Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

ANDROID-ПРИЛОЖЕНИЕ ДЛЯ ПРЕОБРАЗОВАНИЯ РАСЫ НА ИЗОБРАЖЕНИЯХ ЛИЦ С ПОМОЩЬЮ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНОЙ НЕЙРОННОЙ СЕТИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Мачулы Никиты Владимировича

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

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

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Автоматизированная генерация правдоподобных изображений и перенос стиля одного класса изображений на другой — нетривиальные задачи. Существует несколько способов их решения. Один из них — генеративно-состязательные нейронные сети.

Цель бакалаврской работы — создать приложение на платформе Android для преобразования расы на изображениях лиц с помощью генеративно-состязательной нейронной сети.

Поставленные задачи:

- изучить генеративно-состязательные нейронные сети;
- построить модель генеративно-состязательной нейронной сети для решения задачи преобразования изображений из одного домена в другой;
- создать набор данных для обучения нейронной сети;
- обучить построенную модель;
- создать приложение на платформе Android для демонстрации работы модели нейронной сети;
- интегрировать в приложение обученную модель нейронной сети.

Материалы исследования включают статьи, описывающие генеративно-состязательные нейронные сети и наборы данных для их обучения, электронные ресурсы по платформе Android и библиотекам для построения моделей нейронных сетей, решения задач машинного обучения и компьютерного зрения, а также создания программ-серверов.

Работа состоит из трех глав. Глава «Нейронная сеть» содержит теоретические выкладки по генеративно-состязательным нейронным сетям, необходимые для выполнения практической части работы. Глава «Модель нейронной сети и приложение на платформе Android» описывает практическую часть работы. В главе «Результаты» приводятся результаты запусков модели нейронной сети и приложения на платформе Android, а также демонстрируется интеграция двух этих компонент.

1 Нейронная сеть

1.1 Генеративно-состязательная нейронная сеть

Генеративно-состязательная сеть (GAN) — алгоритм машинного обучения без учителя, построенный на комбинации из двух нейронных сетей, одна из которых (сеть G) генерирует образцы, а другая (сеть D) старается отличить настоящие образцы от «поддельных» сгенерированных [1].

Пусть p_{data} — выборка из базы данных реальных изображений, x — изображение, $D(x, \theta_d) \in [0, 1]$ — дискриминатор, представляющий собой нейронную сеть, θ_d — его параметры, $p_g(x)$ — распределение генератора. Пусть $p_z(z)$ — априорное распределение шума, который отображается в пространство изображений генератором $G(z, \theta_g)$, где θ_g — его параметры.

В процессе обучения генеративно-состязательной нейронной сети повышается вероятность правильного определения дискриминатором D класса принимаемых данных — настоящих или сгенерированных.

$$max_D \, \mathbb{E}_{x \sim p_{data}}[log \; D(x)] + \mathbb{E}_{x \sim p_g}[log(1 - D(x))] \tag{1}$$

С точки зрения генератора, необходимо, чтобы дискриминатор как можно хуже отличал фальшивку от реальных изображений, что выражается в формуле 2.

$$min_G \, \mathbb{E}_{z \sim p_z(z)}[1 - D(G(z))] \tag{2}$$

Объединив задачи минимизации генератора и максимизации дискриминатора в одну оптимизационную задачу, получаем минимаксную игру для двух игроков, выражаемую формулой 3.

$$\min_G \max_D \, V(D, G) = \mathbb{E}_{x \sim p_{data}}[log \; D(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))] \tag{3}$$

Преимущества генеративно-состязательных нейронных сетей перед другими подходами к решению задачи генерации:

- для получения градиентов достаточно использовать метод обратного распространения ошибки;
- не требуется использование логических выводов;
- в модель может быть включен широкий класс функций;
- возможность представления распределений с очень большим коэффици-

ентом эксцесса.

Недостатки данного типа нейронных сетей:

— нет явного представления распределения генератора $p_g(x)$;
— во время обучения дискриминатор D постоянно должен быть синхронизированным с генератором G, потому что, в противном случае, возможен сценарий, при котором будут генерироваться почти неотличимые данные.

