Содержание

Об авторе ... 10

О рецензенте ... 11

Предисловие ... 12

Глава 1. Введение в порождающие состязательные сети ... 16
Что такое порождающие сети ... 16
Что такое сеть генератора ... 16
Что такое сеть дискриминатора ... 17
Обучение сети GAN посредством состязательной игры ... 17
Практические применения сетей GAN ... 17
Детализация архитектуры сети GAN ... 18
Архитектура генератора ... 19
Архитектура дискриминатора ... 20
Важные понятия, связанные с сетью GAN ... 21
Алгоритмы оценки ... 23
Варианты сетей GAN ... 25
Глубокие порождающие состязательные сети свертки ... 25
Сеть StackGAN ... 25
Сеть CycleGAN ... 25
Сеть 3D-GAN ... 26
Сеть Age-cGAN ... 26
Сеть pix2pix ... 26
Преимущества сетей GAN ... 27
Проблемы обучения сетей GAN ... 27
Режим коллапса ... 27
Исчезающие градиенты ... 28
Внутренний ковариантный сдвиг ... 28
Решение проблем стабильности при обучении сетей GAN ... 29
Соответствие характеристик ... 29
Мини-пакетная дискриминация ... 29
Усреднение истории ... 31
Одностороннее сглаживание маркировки ... 31
Пакетная нормализация ... 31
Нормализация образцов ... 32
Резюме ... 33

Глава 2. Сеть 3D-GAN – генерация форм 3D с использованием сетей GAN ... 34
Введение в сети 3D-GAN ... 34
Свертки 3D ... 35

Архитектура сети 3D-GAN ... 35
Целевая функция ... 40
Обучение сетей 3D-GAN ... 40
Создание проекта ... 40
Подготовка данных ... 41
Загрузка и извлечение набора данных ... 41
Изучение набора данных ... 42
Реализация сети 3D-GAN в Keras ... 45
Сеть генератора ... 45
Сеть дискриминатора ... 46
Обучение сети 3D-GAN ... 48
Обучение сетей ... 48
Сохранение моделей ... 51
Тестирование моделей ... 51
Визуализация потерь ... 52
Визуализация графов ... 53
Оптимизация гиперпараметров ... 53
Практическое применение сетей 3D-GAN ... 54
Резюме ... 55

Глава 3. Старение лица с использованием условной сети cGAN ... 56
Введение в сети cGAN для старения лица ... 56
Понимание сетей cGAN ... 57
Архитектура сети Age-cGAN ... 58
Этапы обучения сети Age-cGAN ... 59
Создание проекта ... 60
Подготовка данных ... 61
Загрузка набора данных ... 62
Извлечение набора данных ... 62
Реализация сети Age-cGAN в Keras ... 63
Сеть кодировщика ... 64
Сеть генератора ... 66
Сеть дискриминатора ... 69
Обучение сетей cGAN ... 71
Обучение сети cGAN ... 71
Аппроксимация начального скрытого вектора ... 77
Оптимизация скрытого вектора ... 79
Визуализация потерь ... 81
Визуализация графов ... 82
Практические применения сетей Age-cGAN ... 82
Резюме ... 83

Глава 4. Создание анимационных персонажей с использованием сети DCGAN ... 84
Введение в сети DCGAN ... 85
Детали архитектуры сети DCGAN ... 85
Создание проекта ... 92

Загрузка и подготовка набора данных анимационных персонажей ... 93
Загрузка набора данных ... 93
Изучение набора данных ... 93
Обрезка и изменение размера изображений в наборе данных ... 94
Реализация сети DCGAN с использованием Keras ... 96
Генератор ... 96
Дискриминатор ... 98
Обучение сети DCGAN ... 101
Загрузка образцов ... 101
Построение и компиляция сетей ... 102
Обучение сети дискриминатора ... 104
Обучение сети генератора ... 104
Генерация изображений ... 105
Сохранение модели ... 106
Визуализация генерированных изображений ... 107
Визуализация потерь ... 108
Визуализация графов ... 109
Настройка гиперпараметров ... 109
Практические применения сети DCGAN ... 110
Резюме ... 111

Глава 5. Использование сетей SRGAN для создания реалистичных фотоизображений ... 112
Введение в сети SRGAN ... 113
Архитектура сети SRGAN ... 113
Целевая функция обучения ... 117
Создание проекта ... 118
Загрузка набора данных CelebA ... 119
Реализация сети SRGAN в Keras ... 120
Сеть генератора ... 120
Сеть дискриминатора ... 124
Сеть VGG19 ... 127
Состязательная сеть ... 128
Обучение сети SRGAN ... 129
Построение и компиляция сетей ... 129
Обучение сети дискриминатора ... 132
Обучение сети генератора ... 132
Сохранение моделей ... 133
Визуализация генерированных изображений ... 134
Визуализация потерь ... 135
Визуализация графов ... 136
Практическое применение SRGAN ... 137
Резюме ... 137

Глава 6. Сети StackGAN – синтез текста в реалистичные фотоизображения ... 138
Введение в сети StackGAN ... 138
Архитектура сети StackGAN ... 139

Сеть кодировщика текста 140
Блок расширения условий 140
Этап I 141
Этап II 145
Создание проекта 151
Подготовка данных 152
Загрузка набора данных 152
Извлечение набора данных 152
Изучение набора данных 153
Реализация сети StackGAN в Keras 153
Этап I 153
Этап II 161
Обучение сети StackGAN 169
Обучение сети StackGAN этапа I 169
Обучение сети StackGAN этапа II 176
Визуализация генерируемых изображений 180
Визуализация потерь 181
Визуализация графов 182
Практические применения сети StackGAN 182
Резюме 183

Глава 7. Сети CycleGAN – превращение картин в фотографии 184
Введение в сети CycleGAN 185
Архитектура сети CycleGAN 186
Целевая функция обучения 189
Настройка проекта 191
Загрузка набора данных 192
Реализация сети CycleGAN с Keras 192
Сеть генератора 193
Сеть дискриминатора 195
Обучение сети CycleGAN 197
Загрузка набора данных 197
Построение и компиляция сетей 198
Начало обучения 201
Сохранение модели 203
Визуализация генерируемых изображений 204
Визуализация потерь 205
Визуализация графов 206
Практическое применение сетей CycleGAN 207
Резюме 208
Дальнейшее чтение 208

Глава 8. Условная сеть GAN – преобразование изображения в изображение с использованием условных состязательных сетей 209
Введение в сети pix2pix 210
Архитектура сети pix2pix 210
Целевая функция обучения 216
Создание проекта 217

Подготовка данных....218
Визуализация изображений....220
Реализация сети pix2pix в Keras....222
Сеть генератора....222
Сеть дискриминатора....228
Состязательная сеть....232
Обучение сети pix2pix....234
Сохранение моделей....238
Визуализация генерированных изображений....239
Визуализация потерь....240
Визуализация графов....241
Практические применения сети pix2pix....241
Резюме....242

Глава 9. Прогнозирование будущего сетей GAN....243
Наш прогноз будущего сетей GAN....244
Совершенствование существующих методов глубокого обучения....244
Эволюция коммерческих приложений сетей GAN....245
Совершенствование процесса обучения сетей GAN....245
Потенциальные будущие применения сетей GAN....245
Создание инфографики из текста....245
Создание дизайна сайта....245
Сжатие данных....246
Открытие и разработка лекарственных препаратов....246
Сети GAN для генерации текста....246
Сети GAN для генерации музыки....246
Изучение сетей GAN....246
Резюме....247

Предметный указатель....248

Об авторе

Кайлаш Ахирвар (Kailash Ahirwar) – энтузиаст машинного обучения и глубокого обучения. Он работал во многих областях искусственного интеллекта (ИИ) – от обработки естественного языка и компьютерного зрения до моделирования с использованием GAN. Является соучредителем и техническим директором компании Mate Labs. Ахирвар применяет GAN для построения различных моделей, таких как превращение рисунков в фотографии и управление глубоким синтезом изображений с помощью текстурных исправлений.

