|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***"Стилизация изображений с использованием методов глубокого обучения"***

Студент ИУ5-81Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Журавлева П.В.\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Гапанюк Ю.Е.\_ \_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Кротов Ю.Н. \_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2021 г.*

# АННОТАЦИЯ

Расчетно-пояснительная записка 58 с. (с учетом приложений 81 с.), 36 рис., 7 табл., 14 источников, 3 приложения.

Выпускная квалификационная работа бакалавра на тему "Стилизация изображений с использованием методов глубокого обучения" посвящена разработке веб-приложения, предназначенного для стилизации изображения методом нейронного переноса стиля. Стиль с одного изображения переносится на другое с сохранением содержания второго.

Расчетно-пояснительная записка состоит из трех частей.

Постановка задач разработки содержит описание предметной области, функциональных возможностей системы, а также анализ аналогов и сравнение с ними разрабатываемой системы.

Исследовательская часть посвящена изучению понятий нейронных сетей и свёрточных нейронных сетей, архитектуры сетей VGG, а также рассмотрению алгоритма нейронного переноса стиля.

Конструкторско-технологическая часть описывает процесс выбора программных и аппаратных средств, процесс разработки веб-сервиса, включающий в себя реализацию алгоритма нейронного переноса стиля и разработку интерфейса взаимодействия с пользователем, а также анализ использования различных оптимизаторов.

СОДЕРЖАНИЕ

[АННОТАЦИЯ 2](#_Toc73366591)

[СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ 5](#_Toc73366592)

[ВВЕДЕНИЕ 6](#_Toc73366593)

[1 ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ 7](#_Toc73366594)

[1.1 Описание предметной области 7](#_Toc73366595)

[1.2 Функциональные возможности разрабатываемого веб-приложения 7](#_Toc73366596)

[1.3 Выбор критериев качества разрабатываемого веб-приложения 8](#_Toc73366597)

[1.4 Анализ аналогов 10](#_Toc73366598)

[2 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 13](#_Toc73366599)

[2.1 Понятие нейронной сети 13](#_Toc73366600)

[2.2 Понятие свёрточной нейронной сети 15](#_Toc73366601)

[2.2.1 Слой свёртки СНС 15](#_Toc73366602)

[2.2.2 Слой активации СНС 16](#_Toc73366603)

[2.2.3 Пулинг или слой субдискретизации (подвыборки) СНС 17](#_Toc73366604)

[2.2.4 Полносвязная НС 19](#_Toc73366605)

[2.2.5 Часто используемые техники 20](#_Toc73366606)

[2.2.5.1 Паддинг 20](#_Toc73366607)

[2.2.5.2 Страйд 21](#_Toc73366608)

[2.3 Алгоритм нейронного переноса стиля (НПС) 21](#_Toc73366609)

[2.3.1 Использование свёрточной нейронной сети VGG 22](#_Toc73366610)

[2.3.2 Выделение признаков с помощью VGG19 25](#_Toc73366611)

[2.3.3 Определение функции потерь 26](#_Toc73366612)

[2.3.3.1 Функция потерь для содержания 26](#_Toc73366613)

[2.3.3.2 Функция потерь для стиля 27](#_Toc73366614)

[2.3.3.3 Итоговая функция потерь 29](#_Toc73366615)

[3 КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 31](#_Toc73366616)

[3.1 Конструкторская часть 31](#_Toc73366617)

[3.1.1 Выбор программных средств 31](#_Toc73366618)

[3.1.1.1 Выбор языка программирования для разработки 31](#_Toc73366619)

[3.1.1.2 Выбор фреймворков для разработки 32](#_Toc73366620)

[3.1.1.3 Выбор среды разработки 33](#_Toc73366621)

[3.1.1.4 Выбор технологии развёртывания веб-приложения 34](#_Toc73366622)

[3.1.2 Выбор аппаратных средств 35](#_Toc73366623)

[3.2 Технологическая часть 36](#_Toc73366624)

[3.2.1 Разработка веб-сервиса 36](#_Toc73366625)

[3.2.1.1 Реализация алгоритма НПС 36](#_Toc73366626)

[3.2.1.1.1 Предварительная обработка загруженных изображений и обратное преобразование изображений 36](#_Toc73366627)

[3.2.1.1.2 Выбор необходимых промежуточных слоёв и построение модели 37](#_Toc73366628)

[3.2.1.1.3 Определение функции потерь 38](#_Toc73366629)

[3.2.1.1.4 Полезные функции 41](#_Toc73366630)

[3.2.1.1.5 Функция запуска алгоритма НПС 42](#_Toc73366631)

[3.2.1.2 Разработка интерфейса взаимодействия с пользователем 43](#_Toc73366632)

[3.2.2 Анализ использования различных оптимизаторов 52](#_Toc73366633)

[3.2.3 Разработка диаграмм в нотации IDEF0 54](#_Toc73366634)

[ЗАКЛЮЧЕНИЕ 56](#_Toc73366635)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 57](#_Toc73366636)

[ПРИЛОЖЕНИЕ A 59](#_Toc73366637)

[ГРАФИЧЕСКИЕ МАТЕРИАЛЫ 59](#_Toc73366638)

[Примеры стилизованных изображений 60](#_Toc73366639)

[Диаграммы в нотации IDEF0 62](#_Toc73366640)

[ПРИЛОЖЕНИЕ B 66](#_Toc73366641)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 66](#_Toc73366642)

[ПРИЛОЖЕНИЕ C 74](#_Toc73366660)

[ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 74](#_Toc73366661)

# СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

1. **НС** – нейронная сеть
2. **СНС (CNN)** – сверточные нейронные сети (convolutional neural networks).
3. **НПС (NST) –** нейронный перенос стиля (neural style transfer)
4. **НАХС (NAAS)** – нейронный алгоритм художественного стиля (a neural algorithm of artistic style)
5. **VGG (Visual Geometry Group) –** архитектура СНС
6. **Python** – высокоуровневый язык программирования
7. **ИСР (IDE)** – интегрированная среда разработки (Integrated development environment)

# ВВЕДЕНИЕ

Стилизация в изобразительном искусстве – воспроизведение или "имитация образной системы и формальных особенностей одного из стилей прошлого, использованных в новом художественном контексте".[1] Это понятие напрямую связано с термином "стиль", который, в свою очередь, в одной из формулировок означает особое качество формы произведения искусства, достигаемое целостностью творческого метода, способов формообразования, приёмов композиции, индивидуальной манеры и техники, свойственных художникам определённого исторического периода. [2]

Стиль на одном уровне с содержанием изображения используется человеком для выражения своих мыслей, чувств и задумок на бумажном или электронном холсте, однако для полноценной и качественной реализации своей идеи ему необходимо потратить достаточно большое количество времени – навык рисования нарабатывается в течение не одного года. Даже повторение чужого стиля – достаточно сложная задача, и в этом случае становится актуальным вопрос стилизации изображения, с помощью которой человек может превратить свою фотографию в произведение искусства, при этом не обладая необходимыми для этого навыками.

Стилизовать изображение можно разными способами: например, наложить на него фильтр или же перенести стиль с одного изображения на другое. Второй способ можно выполнить с использованием нейросетей, именно поэтому он называется нейронным переносом стиля. Перенос стиля – одно из наиболее креативных применений сверточных нейронных сетей.[3] Два изображения объединяются в одно целое с использованием контента одного из них и стиля другого. В настоящее время данная тема является достаточно актуальной и популярной.

# ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ

# Описание предметной области

Предметной областью данной системы является изобразительное искусство, творческий процесс. В процессе творчества создаются новые объекты, которые зависят исключительно от их автора. Именно он придумывает идею, определяет и выполняет процесс, пользуется конкретными инструментами для создания чего-то нового. Творчество помогает человеку выразить себя, свои чувства и мысли, реализовать задумки. Оно приносит ему чувство радости и удовлетворения. Разрабатываемый сервис позволит пользователю без особых усилий попробовать сотворить что-то новое, прикоснуться к одному из вариантов самовыражения, вдохновиться на будущие творческие разработки. Он сможет почувствовать себя настоящим художником и превратить свою фотографию в художественное произведение.

# Функциональные возможности разрабатываемого веб-приложения

Функциональные возможности разрабатываемого веб-сервиса описаны в Техническом Задании. Ниже приведен список функционала приложения с пояснениями.

1. Загрузка двух исходных изображений – загрузка исходного изображения, которое пользователь хочет обработать, и исходного стилизованного изображения, с которого пользователь хочет взять стиль;
2. Изменение загруженных изображений – возможность замены каждого из изображений, если пользователю не захочется использовать уже загруженные;
3. Удаление загруженных изображений – возможность удаления каждого из изображений без загрузки нового (для дальнейшей работы нужно будет загрузить исходное изображение отдельно);
4. Выбор числа итераций цикла стилизации – чем большее число выбрано, тем лучше получится итоговый результат (однако время выполнения стилизации будет увеличиваться);
5. Создание одного изображения в стиле другого – собственно сам перенос стиля с одного изображения на другое;
6. Отображение прогресса выполнения переноса стиля – отображение индикатора выполнения задачи, показывающего, какая часть переноса уже выполнена;
7. Отображение приблизительного времени выполнения задачи – примерная оценка оставшегося времени выполнения переноса стиля;
8. Отображение промежуточного результата выполнения программы – вывод на экран текущего изображения в процессе переноса стиля;
9. Отображение итогового стилизованного изображения на экране – вывод на экран полученного изображения;
10. Сохранение полученного изображения – наличие возможности сохранить результат в файловую систему.

# Выбор критериев качества разрабатываемого веб-приложения

Для оценки качества работы системы и сравнения ее с аналогами будут рассмотрены следующие критерии:

1. Удобство использования
2. Наличие необходимого функционала (возможности выбора собственного стиля)
3. Возможность выбора степени стилизации изображения
4. Качество итогового результата

Такой важный критерий, как время выполнения алгоритма, я рассматривать не буду, так как он не является стабильным и постоянным, и его достаточно сложно оценить. Например, на одном из сервисов-аналогов время выполнения переноса может быть буквально пару минут или же доходить до 20 в зависимости от нагрузки на сервис.

Оценим коэффициенты важности критериев при помощи метода балльной оценки [4] (таблица 1).

Таблица 1 – Метод балльной оценки

|  |  |  |  |
| --- | --- | --- | --- |
| Код фактора | Название критерия | Балл | Коэффициент важности локального критерия по методу бальной оценки ( |
| Х1 | Удобство использования | 80 | 0,242 |
| Х2 | Наличие необходимого функционала | 100 | 0,303 |
| Х3 | Возможность выбора степени стилизации изображения | 60 | 0,181 |
| Х4 | Качество итогового результата | 90 | 0,273 |

Коэффициент важности локального критерия по методу бальной оценки ( высчитываются по формуле:

где – балл критерия,

– количество критериев.

# Анализ аналогов

В качестве аналогов будут рассмотрены следующие веб-сервисы:

1. deepart.io – В1
2. neuralstyle.art – В2
3. instapainting.com/ai-painter – В3

Данные сервисы были найдены при помощи поисковой системы Google по запросу “style transfer online”. Рассматривать будем только бесплатный функционал данных сервисов.