1.2 CycleGAN

CycleGAN — архитектура генеративно-состязательной нейронной сети, предназначенной для решения задач, для которых невозможно построить биективное отображение между двумя доменами. Задача нейронных сетей такой архитектуры — научиться строить такое отображение $G : X \to Y$ для доменов X и Y, что $G(Y)$ неотличимо от Y, используя состязательную функцию потерь. При этом строится обратное отображение $F : Y \to X$ такое, что $F(G(X)) \approx X$, используя цикличную связную функцию потерь [2].

Обучаются две сети генераторов G и F. G учится по входному изображению из домена X генерировать изображение из домена Y. F, наоборот, учится по входному изображению из домена Y генерировать изображение из домена X. Дискриминаторы D_X и D_Y помогают обучению.

Состязательная функция потерь выражается формулой 4. Генератор $G : X \to Y$ пытается минимизировать, тогда как дискриминатор D — максимизировать результат $min_G\, max_{D_Y}\, L_{GAN}(G, D_Y, X, Y)$.

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[log\ D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[log(1 - D_Y(G(x)))] \quad (4)$$

Также вводится цикличная последовательная функция потерь по формуле 5. Назначение ее состоит в том, чтобы изображение из домена X, последовательно пройдя через G и F, было максимально похожим на исходное, или $F(G(x)) \approx x$.

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|] + \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] \quad (5)$$

Таким образом, целевая функция выражается формулой 6, где λ — гипер-

параметр, контролирующий вес цикличной последовательной функции потерь.

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F) \quad (6)$$

Задача оптимизации определяется по формуле 7.

$$G^*, F^* = arg \min_{F,G} \max_{D_X, D_Y} L(G, F, D_X, D_Y) \quad (7)$$

Также решается проблема изменения генераторами цветовой гаммы оригинального изображения с помощью введения дополнительной функции потерь по формуле 8 для случаев, когда на вход генератора некоторого домена подается изображение того же домена.

$$L_{identity}(G, F) = \mathbb{E}_{y \sim p_{data}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{data}(x)}[\|F(x) - x\|_1] \quad (8)$$

На практике генератор состоит из трех частей — кодирования, преобразования и декодирования. Часть кодирования состоит из трех сверточных слоев с шагом 2. Часть преобразования состоит из нескольких остаточных слоев. Обычно их 9, но точное количество зависит от задачи. Часть декодирования состоит из двух обратных сверточных слоев и одного сверточного слоя. Дискриминатор состоит четырех сверточных слоев с полулинейной функцией активации с «утечкой», которая определяется по формуле $max(x, l)$, где x — вес, l — параметр «утечки».

Для стабилизации обучения в формуле 4 отрицательный логарифм правдоподобия заменяется на среднюю квадратичную ошибку и дискриминатор обновляется при помощи буфера из 50 изображений, сгенерированных ранее, а не тех, что были созданы последним в ходе выполнения процедуры обучения генератором. Значение λ в формуле 6 принимают равным 10, используется оптимизатор Адама с размером выборки, равным 1, и коэффициент скорости обучения принимают за 0.0002.

Авторы CycleGAN продемонстрировали результаты работы нейронной сети такой архитектуры на примере переноса стиля между несколькими парами доменов, например, между доменами изображений лошадей и зебр [3], используя для реализации библиотеку PyTorch, обладающую большой гибкостью.

2 Модель нейронной сети и приложение на платформе Android

2.1 Модель нейронной сети

Библиотека Tensorflow [4] используется для реализации модели нейронной сети. Сети генератора и дискриминатора описаны в модуле models. В модуле train описаны узлы статического графа вычислений. В модуле ops реализованы вспомогательные функции среднего квадратичного отклонения и среднего отклонения.

Для обучения созданной модели генеративно-состязательной нейронной сети необходим набор данных. В качестве источников изображений используется UTKFace — набор данных, представленный в двух вариациях: необработанные изображения и центрированные по лицу, обрезанные по длине и ширине до 200 пикселей изображения [5]. Также UTKFace сопровожден дополнительной информацией по каждому изображению, такой как пол, возраст, раса, имя файла.

