МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ ГЛУБИННОГО ОБУЧЕНИЯ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ ДЛЯ СТИЛИЗАЦИИ ГРАФИЧЕСКИХ ОБРАЗОВ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

Студентки 2 курса 271 группы
направления 09.04.01 — Информатика и вычислительная техника
факультета КНиИТ
Исаевой Татьяны Евгеньевны

Научный руководитель
д.э.н. , к.ф.м.н, профессор _____ Л. В. Кальянов

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2023

ВВЕДЕНИЕ

Актуальность темы. В настоящее время глубинное обучение становится все более популярным и широко используемым методом в области машинного обучения.

Эта техника позволяет строить сложные модели для решения различных задач, таких как классификация, сегментация и детекция объектов. Благодаря своей надежности и высокой точности, алгоритмы глубинного обучения нашли широкое применение в различных областях, включая мобильную разработку[1,2].

В данной магистерской работе рассматриваются основные принципы работы алгоритмов глубинного обучения, а также их применение в мобильной разработке.

Результаты данной работы могут быть полезными для компаний, занимающихся мобильной разработкой, а также могут быть использованы для создания новых и улучшения существующих мобильных приложений.

Сейчас мобильный и быстрый доступ к информации предпочитают все больше людей. С каждым годом процент пользователей мобильных версий сайтов растет, и, следовательно, компьютерные версии становятся все менее популярными

На основании изложенного, можно сделать вывод, что разработка мобильных приложений с использованием методов глубинного обучения является актуальным.

Цель магистерской работы — изучение и применение технологий глубинного обучения в разработке мобильных приложений для стилизации изображений.

В соответствии с поставленной целью определены **следующие задачи:**

— Изучение концепции разработки мобильных приложений;
— Освоение основной теории нейронных сетей;
— Проведение сравнительного анализа моделей нейронных сетей;
— Проектирование и разработка собственной модели нейронной сети;
— Реализация приложения с ипользованием полученных знаний.

Структура и объём работы. Магистерская работа состоит из введения, 5 глав, заключения и списка литературы. Общий объем работы состав-

ляет 71 страницы, объем списка литературы – 20 источников.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теория разработки мобильных приложений» состоит из четырех подразделов, посвященных теории необходимой для создания мобильного приложения.

В подразделе 1.1 рассматриваются основные концепции и особенности мобильной разработки.

В подразделе 1.2 описаны основные этапы жизненного цикла разработки мобильного продукта.

В подразделе 1.3 показаны основные инструменты разработки мобильных приложений с учетом особенностей работы той или иной платформы.

В подразделе 1.4 приведена классификация и анализ популярности тех или иных приложений, в зависимости от области их использования.

Второй раздел «Основы теории нейронных сетей» включает в себя два подраздела, рассказывающих теоретическую базу области изучения нейронных сетей [1-7].

В подразделе 2.1 рассмотрены основные модели нейронных сетей, их особенности, структура и отличия.

В подразделе 2.2 приведен обзор алгоритмов обучения и области их применения.

Раздел 3 «Анализ глубинных нейронных сетей» состоит из двух подразделов, посвященных описанию многослойных нейронных сетей, а также их особенностях и применении в глубинном обучении.

Подраздел 3.1. содержит в себе описание *Convolutional Neural Networks (CNNs)*, которая является классом глубинных нейронных сетей, которые могут распознавать и классифицировать определенные функции изображений и широко используются для анализа графических образов[8,9].

Свое применение они находят не только в распознавании изображений, а также видео, позволяют классифицировать изображения по критериям, анализировать медицинские изображения, применяться в компьютерном зрении и обработке естественного языка.

Архитектура CNN состоит из двух основных частей:

1. Инструмент свертки, который разделяет и идентифицирует различные элементы изображения для анализа в процессе извлечение признаков.

2. Полносвязный слой, который использует выходные данные процесса свертки и предсказывает класс изображения на основе извлеченных ранее признаков.

Также в архитектуре CNN присутствует три типа слоев: сверточные слои, подвыборочные слои и полносвязные (FC) слои. Когда эти слои сложены, будет сформирована архитектура CNN. В дополнение к этим трем слоям есть еще два важных параметра: слой отсева и функция активации, которые определены ниже.

В общем виде сверточный слой можно описать так

$$x^l = f(x_{l-1} * k^l + b^l),$$

где, x^l – выход слоя l;
$f(x)$ – функция активации;
b^l – коэффициент сдвига слоя l;
$*$ – операция свертки входа x с ядром k.

Подвыборочный слой обычно служит мостом между сёрточным слоем и полносвязным слоем (FC).