Проведем сравнение аналогов с веб-сервисом, разрабатываемым в рамках этой работы (В4), с использованием метода взвешенной суммы нормированных показателей сравнения.[4]

Таблица 2 – Расчет весовых коэффициентов выбранных критериев качества

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Код фактора | Критерий | Весовой коэффициент | В1 | В2 | В3 | В4 |
| Х1 | Удобство использования | 0,242 | оч.хор | уд | отл | отл |
| Х2 | Наличие необходимого функционала | 0,303 | да | да | нет | да |
| Х3 | Возможность выбора степени стилизации изображения | 0,181 | нет | нет | да | да |
| Х4 | Качество итогового результата | 0,273 | отл | хор | оч.хор | оч.хор |

Таблица 3 – Сравнение вариантов на Парето-оптимальность

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | В1 | В2 | В3 | В4 |
| В1 | 0 | 1 | 0 | 0 |
| В2 | 0 | 0 | 0 | 0 |
| В3 | 0 | 0 | 0 | 0 |
| В4 | 0 | 1 | 1 | 0 |
| Результат сравнения | 0 | 1 | 1 | 0 |
| Парето-оптимальность варианта | да | нет | нет | да |

Варианты В2 и В3 не являются парето-оптимальными, потому мы можем отбросить их.

Таблица 4 – Вербально-числовая шкала для X1 (удобство использования) и X4 (качество итогового результата)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Отлично | Очень хорошо | Хорошо | Удовлетворительно | Плохо |
| 1 | 0,9 | 0,75 | 0,6 | 0,5 |

Таблица 5 – Вербально-числовая шкала для X2 (наличие необходимого функционала) и X3 (возможность выбора степени стилизации изображения)

|  |  |
| --- | --- |
| Да | Нет |
| 1 | 0 |

Таблица 6 – Нормированные значения сравниваемых вариантов

|  |  |  |  |
| --- | --- | --- | --- |
| Код фактора | Весовой коэффициент | Нормированные  значения показателей | |
| В1 | В4 |
| Х1 | 0,242 | 0,9 | 1 |
| Х2 | 0,303 | 1 | 1 |
| Х3 | 0,181 | 0 | 1 |
| Х4 | 0,273 | 1 | 0,9 |
|  | | 0,7938 | 0,9717 |

Ранжирование вариантов по предпочтительности:

В результате применения метода взвешенной суммы нормированных показателей сравнения можно сделать вывод, что разрабатываемый веб-сервис является одним из самых оптимальных вариантов среди рассмотренных.

Варианты В1 и В4 практически идентичны по выбранным критериям, однако у варианта В4 есть значительное преимущество – наличие возможности выбора степени стилизации изображения.

## **ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ**

# Понятие нейронной сети

Нейронная сеть – это программное или аппаратное воплощение математической модели, построенной по принципу функционирования сетей нервных клеток живых организмов. Это понятие возникло при попытке описания процессов, происходящих в мозгу человека.[5]

Нейронная сеть состоит из простейших вычислительных элементов – искусственных *нейронов*. Каждый нейрон имеет несколько входных и одну выходную связь.[6] Нейрон обычно представляется в виде линейной комбинации всех входных сигналов. *Передаточная функция (функция активации)* определяет зависимость сигнала на выходе нейрона от полученной линейной комбинации сигналов на его входах; это способ нормализации входных данных.

Все нейроны объединены в одну сеть – выходы одних нейронов соединяются со входами других. С единственного выхода нейрона сигнал может поступать на множество входов других нейронов. Связи между нейронами принято называть *синапсами*.[7] Каждая связь характеризуется своим *весом*. Благодаря ему, входная информация изменяется, когда передается от одного нейрона к другому.

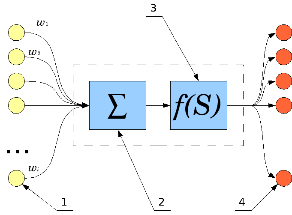


Рис. 1. Схема искусственного нейрона  
1. Нейроны, выходные сигналы которых поступают на вход данному  
2. Сумматор входных сигналов  
3. Вычислитель передаточной функции  
4. Нейроны, на входы которых подаётся выходной сигнал данного  
5. {\displaystyle w\_{i}}  – *веса* входных сигналов

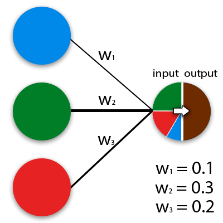
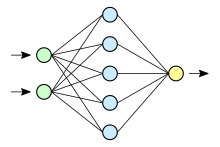


Рис. 2. Синапсы, каждый из которых характеризуется весовым коэффициентом – информация передается с трех нейронов на один

Нейроны делятся на три основных типа: входной, скрытый и выходной. Соответственно, есть входной слой, который получает информацию, n скрытых слоев, которые ее обрабатывают, и выходной слой, который выводит результат.

  
Рис. 3. Схема простой нейросети.   
Зелёным цветом обозначены *входные* нейроны,   
голубым — *скрытые* нейроны, жёлтым — *выходной* нейрон

# Понятие свёрточной нейронной сети

Свёрточная нейронная сеть (СНС или CNN – convolutional neural network) – специальная архитектура нейронных сетей, нацеленная на распознавание и классификацию образов.[8] Она входит в состав технологий глубокого обучения, которые, в свою очередь, являются совокупностью методов машинного обучения, основанных на обучении представлениям. С помощью обучения представлениям (или признакам) система может автоматически обнаружить представления, необходимые для выявления различных признаков. Работа СНС зачастую интерпретируется как переход от конкретных особенностей изображения к всё более и более абстрактным деталям.

# Слой свёртки СНС

Идея СНС заключается в чередовании свёрточных, субдискретизирующих (или слоёв подвыборки, пулинга) и "обычных" слоёв нейронной сети. В операции свёртки используется одна и та же матрица весов (так называемое *ядро свёртки*) для различных нейронов, ее "передвигают" по обрабатываемому слою, посылая после каждого сдвига сигнал для нейрона следующего слоя на аналогичной позиции.[9] Во время этого процесса мы получаем карту признаков (уникальные черты объекта) для обрабатываемого слоя. Выделение признаков на входном изображении и формирование карты признаков – основное назначение свёрточного слоя.[10] В СНС используется целое множество ядер свёртки для кодирования элементов изображения.

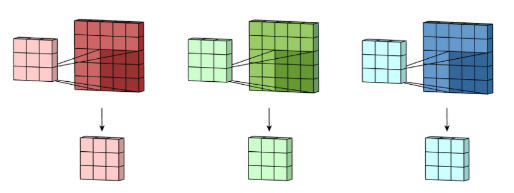


Рис. 4. Операция свёртки

Ядра свёртки формируются во время классического метода обучения сети – *метода обратного распространения ошибки* для обновления весов в нейронной сети. Основная идея этого метода заключается в распространении сигналов ошибки от выходов сети к ее входам. Направление распространения сигнала в данном случае обратно по отношению к прямому распространению сигнала в обычном режиме работы сети. Простыми словами, после каждого прохода по сети обратное распространение выполняет проход в обратную сторону и регулирует параметры модели для уменьшения величины функции ошибки. На каждом слое вычисляются градиенты обучаемых параметров, и в конце обратного распространения они используются для обновления параметров модели (например, весов) с помощью градиентного спуска.

# Слой активации СНС

Скалярный результат каждой свёртки попадает на [функцию активации](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%B0%D1%86%D0%B8%D0%B8). Слой активации обычно логически объединяют со слоем свёртки (считают, что функция активации встроена в слой свёртки). Передаточная функция может быть любой, однако в свёрточных нейронных сетях наиболее часто используется так называемая функция ReLU (от англ. rectified linear unit, что переводится как выпрямленный линейный блок), ее также называют "выпрямитель".[10] Формула ReLU следующая:

(2)

Эта функция возвращает 0, если аргумент отрицательный, и сам аргумент, если он положительный – то есть она, по сути, отсекает отрицательную часть скалярной величины.



Рис. 5. Функция активации ReLU

У этой передаточной функции можно выделить следующие преимущества:

1. Быстро и просто считается производная (для отрицательных значений она равна нулю, для положительных – единице);
2. Разреженность активации – количество включаемых нейронов меньше, нежели при использовании других функций (например, сигмоидной функции или функции гиперболического тангенса), что улучшает производительность обучения модели, и сама сеть становится легче.

Также у нее есть и недостаток, который называется проблемой умирающего ReLU. Если часть производной функции равна нулю, то и градиент для нее будет нулевым. Это приведет к тому, что веса в сети не будут изменяться во время градиентного спуска и нейронная сеть перестанет обучаться.

ReLU стоит использовать при отсутствии особых требований на выходное значение нейрона. Если после обучения результаты модели оказались не оптимальными, стоит рассмотреть другие функции активации, чтобы получить наилучший результат.

# Пулинг или слой субдискретизации (подвыборки) СНС

Операция подвыборки выполняет уменьшение размерности полученных карт признаков сети.[8] Слой субдискретизации уменьшает пространство признаков, сохраняя наиболее важную информацию. Эта операция помогает ускорить дальнейшие вычисления и делает сеть более инвариантной к размерам входного изображения.

Существует несколько версий пулинга:

1. Максимальный пулинг



Рис. 6. Карта признаков и максимальный пулинг

1. Средний пулинг

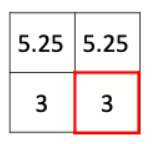


Рис. 7. Карта признаков и средний пулинг

1. Пулинг суммы

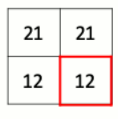


Рис. 8. Карта признаков и пулинг суммы

Наиболее часто используется максимальный пулинг. Слою подвыборки требуется только один гиперпараметр – шаг пулинга, т.е. число раз, в которое нужно сократить исходную размерность. Чаще всего входной тензор уменьшают в два раза.

# Полносвязная НС

После прохождения нескольких слоёв свёртки и пулинга система по факту переходит от четкой сетки пикселей с высоким разрешением (качеством) к более абстрактным картам признаков. Обычно на каждом следующем слое число каналов увеличивается, а размерность изображения в каждом канале уменьшается. В итоге остается большое число каналов с малым числом данных, которые интерпретируются как наиболее абстрактные понятия, полученные из исходного изображения.[8]

Полученные данные объединяются и передаются на обычную полносвязную НС, которая также может состоять и множества слоёв. Полносвязные слои утрачивают пространственную структуру пикселей и обладают сравнительно небольшой размерностью по отношению к числу пикселей исходного изображения.

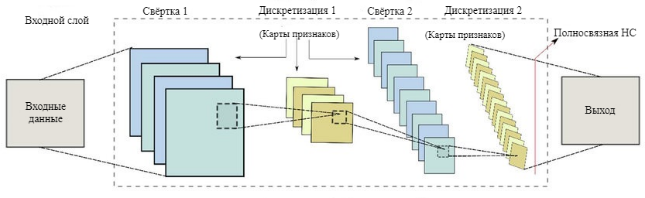


Рис. 9. Архитектура свёрточной нейронной сети (СНС)

# Часто используемые техники

# Паддинг

При движении ядра по исходной матрице крайние пиксели никогда не попадают на центр ядра. Это неприемлемо, если размеры исходной и результирующей матрицы должны быть одинаковы. Эта проблема решается с помощью паддинга, который добавляет дополнительные ("ложные") пиксели вокруг исходной матрицы.[9] Чаще всего значение этих пикселей равно нулю, и такой паддинг называется нулевым. Благодаря паддингу, ядро может обойти исходную матрицу полностью (рис. 10).

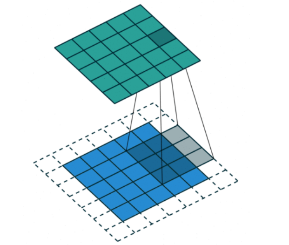


Рис. 10. Техника паддинга

# Страйд

Страйд по своей сути похож на операцию пулинга. В основе принципа страйда лежит идея пропуска ядром определенного числа шагов при его перемещении. Страйд со значением 1 говорит о том, что ядро перемещается на один пиксель, и это приводит нас к стандартной свертке.[9] При страйде со значением 2 ядро смещается на два пикселя, уменьшая размер результирующей матрицы по сравнению с предыдущим вариантом примерно вдвое.

