МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОПРЕДЕЛЕНИЯ ПРОФПРИГОДНОСТИ ЧЕЛОВЕКА ПО ЕГО ФОТОГРАФИЯМ В СОЦИАЛЬНЫХ СЕТЯХ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Шапкиной Дарьи Игоревны

Научный руководитель
к.ф.-м.н., доцент _____ А. С. Иванов

Заведующий кафедрой
к.ф.-м.н., доцент _____ С. В. Миронов

Саратов 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Многие западные компании и фирмы при приеме сотрудников на работу применяют не только классические методы анализа кандидата на его профессиональную пригодность, такие как интервью и резюме (на которых можно немного подделать результаты, чтобы произвести лучшее впечатление на работодателя), но и современные — психологический поиск маркеров в социальных сетей людей, где они более честные и искренние. В России такая тенденция только начинает внедряться в отделы кадров и найма персонала. Актуальность данной работы обусловлена тем, что нет аналогов приложений, алгоритмов или программ, которые бы помогали психологом массово и быстро обрабатывать социальные профили кандидатов, что в свою очередь занимает много времени; также нейронные сети и компьютерное зрение являются одними из самых актуальных тем для научного исследования в сфере IT, что подтверждается массовыми исследованиями, публикациями и разработками в этой области, а также практическим применением во многих приложениях. Данная работа поможет обрабатывать социальные сети в разы быстрее, чтобы отдел по работе с персоналом мог делать анализ более продуктивно и оперативно.

Целью выпускной квалификационной работы является реализация приложения по автоматической обработке психологических маркеров профиля человека в его социальной сети на основе исследований психологов. В качестве примера для изучения и разработки была взята платформа Instagram, так как именно там есть необходимое количество фотографий для анализа потенциального претендента.

Приложение на вход получает название профиля потенциального кандидата, а в ответ должна автоматически выделять психологические маркеры и паттерны в фотографиях человека для того, чтобы составить представление о его профессиональной пригодности. С точки зрения психологии такими маркерами являются доминирующие цвета профиля, эмоции на фотографиях (положительные: улыбка, удивление, радость; отрицательные: грусть, страх, печаль; а также нейтральное выражение лица). Алгоритмы будут автоматически выделять данные маркеры и показывать собранную информацию в виде выведенной в окно приложения информации. По ним психологи или люди, которые работают в отделе кадров смогут делать дополнительные профессио-

нальные выводы насчет кандидата.

В качестве языка разработки был выбран Python, так как он располагает библиотеками по разработке нейронных сетей и алгоритмами по работе с задачами компьютерного зрения. Задачи будут разделены на три глобальные: работа с цветовыми пространствами для последующего извлечения доминирующего цвета наиболее оптимальным методом, разработка модели нейронной сети и реализация интерфейса.

Исходя из поставленных целей, необходимо выполнить следующие задачи:

- Автоматическое извлечение постов из Instagram и их последующая обработка
- Алгоритмическая реализация формулы цветового отличия CIEDE2000
- Выделение преобладающего цвета из профиля
- Выбор подходящей архитектуры сверточной нейронной сети
- Обучение нейронной сети для распознавания эмоции человека
- Реализация интерфейса приложения

Работа состоит из введения, трех глав, заключения, списка из 22 источников и четырех приложений.

В главе «Модель представления и обработки цвета» описывается основная концепция цветового представления в компьютерном зрении. Рассматривается выбор цветового пространства, формулы и методов для выделения преобладающих цветов в профиле.

В главе «Архитектура сверточной нейронной сети» рассматриваются концепции нейронных сетей ResNet и MobileNet, их достоинства и недостатки. Объясняется причина выбора архитектуры ResNet.

В главе «Разработка приложения» рассматривается внутренняя структура программы, а также подробно объясняется работа ее модулей, классов и функций. Демонстрируются тестовые запуски работы приложения и иллюстрируется интерфейс. Также затрагивается тема каскадов Хаара, их функциональное применение в работе по выделению эмоций.

1 Модель представления и обработки цвета

Восприятие цвета — сложная и малоизученная проблема, которая, не поддается даже самым сложным математическим выражениям. Изучая реализацию алгоритмов квантования цвета, были сделаны выводы относительно цветовых пространств. Оценка цветовых различий является субъективной: когда людей просят выбрать «самое близкое» соответствие для определенного цвета из небольшой палитры, выбор часто отличается. Все зависит от цветового восприятия, а также доказано, что женский глаз более чувствителен к разным оттенкам.

