

**РЕФЕРАТ**

Выпускная квалификационная работа 45 с., 3 ч., 14 рис., 4 источников.

НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ, ОБРАБОТКА ИЗОБРАЖЕНИЙ

Объектом исследования является процесс распознавания изображений с использованием генеративно-состязательных сетей (GAN). Конкретно, исследование направлено на разработку и обучение GAN для распознавания изображений, тестирование GAN.

Цель выпускной квалификационной работы – создание генеративно-состязательной нейронной сети, её обучение, создание приложения для использования нейронной сети, а также проверка конечного результата, апробация.

В результате выпускной квалификационной работы была создана генеративно-состязательная нейронная сеть, а также было написано приложение “GAN Lab” для удобной работы с ней.

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc137579268)

[1 Обзор архитектур нейронных сетей для распознавания изображений и различных типов нейронных сетей, используемых для распознавания изображений 6](#_Toc137579269)

[1.1 Обзор архитектуры нейронной сети 6](#_Toc137579270)

[1.2 Функция активации 9](#_Toc137579271)

[1.3 Типы ошибок в задачах обучения нейронных сетей 12](#_Toc137579272)

[1.4 Функция потерь 13](#_Toc137579273)

[1.5 Типы нейронных сетей, используемых для распознавания изображений 15](#_Toc137579274)

[1.6 Переобучение 19](#_Toc137579275)

[2 Методы обучения нейронных сетей для распознавания изображений 20](#_Toc137579276)

[2.1 Методы обучения 20](#_Toc137579277)

[2.2 Предварительная обработка изображения, используемых для улучшения качества и точности распознавания изображений 22](#_Toc137579278)

[2.3 Оценка и сравнение производительности нейронных сетей 23](#_Toc137579279)

[2.4 Применение нейронных сетей для распознавания изображений в реальных приложениях 25](#_Toc137579280)

[2.5 Библиотеки открытого исходного кода для машинного обучения и глубокого обучения на языке программирования Python 27](#_Toc137579281)

[3 Реализация GAN и её обучение 31](#_Toc137579282)

[3.1 Реализация дискриминатора 31](#_Toc137579283)

[3.2 Реализация генератора 32](#_Toc137579284)

[3.3 Создание интерфейса 35](#_Toc137579285)

[3.4 Выбор данных для обучения нейронной сети и их подготовка 36](#_Toc137579286)

[3.5 Функция потерь и оптимизатор 38](#_Toc137579287)

[3.6 Процесс обучения 40](#_Toc137579288)

[Заключение 43](#_Toc137579289)

[Список использованных источников 45](#_Toc137579290)

# **ВВЕДЕНИЕ**

Ключевой сферой компьютерного зрения, находящей свое применение во многих сферах, включая медицину, автоматическое управление, автомобильную промышленность, робототехнику и многие другие, является распознавание изображений. При этом, задача распознавания объектов на изображениях является достаточно трудной из-за сложности и разнообразия изображений, которые необходимо обработать.

До того, как нейронные сети получили широкое распространение, распознавание объектов на изображениях выполнялось с помощью различных алгоритмов компьютерного зрения, которые были основаны на извлечении определенных признаков из изображения. Однако, данные методы имели свои недостатки и ограничения: невозможность обработки больших объемов данных, сложность в анализе изображений с различным контекстом и слабая возможность к обнаружению неизвестных объектов.

Введение нейронных сетей в область распознавания изображений позволило решить многие проблемы. Нейронные сети, в отличие от классических методов, не требуют извлечения признаков из изображений вручную, а обучаются на больших объемах данных с известными метками. Они также позволяют обрабатывать изображения с различным контекстом, таким как фон, освещение и масштаб, и могут обнаруживать неизвестные объекты.

Существует несколько типов нейронных сетей, которые применяются в задаче распознавания объектов на изображениях: сверточные нейронные сети (Convolutional Neural Networks, CNNs), рекуррентные нейронные сети (Recurrent Neural Networks, RNNs) и глубокие нейронные сети (Deep Neural Networks, DNNs). Каждый тип нейронной сети имеет свои особенности и применяется для решения соответствующих задач.

Нейронные сети также имеют свои ограничения. К таковым относят необходимость большого количества данных для обучения и наличие большой вычислительной мощности. Так же, нейронные сети могут страдать от проблемы переобучения.

