Отзыв научного руководителя
о курсовой работе студента Шпуганича А А.
«Генерация изображений с использованием
генеративно-состязательной нейронной сети»

Перед бакалавром Шпуганичем А. А. была поставлена задача исследовать методы генерации изображений с применением генеративно-состязательных нейронных сетей, а также выполнить программную реализацию одного из этих методов и проанализировать полученные результаты.

С поставленной задачей бакалавр Шпуганич А. А. полностью справился, проявив высокий уровень самостоятельности и заинтересованности.

В работе приводится подробное описание полученных результатов.

Работал бакалавр Шпуганич А. А. регулярно и добросовестно. Поставленная задача решена полностью.

Считаю, что курсовая работа бакалавра Шпуганич А. А. на тему «Генерация изображений с использованием генеративно-состязательной нейронной сети» полностью отвечает требованиям, предъявляемым к курсовой работе 3 курса, и заслуживает оценки «отлично» (100 баллов).

К. ф.-м. н., доцент
кафедры информатики
и вычислительного эксперимента  *А.В. Абрамян*

АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

СПРАВКА

о результатах проверки текстового документа
на наличие заимствований

Южный федеральный университет

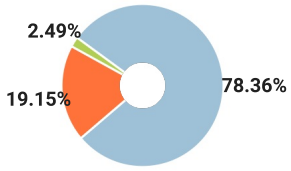
ПРОВЕРКА ВЫПОЛНЕНА В СИСТЕМЕ АНТИПЛАГИАТ.ВУЗ

Автор работы: Шпуганич Алексей Алексеевич
Самоцитирование рассчитано для: Шпуганич Алексей Алексеевич
Название работы: ВКР_Шпуганич
Тип работы: Курсовая работа
Подразделение: Институт математики, механики и компьютерных наук им. И.И.Воровича

РЕЗУЛЬТАТЫ

СОВПАДЕНИЯ	19.15%
ОРИГИНАЛЬНОСТЬ	78.36%
ЦИТИРОВАНИЯ	2.49%
САМОЦИТИРОВАНИЯ	0%

ДАТА ПОСЛЕДНЕЙ ПРОВЕРКИ: 04.06.2024



Структура документа: Проверенные разделы: библиография с.21, титульный лист с.1, содержание с.2, основная часть с.3-20
Модули поиска: Цитирование; Патенты СССР, РФ, СНГ; ИПС Адилет; Диссертации НББ; Коллекция НБУ; Медицина; Публикации eLIBRARY (переводы и перефразирования); Переводные заимствования*; Библиография; Шаблонные фразы; Кольцо вузов; Перефразирования по коллекции издательства Wiley; Издательство Wiley; СПС ГАРАНТ: аналитика; СМИ России и СНГ; Перефразирования по коллекции IEEE; Публикации РГБ; Публикации eLIBRARY; СПС ГАРАНТ: нормативно-правовая документация; Переводные заимствования по Интернету (EnRu); Сводная коллекция ЭБС; Перефразированные заимствования по коллекции Интернет в английском сегменте; Переводные заимствования издательства Wiley; IEEE; Переводные заимствования IEEE; Кольцо вузов (перефразирования); Переводные заимствования по коллекции Гарант: аналитика;

Работу проверил: Абрамян Анна Владимировна

ФИО проверяющего

Дата подписи:

Подпись проверяющего

Чтобы убедиться
в подлинности справки, используйте QR-код,
который содержит ссылку на отчет.

Ответ на вопрос, является ли обнаруженное заимствование корректным, система оставляет на усмотрение проверяющего. Предоставленная информация не подлежит использованию в коммерческих целях.

Задание на курсовую работу бакалавра

Направление подготовки: 02.03.02 — Фундаментальная информатика и информационные технологии

Студент: Шпуганич А. А.

Научный руководитель: к.ф.-м.н., доцент Абрамян А.В.

Год написания курсовой: 2024

Тема работы: Генерация изображений с использованием генеративно-состязательной нейронной сети.

Цель: исследовать методы генерации изображений с применением генеративно-состязательных нейронных сетей, а также выполнить программную реализацию одного из этих методов и проанализировать полученные результаты.

Задачи работы:

1. Проанализировать архитектуру и особенности применения несколь генеративно-состязательных нейронных сетей, используемых для генерации изображений;
2. Построить модель и обучить ее.
3. Проанализировать полученные результаты.

Научный руководитель Абрамян А. В.

Студент Шпуганич А. А.

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«Южный федеральный университет»

Институт математики, механики
и компьютерных наук им. И. И. Воровича

Кафедра информатики и вычислительного эксперимента

Шпуганич Алексей Алексеевич

Генерация изображений с использованием генеративно-состязательной нейронной сети

КУРСОВАЯ РАБОТА
по направлению подготовки
02.03.02— Фундаментальная информатика и информационные технологии

Научный руководитель –
доц., к. ф.-м. н. Абрамян Анна Владимировна

_____ _____
оценка (рейтинг) подпись руководителя

Ростов-на-Дону – 2024

Оглавление

Постановка задачи

Целью этой курсовой работы является исследование методов генерации изображений с применением генеративно-состязательных нейронных сетей, а также программная реализация данных методов с последующим анализом результатов.