Преобразованный набор данных содержит 4 части — обучающая и тестовая для первой расы, обучающая и тестовая для второй расы. Изображения отфильтрованы по расе (европеоидная, негроидная и монголоидная) и по возрасту с интервалом от 18 до 50 лет.

Для обучения модели используется облачная платформа FloydHub, предоставляющая платные услуги по аренде серверов, подходящих для задач обучения, тестирования моделей нейронных сетей, требующих больших вычислительных ресурсов [6]. Управление платформой производится посредством командной строки.

Модуль train предназначен для запуска обучения на платформе. С помощью вспомогательных модулей data и utils производится загрузка изображений.

Модуль test предназначен для запуска тестирования. Результаты запуска тестирования обученной модели нейронной сети представлены в разделе 3

2.2 Приложение на платформе Android

Платформа Android — операционная система для смартфонов, планшетов, электронных книг, цифровых проигрывателей, наручных часов, фитнес-браслетов, игровых приставок, ноутбуков и многих других устройств. Основана на ядре Linux и собственной реализации виртуальной машины Java от

Google [7].

Структура проекта приложения строго регламентирована. В папке java и ее подпапках находится весь код приложения. Папка res используется для файлов-ресурсов различного типа, например, рисунков, констант и макетов.

Во многих популярных приложениях пользователь может выбирать изображение из галереи устройства, а также пользоваться камерой устройства для того, чтобы получить изображение «на месте».

Макет экрана выбора изображения описан в файле activity_camera.xml и содержит несколько элементов. Центральный элемент — объект CameraView, предоставляющий возможность работать с камерой устройства, выводить поток данных, поступаемый с камеры, на дисплей, переключать камеры, фокусировку и вспышку. Дополнительные элементы — кнопка перехода в галерею устройства, кнопка переключения камеры, а также кнопка генерации снимка. Эти элементы реализованы с помощью объекта представления ImageButton. Размещение всех элементов происходит внутри корневого ConstraintLayout. Демонстрация экрана выбора изображения приведена в разделе 3.

Поведение экрана выбора изображения activity_camera описывается в классе CameraActivity. Объектам макета соответствуют java-объекты, которые имеют доступ к их свойствам.

На экране преобразования изображения пользователь должен иметь возможность выбрать целевую расу и получить результат преобразования.

Макет экрана преобразования изображения описан в файле формата XML activity_transfer и содержит два основных элемента — объект ImageView, являющийся контейнером для объекта класса Bitmap, и объект ListView, представляющий собой список элементов с возможностью выбора одного или нескольких из них. Размещение всех элементов происходит внутри корневого ConstraintLayout. Демонстрация экрана преобразования изображения приведена в разделе 3.

Поведение экрана преобразования изображения описывается в классе TranserActivity . Этот класс переопределяет несколько методов, каждый из которых запускается в определенные моменты в ответ на действия пользователя.

2.3 Интеграция

Чтобы связать приложение на платформе Android с обученной моделью генеративно-состязательной нейронной сети, необходимо создать двусто-

роннее взаимодействие, с помощью которого приложение будет отправлять данные модели, а она, в свою очередь, будет принимать и преобразовывать данные, а после отправлять результат своей работы приложению.

Для реализации подобного взаимодействия подходит архитектура «клиент-сервер». В качестве клиента будет выступать само приложение, а в качестве сервера — программа-сервер написанная на языке Python, которая после предварительной обработки входных данных будет запускать преобразование этих данных с помощью обученной модели нейронной сети. Взаимодействие реализуется через протокол HTTP прикладного уровня модели OSI.

Так как передача данных посредством протокола HTTP является достаточно продолжительной по времени операцией, то ее выполнение заблокирует главный поток, в котором происходит взаимодействие пользователя с приложением. Чтобы избежать данной проблемы, можно выполнить код в другом потоке, для чего используется класс AsyncTask. Для реализации механизма отправки выбранного изображения предназначен метод transformFaceBitmap класса ImageTransformer — наследника класса AsyncTask.