Несмотря на эти ограничения, нейронные сети продолжают оставаться одним из наиболее эффективных методов для распознавания объектов на изображениях, находящих свое применение во многих областях, продолжая развиваться, благодаря появлению новых алгоритмов.

Актуальность выпускной квалификационной работы обусловлена тем, что на сегодняшний день нейросеревые технологий обработки изображений охватывают множество областей. Они позволяют автоматически анализировать, классифицировать и понимать содержимое изображений. И их развитие играет важную роль в жизни человека.

Цель выпускной квалификационной работы – создание генеративно-состязательной нейронной сети, а также создание приложения для удобной работы с ней.

Исходя из этой цели необходимо выполнить следующие задачи:

1. Изучить строение нейронной сети и рассмотреть особенности генеративно-состязательной нейронной сети.
2. Создать генеративно-состязательную нейронную сеть и затем обучить её на заранее подготовленном наборе данных.
3. Написать приложение для удобной работы с генеративно-состязательной нейронной сетью.

# **1 Обзор архитектур нейронных сетей для распознавания изображений и различных типов нейронных сетей, используемых для распознавания изображений**

## **1.1 Обзор архитектуры нейронной сети**

Нейронные сети − это математические модели, созданные для обработки информации, которые являются аналогом структуры и функций нервной системы живых организмов, состоящие из нейронов, взаимосвязанных между собой и выполняющие функцию обработки данных. В данном случае, основу представляют нейроны, моделирующие функциональные элементы нервной системы. После принятия входных сигналов, нейрон обрабатывает их и выдает выходной сигнал. При этом каждый входной сигнал должен быть соответственно умножен на весовой коэффициент, определяющий важность данного сигнала для выходного. После, сумма произведений входных сигналов и их весовых коэффициентов подается на функцию активации, определяющую выходной сигнал нейрона [1].

Функция активации склонна иметь различные формы, такие как: сигмоидальную или гиперболический тангенс. Они применяются для ограничения значения выходного сигнала нейрона в определенном диапазоне и уменьшения влияния выбросов во входных данных.

Нейроны объединяются в слои, а те в свою очередь, могут быть как скрытыми, так и выходными. Скрытые слои, обрабатывающие данные между входным и выходным слоями, и являются ключевой особенностью нейронных сетей. Количество скрытых слоев и количество нейронов в каждом слое зависят от конкретной задачи (рисунок 1).

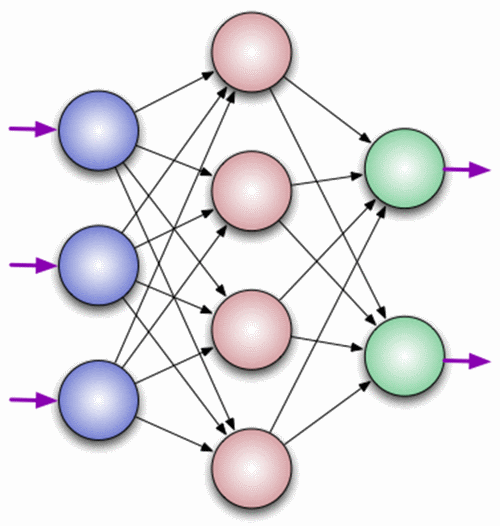


Рисунок 1 – Графическое изображение нейронной сети

Обучение нейронной сети происходит благодаря настройке весовых коэффициентов, происходящей на основе входных данных и ожидаемых выходных значений. Ошибка выходных данных сравнивается с ожидаемыми значениями, применяясь для корректировки весовых коэффициентов, путем метода обратного распространения ошибки (рисунок 2).

Обученная нейронная сеть способна применяться для решения широкого спектра задач. К таковым относятся: распознавание образов, классификация, регрессия и другие. Так же существуют различные типы нейронных сетей: сверточные нейронные сети, рекуррентные нейронные сети и глубокие нейронные сети, применимые в различных областях: от распознавания изображений до обработки естественного языка.