Необходимо провести исследование существующих подходов в проектировании генеративно-состязательных нейронных сетей, рассматривая книги и публикации от авторитетных источников. На основании этого исследования требуется реализовать генеративно-состязательные модели, которые смогут выполнять задачу генерации изображений рукописных цифр MNIST. Помимо этого следует провести тестирование модели на более сложном наборе данных - наборе данных изображений лиц знаменитостей CelebA.

На основании результатов полученных при реализации методов генерации требуется провести сравнение изучаемых моделей и сделать выводы об их эффективности в задаче генерации изображений.

Введение

Создание изображений с применением генеративно-состязательных нейронных сетей или GAN (Generative Adversarial Network) является одним из наиболее перспективных направлений в области машинного обучения.

Генеративно-состязательные сети были представлены Яном Гудфеллоу и его коллегами в 2014 году. Эта архитектура нейронных стала эффективным и удобным инструментом для производства различных данных, применимых как в исследовательской работе, так и в производстве.

Суть GAN заключается в “состязании” или “противоборстве” двух сетей. Первая из них – это дискриминатор. Эта сеть обучена различать реальные образцы от сгенерированных второй сетью. Вторая нейронная сеть – это, соответственно, генератор. Задача генератор состоит в создании образцов, которые дискриминатор не сможет отличить от изначального набора. Такой состязательный процесс обучения приводит к постоянному улучшению качества генерируемых данных.

Актуальность темы данной курсовой работы обусловлена быстрым увеличением потребности в создании большого количества правдоподобных высококачественных изображений. Эта потребность проявляется в различных сферах: от создания компьютерных игр и другого развлекательного контента до автоматизации процессов обработки и улучшения изображений в науке и медицине.

1. Генеративно-состязательная нейронная сеть (GAN)

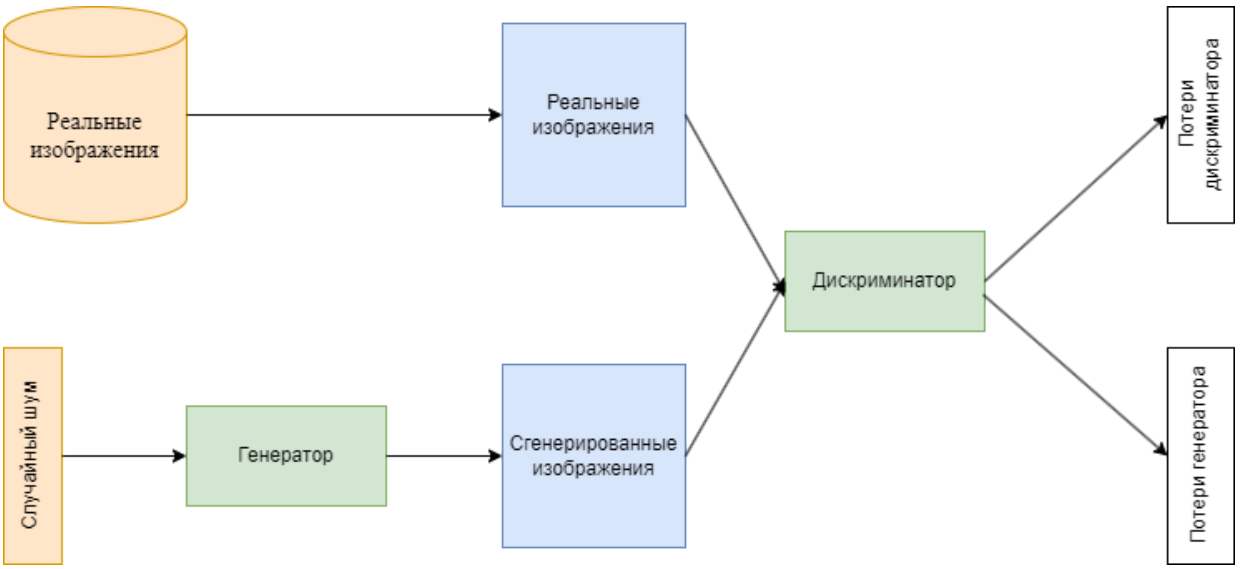
1.1. Обзор генеративно-состязательных нейронных сетей (GAN)

Генеративно-состязательные сети – это архитектура глубоких нейронных сетей, состоящая из двух моделей: генератор и дискриминатор (см. рис. 1). В процессе их совместного обучения генератор создает все более убедительные данные, а дискриминатор учится их эффективнее распознавать.

Модель генератора в GAN обучается на основании имеющихся образцов создавать новые данные. Задача этой модели – производство новых данных из набора случайных чисел (случайного шума), который формирует скрытое пространство (latent dimension) [3].

Модель дискриминатора в GAN занимается различением подлинных данных от тех, что были синтезированы генератором. Её задача – анализировать и классифицировать информацию, поступающую от генератора, согласно установленным категориям. Эта модель способна проводить как многоклассовую, так и бинарную классификацию. Обычно в контексте GAN применяется именно бинарная классификация [3].

Рис.1. Схема генеративно-состязательной нейронной сети



По мнению Дэвида Фостера, высказанного им в книге “Генеративное глубокое обучение. Творческий потенциал нейронных сетей”, ключом к

пониманию генеративно-состязательных нейронных сетей является понимание процесса их обучения.

Обучение дискриминатора требует создания смешанного обучающего набора, в котором часть изображений будут настоящими, а часть сгенерированными. Ответ дискриминатора для настоящих изображений должен быть равен 1, а для поддельных – 0. Таким образом нужно обучить дискриминатор отличать исходные изображения от поддельных, выводя значения близкие к единице для реальных изображений и значения близкие к нулю для сгенерированных [1].