Он очень оптимистичен в отношении AGI и считает, что искусственный интеллект станет рабочей лошадкой эволюции человека.

Эта книга не была бы возможна без помощи моей семьи. Она поддерживала и поощряла меня во время этой работы. Я хотел бы поблагодарить Рахула Вишвакарму (Rahul Vishwakarma) и всю команду Mate Labs за их поддержку. Кроме того, большое спасибо Руби Мохан (Ruby Mohan), Ниту Даниэль (Neethu Daniel), Абхишеку Кумару (Abhishek Kumar), Танаи Агарвалу (Tanay Agarwal), Амаре Ананд Кумар (Amara Anand Kumar) и другим за их ценный вклад.

О рецензенте

Джалай Танаки (Jalaj Thanaki) – известный ученый, имеющая опыт работы в сфере информационных технологий, издательской деятельности и финансов. Она является автором книги «Обработка естественного языка и решение задач машинного обучения на языке программирования Python» (Python Natural Language Processing and Machine Learning Solutions), опубликованной издательством Packt Publishing.

Ее научные интересы лежат в области обработки естественного языка, машинного обучения, глубокого изучения и анализа больших данных. Джалай также является путешественником и любителем природы.

Предисловие

Порождающие состязательные сети (GAN) являются потенциальной перспективой построения сетевых моделей следующего поколения, поскольку обладают возможностью имитировать любые распределения данных. В этой быстро растущей области машинного обучения (Machine Learning, ML) ведутся многочисленные исследования. В данной книге приведены сквозные проекты построения сетей GAN с обучением без учителя.

Проекты порождающих состязательных сетей в книге начинаются с изложения концепций, инструментов и библиотек, применяющихся для создания эффективного проекта. В проектах используются различные наборы данных. Уровень сложности операций, необходимых для реализации проекта, с каждой главой увеличивается, что способствует лучшему пониманию этих проектов.

Вы познакомитесь с практической реализацией популярных проектов, таких как сети 3D-GAN, DCGAN, StackG иА NCycleGAN, их архитектурой и функционированием моделей.

Изучив проекты этой книги, вы будеты готовы создавать, обучать и оптимизировать в своих проектах сквозные модели сетей GAN.

Для кого эта книга

Если вы – специалист по данным, разработчик ML, специалист по глубокому обучению или энтузиаст искусственного интеллекта (AI) и ищете руководство по проекту, чтобы расширить свои знания и опыт в создании реальных моделей сетей GAN, – эта книга для вас.

Содержание книги

Глава 1 «Введение в порождающие состязательные сети» начинается с концепции сетей GAN. Читатели узнают, что такое дискриминатор, что такое генератор и что такое теория игр. Следующие несколько тем будут охватывать архитектуру генератора, архитектуру дискриминатора, целевые функции для генераторов и дискриминаторов, алгоритмы обучения сетей GAN, расходимости Кульбака–Лейблера и Дженсена–Шеннона, матрицы оценки для GAN, различные проблемы с GAN, проблемы исчезающих и взрывных градиентов, равновесие Нэша, пакетную нормализацию и регуляризацию в сетях GAN.

Глава 2 «Сеть 3D-GAN – генерация форм 3D с использованием сетей GAN» начинается с краткого введения в 3D-GAN и различных архитектурных деталей. В этой главе мы будем обучать 3D-GAN генерировать реальные 3D-формы. Мы создадим код для сбора набора данных 3D Shapenet, его очистки и подготовки к обучению. Затем напишем код для 3D-GAN с библиотекой глубокого обучения Keras.

В главе 3 «Старение лица с использованием условной сети cGAN» читатели знакомятся с условными порождающими состязательными сетями (cGAN) и Age-cGAN. Мы изучим различные этапы подготовки данных, такие как загрузка, очистка и форматирование данных. Будем использовать набор данных IMDb Wiki Images. Напишем код для сети Age-cGAN с применением инфраструктуры Keras. Далее мы обучим сеть на наборе данных IMDb Wiki Images. Наконец, мы будем генерировать изображения, используя нашу обученную модель и возраст в качестве нашего условного аргумента. Обученная модель будет генерировать изображения лица человека в разных возрастах.

Глава 4 «Создание анимационных персонажей с использованием сети DCGAN» начинается с введения в сеть DCGAN. Мы изучим различные этапы подготовки данных, такие как сбор данных аниме-персонажей, очистка набора данных и подготовка его к обучению. Рассмотрим реализацию Keras для сети DCGAN в ноутбуке Jupyter. Далее изучим различные способы обучения DCGAN и выберем для нее различные гиперпараметры. Наконец, мы будем генерировать аниме-персонажей, используя нашу обученную модель. Также обсудим практическое применение DCGAN.

Глава 5 «Использование сети SRGAN для создания реалистичных фотоизображений» объясняет, как обучить SRGAN для генерации фотореалистичных изображений. Первым шагом в процессе обучения является сбор набора данных с последующей его очисткой и форматированием для обучения. Читатели узнают, где собрать набор данных, как его очистить и как перевести в нужный для обучения формат.

Глава 6 «Сеть StackGAN – синтез текста в реалистичные фотоизображения» начнется с введения в сеть StackGAN. Сбор данных и подготовка данных являются важными шагами, и мы изучим процесс сбора и подготовки набора данных, его очистки и форматирования. Мы напишем код для сети StackGAN в Keras внутри ноутбука Jupyter. Далее обучим сеть на наборе данных CUB. Наконец, после того как закончим обучение модели, мы сгенерируем фотореалистичные изображения из текстовых описаний. Обсудим различные отраслевые приложения StackGAN и способы их использования в производстве.

Глава 7 «Сеть CycleGAN – превращение картин в фотографии» объясняет, как обучить CycleGAN превращать картины в фотографии. Мы начнем с введения в CycleGAN и рассмотрим их различные приложения. Разберем различные методы сбора данных, очистки данных и формирования данных. Далее напишем коды реализации CycleGAN в Keras и получим подробное объяснение кода в ноутбуке Jupyter. Мы будем обучать сеть CycleGAN на подготовленном нами наборе данных и протестируем нашу обученную модель, чтобы превратить картины в фотографии. Наконец, рассмотрим практическое применение CycleGAN.

Глава 8 «Условная сеть GAN – преобразование изображения в изображение с использованием условных состязательных сетей» рассказывает, как подготовить условную сеть GAN для трансляции изображения в изображение. Мы начнем с введения в условные сети GAN и описания различных методов подготовки данных, таких как сбор данных, очистка данных и форматирование данных.