Цветовая модель RGB — один из наиболее широко используемых методов представления цвета в компьютерной графике. Он использует цветовую систему координат с тремя основными цветами: R (красный), G (зеленый), B (синий) [1].

Каждый основной цвет может принимать значение интенсивности от 0 (самый низкий) до 1 (самый высокий). Смешивание этих трех основных цветов с разными уровнями интенсивности дает множество цветов. Совокупность всех цветов, полученных в результате такой линейной комбинации красного, зеленого и синего, образует цветовое пространство RGB в форме куба. В этом процессе цветовой модели RGB, если три цвета накладываются друг на друга с наименьшей интенсивностью, тогда формируется черный цвет [2], а если он добавляется с полной интенсивностью света, то формируется белый цвет. Чтобы получить другой набор цветов, эти основные цвета должны быть наложены друг на друга с разной интенсивностью. Согласно некоторым исследованиям, интенсивность каждого основного цвета может варьироваться от 0 до 255, в результате чего получается почти 16 777 216 цветов. Как и у любой модели у нее есть достоинства и недостатки. [3]

Цветовая модель CIELAB или просто LAB (называют сокращенно) является наиболее точной моделью для отражения восприятия цветов человеком. Однако она также трудна в представлении и реализации. Модель выражает цвет в виде трех значений: светлотой (L-Lightness) и двумя хроматическими компонентами. Канал a — это цвета от темно-зеленого через серый до пурпурного цвета, канал b — это цвета от синего через серый до желтого. Каналы a и b меняются от -128 до 127, а параметр L — от 0 до 100. Нулевое значение цветовых компонентов при яркости 50 соответствует серому цвету в модели

RGB (119, 119, 119). При значении яркости 100 получается белый цвет, при 0 — черный. Сейчас данное пространство является эталоном.

Цветовое пространство CIELAB представляет собой аппаратно-независимую модель «стандартного наблюдателя». Цвета, которые она определяет, не связаны с каким-либо конкретным устройством.

Пространство CIELAB трехмерно и охватывает весь диапазон человеческого восприятия цвета или гаммы. Она основана на цветовой модели человеческого зрения. Оси a* и b* не ограничены, и в зависимости от эталонного белого они могут легко превышать по модулю 150, чтобы охватить человеческий диапазон. Тем не менее, программные реализации часто ограничивают эти значения по практическим причинам. Например, если используется целочисленная математика, обычно фиксируют a* и b* в диапазоне от -128 до 127.

Распространенные определения цветового различия обычно используют формулу вычисления расстояния в евклидовом пространстве, однако стоит заметить что при этом не каждое цветовое пространство является евклидовым со строгой математической точки зрения.

В качестве метода для выделения доминирующих цветов был выбран метод кластеризации k-means — метод k-средних. Идея метода при кластеризации любых данных заключается в том, чтобы минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров. На первом этапе выбираются случайным образом начальные точки (центры масс) и вычисляется принадлежность каждого элемента к тому или иному центру. Затем на каждой итерации выполнения алгоритма происходит перевычисление центров масс — до тех пор, пока алгоритм не сходится [4].

Касательно изображений и фотографий данный метод работает так: каждый пиксель позиционируется в трехмерном пространстве RGB (от 0 до 255 три координаты), где вычисляется расстояние до центров масс. Для оптимизации каждое изображение уменьшается до размера 100 на 100, что сокращает время работы программы в десятки раз [5].

Алгоритм представляет собой версию EM-алгоритма, применяемого также для разделения смеси гауссиан. Он разбивает множество элементов векторного пространства на заранее известное число кластеров k.

Основная идея заключается в том, что на каждой итерации перевычис-

ляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике [6].

Алгоритм завершается, когда на какой-то итерации не происходит изменения внутрикластерного расстояния. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное отклонение V уменьшается, поэтому зацикливание невозможно.

После того, как массив с информацией по точкам получен вызываем метод kmeans, который и будет высчитывать центры масс. В данном методе n - минимальная дистанция до центра масс, чем она больше, тем быстрее работает алгоритм, но и точность в свою очередь меньше. Путем тестирования оптимально было принято использовать n = 10. k - количество кластеров. Для каждой точке в массиве будем искать минимальное расстояние с помощью Евклидовой метрики. Метрика является простой, но не точной и подходит для RGB, так как модель представима в виде кубического трехмерного пространства.