Генератор обучается создавать изображения, которые дискриминатор не сможет отличить от исходного набора. Это означает, что при анализе сгенерированного изображения дискриминатор должен возвращать значение близкое к 1. Обычно для решения задачи обучения генератора создают комбинированную сеть. Генератор такой комбинированной сети получает на вход вектор шума, из которого производит изображение, поступающие на вход дискриминатора комбинированной модели, после чего дискриминатор определяет вероятность того, что изображение было сгенерировано. На время обучения этой комбинированной модели обучение дискриминатора отключается, так как в ином случае дискриминатор начнет регулировать веса так, что он с большей вероятностью будет определять сгенерированные изображения как реальные [1].

Для того, чтобы измерить степень сходства генерируемых изображений и реальных данных используют целевые функции [3], которые были подробно описаны в статье Яна Гудфеллоу и его коллег под названием “Generative Adversarial Networks” [4]. У каждой сети есть целевая функция, которую в процессе обучения нейронная сеть пытается минимизировать. Конечная целевая функция для GAN имеет следующий вид:

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

Здесь x – это реальные данные, а z – вектор шума. D(x) – это вероятность, что реальные данные являются реальными, т.е. результат анализа реальных данных дискриминатором. G(z) – это сгенерированные данные. Тогда D(G(Z)) – это результат анализа сгенерированных данных дискриминатором. P_{data} – реальное распределение данных, а P_z – распределение данных, созданных генератором. E – ожидаемый выход [3].

В процессе обучения дискриминатор старается максимизировать результат, а генератор – минимизировать. Таким образом суть обучения генеративной сети сводится к достижению равновесия между дискриминатором и генератором [4]. Если равновесие достигается, то говорят что модель сошлась. Это равновесие обычно называют равновесием Нэша [3].

1.2. Применение GAN для генерации изображений рукописных цифр MNIST

В качестве языка программирования в этом примере и в дальнейших будет использоваться Python. Для реализации моделей применяются библиотеки Tensorflow и Keras. При разработке я пользовался средой Jupyter Notebook. Для демонстрации простой GAN я выбрал задачу генерации изображений на основе популярного датасета рукописных цифр MNIST.

Листинг 1. Подготовка пакетов данных для обучения сети GAN.

```
def prepare_mnist_batch(x_full, y_full,  num_label, batch_size):
    x_label = x_full[np.where(y_full == num_label)]
    dataset = tf.data.Dataset.from_tensor_slices(x_label)
    dataset = dataset.shuffle(1024)
    dataset = dataset.batch(batch_size, drop_remainder =
True).prefetch(1)
    return dataset
```

Первой задачей является предобработка данных датасета. Сначала я загружаю датасета из библиотеки Keras и объединяю тренировочный и тестовый набор, так как в этой задаче нет необходимости в разделении. Далее я нормализую пиксели в диапазоне [-1, 1] для дальнейшей работы с ними.

Потом отделяю только изображения необходимой мне цифры, потому что для каждой цифры я обучаю отдельную модель. Завершаю подготовку данных (см. листинг 1) созданием и группировкой в пакеты фиксированного размера объекта типа Dataset.

Листинг 2. Сеть генератора GAN.

```
def build_generator():
    return keras.models.Sequential([
    Input(shape = [NOISE_SIZE]),
    Dense(64),
    LeakyReLU(alpha = 0.2),
    BatchNormalization(momentum = 0.8),
    Dense(128),
    LeakyReLU(alpha = 0.2),
    BatchNormalization(momentum = 0.8),
    Dense(256),
    LeakyReLU(alpha = 0.2),
    BatchNormalization(momentum = 0.8),
    Dense(512),
    LeakyReLU(alpha = 0.2),
    BatchNormalization(momentum = 0.8),
    Dense(np.prod(IMG_SHAPE), activation = 'tanh'),
    Reshape(IMG_SHAPE)
], name = 'generator')
```

Модель генератора (см. листинг 2) состоит из полносвязных слоев Dense с различным числом нейронов. Входной слой сети – это слой Input, который принимает вектор шума размера 32. Также применяются слои активации LeakyReLU с небольшим отрицательным наклоном, что помогает избежать проблемы “умирающих нейронов”. Слои BatchNormalization нормализуют активацию предшествующих им слоев по каждому пакету, что помогает ускорить и стабилизировать процесс обучения. Архитектуру сети генератора можно разделить на блоки, состоящие из полносвязного слоя, слоя активации и слоя нормализации. Выходом сети является полносвязный слой с функцией активации гиперболический тангенс, а также слоя преобразования одномерного выхода предшествующего полносвязного слоя в изображение.

Модель дискриминатор (см. листинг 3) тоже состоит из полносвязных слоев Dense, включающих в себя выходной слой с сигмоидной функцией активации и одним нейроном. Помимо этого, как и в сети генератора,

используются слои активации LeakyReLU. Слой Flatten преобразует входное изображение формы [28, 28, 1] в одномерный вектор для дальнейшей обработки изображения полносвязным слоем. Кроме того, слои отсева Dropout случайным образом “выключают” часть нейронов во время обучения, что дает позволяет в определенных ситуациях предотвратить переобучение модели. По итогу в сети дискриминатора можно выделить блоки состоящие из полносвязного слоя, слоя активации и слоя отсева.