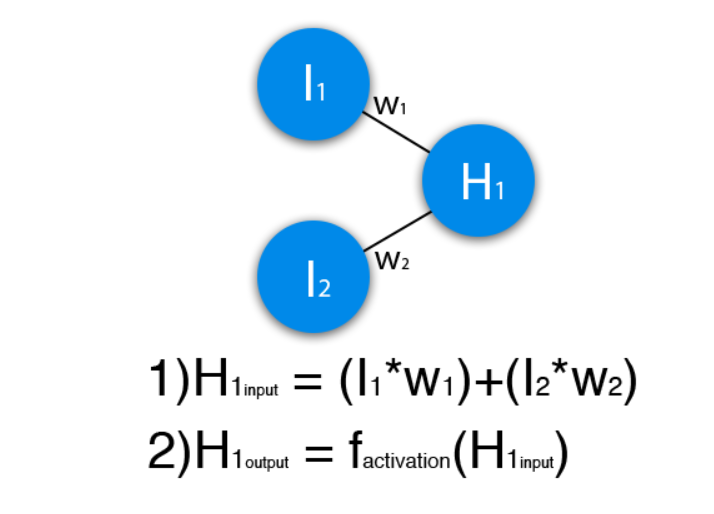


Рисунок 2 – Подсчёт выходных данных на нейроне

Сверточные нейронные сети (Convolutional Neural Networks, CNN) используются для анализа изображений. Состоят из нескольких слоев, включая в себя сверточные слои, выполняющие фильтрацию изображения,   
и слои пулинга, уменьшающие размерность данных. Эти слои позволяют сверточной нейронной сети автоматически извлекать признаки изображения и классифицировать его.

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) широко используются для анализа последовательностей данных: речевых или текстовых. Они обладают памятью и способны сохранять информацию о предыдущих значениях входных данных, благодаря чему могут использовать контекстную информацию. Это делает их особенно полезными для задач, связанных с естественным языком.

Глубокие нейронные сети (Deep Neural Networks, DNN) − это нейронные сети с несколькими скрытыми слоями, обладающие большой вычислительной мощностью и способностью к извлечению признаков, что позволяет им эффективно решать сложные задачи, такие как распознавание речи или обработка изображений.

## **1.2 Функция активации**

Функция активации является одним из важнейших элементов в нейронной сети, используется для нелинейного преобразования входного сигнала нейрона. Она способна определять выходной сигнал нейрона на основе входных данных и параметров весов.

В зависимости от типа нейронной сети и ее задачи, функции активации могут быть различными. Некоторые функций активации представляют:

Линейная функция: данная функция используется крайне редко, за исключением частных случаев, когда необходимо протестировать нейронную сеть или без преобразований передать значение (рисунок 3). Уравнение рассчитывают по формуле (1).

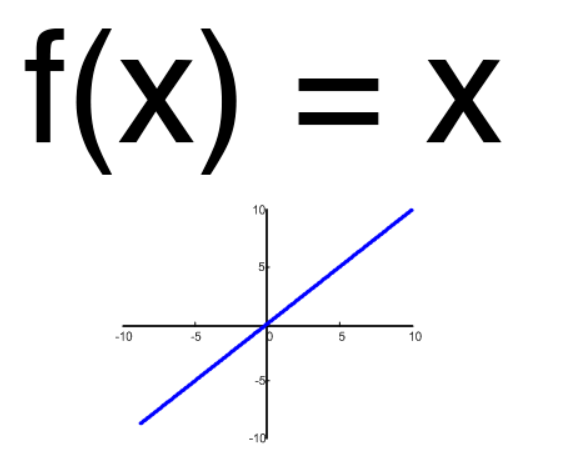


Рисунок 3 – График линейной функции

Сигмоидальная функция (Sigmoid function): эта функция преобразует любое входное значение в диапазон между 0 и 1. В отличие от первой, достаточно часто используется в работе нейронных сетей для бинарной классификации (рисунок 4) и рассчитывается по формуле (2).

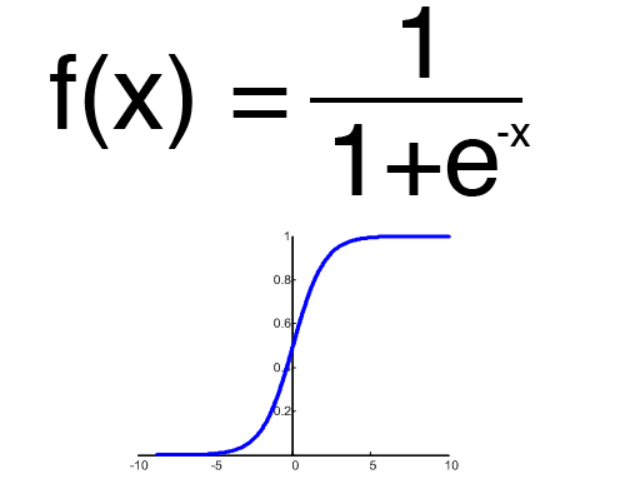


Рисунок 4 – График сигмоидальной функции

Гиперболический тангенс (Hyperbolic Tangent, Tanh): функция также преобразует входное значение в диапазон между -1 и 1. Данная функция представляет собой функцию активации и часто используется в рекуррентных нейронных сетях (рисунок 5). Уравнение функции выражается в виде формулы (3).