Полносвязный (FC) слой состоит из весов и смещений вместе с нейронами и используется для соединения нейронов между двумя различными слоями. Эти слои обычно располагаются перед выходным слоем и образуют несколько последних слоёв архитектуры CNN.

Входное изображение из предыдущих слоев выравнивается и подается на FC слой. Затем одномерный вектор проходит еще несколько FC-слоев, где обычно проводятся операции математических функций. На этом этапе начинается процесс классификации.

Последним важным компонентом модели CNN является *функция активации*, которая определяет функциональные возможности нейронной сети, а также какой метод обучения будет применим.

В подразделе 3.2 приведена рекуррентная нейронная сеть, представляющая собой нейронную сеть, специализирующуюся на обработке последовательности данных $x(t) = (x1), ..., x(\tau)$ с индексом шага времени t в диапазоне от 1 до τ.

Отличием рекуррентных нейронных сетей является то, что они выполняют одну и ту же задачу для каждого элемента последовательности, с выходом, зависящим от предыдущих вычислений[10-15]. Другой способ мышления о РНК заключается в том, что они имеют некую «память», которая захватывает информацию о том, что было вычислено до сих пор.

Входные данные: $x(t)$ берутся в качестве входных данных сети на временном шаге t. Например, $x1$, может быть одномерным вектором, например, слово в предложении.

Скрытое состояние: $h(t)$ представляет скрытое состояние в момент t и действует как «память» сети. h(t) вычисляется на основе текущего входа и скрытого состояния предыдущего шага времени: $h(t) = f(Ux(t)+Wh(t-1))$, где функция f есть нелинейное преобразование, такое как $tanh$, $ReLU$.

Веса: RNN имеет параметризованную матрицу веса U – вход к скрытым соединениям, параметризованную матрицу веса W – к повторяющимся, т.е. скрытый к скрытому, соединениям, и матрицу V – от скрытого к выходу соединения. Все эти весовые коэффициенты (U, V, W) делятся во времени.

Выходные данные: $o(t)$ иллюстрируют выход из узла. На рисунке 4 изображена стрелка после $o(t)$, которая часто подвергается нелинейности, особенно когда сеть содержит последующие слои.

Прямой проход

Модель не предопределяет заранее выбор функции активации для скрытых единиц.

Поэтому, рассмотрим два предположения:

1) функцией активации, будет функция – гиперболический тангенс: $f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$

2) Вывод является дискретным, как если бы RNN использовался для предсказания слов или символов. Естественным способом представления дискретных переменных является рассмотрение вывода o, в качестве дающего ненормализованные логарифмические вероятности для каждого возможного значения дискретной переменной. Затем мы можем применить операцию $softmax$ в качестве шага после обработки для получения вектора $\hat{y}$ нормализованных вероятностей над выходом.

Таким образом, прямой проход RNN может быть представлен набором

уравнений, представленных ниже:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

$$h^{(t)} = tanh(a^{(t)})$$

$$o^{(t)} = c + Vh^{(t)}$$

$$\hat{y}^{(t)} = softmax(o^{(t)})$$

Данные уравнения являются примером рекуррентной сети, которая отображает входную последовательность на выходную той же длины.

Полная потеря для заданной последовательности значений x в сочетании с последовательностью значений y будет сумма потерь за все шаги времени.

Обратный проход

Вычисление градиента или обратный проход включает в себя выполнение прохода вперед, проходящего слева направо метод описан выше, за которым следует обратный проход, движущийся справа налево через граф.

Время выполнения – $O(\tau)$, которое не может быть уменьшено путём распараллеливания, так как прямой граф распространения по своей сути является последовательным; каждый шаг времени может быть вычислен только после предыдущего.

Состояния, вычисленные в прямом проходе, должны храниться до тех пор, пока они не будут повторно использованы во время обратного прохода, поэтому стоимость памяти также равна $O(\tau)$.

Поскольку параметры являются общими для всех временных шагов в сети, градиент на каждом выходе зависит не только от вычислений текущего шага времени, но и от предыдущих шагов времени.

Вычисление градиентов Учитывая функцию потери L, нужно рассчитать градиенты для трех весовых матриц U, V, W и меток смещения b, c и обновить их с скоростью обучения α. Подобно нормальному обратному распространению, градиент дает нам представление о том, как изменяется потеря по отношению к каждому весовому параметру. Обновить веса W, для

минимизации потерь можно с помощью следующего уравнения:

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W}$$

Аналогичные действия должны быть сделаны для других весов U, V, b, c.

Теперь вычислим градиенты по $BPTT$ для уравнений RNN выше. Узлы вычислительного графа содержат характеристики U, W, V, B и c, а также последовательность вершин, индексируемых t для $x(t)$, $o(t)$, $h(t)$ и $L(t)$.