Листинг 3. Сеть дискриминатора GAN.

```
def build_discriminator():
    return keras.models.Sequential([
    Flatten(input_shape = IMG_SHAPE),
    Dense(512),
    LeakyReLU(alpha = 0.2),
    Dropout(rate = 0.3),
    Dense(256),
    LeakyReLU(alpha = 0.2),
    Dropout(rate = 0.3),
    Dense(1, activation = 'sigmoid')
], name = 'discriminator')
```

Для обучения генератора и дискриминатора я применяю метод оптимизации Adam. Adam [2] – это алгоритм, сочетающий лучшие аспекты методов оптимизации RMSProp и Momentum. Адаптивная скорость обучения, стабильность, а также то, что Adam требует меньше настройки параметров, чем другие методы оптимизации, делает его удобным и эффективным для использования в практических задачах, таких как обучение генеративно-состязательных нейронных сетей. Скорости обучения learning_rate присвоено значение 0.0002, а параметру beta_1 значение 0.5. Скорость обучения определяет размер шагов, с которыми оптимизатор обновляет веса в сети. Параметра beta_1 контролирует экспоненциальное затухание скорости для оценки первого момента градиентов.

На роль функции потерь был выбрал метод бинарной кросс-энтропии (Binary Crossentropy) [2]. Выбор обусловлен тем, что дискриминатор выполняет бинарную классификацию, определяя является ли изображение истинным или поддельным. Помимо этого этот метод представляет собой

меру расхождения между предсказаниями дискриминатора и истинным распределением. Это соответствует цели обучения генеративной сети – заставить распределение, создаваемое генератором, приблизиться к распределению настоящих данных. Бинарная кросс-энтропия вычисляется для каждого наблюдения по следующей формуле:

$$L(y, \widehat{y}) = -ylog(\widehat{y}) - (1 - y)log(1 - \widehat{y})$$

где y – это истинная метка (0 или 1), а $\widehat{y}$ – это предсказанная моделью вероятность. Эта функция потерь штрафует модель, если предсказанные вероятности далеки от истинных меток.

Рис. 2. Кривая потерь дискриминатора и генератора GAN, генерирующей цифры 1.

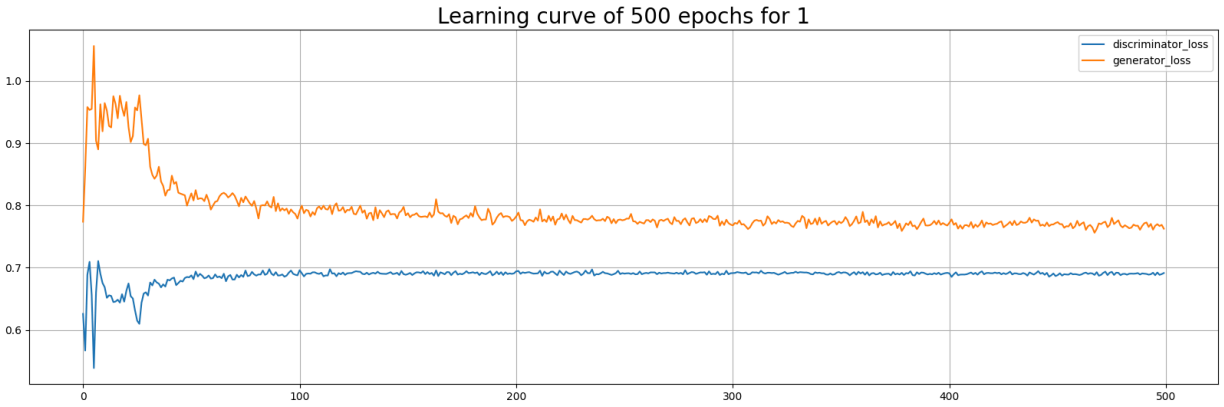
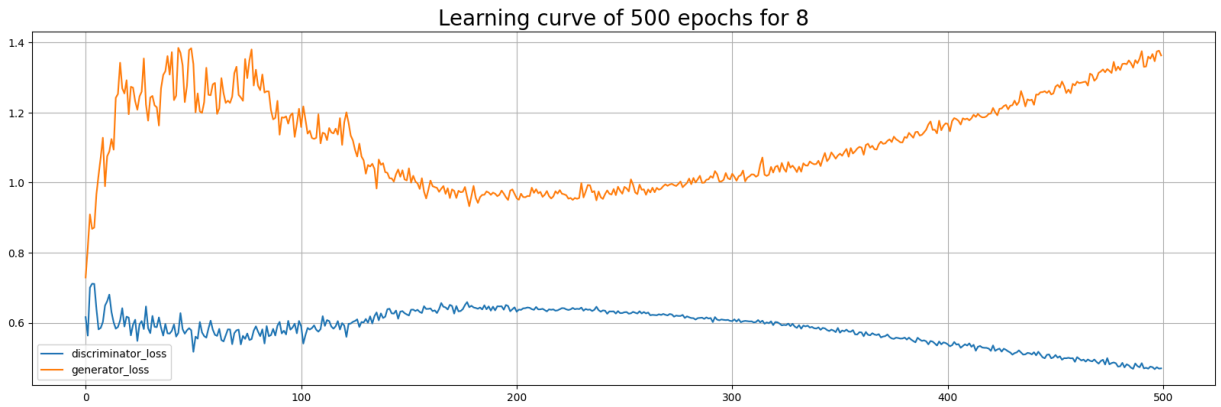


Рис. 3. Кривая потерь дискриминатора и генератора GAN, генерирующей цифры 8.