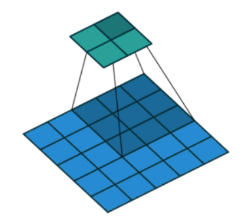
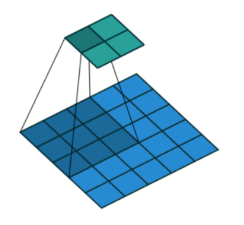


Рис. 11. Техника страйда со значением 2

В некоторых архитектурах нейронных сетей полностью отказались от слоёв пулинга в пользу техники страйда.

# Алгоритм нейронного переноса стиля (НПС)

Для решения поставленной задачи стилизации изображения я выбрала алгоритм нейронного переноса стиля (NST) – этот алгоритм можно считать классическим и первым для решения задачи переноса стиля с использованием свёрточных нейронных сетей. Данный алгоритм впервые был опубликован в [11] и также называется нейронным алгоритмом художественного стиля (НАХС или NAAS – a Neural Algorithm of Artistic Style). Алгоритм использует два изображения: изображение контента (англ. Content Image), в котором нас интересует содержание (обычно пользователь использует фотографии), и изображение стиля (англ. Style Image), в котором нас интересует художественный стиль (обычно в качестве изображения стиля пользователь выбирает картины известных художников).[12] После этого алгоритм изменяет входные данные так, чтобы они соответствовали содержанию изображения контента и художественному стилю изображения стиля. Алгоритм выполняется следующим образом:

1. Оба исходных изображения прогоняются через СНС. Авторами [11] было предложено использовать сеть VGG (16 или 19);
2. После прохождения исходных изображений через сеть выделяются карты признаков стиля и содержания исходных изображений;
3. Определяется функция потерь, включающая в себя потери отдельно стиля и содержания;
4. Запускается цикл оптимизации, в котором функцию потерь необходимо устремить к нулю. В этом цикле итоговое изображение обновляется таким образом, чтобы содержание и стиль максимально соответствовали содержанию первого исходного изображения и стилю второго исходного изображения.

# Использование свёрточной нейронной сети VGG

VGG – это классическая архитектура сверточной нейронной сети. Изначально сеть была основан на анализе того, как увеличить глубину таких сетей. В сети используются небольшие фильтры размера 3 x 3. Сеть отличается своей простотой: единственными другими компонентами являются слои пулинга и полносвязные слои.[13]

Сеть VGG была натренирована на ImageNet – это набор данных, состоящий из более чем 15 миллионов размеченных высококачественных изображений, разделенных на 22 тысячи категорий. Изображения были взяты из Интернета и размечены вручную людьми-разметчиками.

Архитектура VGG выглядит следующим образом:

1. На вход VGG принимает RGB-изображение размером 224x224 пикселя;
2. Свёрточные слои в VGG используют очень маленькое рецептивное поле (3x3). Рецептивное поле – часть матрицы входных сигналов, подвергаемая “свертыванию”;
3. VGG имеет три полносвязных уровня: первые два имеют 4096 нейронов каждый, а третий - 1000 нейронов (1000 потому, что сеть обучалась классифицировать данные на 1000 разных классов);
4. Все скрытые слои VGG используют функцию активации ReLU.

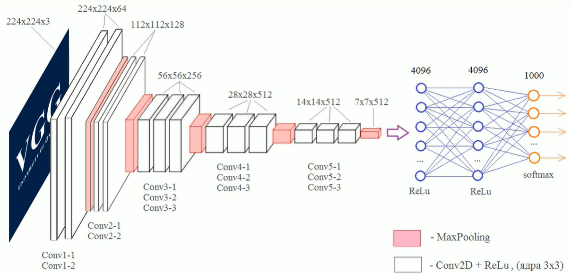


Рис. 12. Общая структура сетей VGG

На рис. 12 изображена общая структура сетей VGG. Первые два числа, указанные над слоями, (224 x 224, 112 x 112 и т.д.) означают размерность в пикселях, а третье означает число каналов/фильтров. Красным цветом по казаны слои максимального пулинга, они уменьшают размерности полученных на вход карт признаков в два раза. На выходе мы получаем тензор размерностью 7 x 7 с 512 каналами. Полученные данные подаются на вход полносвязной нейронной сети. Различие между сетями VGG16 и VGG19 указаны на рис. 13.



Рис. 13. Различие между архитектурами сетей VGG16 и VGG19

По этому рисунку видно, что сеть VGG19 отличается от сети VGG16 только лишь наличием нескольких дополнительных слоёв свёртки. Соответственно названия данным сетям даны по количеству слоёв свёртки и полносвязных слоёв (VGG16 = 13 слоёв свёртки + 3 полносвязных слоя, VGG19 = 16 слоёв свёртки + 3 полносвязных слоя).

У сетей VGG есть два недостатка:

1. Очень медленная скорость обучения;
2. Архитектура сети весит слишком много.

Несмотря на недостатки, VGG является отличной сетью для обучения, так как её легко реализовать. VGG существенно превосходит в производительности прошлые поколения моделей, а глубина представления положительно влияет на точность классификации.[13]

# Выделение признаков с помощью VGG19

Чтобы получить представление содержания и стиля картинки, в первую очередь нужно посмотреть на промежуточные слои модели. Промежуточные слои представляют собой карты признаков, которые по мере углубления становятся более упорядоченными. Мы используем предварительно обученную сетевую архитектуру VGG19. Промежуточные слои играют важную роль в определении представлений. Для входного изображения нужно сопоставить соответствующие представления на этих промежуточных слоях.

Почему эти промежуточные выводы дают возможность определить стиль и контент изображения? Чтобы сеть могла классифицировать изображение (а именно этому она обучена), она должна понимать это изображение. Это включает в себя построение из группы пикселей сложных представлений объектов на изображении. Отчасти это объясняет, почему свёрточные нейронные сети могут хорошо обобщать: они способны заметить постоянство и определить особенности, характерные для какого-либо класса (чтобы отличить, например, кота от собаки), не обращая внимания на фоновый шум. Таким образом, где-то между подачей изображения на вход и выводом результата классификации этого изображения, стоит модель, которая находит признаки во входных данных. Соответственно, обращаясь к этой самой промежуточной точке (т. е. слоям), можно без труда получить представление стиля и содержания изображения.

Для выделения признаков содержания мы будем использовать последний канал VGG19 – Conv5-2. Это тензор размерностью 14 x 14, который содержит 512 каналов.

Стиль, в отличие от содержания, имеет важные детали на разных уровнях абстракции, поэтому для выделения признаков стиля мы будем рассматривать несколько слоёв модели VGG19, а именно: Conv1-1, Conv2-1, Conv3-1, Conv4-1 и Conv5-1.

# Определение функции потерь

# Функция потерь для содержания

Определить функцию потерь для содержимого на самом деле довольно просто. Нужно передать сети два изображения: изображение желаемого содержания и исходное. Благодаря этому мы получим данные из промежуточных слоёв (определённых выше) нашей модели. И единственное, что остаётся, это рассчитать Евклидово расстояние между двумя промежуточными представлениями этих изображений.

Более формально, потеря содержимого – это функция, которая описывает расстояние от содержимого выходного изображения и изображения содержимого . Обозначим предварительно обученную СНС (в нашем случае опять же VGG19) как . За обозначим любое изображение, тогда будет означать переданное сети изображение. Пусть и описывают соответствующее промежуточное представление признаков в сети с входами (выходное изображение) и (изображение содержимого) на слое под номером . Тогда функцию потери содержимого (или расстояния ) можно записать следующим образом:

(3)

# Функция потерь для стиля

Вычисление потери стиля немного сложнее, но оно действует по тому же принципу; на этот раз нужно передать сети следующие изображения: изображение желаемого стиля и исходное. Однако вместо сравнения необработанных промежуточных выходных данных мы сравниваем матрицы Грама двух выходных данных.

Возьмем, например, слой Conv5-1 сети VGG19. Его размерность 14 x 14 (), и содержит он 512 каналов (). Для удобства представим тензор в виде матрицы (рис. 14).

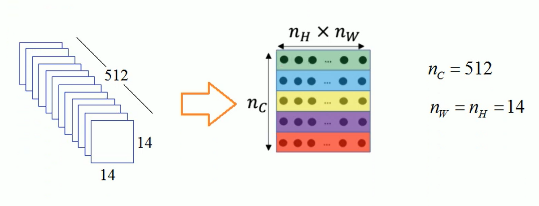


Рис. 14. Тензор признаков изображения, представленный в виде матрицы

Полученную матрицу умножаем на её транспонированный вариант (меняем строки на столбцы) для получения матрицы Грама. Фактически матрица Грама содержит скалярное произведение векторов признаков – мы умножаем каждую строку одной матрицы на каждый столбец другой матрицы (рис. 15).

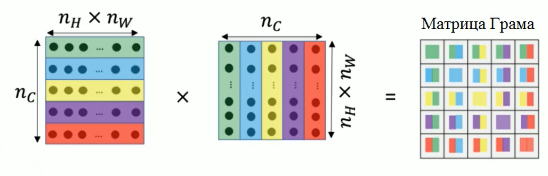


Рис. 15. Получение матрицы Грама

Полученная матрица Грама характеризует схожесть признаков между каналами исходного тензора. В нашем случае это будет характеристика стиля на текущем слое сети VGG19.

Такую матрицу мы составляем для обоих изображений (исходного и желаемого стиля) на каждом из перечисленных выше слоёв. В итоге на каждом слое у нас получается две матрицы Грама – для формируемого изображения и для изображения стиля. После этого мы находим рассогласование между ними по следующей формуле:

, (4)

где – матрица Грама для формируемого изображения,

– матрица Грама для изображения стиля.

Фактически мы здесь получили квадрат Евклидова расстояния между двумя матрицами Грама. Формулу (4) мы вычисляем для каждого из выбранных выше слоёв сети VGG19, то есть мы рассчитаем следующие величины:

После этого полученные величины мы сложим, умножив каждую из них на ее вес:

(5)

В рамках данной задачи возьмем все весовые коэффициенты одинаковыми и равными , то есть, по сути, мы усредним все вычисленные величины.

# Итоговая функция потерь

С учетом полученных значений и итоговую функцию потерь (общий "показатель качества") можно определить следующим образом:

, (6)

где и – некие веса, показывающие, насколько важно учитывать соответственно содержимое и стиль исходных изображений.

Подбирая различные значения и , мы можем получать разную степень стилизации итогового изображения.

Нам необходимо постоянно обновлять изображение, устремляя функцию потерь к нулю. Метод обновления изображения будет разобран в следующем разделе расчётно-пояснительной записки.

# КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

# **Конструкторская часть**

# **Выбор программных средств**

# Выбор языка программирования для разработки

В качестве языка программирования для разработки был выбран язык Python. Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным – всё является объектами Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Python считается самым простым языком программирования – именно поэтому он самый распространенный.

Python является одним из самых популярных языков для решения задач анализа данных и машинного обучения. У этого языка отличная производительность при обработке данных.

Одна из основных причин, почему Python используется для машинного обучения состоит в том, что у него есть множество фреймворков, которые упрощают процесс написания кода и сокращают время на разработку.