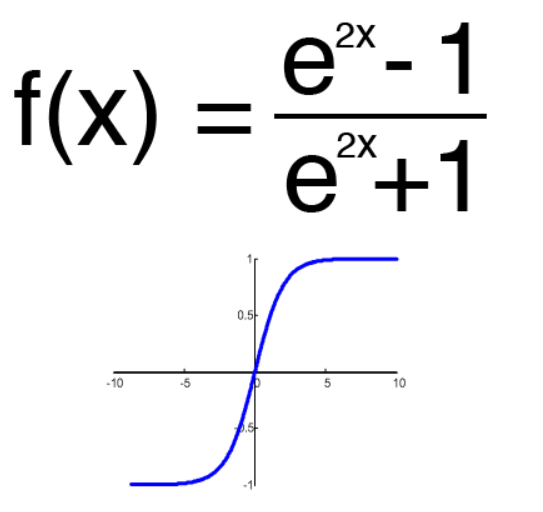


Рисунок 5 – График гиперболического тангенса

Функция ReLU (Rectified Linear Unit): возвращает 0 для всех отрицательных значений и возвращает входное значение для всех положительных значений и рассчитывается по формуле (4). ReLU эффективна для глубоких нейронных сетей, так как способствует быстрому их обучению и сходимости алгоритма обратного распространения ошибки.

Функция Softmax: функция используется для классификации на несколько классов и преобразует входное значение в вероятность классов.

Каждая функция имеет свои преимущества и недостатки, а их выбор зависит от конкретной задачи и архитектуры сети. Правильный выбор функции активации значительно повышает эффективность и работоспособность нейронной сети.

Выбор функции активации зависит от нескольких факторов: тип задачи, данные, используемые для обучения, архитектура нейронной сети и т.д.

Например, если задача состоит в бинарной классификации, то функция активации типа Sigmoid или Tanh может быть наиболее подходящей, так как они преобразуют входное значение в диапазон между 0 и 1 или между -1 и 1 соответственно, что может использоваться для предсказания вероятности принадлежности к классу.

Если же речь идет о многоклассовой классификации, то функция активации типа Softmax может быть более подходящей, так как позволяет преобразовывать входные значения в вероятности принадлежности к каждому из классов.

Функция активации ReLU может быть полезной для глубоких нейронных сетей, так как способствует более быстрому обучению и сходимости алгоритма обратного распространения ошибки. Также, данная функция может быть предпочтительнее других функций для решения задачи обнаружения объектов на изображениях.

Помимо этого, при выборе функции активации важно учитывать наличие градиентов, ибо алгоритм обратного распространения ошибки требует градиентов для нахождения оптимальных значений весов в нейронной сети.

Следовательно, выбор функции активации напрямую зависит от конкретной задачи и архитектуры нейронной сети, и требует тщательного анализа и экспериментов.

## **1.3 Типы ошибок в задачах обучения нейронных сетей**

Ошибка (Error) - это отклонение между предсказанным и истинным значением целевой переменной в задаче обучения нейронной сети. В машинном обучении мы хотим обучить модель, которая имеет возможность предсказывать целевую переменную на основе поступающих данных, и наша задача заключается в минимизировании ошибки.

Существуют метрики, которые позволяют измерять ошибки в задачах машинного обучения. Таковыми являются:

Среднеквадратичная ошибка (Mean Squared Error, MSE) – одна из самых распространенных метрик для регрессии. Она вычисляется как среднее значение квадратов отклонений между предсказанными и истинными значениями.

Средняя абсолютная ошибка (Mean Absolute Error, MAE) - это также метрика для регрессии, но в свою очередь вычисляется как среднее значение абсолютных отклонений между предсказанными и истинными значениями.