Модель обучалась на протяжении 500 эпох. По графику потерь (см. рис. 2) для модели, обученной генерировать цифры 1, можно увидеть, что потери генератора и дискриминатора стабилизировались в пределах определенных

значений. Однако по графику потерь (см. рис. 3) для модели генерирующей цифру 8 очевидно, что потери генератора неуклонно увеличиваются в то время, как потери дискриминатора уменьшаются. Также на созданных изображениях цифры 8 (см. рис. 5) присутствует больше случайных точек, чем на сгенерированных единицах (см. рис. 4). Эти проблемы связаны с неспособностью такой архитектуры модели GAN воспроизводить сложные узоры и изображения. Для улучшения результатов генерации следует использовать модификацию GAN, которая называется DCGAN и применяет сверточные слои.

Рис. 4. Изображения цифры 1, сгенерированные после 500 эпох обучения GAN

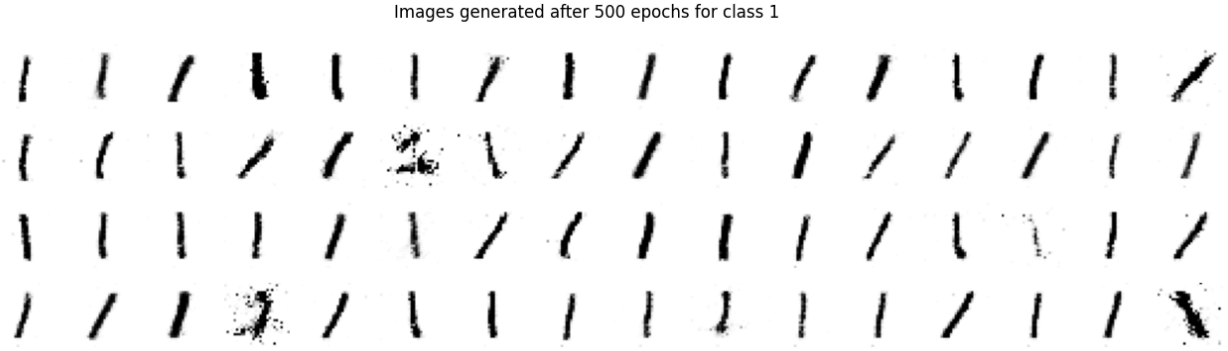
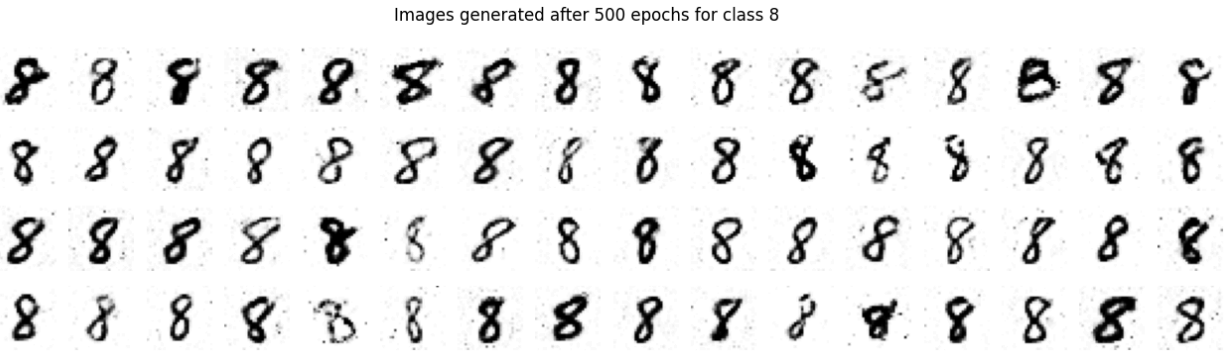


Рис. 5. Изображения цифры 8, сгенерированные после 500 эпох обучения GAN



2. Глубокая сверточная генеративно-состязательная нейронная сеть (DCGAN)

2.1. Обзор сверточных нейронных сетей (CNN)

Сверточные нейронные сети или CNN (convolutional neural networks) – это вид нейронных сетей, применяемых в обработке данных с сеточной топологией. Примером таких данных могут служить изображения. В таком случае изображения рассматриваются как двумерная сетка пикселей [2].

Этот вид нейронных сетей был подробно рассмотрен в книге “Глубокое обучение” за авторством Яна Гудфеллоу. Можно сказать, что сверточная нейронная сеть – это сеть, в которой хотя бы в одном слое применяется операция свертки [2].

Свертка – это в общем виде операция над двумя функциями вещественного аргумента:

$$(x * w)(t) = \int x(a)w(t - a)da$$

Здесь x и w – это функции над которыми производится свертка, t – это переменная, по которой производится свертка и a – это переменная интегрирования.

В контексте сверточных нейронных сетей функция x обычно называется входом, а функция w – ядром. Выход называется картой признаков.

Если предположить, что переменная t может принимать только целые значения, а также что x и w определены исключительно для целых значений t, то можно получить определение дискретной свертки:

$$(x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t - a)da$$

В задачах машинного обучения входом обычно служит тензор данных, а ядром – тензор параметров. Обычно предполагается, что данные функции w и x равны нулю всюду, кроме конечного множества точек.