Давайте обсудим, какие именно библиотеки и фреймворки Python используются в машинном обучении. В научных расчетах используется Numpy, в продвинутых вычислениях — SciPy, в извлечении и анализе данных — SciKit-Learn. Эти библиотеки работают в таких фреймворках, как TensorFlow, CNTK и Apache Spark.

Существует фреймворк для Python, разработанный специально для машинного обучения — это PyTorch.

Кроме того, Python хорошо подходит для машинного обучения, потому что сами алгоритмы машинного обучения сложны для понимания. При работе с Python разработчику не нужно уделять много внимания непосредственно написанию кода: все внимание он может сосредоточить на решении более сложных задач, связанных с машинным обучением.

# Выбор фреймворков для разработки

В процессе решения задач разработки были использованы следующие фреймворки и библиотеки: TensorFlow, Keras, NumPy, Streamlit.

TensorFlow – это комплексная платформа для машинного обучения с открытым исходным кодом. Как и большинство фреймворков глубокого обучения, TensorFlow имеет API на Python поверх механизма C и C ++, что ускоряет его работу.

TensorFlow имеет гибкую экосистему инструментов, библиотек и ресурсов сообщества. Это позволяет исследователям использовать самые современные МО-технологии, а разработчикам – создавать и развёртывать приложения на базе машинного обучения.

Keras – открытая среда глубокого обучения, написанная на Python. Фреймворк нацелен на оперативную работу с нейросетями и является компактным, модульным и расширяемым. Подходит для небольших проектов, так как создать что-то масштабное на нём сложно. Keras работает поверх TensorFlow. Фреймворк содержит многочисленные реализации широко применяемых строительных блоков нейронных сетей, таких как слои, целевые и передаточные функции, оптимизаторы, а также множество инструментов для упрощения работы с изображениями и текстом.

NumPy – библиотека с открытым исходным кодом для языка программирования Python. Её возможности включают в себя поддержку многомерных массивов (включая матрицы) и поддержку высокоуровневых математических функций, предназначенных для работы с многомерными массивами. Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами.

Streamlit – это фреймворк с открытым кодом, специально разработанный для инженеров машинного обучения, работающих с Python. Этот фреймворк является одним из лучших для демонстрации проектов, связанных с анализом данных (data science). Он позволяет создавать интерактивные веб-приложения благодаря буквально нескольким строкам кода. Streamlit превращает скрипты данных в веб-приложения для совместного использования за считанные минуты. В сущности, каждое веб-приложение Streamlit – это скрипт Python.

# Выбор среды разработки

В качестве среды разработки была выбрана PyCharm. PyCharm – интегрированная среда разработки для языка программирования Python. Понятие интегрированной среды разработки (англ. Integrated development environment – IDE) означает комплекс программных средств, используемый программистами для разработки программного обеспечения (ПО). Обычно среда разработки включает в себя:

1. Текстовый редактор;
2. Транслятор (компилятор и/или интерпретатор);
3. Средства автоматизации сборки;
4. Отладчик.

Среда разработки PyCharm была выбрана потому, что весь веб-сервис написан на языке Python. Также с помощью среды разработки можно было легко создать виртуальное окружение (venv) для установки необходимых библиотек, фреймворков и т.п. Виртуальное окружение – это изолированное окружение среды (в нашем случае это окружение Python), которое позволяет нам использовать определенные версии приложений и пакетов. С его помощью можно устанавливать различные пакеты для конкретных проектов без установки их на всю систему. Также его удобно использовать в том случае, если для разных проектов нужны различные версии каких-либо библиотек.

Также PyCharm поддерживает распределённую систему управления версиями git, благодаря чему можно было с лёгкостью разместить проект на GitHub для его дальнейшего развёртывания, а также переносить все изменения в исходном коде.

# Выбор технологии развёртывания веб-приложения

Для развёртывания веб-сервиса была выбрана технология Streamlit Sharing, так как интерфейс веб-сервиса написан с использованием фреймворка Streamlit. Streamlit Sharing можно использовать для быстрого и простого развёртывания Streamlit-приложения – это идеальное решение, если приложение размещено в общедоступном репозитории GitHub.

Развёртывание приложения с помощью Streamlit Sharing выглядит следующим образом:

1. Необходимо запросить приглашение на использование этого сервиса;
2. Разместить Streamlit-приложение в общедоступном репозитории GitHub;
3. Добавить файл requirements.txt, содержащий все внешние зависимости, которые необходимо установить;
4. Зайти на свой аккаунт Streamlit Sharing, к которому привязан аккаунт на GitHub;
5. Выбрать необходимый репозиторий, ветку и файл и нажать "Deploy", то есть развернуть приложение.

После выполненных действий приложение становится доступным для использования на удалённом сервере, и его можно открыть на любом устройстве, имеющем доступ в сеть Интернет.

Если необходимо сделать какие-либо изменения в исходном коде, то достаточно "запушить" (команда git push) их на свой GitHub. Развёрнутое приложение обновится мгновенно.

# **Выбор аппаратных средств**

Так как веб-сервис развёрнут удалённо, то для его использования достаточно наличие устройства с выходом в Интернет. Локально исходный код веб-сервиса запускался на ноутбуке со следующими характеристикам:

1. Оперативная память – 8 ГБ
2. Процессор Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80GHz
3. Тип системы – 64-разрядная операционная система, процессор x64
4. Видеокарта NVIDIA GeForce GTX 1050

Тестовые испытания веб-сервиса показали, что удалённо выполнение стилизации изображение происходит быстрее, нежели локально. Это означает, что характеристики удалённого сервера превосходят характеристики данного ноутбука, однако информацию о них в сети Интернет найти не удалось.

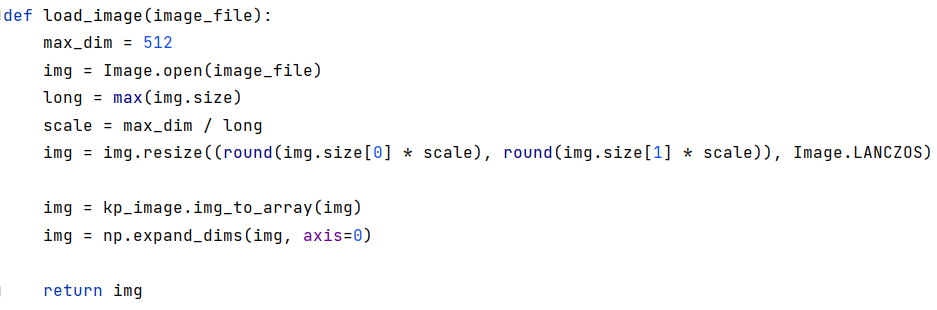
# Технологическая часть

# Разработка веб-сервиса

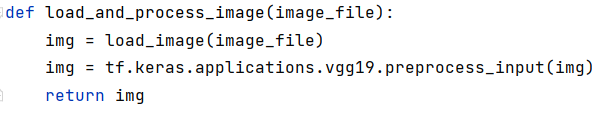
# Реализация алгоритма НПС

# Предварительная обработка загруженных изображений и обратное преобразование изображений

Подготовка данных (то есть предобработка исходных изображений) включает в себя их масштабирование, преобразование изображений в тензоры и добавление нулевой оси:



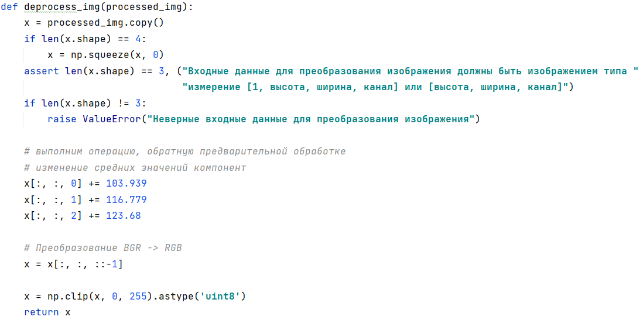
А также смену формата (преобразование цветовой модели):



Функция keras.applications.vgg19.preprocess\_input позволяет перевести изображение из формата RGB в формат BGR, а также меняет средние значения цветовых компонент следующим образом: B(103.939), G(116.779), R(123.68) – вычитает указанные в скобках величины для каждого из каналов.

Все вышеперечисленные действия выполняются для того, чтобы исходные изображения могли восприниматься сетью VGG19.

Для преобразования изображения из формата BGR в формат RGB введём вспомогательную функцию, выполняющую действия, обратные предварительной обработке загруженных изображений. Это необходимо нам для того, чтобы изображение можно было вернуть в "обычное" состояние и вывести его на экран:



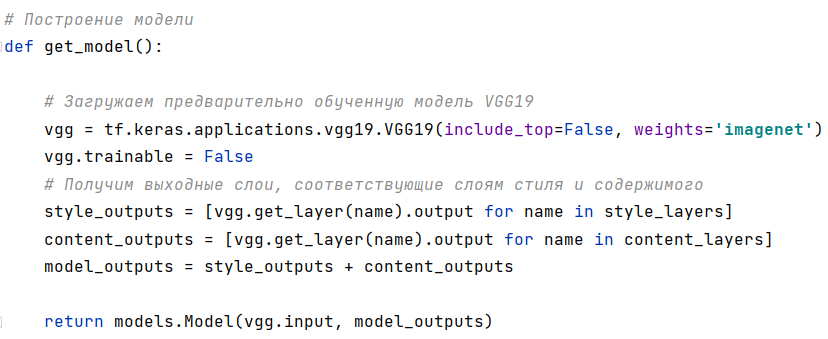
# Выбор необходимых промежуточных слоёв и построение модели

Определяем вспомогательные коллекции с именами слоёв, которые мы будем выделять из сети VGG19, и сразу определяем их количество.

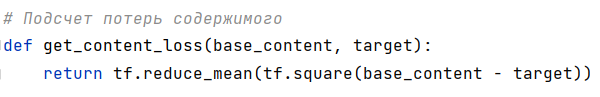


Создадим нашу модель с доступом к промежуточным слоям. Эта функция загрузит модель VGG19 и получит доступ к промежуточным слоям. Затем эти слои будут использоваться для создания новой модели, которая будет принимать входное изображение и возвращать выходные данные этих промежуточных слоев из модели VGG.

Функция возвращает модель keras, которая принимает входные данные изображения и выводит промежуточные слои стиля и содержимого.

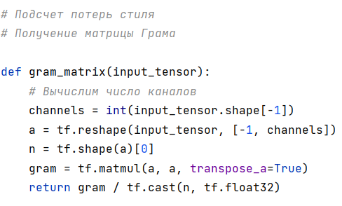


# Определение функции потерь

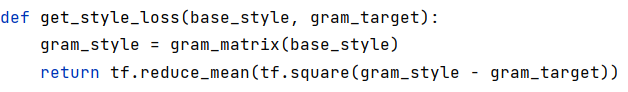
Функция потерь для содержимого будет выглядеть следующим образом:

Эта функция с помощью операций tensorflow вычисляет значение, соответствующее формуле (3).