Далее напишем код для условной сети GAN в Keras внутри ноутбука Jupyter. Потом узнаем, как обучить условную сеть GAN на подготовленном нами наборе данных. Мы будем изучать различные гиперпараметры для обучения. Наконец, протестируем условную сеть GAN и обсудим различные варианты использования преобразования изображения в изображение в реальных приложениях.

Глава 9 «Прогнозирование будущего сетей GAN» является последней главой. Изучив основы сетей GAN и приведенные в книге проекты, в этой главе читатель получит представление о будущем сетей GAN. Здесь мы рассмотрим, как в последние 3–4 года внедрение сетей GAN было феноменальным и насколько хорошо индустрия приняла его. Я расскажу также о моем личном видении будущего сетей GAN.

Чтобы получить максимальную отдачу от этой книги

Требуется знание глубокого обучения, библиотеки Keras и некоторые предварительные знания TensorFlow. Опыт кодирования в Python 3 был бы полезен.

Используемые условные обозначения

В этой книге используется ряд текстовых обозначений.

`Код в тексте`: указывает кодовые слова в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, пути, фиктивные URL-адреса, ввод пользователя и маркеры Twitter. Вот пример: «Используйте функцию `loadmat()` из `scipy` для извлечения вокселей».

Блок кода устанавливается следующим образом:

```
import scipy.io as io
voxels = io.loadmat ("путь к .mat file") ['пример']
```

Любой ввод или вывод командной строки записывается так:

```
pip install -r requirements.txt
```

Полужирный: обозначает новый термин, важное слово или слова, которые вы видите на экране.

Так будут оформляться предупреждения и важные примечания.

Так будут оформляться советы или рекомендации.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте **www.dmkpress.com**, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Так-

же можно послать письмо главному редактору по адресу **dmkpress@gmail.com**; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу **http://dmkpress.com/authors/publish_book/** или напишите в издательство по адресу **dmkpress@gmail.com**.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте **www.dmkpress.com** или **www.дмк.рф** на странице с описанием соответствующей книги.

Пакет кода для книги также размещен на GitHub на **https://github.com/PacktPublishing/Generative-Adversarial-Networks-Projects**.

В случае обновления кода он будет обновлен в существующем репозитории GitHub.

У нас есть другие комплекты кода из нашего богатого каталога книг и видео, доступных по адресу: **http://github.com/PacktPublishing/**. Проверьте их!

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу **dmkpress@gmail.com**, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты **dmkpress@gmail.com**.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Глава 1

Введение в порождающие состязательные сети

В этой главе мы рассмотрим **порождающие состязательные сети** (Generative Adversarial Networks, в дальнейшем GAN). Это тип архитектуры глубоких нейронных сетей, использующий для генерации данных обучение без учителя. Они были предложены в 2014 году в работе Яна Гудфеллоу (Ian Goodfellow), Йошуа Бенжио (Yoshua Bengio) и Аарона Курвиля (Aaron Courville), которую можно найти по адресу: **https://arxiv.org/pdf/1406.2661**. Сети GAN имеют много применений, включая такие, как генерирование изображений и разработка лекарственных средств.

В этой главе вы познакомитесь с основными компонентами GAN. Мы расскажем в ней, как работает каждый компонент, а также о важных концепциях и технологиях, лежащих в основе GAN. В главе будет дан краткий обзор преимуществ и недостатков использования сетей GAN и обзор ряда реальных приложений.

В главе эти вопросы рассматриваются в следующей последовательности:

- что такое GAN;
- архитектура GAN;
- важные понятия, связанные с GAN;
- различные разновидности GAN;
- преимущества и недостатки GAN;
- практическое применение GAN.

Что такое порождающие сети

Сеть GAN – это архитектура глубоких нейронных сетей, состоящая из двух сетей, сети генератора и сети дискриминатора. Посредством ряда циклов генерации и дискриминации обе сети обучают друг друга, пытаясь перехитрить друг друга.

Что такое сеть генератора

Сеть генератора использует существующие данные для генерации новых данных. Она может, например, использовать существующие изображения для соз-

дания новых изображений. Основная цель генератора – генерировать данные (например, изображения, видео, аудио или текст) из случайно генерированного вектора чисел, называемого **скрытым пространством** (latent space). При создании сети генератора нам необходимо указать цель сети. Это может быть генерация изображения, генерация текста, генерация аудио, видео и т. д.

Что такое сеть дискриминатора

Сеть дискриминатора пытается отличить реальные данные от данных, генерируемых сетью генератора. Сеть дискриминатора пытается сопоставить поступающие от генератора данные и заранее определенные классы. Она может выполнять как классификацию по многим классам, так и бинарную классификацию. Как правило, в сетях GAN выполняется бинарная классификация.

Обучение сети GAN посредством состязательной игры

1. Первая сеть, сеть генератора, никогда не видела некого реального произведения искусства, но пытается создать такое произведение искусства, которое выглядит как это реальное.
2. Вторая сеть, дискриминатор, пытается определить, является ли произведение искусства оригиналом или это подделка.
3. Генератор, в свою очередь, пытается обмануть дискриминатор, с тем чтобы тот принимал его подделку за оригинал, создавая в процессе итераций все более реалистичные изображения.
4. Дискриминатор пытается перехитрить генератор, продолжая совершенствовать свой собственный критерий определения подделки.
5. Посредством обратной связи они получают друг от друга успешные изменения, создаваемые каждым из них в процессе каждой итерации. В целом этот процесс и является обучением сети GAN.
6. В конечном итоге дискриминатор обучает генератор до стадии, в которой тот уже не может больше отличить реальное произведение искусства от подделки.

В этой игре обе сети обучаются одновременно. Когда мы достигнем стадии, в которой дискриминатор не может различить настоящие и поддельные произведения искусства, сеть достигает состояния, известного как равновесие Нэша. Мы обсудим это позже в данной главе.

Практические применения сетей GAN

GAN имеют ряд весьма полезных практических приложений, которые включают следующие.

- **Генерация изображений:** состязательные сети могут быть использованы для создания реалистичных изображений после обучения на образцах изображений. Например, если мы хотим создать новые изображения собак, мы можем обучить сеть GAN на нескольких тысячах изображений собак. Как

только обучение закончится, порождающая сеть будет иметь возможность генерировать новые изображения, которые отличаются от изображений в обучающем наборе. Генерация изображений используется в маркетинге, создании логотипов, развлечениях, социальных сетях и т. д. В следующей главе мы будем генерировать лица анимационных персонажей.