Для каждого узла n необходимо вычислить градиент ∇nL рекурсивно, на основе градиента, вычисленного на узлах, следующих за ним в графе.

Градиент по отношению к выходу $o(t)$ вычисляется в предположении, что $o(t)$ используется в качестве аргумента для функции $softmax$ для получения вектора $\hat{y}$. Предположим, что потеря является отрицательной логарифмической вероятностью истинного объекта $\hat{y}$.

$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L(t)} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i=y^{(t)}}$$

Теперь рассмотрим, как градиент протекает через скрытое состояние $h(t)$.

Производится обратная проработка последовательности, начиная с конца последовательности. На последней итерации τ, $h(\tau)$ имеет только $o(\tau)$ как потомок, следовательно его градиент вычисляется легко:

$$\nabla_{\boldsymbol{h}^{(\tau)}} L = \boldsymbol{V}^\top \nabla_{\boldsymbol{o}^{(\tau)}} L$$

Затем вернемся на итерацию назад, для обратного распространения градиентов во времени, от $t = \tau - 1$ до $t = 1$, отмечая, что $h(t)$ (для $t < \tau$) имеет как потомков как $o(t)$ так и $h(t+1)$. Таким образом, градиент определяется:

$$\begin{aligned}\nabla_{\boldsymbol{h}^{(t)}} L &= \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^{\top} \left(\nabla_{\boldsymbol{h}^{(t+1)}} L\right) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^{\top} \left(\nabla_{o^{(t)}} L\right) \\ &= \boldsymbol{W}^{\top} \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t+1)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t+1)}} L\right) + \boldsymbol{V}^{\top} \left(\nabla_{\boldsymbol{o}^{(t)}} L\right)\end{aligned} \tag{1}$$

Как только градиенты на внутренних узлах вычислительного графа получены, получим градиенты на узлах параметров. Расчет градиента с использованием правила цепи для всех параметров:

$$\begin{aligned}\nabla_c L &= \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}}\right)^{\top} \nabla_{\boldsymbol{o}^{(t)}} L = \sum_i \nabla_{\boldsymbol{o}^{(t)}} L \\ \nabla_{\boldsymbol{b}} L &= \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}(t)}\right)^{\top} \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \nabla_{\boldsymbol{h}^{(t)}} L \\ \nabla_{\boldsymbol{V}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla_{\boldsymbol{V}^{(t)}} o_i^{(t)} = \sum_t \left(\nabla_{o^{(t)}} L\right) \boldsymbol{h}^{(t)^{\top}} \\ \nabla_W L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_W^{(t)} h_i^{(t)} = \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \boldsymbol{h}^{(t-1)^{\top}} \\ \nabla_U L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)} = \sum_t \operatorname{diag}\left(1 - \left(h^{(t)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \boldsymbol{x}^{(t)^{\top}}\end{aligned} \tag{2}$$

Четвертый раздел «Проектирование и реализация глубинной нейронной сети» содержит в себе описание алгоритма для стилизации графических образов, а также его реализацию.

Подраздел 4.1 содержит в себе описание процесса переноса нейронного стиля. Сеть использует нейронные представления для разделения и рекомбинации содержания и стиля произвольных изображений, обеспечивая нейронный алгоритм для создания художественных изображений.

Реконструкция содержания. Визуализация информации на различных стадиях обработки в CNN, реконструируя входное изображение, зная только ответы сети на конкретном уровне. Входное изображение обрабатывается на слоях: $conv1\_1\,(a)$, $conv2\_1\,(b)$, $conv3\_1\,(c)$, $conv4\_1\,(d)$ и $conv5\_1\,(e)$

оригинальной сети. Слои (a, b, c).

На более высоких уровнях сети теряется подробная информация о пикселях, в то время как высокоуровневое содержание изображение сохраняется (d, e).

Реконструкция стиля. Помимо оригинальных представлений CNN необходимо построить новое пространство, которое захватывает стиль входного изображения. Представление стиля вычисляет корреляции между различными признаками в различных слоях CNN.

Реконструкция стиля входного изображения из стилевых представлений, построенных на различных подмножествах слоёв CNN ($conv1\_1\,(a)$, $conv1\_1$ и $conv2\_1\,(b)$, $conv1\_1$, $conv2\_1$ и $conv3\_1\,(c)$, $conv1\_1$, $conv2\_1$ $conv3\_1$ и $conv4\_1\,(d)$). Это создает изображения, которые соответствуют стилю данного изображения при увеличении масштаба.