Перед передачей через HTTP-соединение объект класса Bitmap должен быть преобразован в массив байтов в учетом формата принимаемых моделью нейронной сети изображений — JPEG. Отправка изображения реализуется посредством метода запроса POST и класса HttpURLConnection.

Для создания сервера используется библиотека Flask — микрофреймворк для создания веб-сайтов, использующий Werkzeug, набор инструментов WSGI, стандартного интерфейса Python для развертывания веб-приложений и взаимодействия между ними и различными серверами разработки, а также шаблонизатор Jinja2 [8].

Код сервера описан в модуле server. После загрузки изображения на сервер функция get_prediction считывает изображение в массив ndarray библиотеки numpy [9]. Функция tranform_image находит положение лица на изображении при помощи модуля face_detector, вырезает найденное лицо с помощью модуля image_cropper, приводя его к размеру 200×200 пикселей, преобразует лицо, используя модуль запуска нейронной сети domain_transfer, и вставляет преобразованное лицо в исходное изображение при помощи модуля image_blender. При успешном преобразовании сервер посылает клиенту файл изображения.

3 Результаты

Результаты преобразования из европеоидной в негроидную расу представлены на рисунках 1 и 2:

Рисунок 1 – Исходное изображение лица европеоидной расы, изображение, преобразованное из исходного в негроидную расу и изображение лица исходной расы, преобразованное из негроидной расы

Рисунок 2 – Исходное изображение лица негроидной расы, изображение, преобразованное из исходного в европеоидную расу и изображение лица исходной расы, преобразованное из европеоидной расы

Результаты преобразования из европеоидной в монголоидную расу представлены на рисунках 3 и 4:

Демонстрация интеграции приложения с обученной моделью генеративно-состязательной нейронной сети производится путем загрузки изображения в приложение с помощью первого экрана. На рисунке 5 продемонстрированы экраны выбора и преобразования изображений, а также продемонстрирован процесс преобразования изображения лица европеоидной расы при выборе элемента списка «Негроид».

Рисунок 3 – Исходное изображение лица европеоидной расы, изображение, преобразованное из исходного в монголоидную расу и изображение лица исходной расы, преобразованное из монголоидной расы

Рисунок 4 – Исходное изображение лица монголоидной расы, изображение, преобразованное из исходного в европеоидную расу и изображение лица исходной расы, преобразованное из европеоидной расы