- **Синтез текста в изображение:** создание изображений из текстовых описаний является интересным вариантом использования сетей GAN. Поскольку сети GAN способны генерировать новые данные на основе написанного текста, это может быть применено в киноиндустрии – при создании комиксов можно автоматически генерировать последовательные эпизоды.
- **Старение лица:** может быть очень полезно как для развлечений, так и в промышленности. Особенно оно полезно для распознавания лиц стареющих работников компании, потому что в этом случае компании не нужно менять из-за этого свои охранные системы. Сеть Age-cGAN может генерировать изображения лиц в разном возрасте, что затем можно использовать для обучения надежной модели проверки лиц.
- **Перевод изображения в изображение:** может использоваться для преобразования фотографии, снятой днем, в фотографию, снятую ночью, для преобразования эскизов в картины, стилизации изображения, чтобы оно выглядели как, например, картины кисти Пикассо или Ван Гога, автоматически преобразовывать аэрофотоснимки спутниковых изображений, а также преобразовывать изображения лошадей в изображения зебр. Эти варианты использования являются новаторскими и позволяют существенно экономить время.
- **Синтез видео:** можно использовать для создания видео. Сети GAN могут генерировать контент за меньшее время, чем если бы он создавался вручную. Они могут повысить производительность создателей фильмов, а также расширить возможности любителей, которые хотят в свободное время создавать креативные видео.
- **Генерация изображений с высоким разрешением:** если у вас есть фотографии, сделанные с низким разрешением камеры, сети GAN могут помочь вам генерировать изображения с высоким разрешением и не терять существенных деталей. Это может быть полезно при создании веб-сайтов.
- **Заполнение недостающих частей изображений:** если у вас есть изображение с некоторыми недостающими деталями, GAN могут помочь вам восстановить их.

Детализация архитектуры сети GAN

Архитектура сети GAN имеет два основных элемента: сеть генератора и сеть дискриминатора. Каждая сеть может быть любой нейронной сетью, такой как **искусственная нейронная сеть** (Artificial Neural Network, ANN), **нейронная сеть свертки** (Convolutional Neural Network, CNN), **рекуррентная нейронная сеть** (Recurrent Neural Network, RNN) **или сеть с долговременной кратковре-**

менной памятью (Long Short Term Memory, LSTM). Дискриминатор должен иметь полносвязные слои с классификатором в конце.

Давайте подробнее рассмотрим компоненты архитектуры GAN. В этом примере будем полагать, что создается фиктивная сеть GAN.

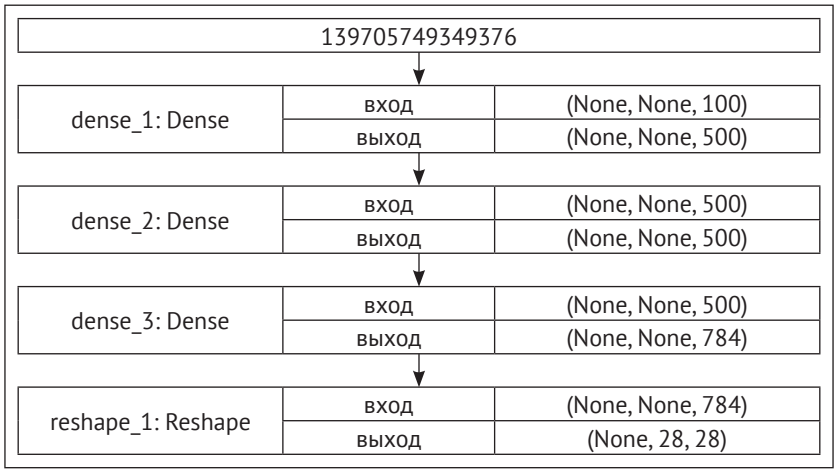
Архитектура генератора

Сеть генератора в нашей фиктивной сети GAN представляет собой простую нейронную сеть прямого распространения с пятью слоями: входной слой, три скрытых слоя и выходной слой. Давайте подробнее рассмотрим конфигурацию такой фиктивной сети генератора.

№ слоя	Наименование слоя	Конфигурация
1	Input layer (входной слой)	`input_shape=(batch_size, 100), output_shape=(batch_size, 100)`
2	Dense layer (плотный слой)	`neurons=500, input_shape=(batch_size, 100), output_shape=(batch_size, 500)`
3	Dense layer (плотный слой)	`neurons=500, input_shape=(batch_size, 500), output_shape=(batch_size, 500)`
4	Dense layer (плотный слой)	`neurons=784, input_shape=(batch_size, 500), output_shape=(batch_size, 784)`
5	Reshape layer (слой восстановления)	`input_shape=(batch_size, 784), output_shape=(batch_size, 28, 28)`

Эта таблица показывает конфигурацию скрытых слоев, а также входного и выходного слоев сети.

Следующая таблица показывает поток тензоров и форму входного и выходного тензоров для каждого слоя в сети генератора:



Архитектура сети генератора

Давайте рассмотрим, как эта нейронная сеть обрабатывает информацию во время прямого распространения данных.

- Входной слой принимает 100-мерный вектор, выбранный из гауссова (нормального) распределения, и передает тензор в первый скрытый слой без каких-либо изменений.
- Три скрытых слоя – это плотные слои с 500, 500 и 784 единицами соответственно. Первый скрытый слой (плотный слой) преобразует тензор формы `(batch_size, 100)` в тензор формы `(batch_size, 500)`.
- Второй плотный слой генерирует тензор формы `(batch_size, 500)`.
- Третий скрытый слой генерирует тензор формы `(batch_size, 784)`.
- В последнем выходном слое этот тензор преобразуется из формы `(batch_size, 784)` в форму `(batch_size, 28, 28)`. Это означает, что наша сеть будет генерировать пакет изображений, где одно изображение будет иметь форму (28, 28).

Архитектура дискриминатора

Дискриминатор нашей сети GAN является нейронной сетью прямого распространения с пятью слоями, включая входной и выходной слои и три плотных слоя. Сеть дискриминатора является классификатором и несколько отличается от сети генератора. Она обрабатывает изображение и выводит вероятность того, что изображение принадлежит определенному классу.

Следующая таблица показывает поток тензоров, а также формы ввода и вывода тензоров для каждого слоя в сети дискриминатора:

139906652570288		
flatten_1: Flatten	вход	(None, 28, 28)
	выход	(None, 784)
dense_4: Dense	вход	(None, 784)
	выход	(None, 500)
dense_5: Dense	вход	(None, 500)
	выход	(None, 500)
dense_6: Dense	вход	(None, 500)
	выход	(None, 1)

Архитектура сети дискриминатора

Давайте рассмотрим, как дискриминатор обрабатывает данные при прямом распространении во время обучения сети.

1. Первоначально он получает входные данные в форме 28×28.
2. Входной слой принимает входной тензор, который является тензором с формой (`batch_sizex28x28`), и передает его первому скрытому слою без каких-либо изменений.
3. Затем плоский слой создает тензор 784-мерного вектора, который передается первому скрытому (плотному) слою. Первый и второй скрытые слои создают 500-мерный вектор.
4. Последний слой – это выходной слой, который тоже является плотным, с одним элементом (нейроном) и сигмоидом в качестве функции активации. Он выводит одно значение, либо 0, либо 1. Значение 0 указывает, что предоставленное изображение является поддельным, а значение 1 указывает, что предоставленное изображение является подлинным.

Важные понятия, связанные с сетью GAN

Теперь, когда мы обсудили архитектуру GAN, давайте сделаем краткий обзор некоторых важных концепций. Сначала мы рассмотрим **расходимость Кульбака–Лейблера** (KL divergence). Очень важно также понимать **расходимость Дженсена–Шеннона** (JS divergence), которая является важной мерой оценки качества моделей. Затем мы рассмотрим равновесие по Нэшу, которого стремимся достичь во время обучения. Наконец, более подробно рассмотрим целевые функции, которые очень важны для хорошей реализации сетей GAN.

Расходимость Кульбака–Лейблера

Расходимость Кульбака–Лейблера, также известная как **кросс-энтропия**, представляет собой метод, используемый для определения сходства между двумя вероятностными распределениями. Она определяет, как одно распределение вероятности p отличается от другого ожидаемого p распределения вероятности q.

