированы пути развития данной работы такие как: уменьшение времени генерации изображения и повышение качества создаваемого изображения.

Данная работа продемонстрировала, что при необходимом желании можно найти применение свёрточным нейронным сетям в сфере, для которой они изначально не предназначались.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [Электронный ресурс] Режим доступа: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

2. Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation. DeepLearning. [Электронный ресурс] Режим доступа: http://deeplearning.net/tutorial/lenet.html

3. Convolutional Neural Networks for Visual Recognition [Электронный ресурс] Режим доступа: http://cs231n.github.io/convolutional-networks/

4. How to do Semantic Segmentation using Deep learning [Электронный ресурс] Режим доступа: https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef

5. Just look at the image: viewpoint-specific surface normal prediction for improved multi-view reconstruction [Электронный ресурс] Режим доступа: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Galliani_Just_Look_at_CVPR_2016_paper.pdf

6. Learning Deep Features for Discriminative Localization [Электронный ресурс] Режим доступа: http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf

7. M.D. Zeiler. Visualizing and Understanding Convolutional Networks [Электрон. ресурс] / M.D. Zeiler, R. Fergus. - Режим доступа: https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

8. Neural Style Transfer: Creating Art with Deep Learning using tf.keras and eager execution [Электронный ресурс] Режим доступа: https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-and-eager-execution-7d541ac31398

9. Predicting Video Saliency with Object-to-Motion CNN and Two-layer Convolutional LSTM [Электронный ресурс] Режим доступа: https://arxiv.org/pdf/1709.06316.pdf

10. The history of Neural networks [Электронный ресурс] Режим доступа: https://dataconomy.com/2017/04/history-neural-networks/

11. VGG16 — сверточная сеть для выделения признаков изображений [Электронный ресурс] Режим доступа: https://neurohive.io/ru/vidy-nejrosetej/vgg16-model/

12. Использование GPU для решения задач компьютерного зрения в библиотеке OpenCV [Электронный ресурс] Режим доступа: http://agora.guru.ru/hpc-h/files/Using_GPU_for_computer_vision_in_OpenCV_library.pdf

13. Как работает сверточная нейронная сеть: архитектура, примеры, особенности [Электронный ресурс] Режим доступа: https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/

14. Как работает сверхглубокая нейронная сеть? [Электронный ресурс] Режим доступа: https://postnauka.ru/video/66872

15. Локализация объектов на изображении методом свёрточных нейронных сетей [Электронный ресурс] Режим доступа: http://www.azoft.ru/blog/lokalizaciya-obektov-na-izobrazhenii-metodom-svyortochnyx-nejronnyx-setej/

16. Милютин И. VGG16 — сверточная сеть для выделения признаков изображений [Электронный ресурс] / И. Милютин. - Режим доступа: https://neurohive.io/ru/vidy-nejrosetej/vgg16-model

17. Нейронные сети: полный курс, 2-е издание. : Пер. с англ. – М. : Издательский дом “Вильямс, 2006. 1104 с. : ил. Парал. тит. англ.”

18. Нейронные сети: практическое применение [Электронный ресурс] Режим доступа: https://habr.com/ru/post/322392/

19. Что такое машинное зрение и чем оно отличается от человеческого? [Электронный ресурс] Режим доступа: https://meduza.io/feature/2019/03/30/chto-

takoe-mashinnoe-zrenie-i-chem-ono-otlichaetsya-ot-chelovecheskogo-seychas-ob-yasnim-ponyatno

20. Что такое свёрточная нейронная сеть [Электронный ресурс] Режим доступа: https://habr.com/ru/post/309508/

ПРИЛОЖЕНИЕ А

Результаты третьего тестирования

№ Итерации	Общая ра	Время (се	№ Итера	Общая ра	Время (се	№ Итера	Общая ра	Время (се	№ Итера	Общая ра	Время (се
1	1,29E+13	12	1301	2,52E+10	13	2601	1,92E+10	13	3901	7,04E+09	12
51	7,67E+11	12	1351	1,41E+10	13	2651	8,39E+09	13	3951	8,55E+09	12
101	4,20E+11	12	1401	1,56E+10	13	2701	9,21E+09	13	4001	6,25E+09	12
151	2,66E+11	12	1451	1,40E+10	13	2751	1,46E+10	13	4051	6,21E+09	12
201	1,83E+11	12	1501	1,78E+10	13	2801	8,37E+09	13	4101	7,13E+09	12
251	1,33E+11	12	1551	1,24E+10	13	2851	1,37E+10	13	4151	2,21E+10	12
301	1,01E+11	12	1601	1,21E+10	13	2901	7,56E+09	13	4201	7,24E+09	12
351	8,01E+10	12	1651	1,28E+10	13	2951	7,62E+09	13	4251	7,55E+09	12
401	6,52E+10	12	1701	2,04E+10	13	3001	9,40E+09	13	4301	6,89E+09	12
451	5,45E+10	12	1751	1,17E+10	13	3051	1,10E+10	13	4351	6,18E+09	12
501	4,65E+10	12	1801	1,52E+10	13	3101	7,47E+09	13	4401	7,00E+09	12
551	4,05E+10	12	1851	1,09E+10	13	3151	1,09E+11	13	4451	5,77E+09	12
601	3,58E+10	12	1901	1,03E+10	13	3201	9,99E+09	13	4501	5,71E+09	12
651	3,21E+10	12	1951	1,04E+10	13	3251	8,47E+09	13	4551	6,28E+10	12
701	2,92E+10	12	2001	1,19E+10	13	3301	8,58E+09	12	4601	7,30E+09	12
751	2,83E+10	12	2051	1,31E+10	13	3351	1,39E+10	12	4651	6,31E+09	12
801	2,48E+10	12	2101	7,75E+10	13	3401	9,38E+09	12	4701	6,30E+09	12
851	2,33E+10	12	2151	1,10E+10	12	3451	8,71E+09	12	4751	1,08E+10	12
901	2,48E+10	12	2201	9,84E+09	12	3501	1,52E+10	12	4801	6,17E+09	12
951	2,17E+10	12	2251	9,41E+09	13	3551	9,59E+09	12	4851	6,72E+09	12
1001	1,94E+10	12	2301	9,05E+09	12	3601	9,98E+09	12	4901	5,70E+09	12
1051	1,81E+10	12	2351	8,83E+09	13	3651	6,96E+09	12	4951	5,67E+09	12
1101	1,73E+10	13	2401	1,20E+10	12	3701	1,57E+10	12	5001	1,06E+10	12
1151	1,66E+10	13	2451	8,52E+09	12	3751	6,70E+09	12			
1201	1,98E+10	13	2501	9,06E+09	12	3801	7,34E+09	12			
1251	1,82E+10	13	2551	9,71E+09	12	3851	2,81E+10	12			