Далее пересчитываем минимальное расстояние между точками. Если расстояние меньше минимального, в список с номером кластера (у нас их 3) записываем точку с нужным цветом. И пересчитываем центры масс для всех точек кластеров. Если минимальное расстояние до точек найдено, прерываем алгоритм, иначе — продолжаем искать.

После того, как классы найдены, возвращаем значения цветов. Альтернативным методом в программе является метод библиотеки OpenCV. Сначала мы определим функцию, которая будет преобразовывать RGB в hex так, чтобы мы могли использовать их в качестве меток для круговой диаграммы. При чтении цвета, который находится в RGB, мы возвращаем строку.: 02x — это просто отображает шестнадцатеричное значение для соответствующего цвета.

Алгоритм KMeans создает кластеры на основе предоставленного количества кластеров. В нашем случае это сформирует кластеры цветов, и эти кластеры будут нашими главными цветами. Метод реализован внутри библиотеки OpenCV, и ничем не отличается от метода, реализованного выше. Однако он более прост в использовании, так как требует только встроенной библиотеки. Метод получается компактным.

Этот метод был представлен для сравнения, так как велись исследования для нахождения наилучшего метода, и несет ознакомительный характер, так как возможны различные варианты реализации. Однако в нем есть недостаток, что используется Евклидова метрика, тогда как в собственном алгоритме можно изменять метрики на более оптимальные и практичные.

Одним из стандартов цветовой модели является пространство CIE XYZ. Все остальные модели можно интерпретировать как разные отображения или подмножества цветового пространства CIE XYZ. CIE - сложная модель, и, что наиболее важно, хотя она определяет пространство всех цветов, которые мы можем различить, она не предлагает перцептуально однородного цветового пространства. Таким образом, расстояние между двумя точками в пространстве CIE-XYZ не имеет значения для относительной близости этих цветов. Поэтому используя пространство CIELAB и не евклидову метрику, а CIEDE2000, можно достичь наиболее точного отображения восприятия, как

$\triangle E = \sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2}$ — отражает поворот цветового угла тона (RT), чтобы устранить проблемы в синей области (угол Hue 275°); компенсацию для нейтральных цветов, для светлоты (SL), для насыщенности цвета (SC), для тона (SH).

2 Архитектура сверточной нейронной сети

Структурной единицей нейронной сети является нейрон. На вход нейрону подается n-мерный вектор признакового описания. Каждому входу соответствую весовой коэффициент. Выходом нейрона является значение функции активации от взвешенной суммы входных значений. Также на вход подается свободный член, имеющий постоянное значение, который также имеет вес. Если нейрон находится в скрытом или выходном слое нейронной сети, то на вход ему подаются выходные значения других нейронов.

Соединяя нейроны и образую слои можно получить полносвязную нейронную сеть или — многослойный персептрон. Нейроны располагаются слоями, при этом каждый нейрон из следующего слоя связан со всеми нейронами предыдущего. Выделяют три типа слоя — входной слой неройнов, скрытый слой и выходной слой. Чаще всего используется трехслойный персептрон, то есть с одним скрытым слоем.

Функция активации нейронов (характеристическая функция) — нейлинейный преобразователь выходного сигнала сумматора. Может быть одной и той же для всех нейронов сети. Большинство функций активации являются симметричными и нечетными.

Сверточные нейронные сети берут истоки от обычных нейронных сетей, базовые понятия которых описаны выше, которые используют статистические данные или какие-либо другие. Сверточные нейронные сети также называют английской аббревиатурой CNN, они состоят из нейронов с обучаемыми весами.Каждый нейрон получает некоторый входной сигнал вследствие чего выполняет точечное произведение с последующих выходным сигналом, который осуществляется с помощью функции активации.

Нейронная сеть получает вход (один вектор) и передает его в скрытый слой для конвертации. Каждый скрытый слой состоит из группы нейронов, где каждый нейрон полностью связан со всеми нейронами в предыдущем слое (полносвязная нейронная сеть). Последний слой нейронов зависит от всех остальных и связан с ними, он также называется «выходным слоем», он отвечает за классификацию изображения.