Уравнение, используемое для расчета расхождения KL двух вероятностных распределений $p(x)$ и $q(x)$, выглядит следующим образом:

$$D_{\mathrm{KL}}(p||q) = \int_x p(x)\log\frac{p(x)}{q(x)}dx.$$

Дивергенция KL будет нулевой или минимальной, когда $p(x)$ равно $q(x)$ в любой точке.

Из-за асимметричной природы дивергенции KL эту расходимость не следует использовать для измерения расстояния между двумя вероятностными распределениями и соответственно в качестве метрики расстояний.

Расходимость Дженсена–Шеннона

Расходимость Дженсена–Шеннона (JS), также известная как **радиус информации** (information radius, IRaD), или полная дивергенция к среднему, является еще одной мерой сходства между двумя вероятностными распределениями. Она основана на расходимости KL, однако, в отличие от расходимости KL, рас-

ходимость JS носит симметричный характер и может использоваться для измерения расстояния между двумя вероятностными распределениями. Если мы извлечем квадратный корень из расхождения Дженсена–Шеннона, то получим расстояние Дженсена–Шеннона, то есть метрику расстояния.

Следующее уравнение представляет расхождение Дженсена–Шеннона между двумя вероятностными распределениями p и q:

$$D_{\mathrm{JS}}(p||q) = \frac{1}{2} D_{\mathrm{KL}}\left(p||\frac{p+q}{2}\right) + \frac{1}{2} D_{\mathrm{KL}}\left(q||\frac{p+q}{2}\right).$$

В этом уравнении $(p + q)$ – мера средней точки, а D_{KL} – расхождение Кульбака–Лейблера.

Теперь, когда мы определили KL-расхождение и JS-расхождение, давайте рассмотрим равновесие Нэша (Nash equilibrium) для GAN.

Равновесие Нэша

Равновесие Нэша описывает конкретное состояние в теории игр. Это состояние может быть достигнуто в бескоалиционной игре, в которой каждый игрок пытается выбрать наилучшую возможную стратегию, чтобы получить наилучший возможный результат, исходя из своего ожидания того, что будут делать другие игроки. В конце концов, все игроки достигают состояния, в котором они выбрали для себя наилучшую возможную стратегию на основе решений, принятых другими игроками. В этом состоянии игры они не получат никакой выгоды от изменения своей стратегии. Это состояние является равновесием Нэша.

Известный пример того, как можно достичь равновесия Нэша, – это так называемая дилемма заключенного. В этом примере два преступника (A и B) были арестованы за совершение преступления. Оба были помещены в отдельные камеры без возможности общения друг с другом. У прокурора имеется достаточно доказательств, чтобы осудить их за более мелкое правонарушение, а не за основное преступление, которое может привести к тому, что они надолго попадут в тюрьму. Чтобы получить обвинительный приговор, прокурор делает им предложение:

- если A и B оба признают причастность другого к преступлению, они оба получают 2 года тюрьмы;
- если A признает причастность B, а B хранит молчание, то A будет освобожден, а B получит 3 года тюрьмы (и наоборот);
- если A и B оба хранят молчание, они оба получают только по 1 году тюрьмы по более легкой статье.

Эти три сценария показывают, что наилучший возможный результат для A и B – молчать и получить по 1 году тюрьмы. Однако риск при молчании заключается в том, что можно получить 3 года, поскольку ни A, ни B не может быть твердо уверен, что другой заключенный тоже будет молчать. Таким образом, они находились бы в состоянии, в котором фактическая оптимальная страте-

гия каждого из них была бы признать причастность другого, поскольку именно этот выбор обеспечивает самое высокое вознаграждение при самом низком штрафе. Когда такое состояние достигнуто, ни один из преступников, изменив свою стратегию, не получает преимущества, и, таким образом, оба находятся в состоянии равновесия Нэша.

Целевые функции

Чтобы создать генераторную сеть, которая генерирует изображения, похожие на реальные изображения, мы стараемся повысить сходство данных, генерируемых генератором, с реальными данными. Чтобы измерить степень сходства, используем целевые функции. Обе сети имеют свои целевые функции, и во время обучения каждая пытается минимизировать свою целевую функцию. Следующее уравнение представляет конечную целевую функцию для GAN:

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

В этом уравнении $D(x)$ является моделью дискриминатора, $G(z)$ – модель генератора, P_x – реальное распределение данных, P_z – распределение данных, генерированных генератором и E – ожидаемый выход.

Во время обучения D (дискриминатор) хочет максимизировать весь результат, а G (генератор) хочет минимизировать его, тем самым GAN обучается достигать равновесия между сетями генератора и дискриминатора. Когда равновесие достигается, мы говорим, что модель сошлась. Это равновесие является равновесием Нэша. Когда обучение завершено, мы получаем модель генератора, способную генерировать изображения, похожие на реальные.

Алгоритмы оценки

Вычислить верность GAN просто. Целевая функция для GAN не является специфической функцией, такой как среднеквадратическая ошибка или кросс-энтропия. Сети GAN изучают целевые функции в процессе обучения. Существует много алгоритмов оценки, предложенных исследователями для измерения того, насколько хорошо работает модель. Давайте рассмотрим некоторые алгоритмы оценки детально.

Начальная оценка

Начальная оценка является наиболее широко используемым алгоритмом оценки для сетей GAN. Она использует предварительно обученную начальную сеть V3 (обученную на Imagenet), чтобы извлечь особенности генерируемого и реального изображений. Это было предложено Шейном Барратом (Shane Barrat) и Риши Шармой (Rishi Sharma) в их работе «Заметки о начальной оценке» (*A Note on the Inception Score*, **https://arxiv.org/pdf/1801.01973.pdf**). Начальная оценка, или сокращенно IS (Inception Score), измеряет качество и разнообразие генерируемых изображений. Давайте посмотрим на уравнение для IS:

$$\text{IS}(G) = \exp(\mathbb{E}_{\text{x}\sim p_g} D_{KL}(p(y|\text{x})||p(y))).$$

В этом уравнении x представляет выборку из распределения p_g, а $x \sim p_g$ – ту же концепцию. $p(y|x)$ является условным распределением классов, а $p(y)$ – маргинальным распределением классов. Чтобы рассчитать начальную оценку, надо выполнить следующие действия.

1. Начать с выборки N изображений, генерированных моделью, обозначенных как (x^i).
2. Построить маргинальное распределение классов, используя следующее уравнение:

$$p(y) = \int_x p(y|x)p_g(x).$$

3. Вычислить расходимость KL и ожидаемое улучшение, используя уравнение

$$\text{IS}(G) = \exp(\mathbb{E}_{x \sim p_g} D_{KL}(p(y|x)||p(y))).$$

4. Вычислить экспоненту результата, чтобы получить начальную оценку.

Качество модели хорошее, если у нее высокая начальная оценка. Хотя это является важным показателем, у него есть определенные проблемы. Например, он показывает хорошую оценку верности, даже когда модель генерирует всего одно и то же изображение в классе, то есть у модели не хватает разнообразия. Для решения этой проблемы были предложены другие показатели. Мы рассмотрим один из них в следующем разделе.