Следующие две функции используются для определения потерь по стилю на определённом слое СНС. Одна из них преобразует тензор в матрицу и вычисляет матрицу Грама (рис. 14 и рис.15):



Вторая функция вычисляет стиль для строго определённого слоя нейронной сети (то есть её нужно будет подсчитать для всех выбранных слоёв стиля):



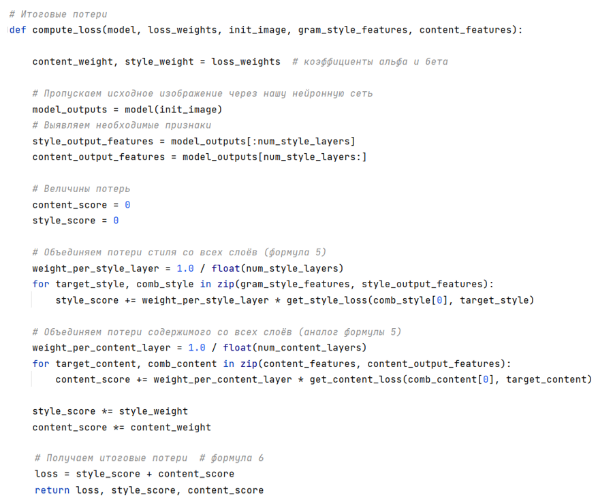
Эта функция с помощью операций tensorflow вычисляет значение, соответствующее формуле (4).

Итоговая функция вычисления потерь принимает на вход следующие параметры:

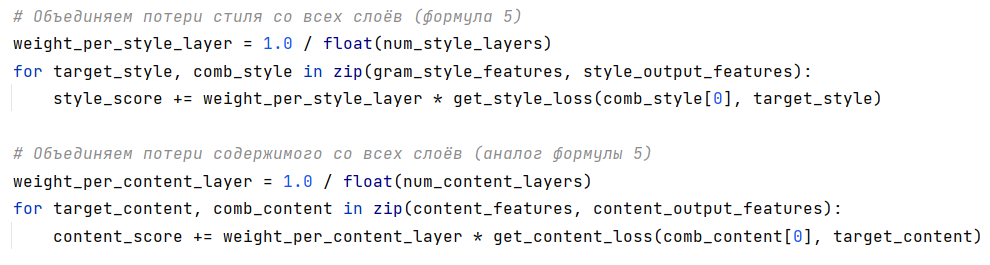
1. model: модель, которая даст нам доступ к промежуточным слоям;
2. loss\_weights: веса, показывающие вклад каждой из функций потерь, это коэффициенты и из формулы (6);
3. init\_image: исходное изображение, которое мы будем обновлять в процессе оптимизации;
4. gram\_style\_features: предварительно вычисленные матрицы Грама, соответствующие выбранным слоям стилей;
5. content\_features: предварительно вычисленные выходные данные из определенных интересующих слоев контента.

И возвращает она значения loss, style\_score, content\_score, соответствующие итоговым потерям и потерям по каждой из рассматриваемых компонент.

Выглядит функция вычисления потерь следующим образом:

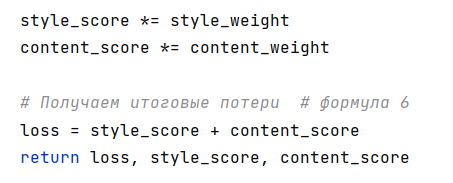


Рассмотрим некоторые части функции более детально. На приведённом ниже кусочке мы вычисляем итоговые потери по стилю, умножая величину, полученную на каждом отдельном слое, на её вес в соответствии с формулой (5):



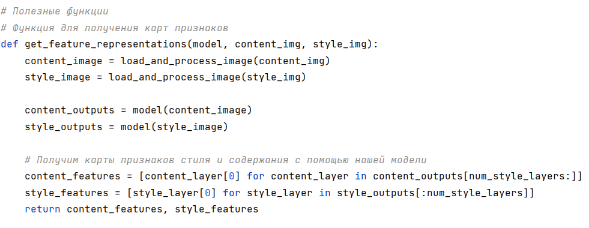
Аналогичную операцию мы выполняем и для стиля содержания. Этот цикл, по сути, выполнится один раз, однако есть возможность использовать его при выборе нескольких слоёв содержания.

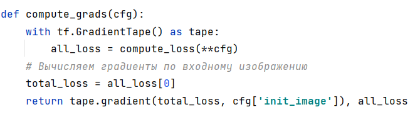
С помощью кода, указанного ниже, мы получаем значение в соответствии с формулой (6), учитывая коэффициенты и :



# Полезные функции

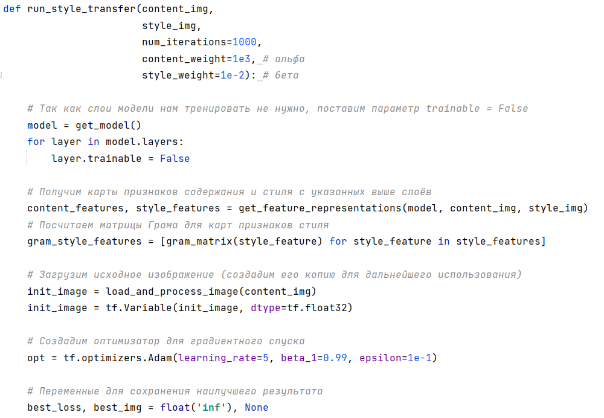
Определим пару полезных функций: первая будет загружать исходные изображения и получать необходимые нам карты признаков, а вторая будет считать градиент для изменения пикселей формируемого изображения в соответствии с функцией потерь. Градиент мы получаем с помощью средств tensorflow, чтобы затем подать его на вход оптимизатору, который, в свою очередь, будет изменять пиксели изображений с целью уменьшения функции потерь.

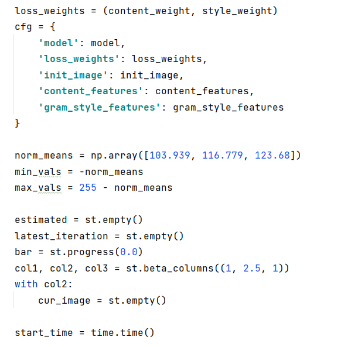


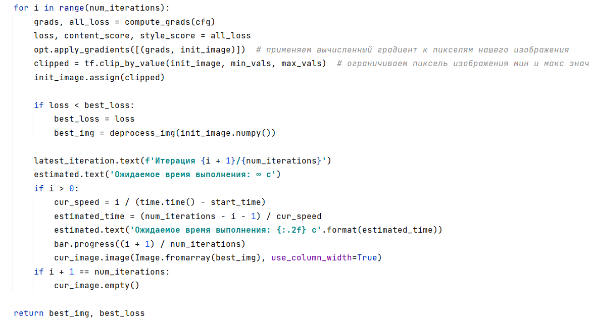


# Функция запуска алгоритма НПС

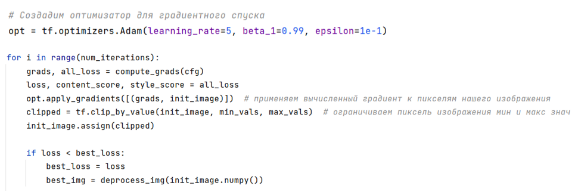
Полная функция запуска алгоритма нейронного переноса стиля выглядит следующим образом:







Рассмотрим некоторые части функции более детально. В приведённом ниже коде мы создаём оптимизатор для алгоритма градиентного спуска, затем в цикле по числу итераций мы вычисляем текущие градиент и потери, с помощью оптимизатора применяем вычисленный градиент к пикселям нашего изображения, тем самым обновляя его. Мы не обновляем веса в сети. После этого мы проверяем, уменьшились ли потери, и если да, то запоминаем текущее значение и обновляем лучшее изображение. Чем больше итераций цикла будет выполнено, тем лучше будет итоговый результат.



# Разработка интерфейса взаимодействия с пользователем

Для предоставления пользователю возможности использования алгоритма нейронного переноса стиля необходимо разработать простой и интуитивно понятный интерфейс. Все требования к функциональным возможностям сервиса, перечисленные в приложении В данной работы, должны быть выполнены.

Пользовательский интерфейс написан с использованием фреймворка Streamlit. Веб-сервис развёрнут с помощью инструмента Streamlit Sharing.

Домашняя страница интерфейса взаимодействия с пользователем изображена на рис. 16.

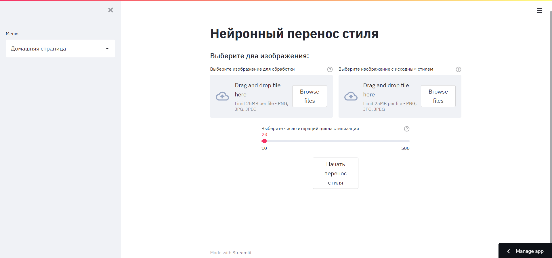


Рис. 16. Домашняя страница веб-сервиса

Через меню слева можно попасть на страницу "О сервисе", где указана основная информация о функционале сервиса и о его разработчике (рис. 17)

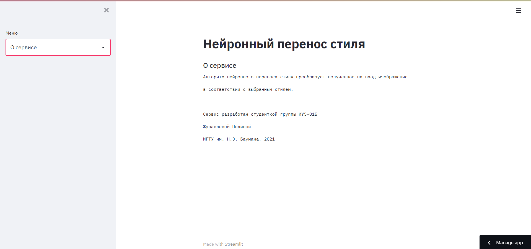


Рис. 17. Страница "О сервисе"

Streamlit поддерживает автоматическое масштабирование контента на странице, поэтому сервисом удобно пользоваться как на персональном компьютере в окнах разного размера (рис. 18), так и на телефоне (рис. 19).

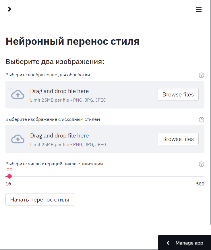
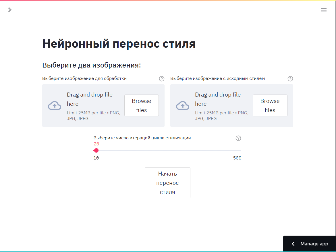


Рис. 18. Изменение размера окна на персональном компьютере

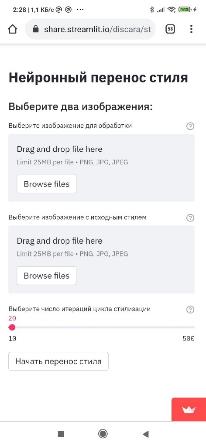


Рис. 19. Интерфейс сервиса на смартфоне

С помощью загрузчиков файлов можно загрузить исходные изображения из файловой системы: изображение для обработки и изображение с исходным стилем (рис. 20).

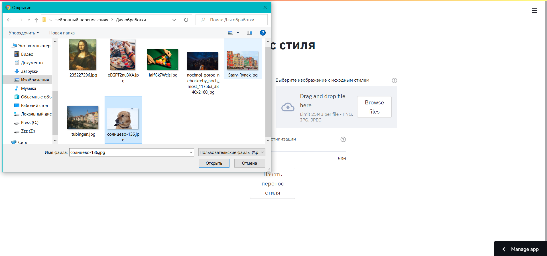


Рис. 20. Выбор изображения из файловой системы

После загрузки выбранные изображения отображаются на странице (рис. 21). Если хотя бы одно (или оба) изображение не загружено, то при попытке запуска переноса стиля появится соответствующее сообщение об ошибке (рис. 22-24).