Нередко производятся свертки сразу по нескольким осям. Так если входом будет двумерное изображение, то и ядро должно быть двумерным:

$$(I * K)(i,j) = \sum_m \sum_n I(m,n)K(i - m, j - n)$$

В связи с тем, что свертка коммутативна, формулу допустимо записывать и в следующем виде:

$$(K * I)(i,j) = \sum_m \sum_n I(i - m, j - n)K(m,n)$$

Именно эту формулу проще реализовать в библиотеках машинного обучения, так как диапазон допустимых значений m и n меньше [2].

Операция свертки в случае обработки изображений в машинном обучении сводится к тому, что ядро двигается по двумерной матрице изображения поэлементно перемножая и суммируя ту часть данных, которую оно в данный момент покрывает. Эти операции повторяются после каждого смещения ядра на значение, называемое шагом свертки. Все матрицы полученные после обработки изображения ядрами объединяются в один тензор. На выходе получается новое изображение другого размера [1].

При работе с одноканальными изображениями термины фильтр и ядро можно приравнять. Однако для цветных изображений это не так, потому что фильтр - это коллекция ядер, где каждое ядро соответствует одному каналу. По итогу обработки изображения формируется общий канал на основании суммирования матриц этой коллекции [1].

Целью операции свертки в нейронных сетях является выделение из исходного изображения высокоуровневых признаков, таких как объекты по краям изображения. Таким образом первые сверточные слои в сети применяются для извлечения низкоуровневых признаков. К низкоуровневым признакам относится, например, цвет. С добавлением большего числа слоев свертки нейронная сеть начинает анализировать признаки все более высокого уровня [1]. По итогу такой операции размерность признака может

уменьшаться, оставаться неизменной или даже увеличиваться в зависимости от задач для которых строится нейронная сеть.

Сверточные нейронные сети могут применяться в решении множества задач связанных с обработкой изображений. Они оказываются полезны как в классификации, так и в генерации и модификации входных данных.

2.2. Обзор глубоких сверточных генеративно-состязательных нейронных сетей (DCGAN)

Изначально сложные нейронные сети, наподобие сверточных нейронных сетей и рекуррентных нейронных сетей не применялись в работе с GAN. Однако результаты сверточных сетей в работе с изображениями привели к развитию глубоких сверточных генеративно-состязательных нейронных сетей или DCGAN (Deep Convolutional Generative Adversarial Network), предложенных Алексом Редфордом и его коллегами в статье “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”.

Архитектура и процесс обучения DCGAN схожи с GAN. Однако в DCGAN применяются сверточные слои, за которыми идут слои нормализации или объединения, а далее следуют слои функции активации. В DCGAN дискриминатор понижает частоту дискретизации изображения с помощью сверточных слоев и нормализации, после чего применяет полносвязный слой для классификации изображения как реального или сгенерированного. Генератор получает произвольный вектор шума из скрытого пространства, увеличивает частоту дискретизации, после чего генерирует изображение [3].

DCGAN может применятся в различных сферах работ с изображениями. Задачи, которые могут выполнять такие нейронные сети варьируется от увеличения разрешения и восстановления изображений до генерации уникального контента, что оказывается полезно в исследованиях процессов обучения нейронных сетей и разработке новых алгоритмов.

2.3. Применение DCGAN для генерации изображений рукописных цифр MNIST

В этом примере предобработка изображения и процесс обучения схожи с предыдущим примером генерации изображений рукописных цифр. Основные различия содержаться в архитектурах дискриминатора и генератора.

Листинг 4. Сеть генератора DCGAN

```
def build_generator():
    return keras.models.Sequential([
    Dense(7 * 7 * 256, input_shape=[NOISE_SIZE]),
    BatchNormalization(momentum = 0.8),
    LeakyReLU(alpha = 0.2),
    Reshape([7, 7, 256]),
    Conv2DTranspose(128, kernel_size = 4, strides = 1, padding =
'same'),
    BatchNormalization(momentum = 0.8),
    LeakyReLU(alpha = 0.2),
    Conv2DTranspose(64, kernel_size = 4, strides = 2, padding =
'same'),
    BatchNormalization(momentum = 0.8),
    LeakyReLU(alpha = 0.2),
    Conv2DTranspose(1, kernel_size = 4, strides = 2, padding =
'same', activation = 'tanh'),
], name = 'generator')
```

Первым слоем сети генератора (см. листинг 4) является полносвязный слой, который преобразует входной вектор шума в тензор формы [None, 7*7*256]. Слои BatchNormalization и LeakyReLU работают соответственно примеру обычной GAN. Слой преобразования Reshape меняет форму входного тензора на [None, 7, 7, 256].