Начальное расстояние Фреше

Чтобы преодолеть недостатки начальной оценки, Мартином Хойзелем (Martin Heusel) и другими была предложена оценка **начального расстояния Фреше** (**the Fréchet Inception Distance, FID**) в их статье «Обучение сетей GAN по правилу обновления двух временны́х масштабов, сходящихся к локальному равновесию Нэша» (*GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*), **https://arxiv.org/pdf/1706.08500.pdf**.

Уравнение для вычисления FID:

$$FID = ||\mu_r - \mu_g||^2 + T_r(\sum_r + \sum_g - 2(\sum_r \sum_g)^{1/2}).$$

Это уравнение представляет оценку FID между реальными изображениями x и генерированными изображениями g. Чтобы рассчитать оценку FID, мы используем начальную сеть для извлечения карты характеристик из промежуточного слоя в начальной сети. Затем моделируем многомерное распределение Гаусса, которое изучает распределение карт характеристик. Это многомерное распределение Гаусса имеет среднее значение μ и ковариацию $\sum$, которые мы используем для вычисления оценки FID. Чем меньше показатель FID, тем лучше модель и тем больше ее способность создавать более разнообразные изображения с более высоким качеством. Идеальная порождающая модель будет иметь оценку FID, равную нулю. Преимущество использования оценки FID перед начальной оценкой в том, что она устойчива к шуму и позволяет легко оценивать разнообразие изображений.

Реализация FID в программной библиотеке TensorFlow находится по следующему адресу: **https://www.tensorflow.org/api_docs/python/tf/contrib/gan/eval/frechet_classifier_distance.pdf**.

Недавно исследователями предложены и другие алгоритмы оценки. Мы не будем рассматривать здесь все. Прежде чем читать дальше, посмотрите на иной алгоритм оценки, который называется Mode Score. Информацию о нем можно найти на сайте **https://arxiv.org/pdf/1612.02136.pdf**.

Варианты сетей GAN

В настоящее время доступны тысячи различных сетей GAN, и это число увеличивается с феноменальной скоростью. В этом разделе мы разберем шесть популярных архитектур сетей GAN, которые более подробно будем рассматривать в последующих главах этой книги.

Глубокие порождающие состязательные сети свертки

Алек Рэдфорд (Alec Radford), Люк Мец (Luke Metz) и Сумит Чинтала (Soumith Chintala) предложили глубокие сети свертки GAN (DCGANs) в статье под названием «Обучение без учителя глубоких порождающих состязательных сетей свертки» (*Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*). Статья доступна по адресу: **https://arxiv.org/pdf/1511.06434.pdf**. Простые сети GAN (vanilla GANs) обычно не имеют нейронных сетей свертки (CNN). Впервые сети свертки были применены в сетях DCGAN. Мы научимся создавать анимационные лица персонажей, используя сеть DCGAN, в главе 3 «Старение лица с использованием условной сети cGAN».

Сеть StackGAN

Сети StackGAN были предложены Хань Чжаном (Han Zhang), Тао Сюй (Tao Xu), Хуншен Ли (Hongsheng Li) и другими в статье «StackGAN: синтез текста в фотореалистичное изображение порождающими состязательными сетями» (*StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative*). Она доступна по адресу: **https://arxiv.org/pdf/1612.03242.pdf**. Авторы использовали StackGAN для изучения синтеза текста в изображения и получили впечатляющие результаты. StackGAN – это две сети, которые генерируют реалистичные изображения, представленные текстовым описанием. Мы научимся создавать реалистичные изображения из текстовых описаний с использованием StackGAN в главе 6 «Сети StackGAN – синтез текста в реалистичные фотоизображения».

Сеть CycleGAN

Сети CycleGAN были предложены Цзюнь-Янь Чжу (Jun-Yan Zhu), Тэсуном Парком (Taesung Park), Филиппом Изолой (Phillip Isola) и Алексеем А. Эфросом (Alexei A. Efros) в статье «Непарный перевод изображения в изображение с использованием согласованного цикла порождающих сетей» (*Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*). Она доступна по

адресу: **https://arxiv.org/pdf/1703.10593**. У сети CycleGAN есть несколько интересных потенциальных применений, таких как преобразование фотографий в живопись и наоборот, преобразование сделанной летом фотографии в фотографию, сделанную зимой, и наоборот, или преобразование изображений лошадей в изображения зебр и наоборот. Мы научимся превращать картины в фотографии с помощью сети CycleGAN в главе 7 «Сети CycleGAN – превращение картин в фотографии».

Сеть 3D-GAN

3D-GAN были предложены Цзяцзюнем Ву (Jiajun Wu), Ченкаем Чжаном (Chengkai Zhang), Тианфаном Ксю (Tianfan Xue), Уильямом Т. Фриманом (William T. Freeman) и Джошуа Б. Тененбаумом (Joshua B. Tenenbaum) в статье «Изучение формы вероятностного скрытого пространственного объекта помощью 3D-модели порождающей состязательной сети» (*Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling*). Она доступна по адресу: **https://arxiv.org/pdf/1610.07584**. Генерация трехмерных моделей объектов имеет множество применений в производстве и практике 3D-моделирования. Сеть 3D-GAN способна генерировать новые 3D-модели различных объектов, однажды обученных на 3D-моделях объектов. Мы научимся создавать 3D-модели объектов с использованием 3D GAN в главе 2 «Сеть 3D-GAN – генерация форм 3D с использованием сети GAN».

Сеть Age-cGAN

Старение лица с использованием условных сетей GAN было предложено Григорием Антиповым (Grigory Antipov), Моэзом Бакушем (Moez Baccouche) и Жан-Люком Дугле (Jean-Luc Dugelay) в статье «Старение лица с помощью условной порождающей состязательной сети», доступной по адресу: **https://arxiv.org/pdf/1702.01983.pdf**. У процедуры старения лица есть много применений в промышленности, включая распознавание лиц по возрасту, поиск потерявшихся детей, и в развлечениях. Мы научимся обучать условную GAN создавать лица заданного возраста в главе 3 «Старение лица с использованием условной cGAN».

Сеть pix2pix

Сеть pix2pix была представлена Цзюнь-Янь Чжу (Jun-Yan Zhu), Тэсуном Парком (Taesung Park), Филиппом Изолой (Phillip Isola) и Алексеем А. Эфросом (Alexei A. Efros) «Трансляция изображения в изображение с помощью условных состязательных сетей» (*Image-to-Image Translation with Conditional Adversarial Networks*). Она доступна по адресу: **https://arxiv.org/abs/1611.07004**. Сеть pix-2pix имеет варианты использования, аналогичные сети CycleGAN. Она может преобразовывать контуры зданий в изображения зданий (мы увидим такой пример в главе о pix2pix), черно-белые изображения в цветные изображения, изображения, сделанные днем, в ночные изображения, превращать эскизы в фотографии и аэрофотоснимки в изображения в виде карт.

Список всех существующих подобных сетей GAN смотрите в статье Авинаша Хиндупура (Avinash Hindupur) «Зоопарк сетей GAN» (*GAN Zoo*), доступной по адресу: **https://github.com/hindupuravinash/thean-zoo**.