Сверточные слои являются основой CNN, поскольку они содержат выученные ядра (они представляют собой матрицу с числами), которые извлекают признаки, отличающие различные изображения друг от друга. Каждое зве-

но представляет собой уникальное ядро, которое используется для операции свертки для получения выходного сигнала текущего сверточного нейрона или карты активации. По своей сути двумерная свертка — довольно простая операция: ядро идет по двумерному изображению, поэлементно выполняя операцию умножения с той частью входных данных, над которой оно сейчас находится, и затем суммирует все полученные значения в один выходной пиксель. Ядро повторяет эту процедуру с каждой локацией. Образы на выходе являются, по существу, взвешенными суммами (где веса являются значениями самого ядра) образов на входе, расположенных примерно в том же месте что и выходной пиксель на входном слое.

Шаг указывает, на сколько пикселей должно быть сдвинуто ядро одновременно. Например, нейронная сеть VGG использует шаг 1 для своих сверточных слоев, что означает, что скалярное произведение выполняется на поле ввода 3x3, чтобы получить выходное значение, а затем сдвигается вправо на один пиксель для каждой последующей операции. Влияние исполнения на CNN аналогично влиянию размера ядра. По мере того как шаг становится меньше, чем больше данных извлекается, тем больше функций исследуется, что также приводит к увеличению выходных слоев.

Так как в современных сетях много слоев, то хранить три канала изображения со всеми векторами признаков (учитывая, что как правило изображения берется в размере 224 на 224 пикселя, которые умножаются на 3 канала) это затратно по всем ресурсам, поэтому в нейронных сетях используют пулинговый слой. Он снижает размерность изображения. Исходное изображение делится на блоки размером $w \times h$ и для каждого блока вычисляется некоторая функция. Чаще всего используется функция максимума (max pooling). Обучаемых параметров у этого слоя нет. Основные цели пулингового слоя ускорить вычисления и уменьшить изображение, чтобы свертки оперировали над большей областью изображения, при этом не меняя размеров фильтра.

2.1 ResNET и MobileNet

ResNet является остаточной сверточной сетью и содержит 152 уровня глубины. Однако эталонной сетью считается ResNet с 30-34 слоями, так как когда более глубокая сеть начинает сворачиваться, возникает проблема: с увеличением глубины сети точность сначала увеличивается, а затем быстро ухудшается. Снижение точности обучения показывает, что не все сети легко

оптимизировать. Поэтому как правило каждый блок ResNet имеет два уровня глубины (используется в небольших сетях, таких как ResNet 18, 34) или 3 уровня (ResNet 50, 101, 152).

Обучение сети производилось на базе ImageNet, по своей природе это набор данных миллионов помеченных изображений с высоким разрешением, относящихся примерно к 22 тысячам категорий. Изображения были собраны из Интернета и помечены людьми. Существует около 1.2 миллиона обучающих образов, 50 тысяч валидаций и 150 тысяч тестовых изображений.

MobileNet более современная и легковесная архитектура, но по своей природе она ничем не уступает ResNet. Для примера выборка в данной работе на базе ResNet училась по 7-8 часов, а на базе MobileNet менее 4 часов, при этом результаты оказались даже лучше. MobileNet также легок в своей архитектуре. Он использует отделимые по глубине свертки, что в основном означает, что он выполняет одну свертку для каждого цветового канала, а не объединяет все три и выравнивает их. Это имеет эффект фильтрации входных каналов. Свертка по глубине применяет один фильтр к каждому входному каналу. Затем поточечная свертка применяет свертку 1×1 для объединения выходов по глубине. Стандартная свертка фильтрует и объединяет входы в новый набор выходов за один шаг. Отделимая по глубине свертка разделяет это на два слоя: отдельный слой для фильтрации и отдельный слой для объединения. Эта факторизация приводит к резкому сокращению вычислений и размеров модели.

Архитектура Mobilenet выглядит следующим образом, имея 30 слоев с:

1. сверточный слой с шагом два
2. глубинный слой
3. точечный слой, который удваивает количество каналов
4. глубинный слой с шагом два
5. точечный слой, который удваивает количество каналов

Данные архитектуры сетей в данном разрабатываемом приложении по своей сути взаимозаменяемы, однако MobileNet обучается быстрее. В зависимости от датасета, и выбранных тестовых изображений сети показывают разный результат, однако их разница составляет менее 0.05. В программе в качестве выбранной архитектуры для приложения будет использован MobileNet версии 2.