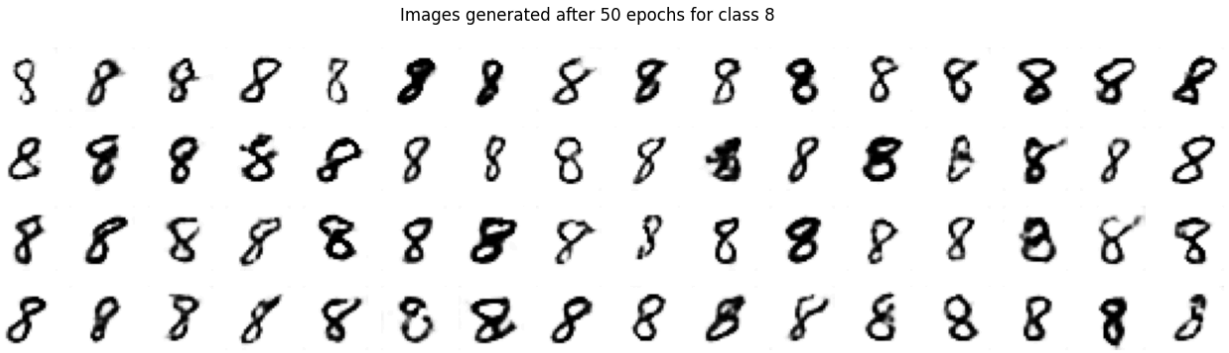
Деконволюционные слои Conv2DTranspose используются для увеличения пространственных размеров тензора, что позволяет преобразовать шум в выходное изображение. Имеется выходной деконволюционный слой с 1 фильтром, размером ядра 4x4 и шагом 2, а также гиперболической функцией активации, которая ограничивает выходные значения в диапазоне [-1, 1].

Листинг 5. Сеть дискриминатора DCGAN

```
def build_discriminator():
    return keras.models.Sequential([
    Conv2D(64, kernel_size = 4, strides = 2, padding = 'same',
input_shape = IMG_SHAPE),
    LeakyReLU(alpha = 0.2),
    Dropout(rate = 0.3),
    Conv2D(128, kernel_size = 4, strides = 2, padding = 'same'),
    LeakyReLU(alpha = 0.2),
    Dropout(rate = 0.3),
    Conv2D(256, kernel_size = 4, strides = 2, padding = 'same'),
    LeakyReLU(alpha = 0.2),
    Dropout(rate = 0.3),
    Flatten(),
    Dense(1, activation = 'sigmoid')
], name = 'discriminator')
```

Сеть дискриминатора (см. листинг 5) начинается со сверточного слоя, применяющего 64 фильтра с размером ядра 4x4 и шагом 2 к входному изображению. Отступ (padding) во всех сверточных слоях сети генератора является “одинаковым” (same), что означает, что к входным данным добавляются дополнительные нули, чтобы после применения сверточного слоя размер выходных данных оставался таким же, как и размер входных данных. Это позволяет анализировать различные признаки изображения, но при этом сохранять пространственные параметры изображения на протяжении всей сети, что не дает сети терять информацию на краях изображения [1]. Далее в сети используются слой активации LeakyReLU и слой отсева Dropout, как в примере простой GAN. После этого следуют еще два блока состоящих из сверточного слоя, слоя активации и слоя отсева. В этих блоках постепенно увеличивается число фильтров, что позволяет анализировать признаки более высоких уровней. Выходной слой представляет собой полносвязный слой с одним нейроном и сигмоидальной функцией активации.

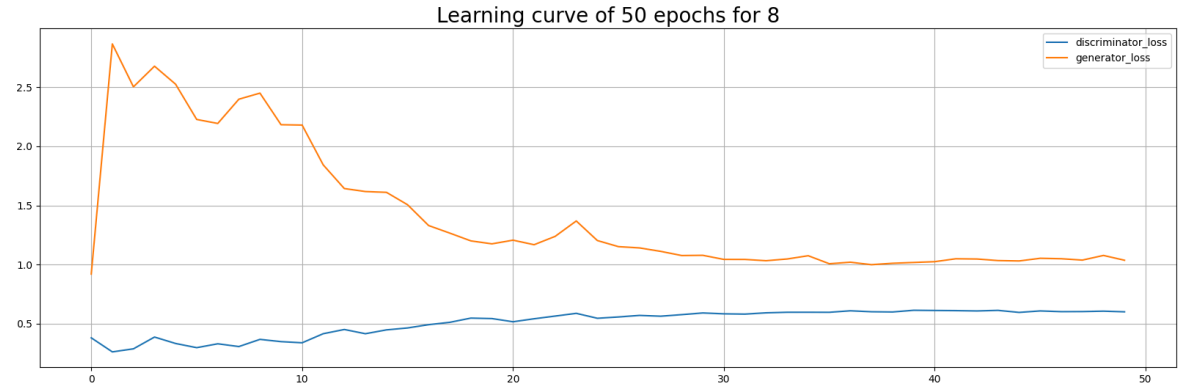
Рис. 6. Изображения цифры 8, сгенерированные
после 50 эпох обучения DCGAN.



Как и в случае с простой GAN применяется оптимизатор Adam и функция потерь BinaryCrossentropy. Модель обучалась на протяжении 50 эпох.

В сравнении с примером обычной генеративно-состязательной сети без сверточных слоев, генератор способен создавать более реалистичные и разнообразные изображения с меньшим количеством артефактов и случайных точек (см. рис. 6). Также можно увидеть, что показатели потерь (см. рис. 7) стабилизируются для всех рукописных цифр от 0 до 9. Результаты показывают все преимущества применения сверточных слоев в задаче генерации изображений.

Рис. 7. Кривая потерь дискриминатора и генератора DCGAN,
генерирующей цифры 8.