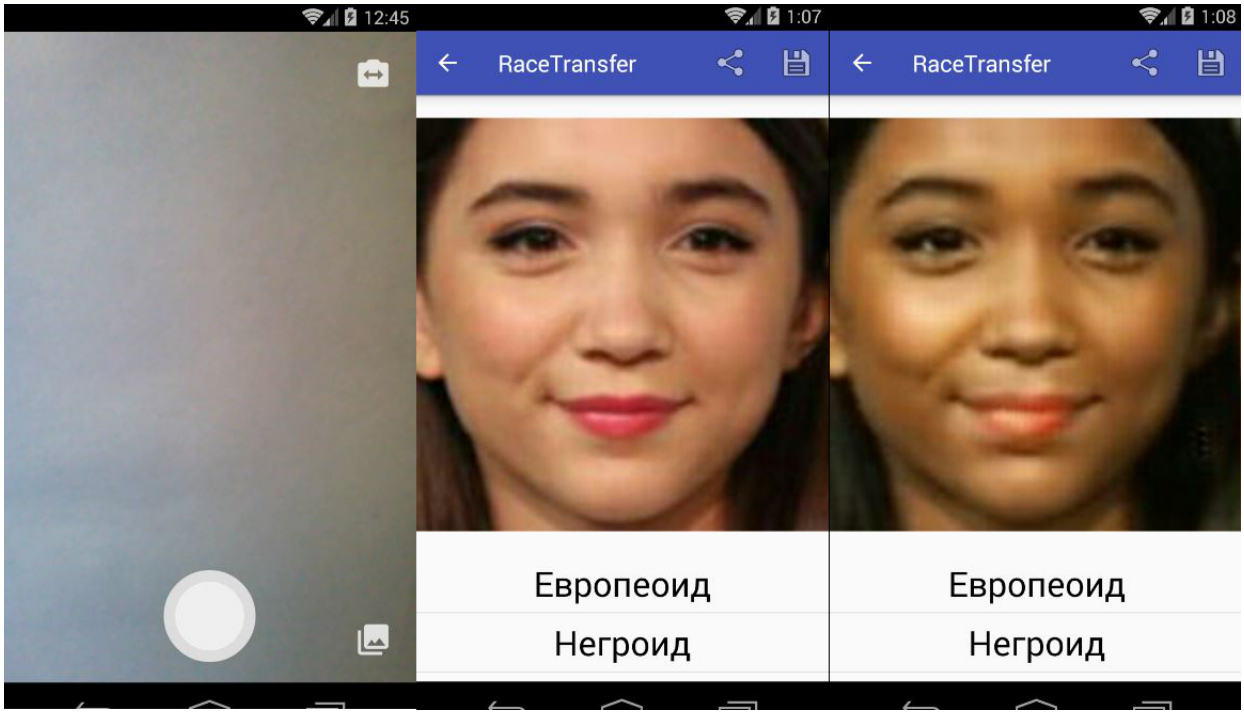

Рисунок 5 – Экраны выбора и преобразования изображений, отображающие процесс преобразования изображения лица европеоидной расы при выборе элемента списка «Негроид»

ЗАКЛЮЧЕНИЕ

Цель — создать приложение на платформе Android для преобразования расы на изображениях лиц с помощью генеративно-состязательной нейронной сети — достигнута.

Все поставленные задачи выполнены. Изучены генеративно-состязательные нейронные сети. Построена модель генеративно-состязательной нейронной сети для решения задачи преобразования изображений из одного домена в другой. Создан набор данных для обучения нейронной сети. Обучена построенную модель. Создано приложение на платформе Android для демонстрации работы модели нейронной сети. Обученная модель нейронной сети интегрирована в приложение.

Работа показала возможность использования генеративно-состязательных нейронных сетей для решения задач автоматизированной генерации правдоподобных изображений и переноса стиля одного класса изображений на другой, в частности, преобразования расы на изображениях лиц.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Generative adversarial networks / I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio.

2 *Zhu, J.* Unpaired image-to-image translation using cycle-consistent adversarial networks / J. Zhu, T. Park, P. Isola, A. A. Efros // *CoRR*. — 2017. — Vol. abs/1703.10593. http://arxiv.org/abs/1703.10593.

3 CycleGAN and pix2pix in PyTorch [Электронный ресурс]. — URL: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix (Дата обращения 19.02.2018). Загл. с экр. Яз. англ.

4 TensorFlow [Электронный ресурс]. — URL: https://www.tensorflow.org/ (Дата обращения 02.03.2018). Загл. с экр. Яз. англ.

5 UTKFace: Large Scale Face Dataset [Электронный ресурс]. — URL: https://susanqq.github.io/UTKFace/ (Дата обращения 27.03.2018). Загл. с экр. Яз. англ.

6 FloydHub - Deep Learning Platform - Cloud GPU [Электронный ресурс]. — URL: https://www.floydhub.com (Дата обращения 08.05.2018). Загл. с экр. Яз. англ.

7 Android Developers [Электронный ресурс]. — URL: https://developer.android.com/ (Дата обращения 09.04.2018). Загл. с экр. Яз. англ.

8 Welcome | Flask (A Python Microframework) [Электронный ресурс]. — URL: http://flask.pocoo.org/ (Дата обращения 21.04.2018). Загл. с экр. Яз. англ.

9 Numpy [Электронный ресурс]. — URL: http://www.numpy.org/ (Дата обращения 18.03.2018). Загл. с экр. Яз. англ.