В подразделе 4.2. показан алгоритм, состоящий из 5 шагов, а также программная реализация нейронной сети.

Для изучения и программной реализации была выбрана глубинная нейронная сеть, которая исходя из нескольких полученных входных графических данных, на выходе отдаст стилизованное графическое изображение.

Принцип переноса нейронного стиля заключается в определении двух функций, одна из которых описывает, как различно содержание двух изображений, $L_{content}$, и одна, которая описывает разницу между двумя изображениями с точки зрения их стиля, L_{style}.

Затем, получив три изображения, желаемое изображение стиля, желаемое изображение содержания и входное изображение (инициализированное с изображением содержимого), попытаться преобразовать входное изображение, чтобы минимизировать расстояние между содержимым и его стилевым изображением.

Т.е. берется базовый входной образ, который сопоставляется, и образ стиля, который с которым нужно сопоставить. Мы преобразуем базовое входное изображение, минимизируя расстояния (потери) содержания и стиля с помощью обратного распространения, создавая изображение, которое соответствует содержимому входному изображения и стилю стилизующего изображения.

Общий алгоритм переноса стиля:

1. Визуализировать данные
2. Базовая предварительная обработка/подготовка наших данных
3. Настраивать функции потерь
4. Создавать модель
5. Оптимизация для функции потерь

Content Loss – есть функция, описывающая расстояние между контентом и нашим выходным изображением x и нашим контентным изображением p. Пусть C_{nn} – предварительно обученная глубинна свёрточная нейронная сеть. Пусть X – любое изображение, тогда C_{nn} – это сеть, подаваемая X.

Пусть $F_{ij}^{l}(x) \in C_{nn}(x)$ и $P_{ij}^{l}(p) \in C_{nn}(p)$ описывают соответствующее промежуточное представление признаков сети с входами x и p на слое l. Затем опишем расстояние содержания (потери) формально как:

$$\mathcal{L}_{\text{content}}(p, x) = \sum_{i,j} \left(F_{ij}^{x} - P_{ij}^{p}\right)^2.$$

Выполняется обратное распространение ошибки, чтобы минимизировать потери контента. Таким образом, меняется начальное изображение до тех пор, пока оно не генерирует аналогичный ответ на определенном уровне (определяемом в $content\_layer$) как исходное содержание образа, т.е. если принимать в качестве ввода карты признаков на уровне L в сети, подаваемой входное изображение x, и p – контент изображения, и вернуть расстояние содержания.

Теперь необходимо вычислить $StyleLoss$ вычисления следуют тому же принципу, что описан выше, с разницей, что сейчас на ход идет изображение для стилизации и вместо сравнения первичных промежуточных выходов базового входного изображения и стиля, нужно сравнивнить матрицы Грама двух выходов.

Математически описывается $StyleLoss$ исходного входного изображения x и изображения стиля a, как расстояние между представлением стиля (грамовыми матрицами) этих изображений[16-18]. Описание представление стиля изображения как корреляцию между различными ответами фильтра, заданными матрицей Грама G^{l}, где G_{ij}^{l} – это внутреннее произведение между векторизированной картой признаков i и j в слое l.

Видно, что G_{ij}^{l}, сгенерирован над картой признаков для данного изоб-

ражения и представляет корреляцию между отображениями признаков i и j.

Чтобы создать стиль для базового входного изображения, необходимо выпонить градиентный спуск из контентного изображения, чтобы преобразовать его в изображение, которое соответствует стилю представления исходного изображения.

Происходит это за счет минимизации среднего квадратического расстояния между корреляцией признаков изображения стиля и входного изображения.

Вклад каждого слоя в общую потерю стиля описывается следующим образом:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

и общая потеря

$$\mathcal{L}_{\text{style}}\left(ca, x\right) = \sum_{l=0}^{L} w_l E_l$$

Где w – есть вес слоя и равен, $\frac{1}{|L|}$.

Далее необходимо осуществить градиентный спуск, Т.е. происходит итеративное обновление входного изображения. Для этого рассчитываются *Style Loss* и *Content Loss*.

В пятом разделе «Описание конечного приложения» Разработанный прототип мобильного приложения является контрольной точкой точкой в реализации прикладного применения технологий глубинного обучения в области мобильной разработки.

В его основе лежит ранее реализованный алгоритм глубинного обучения сверточной нейронной сети, применимый в области обработки графических изображений, а именно, в изменении стилистики изображений.Программа реализована на языке *Kotlin*, с использованием интегрированных библиотек *Java* и *Python*. Средой разработки яаляется инструмент *Android Studio*

ЗАКЛЮЧЕНИЕ

В рамках магистерской работы были выполнены следующие задачи:

— Изучена теория разработки мобильных приложений под платформу Android;

— Рассмотрена основная теория по реализации и проектированию глубинных нейронных сетей;
— Проведен сравнительный анализ моделей глубинных нейронных сетей;
— Написана программная реализация глубинной нейронной сети;
— Разработано мобильное приложение с применением полученных знаний.