3 Разработка приложения

Обучение модели производилось в среде разработке Google Colaboratory, так как там поддерживается ускорители GPU и TPU (графические процессоры, которые ускоряют в том числе обучения моделей, так как без ускорителей модель может обучаться даже в течение 12 часов на наборе данных меньше 20.000 фотографий, что является очень медленным процессом). GPU ускоряет процесс примерно в два раза.

Основой для обучения были выбраны архитектуры ResNet50 и MobileNet. В качестве дополнительных инструментов используется библиотека TensorFlow и надстройка над данной библиотекой - модуль Keras [7], а также встроенный в библиотеку OpenCV алгоритм поиска лиц на фотографии методом каскадов Хаара. [8]

Каскады Хаара – это поиск особенностей на изображении, представляют собой прямоугольные волны одинаковой длины. В двух измерениях, прямоугольная волна являет собой пару соседних прямоугольников – один светлый и другой темный.Фактически прямоугольные комбинации, используемые для визуального обнаружения объекта не являются подлинными вейвлетами Хаара. Вместо этого, они содержат прямоугольные комбинации, которые лучше подходят для визуальных задач распознавания. Наличие функции Хаара определяется посредством вычитания среднего значения области темных пикселей из среднего значения области светлых пикселей. Если разница превышает порог, тогда функция является существующей.

В программе данный метод применяется для того, чтобы выделить область лица, а затем именно выделенную область передать модели.

В переменную detected записываются все области, где были обнаружены лица при помощи встроенной функции detectMultiScale, далее идя по всем областям происходит обработка фотографии: каждое найденное лицо «вырезается» [9] из начального изображения и становится отдельным изображением, которое передается в функцию, где происходит предсказание модели.

После загрузки архива изображений необходимо их преобразовать в подготовленные батчи. Это можно сделать вручную, либо посредством встроенных функций в библиотеку машинного обучения. Один из таких методов - генерация батчей из определенного пути каталога.

После разделения выборки на батчи необходимо загрузить предвари-

тельную модель (либо создать собственную, но так как ResNet и MobileNet проверенные архитектуры, то использовать необходимо их с заменой некоторых слоев). В процессе передачи обучения одна предобученную модель используется для классификации собственных изображений. Изменяя ее последний слой, или несколько слоев снова запускается процесс обучения на новом наборе данных.

В TensorFlow Hub можно найти не только полные предобученные модели (с последним слоем), но и модели без последнего классификационного слоя. Последние могут быть с легкостью использованы для передачи обучения. Продолжим использовать MobileNet v2 (ResNet, результаты работы моделей идентичны). Более того быстродействие таких моделей быстрее, чем написанные с нуля.

Частичную модель с TensorFlow (без последнего классификационного слоя) называется `feature_extractor` — модель принимает на вход данные и преобразуем их до конечного набора выделенных свойств (характеристик). Производится также «заморозка» переменных в слое извлечения свойств для того, чтобы в процессе обучения менялись только значения переменных слоя классификации. Далее необходимо обернуть слой из TensorFlow Hub в tf.keras.Sequential-модель и добавить классификационный слой (делим на 3 класса). Используемая функция активации - softmax.

После настройки происходит переобучение модели. Чтобы сохранить веса, которая получила модель выполняется функция `model.save('model.h5')`. Таким образом модель сохранена и обучена. Теперь осталось только перенести ее в класс, связанный с интерфейсом и заново извлечь модель:

```
model.load_weights ("model.h5")
```

Разработка приложения, которое бы отображало необходимые выходные данные (маркеры) было решено делать на базе открытой библиотеки PySide PySide - это проект привязок Python Qt, предоставляющий доступ ко всей структуре Qt 4.8, а также к инструментам генератора для быстрого создания привязок для любых библиотек C ++ [10]. Основным преимуществом является удобный интерфейс разработки и предоставление возможности использования формального языка описания внешнего вида документов HTML - стилей CSS.

Текстовые поля получают значения логина и пароля, обрабатывают ошибку через конструкцию try-except: из файла экспортируется класс Insta со зна-

чениями полей, если сессия входа успешно завершена, то старое окно закрывается и открывается новое; если ошибка «поймана», то выводится сообщение об ошибки через встроенный класс ошибок – QtWidgets.QErrorMessage ().

Второе окно, где вводится имя пользователя, для которого необходимо провести аналитику, содержит такой же стилистический файл, в котором также два текстовых поля, две надписи, одна кнопка, а в дополнение одно большое текстовое поле, в котором выводится результат. Стили содержат те же стили, как и в предыдущем окне для поддержания структуру.