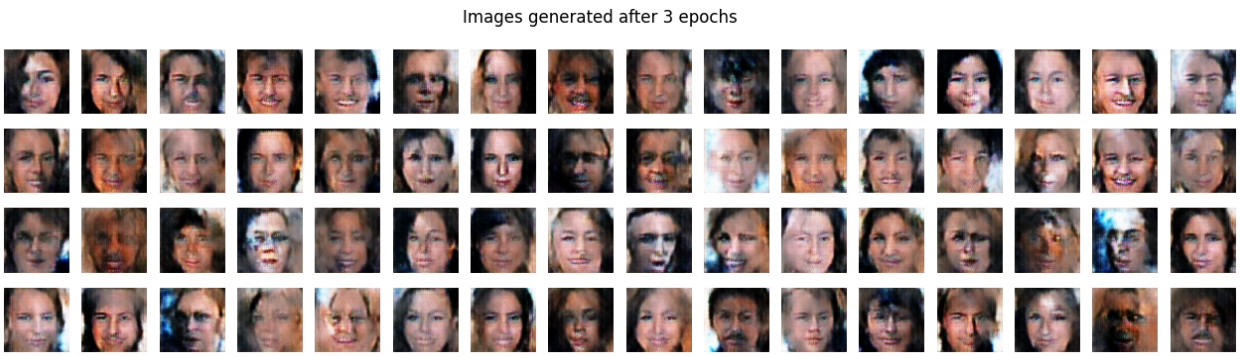


2.4. Применение DCGAN для генерации изображений лиц

Для демонстрации работы DCGAN на более сложном наборе данных я выбрал задачу генерации изображений лиц на основе датасета CelebA, содержащего 200 000 изображений лиц знаменитостей. Предварительно выбрал 50 000 случайных изображений из этого датасета и обрезал изображения так, чтобы на них остались только лица. Далее изменил размер изображения до размера 64x64 пикселей и нормализовал пиксели в диапазоне [0, 1].

Основным отличием модели генератора данной DCGAN от сети для генерации изображений MNIST является наличие большего числа блоков скрытых слоев, а также в функции активации на выходном слое сети, которая была заменена с гиперболического тангенса на сигмоидную в связи с тем, что пиксели изображения нормализуются не в пределах [-1, 1], а в пределах [0, 1]. Помимо этого число нейронов на выходном слое сети было изменено с одного на три из-за того, что теперь генерируются цветные изображения. Единственным изменением в сети дискриминатора является один добавленный блок слоев, состоящий из сверточного слоя, слоя активации и слоя отсева.

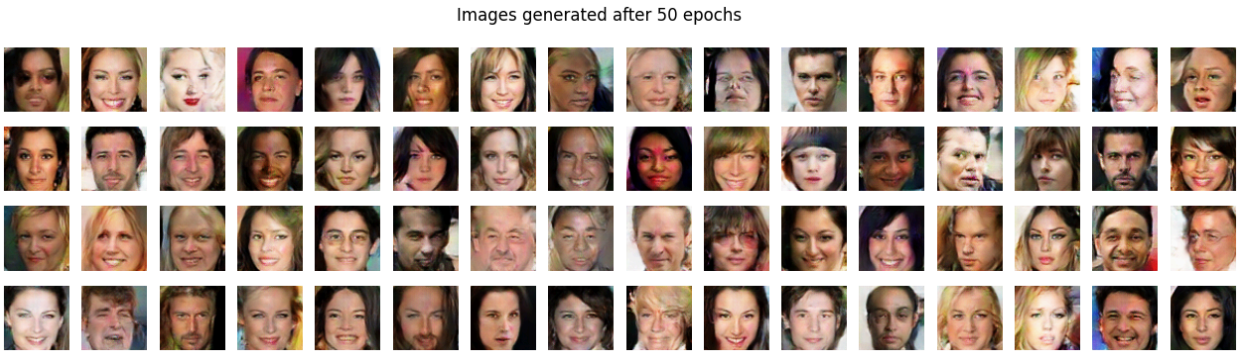
Рис. 8. Изображения лиц, сгенерированные
после 3 эпох обучения DCGAN



По мере обучения уменьшалось число артефактов на изображениях и увеличивалась правдоподобность создаваемых лиц (см. рис. 8). По итогу 50 эпох обучения генератор способен воспроизводить изображения лиц схожие с

содержимым датасета CelebA (см. рис. 9), что показывает эффективность DCGAN в решении задачи генерации сложных изображений.

Рис. 9. Изображения лиц, сгенерированные
после 50 эпох обучения DCGAN.



Заключение

В результате выполнения этой курсовой работы была исследована задача генерации изображений с применением генеративно-состязательных нейронных сетей на примере генерации изображений рукописных цифр MNIST и генерации изображений лиц.

Простая GAN использовалась для генерации изображений рукописных цифр на основе датасета MNIST. Были продемонстрированы базовые принципы GAN и их способность обучаться воспроизводить новые данные на основе простых примеров.

Далее была реализована модель DCGAN для решения той же задачи генерации изображений рукописных цифр. Это позволило наглядно показать преимущества применения сверточных слоев при работе с изображениям. Помимо этого DCGAN была применена для генерации реалистичных изображений лиц. Благодаря сверточным слоям сеть смогла обучиться на более сложном наборе данных и создать реалистичные изображения человеческих лиц.

По итогу демонстрации и сравнения результатов был сделан вывод о малой эффективности GAN без сверточных слоев, по сравнению с DCGAN, в задаче генерации изображений.

Программная реализация проекта выложена на GitHub [5].

Литература

1. Дэвид Фостер «Генеративное глубокое обучение. Творческий потенциал нейронных сетей»
2. Ян Гудфеллоу «Глубокое обучение»
3. Кайлаш Ахирвар «Состязательные сети. Проекты»
4. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio «Generative Adversarial Networks»
5. https://github.com/venexene/GANImgGen