Все поставленные задачи были выполнены и результаты были представлены на научной студенческой конференции (СНК), кафедральный этап от 20 апреля 2023 года.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bengio, Yoshua Greedy Layer-Wise Training of Deep Networks / Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle// 2007. - Universit´e de Montr´eal Montr´eal, Qu´ebec.- 8 p.
2. Гудфеллоу Я. Глубокое обучение сетей. / Гудфеллоу Я., Бенджио И., Курвилль А. - М. : ДМК, 2018. - 646 с.
3. Meier, Reto The Busy Coder's Guide to Advanced Android Development / Reto Meier // 2018. - John Wiley Sons, Inc. - 868 p.
4. Mew, Kyle Mastering Android Studio / Kyle Mew // 2017. - Packt, Inc. - 432 p.
5. Murphy, Mark L PROFESSIONAL ANDROID 4 / Mark L. Murphy//2011. - CommonsWare. - 555 p.
6. Гафаров, Ф. М. Искусственные нейронные сети и их приложения: учеб. пособие. / Ф. М. Гафаров, А. Ф. Гамильянов: Изд-во Издательство Казанского университета, 2018. - 120 с.
7. Hopfield, John J. Neural networks and physical systems with emergent collective computational abilities./ John J Hopfield: Proceedings of the national academy of sciences 79.8: 1982.- 2554-2558 с.
8. Hayes, Brian. First links in the Markov chain / Brian Hayes. - American Scientist 101.2: 2013. - 252 с.
9. LeCun, Yann. Gradient-based learning applied to document recognition./ Yann LeCun : Proceedings of the IEEE 86.11: 1998.- 2278-2324 с.
10. Zeiler, Matthew D., Deconvolutional networks./ Zeiler, D. Matthew : Computer Vision and Pattern Recognition (CVPR): 2010 IEEE Conference on. IEEE, 2010.

11. William M., A Programmer's Guide to Computer Science/ Brit Springer, Nicholas R.— Madison. : Jaxson Media, 2019.— 190 с.
12. Christopher M. Bishop., Pattern Recognition and Machine Learning./ Bishop Christopher M.- : Springer, 2007.- 729 с.
13. Deep Diffeomorphic Transformer Networks [Электронный ресурс]. URL: http:// www2.compute.dtu.dk/ sohau/papers/cvpr2018/detlefsencvp$r$2018.pdf (дата обращения: 10.12.2021). Загл. с экр. яз. англ.
14. Simonyan, K. Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. [Электронный ресурс]. URL: http: //arxiv.org/abs/1409. (дата обращения: 03.04.2022) Загл. с экр. яз. англ.
15. Russakovsky, O. et al. ImageNet Large Scale Visual Recognition Challenge. [Электронный ресурс]. URL: http://arxiv.org/abs/1409.0575. (дата обращения: 03.04.2022) Загл. с экр. яз. англ.
16. Jia, Y. et al. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the ACM International Conference on Multimedia, [Электронный ресурс]. URL: http://dl.acm.org/citation.cfm?id=2654889. (дата обращения: 02.05.2022) Загл. с экр. яз. англ.
17. Adelson, E. H. Bergen, J. R. Spatiotemporal energy models for the perception of motion. [Электронный ресурс]. URL: http://www.opticsinfobase.org/josaa/fulltext.cfm?uri=josaa-2-2-284. (дата обращения: 02.05.2022) Загл. с экр. яз. англ.
18. Xie, X., Tian, F. Seah, H. S. Feature Guided Texture Synthesis (FGTS) for Artistic Style Transfer. In Proceedings of the 2Nd International Conference on Digital Interactive Media in Entertainment and Arts [Электронный ресурс]. URL: http://doi.acm.org/10.1145/1306813.1306830. (дата обращения: 02.05.2022) Загл. с экр. яз. англ.
19. Glek, Pavel. [Электронный ресурс]. URL: https://neurohive.io/ru/osnovy-data-science/glubokoe-obuchenie-deep-learning-kratkij-tutorial/ дата обращения: 03.05.2023)
20. Farraj,A. K.Queue performance measures for cognitive radios in spectrum sharing systems / A. K. Farraj, S. L. Miller, K. A. Qaraqe//IEEE GLOBECOM Workshops, 2011.— VOL.— 9.— P. 997 - 1001.