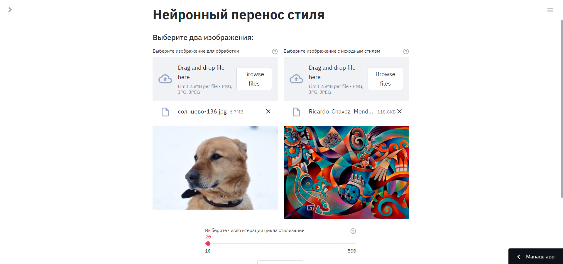


Рис. 21. Отображение загруженных изображений

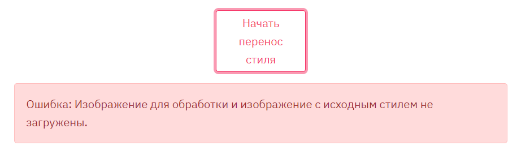


Рис. 22. Сообщение об ошибке, если оба изображения не были загружены

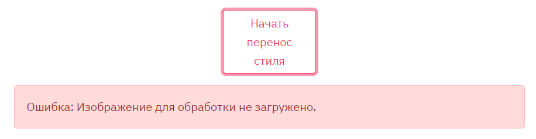


Рис. 23. Сообщение об ошибке, если исходное изображение для обработки не было загружено

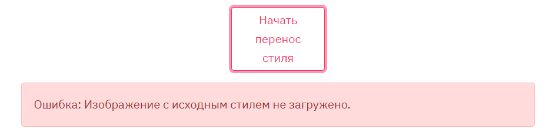


Рис. 24. Сообщение об ошибке, если изображение с исходным стилем не было загружено

Также у каждого загрузчика при наведении на вопросительный знак появляется подсказка о загрузке файлов на русском языке (рис. 25).

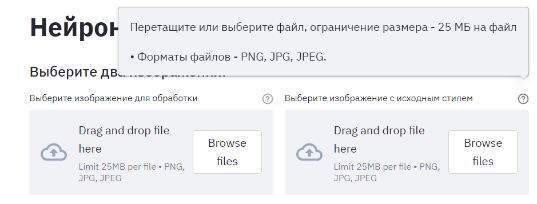


Рис. 25. Подсказка о загрузке файлов на русском языке

Загруженные изображения можно удалить, нажав на крестик (рис. 26), либо заменить, загрузив новое (либо выбрать в файловой системе, либо перенести). Тогда на месте старого изображения появится новое (рис. 27).

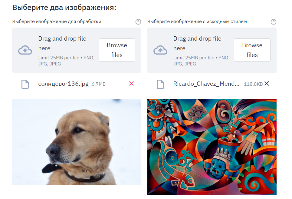
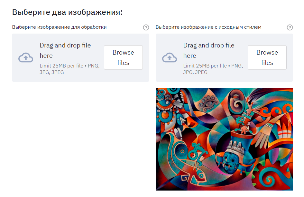
 

Рис. 26. Удаление изображения

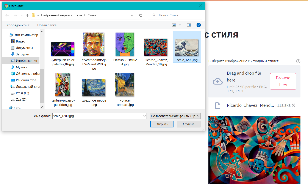
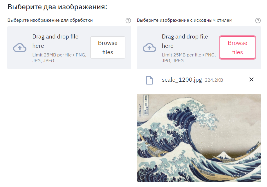
 

Рис. 27. Изменение изображения

После загрузки изображений пользователю предлагается с помощью слайдера выбрать число итераций для цикла стилизации (рис. 28). По умолчанию это число равно 20. При наведении на вопрос рядом со слайдером появляется справочная информация о значении числа итераций (рис. 29).

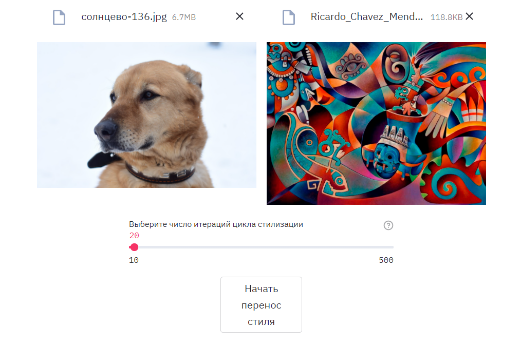


Рис. 28. Слайдер, с помощью которого можно выбрать   
число итераций цикла стилизации

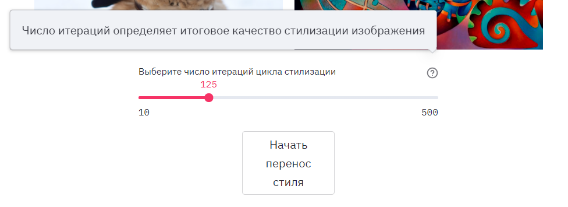


Рис. 29 Справочная информация о значении   
числа итераций цикла стилизации

При нажатии на кнопку "Начать перенос стиля" начинается выполнение нейронного переноса стиля (если загружены оба изображения), при этом на экране начинает отображаться информация о выполнении алгоритма: ожидаемое время выполнения, текущая итерация, индикатор, визуально показывающий прогресс выполнения и текущий вариант изображения (рис. 30). Текущий вариант изображения можно в любой момент сохранить (рис. 31).

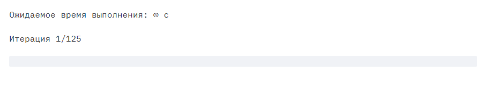




Рис. 30. Информация о выполнении алгоритма

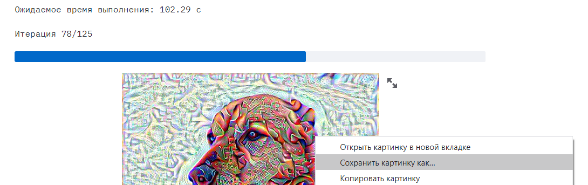


Рис. 31. Возможность сохранения изображения на любой итерации

После окончания выполнения алгоритма на экране остается итоговое изображение (рис. 32), которое можно сохранить двумя способами:

1. Нажать на изображение правой кнопкой мыши и сохранить с использованием контекстного меню (как на рис. 31);
2. Нажать на ссылку, появившуюся под изображением, левой кнопкой мыши – тогда изображение сохранится в папку "Загрузки" под названием result.jpg (рис. 33); либо нажать на нее правой кнопкой мыши и выбрать пункт "Сохранить ссылку как…" В таком случае можно выбрать местоположение для изображения в файловой системе и изменить его название (рис. 34).

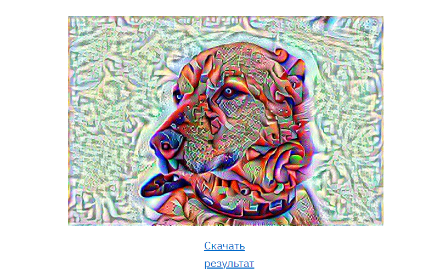


Рис. 32. Итоговое изображение, отображающееся на экране,   
и ссылка на его скачивание

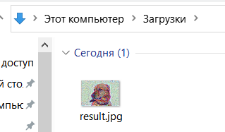
 

Рис. 33. Первый способ сохранения изображения по ссылке

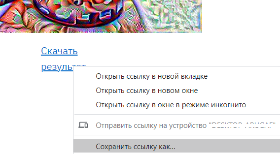
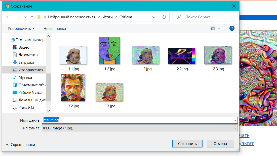
 

Рис. 34. Второй способ сохранения изображения по ссылке

Варианты итоговых стилизованных изображений приведены в приложении А данной расчетно-пояснительной записки.

# Анализ использования различных оптимизаторов

Оптимизаторы определяют оптимальный набор параметров модели, таких как вес и смещение, чтобы при решении конкретной задачи модель выдавала наилучшие результаты.[14] Однако в нашем случае оптимизаторы используются для применения вычисленного средствами tensorflow градиента к пикселям изображения, тем самым изменяя его. Рассмотрим несколько вариантов оптимизаторов, содержащихся в модуле tf.keras.optimizers:

1. SGD;
2. Adagrad;
3. Adam;

У данных оптимизаторов будем менять только параметр learning rate (коэффициент скорости обучения), который влияет на то, насколько сильно изменяется изображение во время каждой итерации. Остальные параметры устанавливаются по умолчанию. Результаты исследования представлены в таблице 7.

Таблица 7 – Сравнение различных оптимизаторов при изменении параметра learning rate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | learning rate = 0.02 | learning rate = 5 | learning rate = 10 | learning rate = 50 |
| SGD | Итерация 15 | Итерация 4 | - | - |
| Adagrad | Итерация 15 | Итерация 15 | Итерация 15 | Итерация 5 |
| Adam | Итерация 15 | Итерация 15 | Итерация 15 | Итерация 11 |

Рассмотрим поближе изображения, полученные во время использования оптимизаторов Adagrad и Adam при равенстве параметра learning rate 10 и 50 (рис. 35-36), чтобы лучше увидеть различия между изображениями.



Рис. 35. Использование оптимизатора Adagrad   
(learning rate =10 и 50, итерации 15 и 5 соответственно)



Рис. 36. Использование оптимизатора Adam   
(learning rate =10 и 50, итерации 15 и 11 соответственно)

Рассматривая результаты, полученные во время анализа разных оптимизаторов, можно сделать вывод, что каждый из оптимизаторов имеет свой минимум параметра learning rate, при котором изменения на изображении становятся заметны, и свой предел повышения параметра learning rate. В какой-то момент изображения становятся испорченными. Оптимизатор SGD уже при значении 5 начал выдавать испорченное изображение, а оптимизаторы Adagrad и Adam при одинаковом значении параметра learning rate (50) начали выдавать испорченный результат на разных итерациях – Adagrad на 5-ой, а Adam – на 11-ой. При этом можно заметить, что при значениях learning rate = 5 и learning rate = 10 на одинаковых итерациях оптимизатор Adam выдает лучшие варианты изображений, нежели оптимизатор Adagrad. Следовательно, рассмотренные оптимизаторы можно поставить в следующем порядке по предпочтительности:

# Разработка диаграмм в нотации IDEF0

Разработанная система полностью соответствует требованиям, описанным в Техническом Задании. Реализованы все заявленные в приложении B функциональные характеристики веб-сервиса.

Основной реализованный функционал системы описан при помощи диаграмм нотации IDEF0. Построенные диаграммы содержатся в приложении А данной расчетно-пояснительной записки.

# ЗАКЛЮЧЕНИЕ

В результате выполнения у веб-сервис, позволяющий пользователю выполнить алгоритм нейронного переноса стиля – то есть перенести стиль с одного изображение на другое с сохранением содержания второго изображения.

В ходе разработки были рассмотрены понятия нейронных и сверточных нейронных сетей, архитектура сетей VGG, был разобран классический алгоритм нейронного переноса стиля. Данный алгоритм был реализован с помощью фреймворков и библиотек TensorFlow, Keras, NumPy, пользовательский интерфейс был создан с использованием фреймворка Streamlit. Также был проведен сравнительный анализ нескольких оптимизаторов, используемых для обновления исходного изображения, и выбран лучший из них – оптимизатор Adam.