Преимущества сетей GAN

Сети GAN имеют определенные преимущества перед другими методами обучения с учителем и без учителя:

- **сеть GAN – это метод обучения без учителя:** маркировка данных является ручным процессом, занимающим много времени. Сети GAN не требуют маркированных данных; они могут быть обучены с использованием немаркированных данных, поскольку обучаются внутренними данными;
- **генерируемые сетью GAN данные:** одним из самых больших преимуществ сетей GAN является то, что они генерируют данные, подобные реальным данным. Это обеспечивает большое количество различных применений в реальном мире. Такие сети могут генерировать изображения, текст, аудио и видео, которые неотличимы от реальных данных;
- **сети GAN изучают плотности распределения данных:** сети GAN изучают внутренние представления данных. Как упоминалось ранее, сети GAN могут изучать сложные распределения данных, и это может быть использовано для решения многих проблем машинного обучения;
- **обученный дискриминатор является классификатором:** после обучения мы получаем дискриминатор и генератор. Сеть дискриминатора является классификатором и может использоваться для классификации объектов.

Проблемы обучения сетей GAN

Как и любая технология, сети GAN имеют ряд проблем. Эти проблемы, как правило, относятся к процессу обучения и включают в себя режим коллапса, внутренние сдвиги ковариации и исчезающие градиенты. Давайте рассмотрим эти проблемы более подробно.

Режим коллапса

Режим коллапса – это проблема, которая относится к ситуации, когда сеть генератора генерирует образцы, имеющие лишь небольшое разнообразие, или когда модель начинает генерировать одно и то же изображение. Иногда распределение вероятностей является мультимодальным и очень сложным по своей природе. Оно может содержать данные из разных наблюдений и иметь несколько пиков для разных подграфов образцов. Иногда сети GAN не могут моделировать мультимодальное распределение вероятностей данных и потому подвержены режиму коллапса. Ситуация, при которой генерированные образцы практически идентичны, является полным коллапсом.

Существует много методов, которые можно использовать для преодоления проблемы режима коллапса. Они включают:

- обучение многих моделей (сетей GAN) в различных режимах;
- обучение сетей GAN различными наборами данных.

Исчезающие градиенты

В процессе обратного распространения градиент распространяется в обратном направлении, от последнего слоя к первому слою. При этом он становится все меньше и меньше. Иногда градиент настолько мал, что начальные слои обучаются очень медленно или перестают обучаться совсем. В этом случае градиент не изменяет значения весов начальных слоев, и поэтому обучение начальных слоев в сети останавливается. Данный коллапс известен как проблема **исчезающих градиентов**.

Эта проблема усугубляется, если мы обучаем большую сеть с помощью методов градиентной оптимизации. Градиентные методы оптимизации оптимизируют значение параметра посредством вычисления изменений на выходе сети при небольшом значении параметра. Если изменение значения параметра вызывает небольшое изменение на выходе сети, то изменение весов будет очень маленьким и сеть перестанет обучаться.

Эта проблема также возникает, когда мы используем функции активации, такие как сигмоид и tanh. Функции активации сигмоид ограничивают значения диапазоном от 0 до 1, преобразуя большие значения x близко к 1, а малые или отрицательные значения x близко к нулю. Функция активации tanh ограничивает входные значения диапазоном от –1 до 1, преобразуя большие входные значения примерно до 1, а небольшие значения примерно до –1. Когда мы применяем обратное распространение, то используем цепное правило дифференцирования, обладающее свойством умножения. При достижении начальных слоев сети градиент (ошибка) уменьшается экспоненциально, вызывая проблему исчезающих градиентов.

Чтобы преодолеть эту проблему, мы можем использовать функции активации, такие как ReLU, ReLU с утечкой и PReLU. Градиенты этих функций активации не насыщаются при обратном распространении, обеспечивая эффективное обучение сети. Другое решение заключается в использовании пакетной нормализации, которая нормализует входы скрытых слоев.

Внутренний ковариантный сдвиг

Внутренний ковариантный сдвиг происходит при изменениях распределения на входе сети. Когда входное распределение изменяется, скрытые слои пытаются обучиться адаптироваться к новому распределению. Это замедляет процесс обучения. При замедлении процесса требуется большее время для обеспечения сходимости к глобальному минимуму. Эта проблема возникает, когда статистическое распределение входных данных сети резко отличается от входных данных, которые поступали в сеть раньше. Пакетная нормализация

и другие методы нормализации могут решить эту проблему. Мы рассмотрим их в следующих разделах.

Решение проблем стабильности при обучении сетей GAN

Стабильность обучения – одна из наибольших проблем, связанных с сетями GAN. Для некоторых наборов данных сеть GAN никогда не сходится из-за проблем такого типа. В этом разделе мы рассмотрим некоторые решения, которые можно использовать для улучшения стабильности сетей GAN.

Соответствие характеристик

Во время обучения GAN мы максимизируем целевую функцию дискриминатора сети и минимизируем целевую функцию генератора сети. Такая целевая функция имеет ряд серьезных недостатков. Например, она не учитывает статистику генерированных данных и реальных данных.

Сопоставление характеристик функций – метод, предложенный Тимом Салимансом (Tim Salimans), Яном Гудфеллоу (Ian Goodfellow) и другими в работе «Улучшенные методы обучения сетей GAN» (*Improved Techniques for Training GANs*), улучшающий сходимость GAN путем введения новой целевой функции. Новая целевая функция для сети генератора побуждает ее генерировать данные со статистикой, которая в большей степени соответствует реальным данным.

При применении этого метода сеть не запрашивает у дискриминатора двоичную маркировку. Вместо этого из сети дискриминатора поступают активации или карты характеристик входных данных, извлеченных из промежуточного слоя сети дискриминатора. Таким образом, в процессе обучения сеть дискриминатора изучает статистику реальных данных, что увеличивает ее способность отличать реальные данные от поддельных, зная их дискриминационные особенности.

Чтобы представить этот метод математически, давайте посмотрим сначала на отличия в обозначениях:

- $f(x)$: активации или карты характеристик реальных данных из промежуточного слоя сети дискриминатора;
- $f(G(z))$: карты активации/характеристик данных, генерированных сетью генератора из промежуточного слоя сети дискриминатора.

Новая целевая функция может быть представлена следующим образом:

$$||\mathbb{E}_{x \sim p_{\text{data}}} \mathrm{f}(x) - \mathbb{E}_{z \sim p_z(z)} \mathrm{f}(G(z))||_2^2.$$

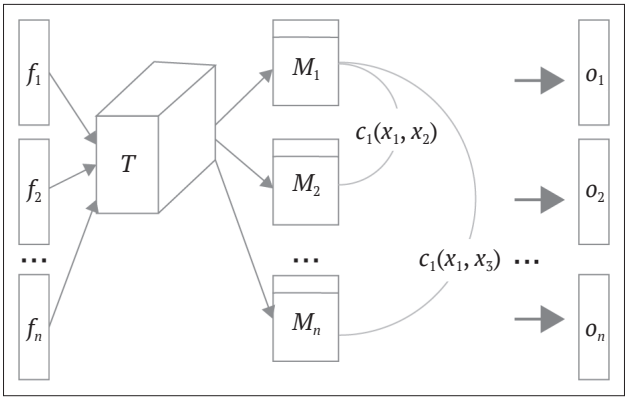
Используя эту целевую функцию, можно получить лучшие результаты, но при этом мы все же не имеем гарантии сходимости.