Далее перейдем непосредственно к основному файлу, где импортируется классы интерфейса (два файла со стилями), а также еще два файла: файл с классами Insta, Picture (данные классы описаны выше), с функциями определения основного цвета профиля, а также другой файл с классом Model, который определяет эмоциональный фон профиля. Файлы экспортируются в начале, и с помощью классов используется вместе с библиотекой PyQT5. Из текстового поля textEdit извлекается название профиля, далее идет обработка через другой файл с классами, фотографии экспортируются из папки, в которую загрузилась вся информация профиля.

Есть поле, в котором вводится количество постов. Поле можно оставить пустым – ошибки не будет, так как в классе Insta в функции loading () есть необязательный параметр, поэтому если значение не передать, то останется данный параметр. Если буду введены не целое число, а любой другой символ, то будет выведено сообщение об ошибке, и пока она не будет исправлена обработка профиля не начнется.

После всех пройденных этапов в основное текстовое поле будет выведена вся собранные психологические маркеры: преобладающий цвет профиля и его характеристика (так как она задана и не меняется), а также преобладающие эмоции в профиле (положительный, нейтральные или отрицательные). Это происходит следующим образом: обученная раннее модель, у которой были сохранены веса, находится в отдельном файле, так как веса просто загружаются и не требуется заново обучать модель. Каждая фотография проходит через модель, в которой определяются вероятности принадлежности к одному из трех классов (положительные, нейтральные и отрицательные эмоции), далее высчитывается процентное соотношение преобладающей эмоции, результат работы файла вызывается в основной программе.

ЗАКЛЮЧЕНИЕ

В ходе работы были рассмотрены следующие теоретические аспекты:

- преимущества и недостатки различных цветовых пространств
- сравнение формул цветового отличия
- метод k-средних
- архитектуры сверточных нейронных сетей ResNet и MobileNet
- выделение признаков с помощью каскадов Хаара
- методы разработки интерфейса с библиотекой PyQt5

Также выполнены следующие задачи:

- Реализовано автоматическое извлечение постов из Instagram и их последующая обработка
- Программно реализована формулы цветового отличия CIEDE2000
- Выделены преобладающего цвета из профиля
- Выполнено обучение нейронной сети для распознавания эмоций человека
- Реализован интерфейс приложения на основе библиотеки PyQt5

Результатом выпускной квалификационной работы является работающее приложение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Цветовосприятие человека [Электронный ресурс]. — URL: http://www.brucelindbloom.com/index.html#BluePurple (Дата обращения 19.02.2021). Загл. с экр. Яз. рус.

2 A guide to Understanding Color Tolerancing [Электронный ресурс]. — URL: https://web.archive.org/web/20151010064956/http://www.. (Дата обращения 16.03.2021). Загл. с экр. Яз. англ.

3 Colour difference [Электронный ресурс]. — URL: https://www.researchgate.net/publication/236023905_Co.. (Дата обращения 05.05.2021). Загл. с экр. Яз. англ.

4 Colour metric [Электронный ресурс]. — URL: https://www.compuphase.com/cmetric.htm (Дата обращения 12.05.2021). Загл. с экр. Яз. англ.

5 The CIEDE2000 Color-Difference [Электронный ресурс]. — URL: http://www2.ece.rochester.edu/~gsharma/ciede2000/cied.. (Дата обращения 08.05.2021). Загл. с экр. Яз. англ.

6 Цветовая идентификация в изображениях [Электронный ресурс]. — URL: https://www.machinelearningmastery.ru/color-identific.. (Дата обращения 24.05.2021). Загл. с экр. Яз. рус.

7 Keras [Электронный ресурс]. — URL: https://keras.io/api/preprocessing/image/ (Дата обращения 24.05.2021). Загл. с экр. Яз. англ.

8 Комплексная платформа машинного обучения с открытым исходным кодом [Электронный ресурс]. — URL: https://www.tensorflow.org/ (Дата обращения 24.05.2021). Загл. с экр. Яз. рус.

9 *Шолле, Ф.* Глубокое обучение на Python / Ф. Шолле. — СПб.: Питер, 2018. — С. 340.

10 Python GUI [Электронный ресурс]. — URL: https://www.machinelearningmastery.ru/color-identific.. (Дата обращения 24.05.2021). Загл. с экр. Яз. рус.