Были выполнены все требования к функциональным характеристикам, основными (самыми главными) из них являются следующие требования:

1. Загрузка двух исходных изображений;
2. Изменение загруженных изображений;
3. Выбор числа итераций цикла стилизации;
4. Создание одного изображения в стиле другого;
5. Отображение промежуточного результата выполнения программы;
6. Отображение итогового стилизованного изображения на экране;
7. Сохранение полученного изображения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Стилизация // Власов В. Г. Новый энциклопедический словарь изобразительного искусства. В 10 Т. СПб.: Азбука-Классика, т. 9, 2008. С.251-259.
2. Стиль // Власов В. Г. Новый энциклопедический словарь изобразительного искусства. В 10 т. — СПб.: Азбука-Классика. — Т. IX, 2008. — С. 460
3. Туториал: перенос стиля изображений с TensorFlow [Электронный ресурс] – URL: <https://neurohive.io/ru/tutorial/tutorial-perenos-stilja-izobrazhenij-s-tensorflow/>, дата обращения 22.05.2021
4. Постников В. М. Основы эксплуатации автоматизированных систем обработки информации и управления. Краткий курс: учеб. пособие. — 177 с.
5. Нейронная сеть // Википедия [Электронный ресурс] – URL: <https://ru.wikipedia.org/wiki/Нейронная_сеть>, дата обращения 22.05.2021
6. Нейросеть [Электронный ресурс] – URL: <https://basegroup.ru/deductor/function/algorithm/neuronet>, дата обращения 22.05.2021
7. Нейронные сети для начинающих. Часть 1 // Хабр [Электронный ресурс] – URL: <https://habr.com/ru/post/312450/>, дата обращения 22.05.2021
8. Свёрточная нейронная сеть // Википедия [Электронный ресурс] – URL: <https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть>, дата обращения 23.05.2021
9. Свёрточная нейронная сеть с нуля. Часть 0. Введение [Электронный ресурс] – URL: <https://programforyou.ru/poleznoe/convolutional-network-from-scratch-part-zero-introduction>, дата обращения 23.05.2021
10. Практики реализации нейронных сетей [Электронный ресурс] – URL: <https://neerc.ifmo.ru/wiki/index.php?title=Практики_реализации_нейронных_сетей#.D0.A4.D1.83.D0.BD.D0.BA.D1.86.D0.B8.D1.8F_ReLU>, дата обращения 22.05.2021
11. Leon A. Gatys, Alexander S. Ecker, Matthias Bethge (2015) A Neural Algorithm of Artistic Style – 16 с.
12. Neural Style Transfer [Электронный ресурс] – URL: <https://neerc.ifmo.ru/wiki/index.php?title=Neural_Style_Transfer>, дата обращения 25.05.2021
13. VGG16 — сверточная сеть для выделения признаков изображений // Neurohive [Электронный ресурс] – URL: https://neurohive.io/ru/vidy-nejrosetej/vgg16-model/
14. Объясняем на пальцах принцип действия оптимизаторов для нейронных сетей: основные алгоритмы, и зачем они нужны1 // Хабр [Электронный ресурс] – URL: <https://habr.com/ru/company/skillfactory/blog/552394/>, дата обращения 26.05.2021

# 

# ПРИЛОЖЕНИЕ A

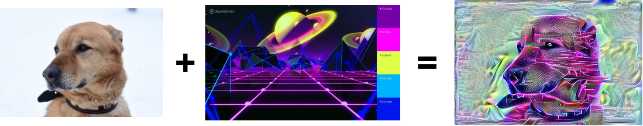
# ГРАФИЧЕСКИЕ МАТЕРИАЛЫ

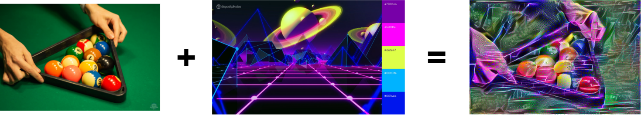
1. Примеры стилизованных изображений
2. Диаграммы в нотации IDEF0

# Примеры стилизованных изображений



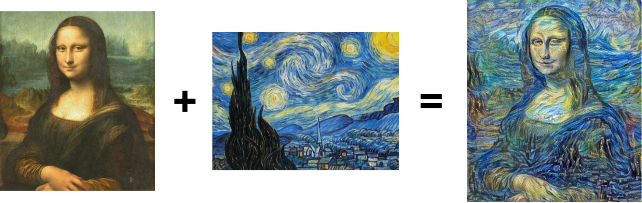










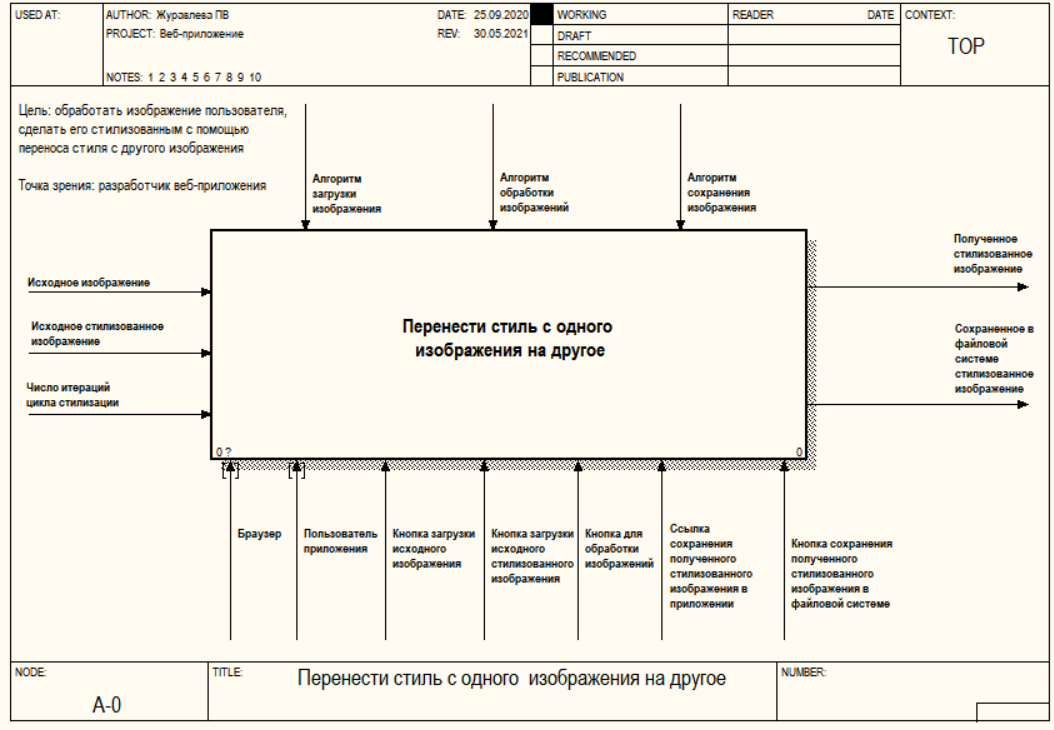


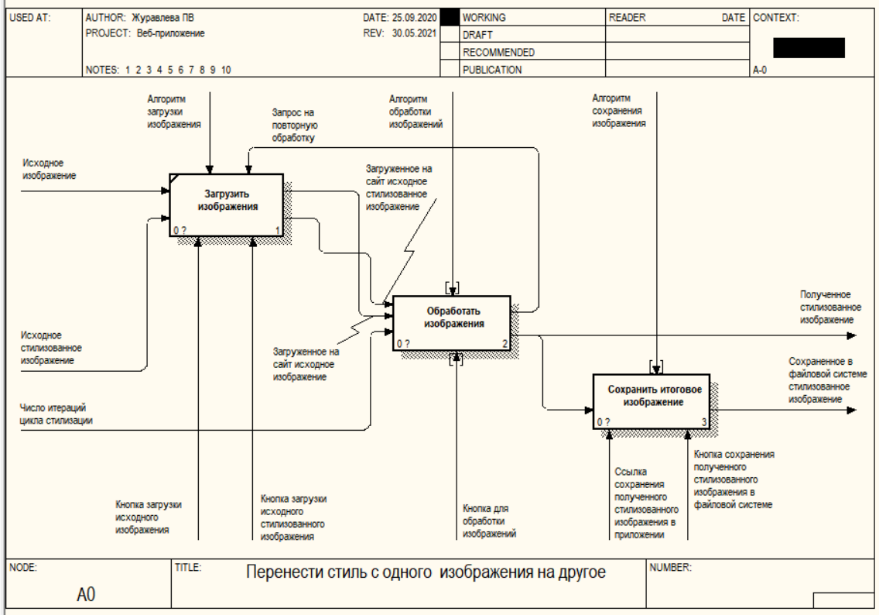


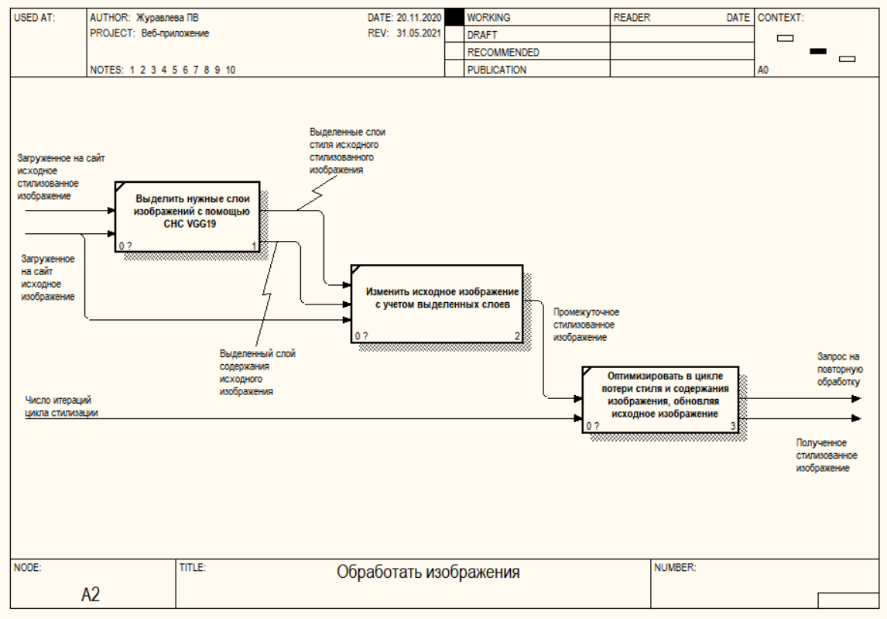


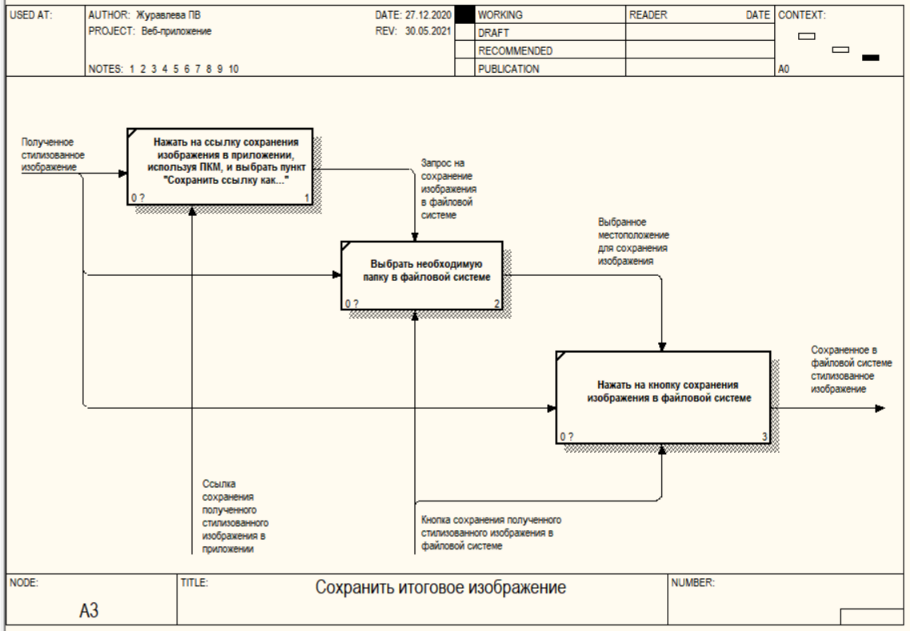


# Диаграммы в нотации IDEF0









# ПРИЛОЖЕНИЕ B

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. Н.Э. Баумана**

Кафедра «Системы обработки информации и управления»

Утверждаю: Гапанюк Ю.Е Согласовано: Гапанюк Ю.Е.