Мини-пакетная дискриминация

Мини-пакетная дискриминация является еще одним методом стабилизации обучения сетей GAN. Она была предложена Гудфеллоу (Goodfellow) и други-

ми в работе «Метод улучшения обучения сетей GAN» (*Improved Techniques for Training GANs*), которая доступна по адресу **https://arxiv.org/pdf/1606.03498.pdf**. Чтобы понять этот подход, давайте сначала рассмотрим детали проблемы. Во время обучения сети GAN, когда мы подаем независимые входы в сеть дискриминатора, координация между градиентами может исчезать, и это мешает сети дискриминатора обучаться дифференцировать различные изображения, генерируемые сетью генератора. Это режим коллапса, проблема, о которой говорилось ранее. Чтобы решить данную проблему, мы можем использовать мини-пакетную дискриминацию.

Следующая диаграмма иллюстрирует данный процесс:



Мини-пакетная дискриминация – многошаговый процесс. Необходимо выполнить следующие шаги, чтобы создать мини-пакетную дискриминацию в вашей сети.

1. Извлечь карты характеристик образца и умножить их на тензор $T \in R^{A \times B \times C}$, генерируя матрицу $M_i \in R^{A \times B}$.
2. Вычислить расстояние L_1 между строками матрицы M_i, используя следующее равенство:

   $c_b(x_i, x_j) = \exp(-||M_{i,b} - M_{j,b}||_{L_1}) \in \mathbb{R}$.

3. Вычислить сумму всех расстояний для конкретного образца x_i:

   $$o(x_i)_b = \sum_{j=1}^{n} c_b(x_i, x_j) \in \mathbb{R}.$$

4. Последовательно связать $o(x_i)$ с $f(x_i)$ и передать следующему слою сети:

   $o(x_i) = [o(x_i)_1, o(x_i)_2, \ldots, o(x_i)_B] \in \mathbb{R}^B$, $o(X) \in \mathbb{R}^{n \times B}$.

Чтобы определить этот метод математически, давайте более детально посмотрим на обозначения:

- $f(x_i)$: активация или карта характеристик для i-го образца из среднего слоя в сети дискриминатора;
- $T \in R^{A \times B \times C}$: трехмерный тензор, полученный умножением тензора на $f(x_i)$;
- $M_i \in R^{A \times B}$: матрица, генерированная при перемножении тензора T на $f(x_i)$;
- $o(x_i)$: выход после суммирования всех расстояний для данного образца x_i.

Мини-пакетная дискриминация помогает предотвратить режим коллапса и улучшает шансы на стабильность обучения.

Усреднение истории

Усреднение истории – это метод усреднения параметров в прошлом и добавления среднего к соответствующим функциям стоимости сетей генератора и дискриминатора. Он был предложен Гудфеллоу (Goodfellow) и другими в упомянутой работе «Метод улучшения обучения сетей GAN» (*Improved Techniques for Training GANs*).

Усреднение истории определяется как

$$\left\| \theta - \frac{1}{t} \sum_{i=1}^{t} \theta[i] \right\|^2.$$

В этом уравнении $\theta[i]$ является величиной параметров в данный момент i. Этот метод также может улучшить стабильность обучения сети GAN.

Одностороннее сглаживание маркировки

Ранее значения маркировки / величины цели для классификатора были равны 0 или 1: для поддельных изображений 0 и для реальных изображений 1. В связи с этим сети GAN были предрасположены к вводимым в нейронную сеть состязательным образцам, что приводило к неверному выходу сети. Метод сглаживания маркировки является методом введения в сеть дискриминатора сглаженных маркировок. То есть мы можем иметь десятичные значения маркировок, такие как 0.9 (истина), 0.8 (истина), 0.1 (ложь) или 0.2 (ложь), вместо маркировки каждого образца либо 1 (истина), либо 0 (ложь). Мы сглаживаем целевые значения маркировок как реального изображения, так и поддельного изображения. Сглаживание маркировок может снизить риск наличия состязательных образцов в сети GAN. Чтобы применить сглаживание маркировок, присваивайте изображениям маркировки 0.9, 0.8 и 0.7, а также 0.1, 0.2 и 0.3. Узнать больше о сглаживании маркировок можно по адресу: **https://arxiv.org/pdf/1606.03498.pdf**.

Пакетная нормализация

Пакетная нормализация – это метод, который нормализует векторы характеристик так, чтобы они не имели среднего значения или единичной дисперсии. Он используется для стабилизации обучения и борьбы с проблемой плохой инициализации веса. Это этап предварительной обработки, который мы при-

меняем к скрытым слоям сети, и он помогает нам уменьшить внутреннее ковариантное смещение.

Иоффе (Ioffe) и Сегеди (Szegedy) представили пакетную нормализацию в 2015 году в статье «Нормализация: ускорение глубокого обучения сети за счет уменьшения внутреннего ковариантного сдвига» (*Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*). Эту работу можно найти по адресу: **https://arxiv.org/pdf/1502.03167.pdf**.

Преимущества пакетной нормализации следующие:

- **сокращает внутренний ковариантный сдвиг.** Пакетная нормализация помогает нам уменьшить внутренний ковариантный сдвиг путем нормализации величин;
- **ускоряет обучение.** Сети будут обучаться быстрее, если величины взяты из нормального (гауссова) распределения. Пакетная нормализация помогает отбелить значения внутренних слоев нашей сети. Общее обучение проходит быстрее, но каждая итерация замедляется из-за того, что задействованы дополнительные вычисления;
- **более высокая верность.** Пакетная нормализация обеспечивает лучшую верность;
- **более высокая скорость обучения.** Обычно, когда мы обучаем нейронные сети, мы используем низкую скорость обучения, которая требует больше времени для сходимости сети. При пакетной нормализации мы можем использовать более высокие темпы обучения, благодаря чему наша сеть быстрее достигает глобального минимума;
- **уменьшение необходимости использования выпадения.** Когда мы используем выпадение, то в некоторой степени искажаем базовую информацию во внутренних слоях сети. Пакетная нормализация действует как регулятор, то есть мы можем обучать сеть без слоя выпадения.

При пакетной нормализации мы применяем нормализацию ко всем скрытым слоям, а не только к входному слою.

Нормализация образцов

Как упоминалось в предыдущем разделе, при пакетной нормализации нормализуется группа образцов с использованием информации только из этого пакета. Нормализация образцов – несколько другой подход. В случае нормализации образцов мы нормализуем каждую карту характеристик, используя информацию только из данной карты. Нормализация образцов была предложена Дмитрием Ульяновым (Dmitry Ulyanov) и Андреа Ведальди (Andrea Vedaldi) в их статье «Нормализация образцов: недостающий ингредиент для быстрой стилизации» (*Instance Normalization: The Missing Ingredient for Fast Stylization*), доступной по адресу: **https://arxiv.org/pdf/1607.08022.pdf**.

Резюме

В этой главе мы узнали о том, что такое сети GAN и какие компоненты составляют их стандартную архитектуру. Мы также рассмотрели различные виды доступных сетей GAN. После определения базовых концепций GAN перешли к рассмотрению основных концепций, лежащих в основе конструкции и обеспечивающих функционирование сетей GAN. Мы узнали о преимуществах и недостатках сетей GAN, а также о решениях, которые помогают преодолеть недостатки. Наконец, мы узнали о различных практических применениях GAN.