"\_\_"\_\_\_\_\_\_\_\_\_\_2021 г. "\_\_"\_\_\_\_\_\_\_\_\_\_2021 г.

**Стилизация изображений с использованием методов глубокого обучения**

Техническое задание

(вид документа)

писчая бумага

(вид носителя)

7

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| Журавлева П.В. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

Москва - 2021

Содержание

[Содержание 60](#_Toc72376249)

[1. Наименование 61](#_Toc72376250)

[2. Основание для разработки 61](#_Toc72376251)

[3. Исполнитель 61](#_Toc72376252)

[4. Назначение и цель работы 61](#_Toc72376253)

[5. Содержание работы 61](#_Toc72376254)

[5.1 Задачи 61](#_Toc72376255)

[5.2 Требования к функциональным характеристикам 62](#_Toc72376256)

[5.3 Требования к архитектуре программного изделия 63](#_Toc72376257)

[5.4 Требования к входным и выходным данным 63](#_Toc72376258)

[5.5 Требования к надежности 64](#_Toc72376259)

[5.6 Лингвистические требования 64](#_Toc72376260)

[5.7 Требования к составу технических средств 64](#_Toc72376261)

[6. Этапы работы 64](#_Toc72376262)

[7. Техническая документация 65](#_Toc72376263)

[8. Порядок приема работы 65](#_Toc72376264)

[9. Дополнительные условия 65](#_Toc72376265)

1. Наименование

Стилизация изображений с использованием методов глубокого обучения.

2. Основание для разработки

Основанием для разработки является задание на выпускную работу, подписанное руководителем выпускной работы и утверждённое заведующим кафедрой. Задание утверждено кафедрой ИУ5 МГТУ им. Н.Э. Баумана.

3. Исполнитель

Студентка четвёртого курса группы ИУ5-81Б Журавлева П.В.

4. Назначение и цель работы

Целью работы является проектирование и разработка системы, позволяющей произвести перенос нейронного стиля, т.е. создать одно изображение в стиле другого.

Назначением работы является создание сервиса, позволяющего пользователю без особых усилий преобразовать его изображение в "шедевр" в желаемом стиле.

5. Содержание работы

5.1 Задачи

5.1.1. Исследовать предметную область, определить функциональные задачи.

5.1.2. Разработать архитектуру программного обеспечения.

5.1.3. Собрать данные предметной области.

5.1.4. Структурировать и подготовить данные предметной области.

5.1.5. Обучить модель глубокого обучения или использовать уже обученную модель, разработать алгоритм решения задачи.

5.1.6. Реализовать веб-сервис с интегрированной моделью глубокого обучения.

5.1.7. Провести тестирование информационно - программного продукта.

5.1.8. Провести отладку программного продукта.

5.1.9. Оформить техническую документацию.

5.2 Требования к функциональным характеристикам

Разрабатываемая система должна выполнять следующие функции:

5.2.1. Загрузка двух исходных изображений.

5.2.2. Изменение загруженных изображений.

5.2.3. Удаление загруженных изображений.

5.2.4. Выбор числа итераций цикла стилизации.

5.2.5. Создание одного изображения в стиле другого.

5.2.6. Отображение прогресса выполнения переноса стиля.

5.2.7. Отображение приблизительного времени выполнения задачи.

5.2.8. Отображение промежуточного результата выполнения программы.

5.2.9. Отображение итогового стилизованного изображения на экране.

5.2.9. Сохранение полученного изображения.

5.3 Требования к архитектуре программного изделия

Программный продукт представляет собой веб-сервис, использующий фреймворк для приложений с открытым исходным кодом Streamlit для создания интерфейса взаимодействия с алгоритмом нейронного переноса стиля, а также сервис Streamlit Sharing для развертывания приложения на удаленном сервере с помощью размещенного на Github кода программы.

5.4 Требования к входным и выходным данным

**5.4.1 Требования к входным данным**

1. Данные, полученные от пользователя:

* Исходное изображение, которое пользователь хочет стилизовать
* Исходное изображение, в стиле которого пользователь хочет получить итоговую работу
* Число итераций цикла стилизации, влияющее на качество итогового изображения

**5.4.2 Требования к выходным данным**

Выходные данные представляют собой изображение, полученное в результате нейронного переноса стиля с одного изображения на другое, а также различные промежуточные данные: ожидаемое время выполнения алгоритма, текущая итерация цикла переноса стиля и изображение, полученное на конкретном этапе исполнения программы.

5.5 Требования к надежности

Не должна выдавать ошибок, не предусмотренных работой; программа должна надежно и устойчиво функционировать. Должна иметь возможность быть восстановленной в течение часа, в случае непредвиденных проблем функционирования.

5.6 Лингвистические требования

Клиентская часть веб-сервиса должна быть русифицирована.

5.7 Требования к составу технических средств

Минимальные системные требования для клиентской части должны обеспечивать возможность установки программного обеспечения, необходимого для работы веб-сервиса.

6. Этапы работы

График выполнения отдельных этапов работ приведен в соответствии с

приказом об организации учебного процесса в 2020/2021 учебном году.

Таблица 1: Этапы разработки

| № п/п | Наименование этапа и содержание работ | Сроки исполнения |
| --- | --- | --- |
| 1 | Разработка и утверждение ТЗ | Март 2021 г. |
| 2 | Исследование предметной области | Апрель 2021 г. |
| 3 | Разработка архитектуры программного обеспечения | Апрель 2021 г. |
| 4 | Создание программ | Апрель — Май 2021 г. |
| 5 | Тестирование и отладка | Май 2021 г. |
| 6 | Оформление документации | Май — Июнь 2021 г. |
| 7 | Защита работы | Июнь 2021 г. |

7. Техническая документация

По окончании работы предъявляется следующая техническая документация:

1. Техническое задание;

2. Расчётно-пояснительная записка;

3. Программа и методика испытаний;

4. Графический материал по проекту в формате презентации.

8. Порядок приема работы

Приём и контроль программного изделия осуществляется в соответствии с методикой испытаний (см. документ «Программа и методика испытаний»).

9. Дополнительные условия

Данное техническое задание может уточняться в установленном порядке.

# ПРИЛОЖЕНИЕ C

# ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Утверждаю: Гапанюк Ю.Е. Согласовано: Гапанюк Ю.Е.

"\_\_"\_\_\_\_\_\_\_\_\_\_2021 г. "\_\_"\_\_\_\_\_\_\_\_\_\_2021 г.

**Стилизация изображений с использованием методов глубокого обучения**

Программа и методика испытаний

(вид документа)

писчая бумага

(вид носителя)

7

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| Журавлева П.В. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

Москва - 2021Содержание

[1. Объект испытаний 69](#_Toc72380917)

[2. Цель испытаний 69](#_Toc72380918)

[3. Состав предъявляемой документации 69](#_Toc72380919)

[4. Технические требования 69](#_Toc72380920)

[4.1. Требования к составу аппаратного обеспечения 69](#_Toc72380921)

[4.2. Требования к составу программного обеспечения 69](#_Toc72380922)

[5. Порядок проведения испытаний 70](#_Toc72380923)

[6. Последовательность проведения испытаний 70](#_Toc72380924)

# 1. Объект испытаний

Объектом испытаний является веб-сервис, позволяющий произвести перенос нейронного стиля, т.е. создать одно изображение в стиле другого.

# 2. Цель испытаний

Испытания проводятся с целью проверки соответствия системы требованиям к функциональным характеристикам, указанным в п. 5.2 технического задания.

# 3. Состав предъявляемой документации

* 1. Техническое задание;
  2. Программа и методика испытаний.

# 4. Технические требования

# 4.1. Требования к составу аппаратного обеспечения

Минимальные требования для работы клиента:

1. Персональный компьютер на операционной системе Windows или мобильный телефон/смартфон;
2. Подключение к интернету.

# 4.2. Требования к составу программного обеспечения

Для доступа к данному веб-сервису необходимо наличие браузера.

# 5. Порядок проведения испытаний

В процессе проведения испытаний необходимо убедиться в работоспособности веб-сервиса, а конкретно следующих функций:

1. Загрузка и изменение исходных изображений;
2. Определение интенсивности нейронного переноса;
3. Отображение промежуточных результатов и полезной информации;
4. Скачивание полученного изображения.

# 6. Последовательность проведения испытаний

| № | Номер и наименование пункта в ТЗ | Действия пользователя | Результат |
| --- | --- | --- | --- |
| 1 | 5.2.1.  Загрузка двух исходных изображений. | С помощью двух загрузчиков файлов выбрать два изображения из файловой системы. | Выбранные изображения отображаются на экране под загрузчиками. |
| 2 | 5.2.2.  Изменение загруженных изображений. | Загрузить изображения(е) при уже имеющихся загруженных изображениях аналогично действиям, выполненным в пункте 1 последовательности проведения испытаний. | Изображение(я) заменено(ы) на новое(ые). |
| 3 | 5.2.3  Удаление загруженных изображений. | Нажать на крестик рядом с названием конкретного изображения. | Изображение(я) удалено(ы). |
| 4 | 5.2.4.  Выбор числа итераций цикла стилизации. | С помощью графического слайдера выбрать необходимое число итераций. | Выбранное число отображается на экране. |
| 5 | 5.2.5.  Создание одного изображения в стиле другого. | Нажать на кнопку "Начать перенос стиля". | Успех: началось выполнение алгоритма переноса стиля с использованием ранее загруженных изображений и выбранного числа итераций.  Неудача: если какое-то из изображений не было загружено, появляется сообщение об ошибке, говорящее пользователю об этом. |
| 6 | 5.2.6.  Отображение прогресса выполнения переноса стиля. |  | При успешном выполнении пункта 5 последовательности проведения испытаний на экране отображается индикатор выполнения (прогресс бар) алгоритма и номер текущей итерации. |
| 7 | 5.2.7. Отображение приблизительного времени выполнения задачи. |  | При успешном выполнении пункта 5 последовательности проведения испытаний на экране отображается оценочное время завершения выполнения алгоритма в секундах. |
| 8 | 5.2.8  Отображение промежуточного результата выполнения программы. |  | При успешном выполнении пункта 5 последовательности проведения испытаний на экране отображается изображение, полученное на конкретном этапе исполнения программы. |
| 9 | 5.2.9.  Отображение итогового стилизованного изображения на экране. |  | При успешном выполнении пункта 5 последовательности проведения испытаний после завершения выполнения алгоритма на экране отображается итоговое стилизованное изображение. |
| 10 | 5.2.10.  Сохранение полученного изображения | Вариант 1:  На персональном компьютере: нажать на изображение правой кнопкой мыши и в контекстном меню выбрать пункт "Сохранить картинку как…". После этого в появившемся окне выбрать необходимое расположение для изображения в файловой системе и нажать кнопку "Сохранить".  На мобильном телефоне/смартфоне: удерживать изображение несколько секунд, в появившемся контекстном меню выбрать пункт "Скачать изображение". | Изображение сохранено в файловой системе устройства пользователя. |
| 11 | 5.2.10.  Сохранение полученного изображения | Вариант 2:  На ПК: нажать на ссылку "Скачать результат" левой кнопкой мыши, либо нажать на нее правой кнопкой, выбрать пункт "Сохранить ссылку как…" и выбрать расположение для изображения в файловой системе аналогично вар.1.  На мобильном телефоне/смартфоне нажать на ссылку "Скачать результат". | Изображение сохранено в файловой системе устройства пользователя. |