Категориальная кросс-энтропия (Categorical Cross-Entropy, CCE) − метрика, которая используется для классификации с несколькими классами. Она как бы находит разницу между предсказанными вероятностями и истинными вероятностями классов.

Бинарная кросс-энтропия (Binary Cross-Entropy, BCE) – данная метрика используется для задач бинарной классификации. Также, как и предыдущая, измеряет разницу между предсказанными вероятностями и истинными вероятностями.

Точность (Accuracy) − это метрика, которая измеряет процент правильных предсказаний.

Для вычисления ошибки необходимо сравнение предсказанных значений с истинными значениями целевой переменной, которые мы знаем из обучающего набора данных. После, мы используем выбранную метрику, с целью вычисления ошибки. Наша цель в данный момент минимизировать ошибку на обучающем наборе данных.

## **1.4 Функция потерь**

Функция потерь (Loss function) − это математическая функция, измеряющая качество предсказаний целевой переменной моделью на тренировочных данных. Представляет собой числовую меру расхождения между предсказанной моделью и реальными значениями целевой переменной.

Функция потерь является основным составляющим обучения модели. Цель − минимизировать ошибку на обучающих данных и создать модель, которая будет делать точные предсказания на новых данных. В процессе обучения, оптимизатор пытается уменьшить значение функции потерь, меняя веса модели.

В машинном обучении множество различных функций потерь, и выбор зависим от: конкретной задачи, типа модели и вида данных.

Наиболее распространенные из них:

Mean Squared Error (MSE) − среднеквадратичная ошибка, используемая для задач регрессии в тех случаях, когда целевая переменная является непрерывной отраженна в формуле (5).

где n − количество наблюдений;

y − реальное значение целевой переменной;

yhat − предсказанное значение целевой переменной.

Binary Cross Entropy Loss − функция потерь для бинарной классификации, используемая в случаях, когда целевая переменная принимает только два значения. Уравнение функции представлено в формуле (6).

где y − реальное значение целевой переменной;

p(yi) − предсказанное значение целевой переменной.

Categorical Cross Entropy Loss − функция потерь для многоклассовой классификации, используемая, когда целевая переменная является категориальной и выражается формулой (7).

где yi − реальное значение целевой переменной для i-го класса;

yhat i − предсказанное значение целевой переменной для i-го класса.

Hinge Loss − функция потерь, используемая для классификации, особенно для опорных векторных машин (SVM). Hinge Loss можно вычислить по формуле (8).

где y − реальное значение целевой переменной (-1 или 1);

yhat − предсказанное значение целевой переменной.

Huber Loss − функция потерь, используемая для регрессии, которая менее чувствительна к выбросам, чем MSE. Данная функция рассчитывается по формуле (9).

где delta − параметр, который определяет порог, на котором переключается функция потерь.

Выбор функции потерь важен в создании нейронной модели. Оптимальную функцию потерь можно подобрать опытным путем, экспериментируя с разными вариантами и выбирая ту, которая дает лучшие результаты на тестовых данных.

## **1.5 Типы нейронных сетей, используемых для распознавания изображений**

Распознавание изображений − наиболее распространенная задача в области машинного обучения, для ее решения используются различные типы нейронных сетей. Пример некоторых:

Сверточные нейронные сети (Convolutional Neural Networks, CNN) − это популярный тип нейронных сетей, используемый для обработки изображений. Представляет собой набор слоев, каждый из которых преобразует входные данные в некоторый выход. Слои сверточной нейронной сети включают в себя: сверточные слои, слои подвыборки (pooling), слои активации и полносвязные слои (рисунок 6). Сверточные слои используются для обнаружения признаков в изображениях, а слои подвыборки для уменьшения размерности данных. Слои активации помогают модели изучить нелинейные зависимости в данных, а полносвязные слои для классификации изображений [4].

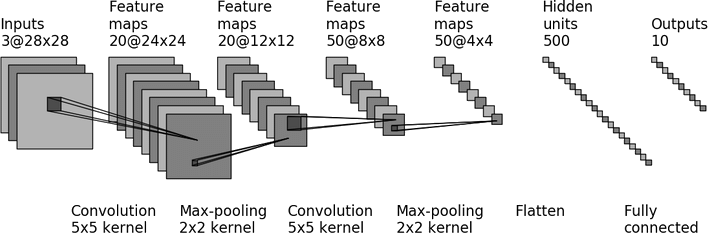


Рисунок 6 – Схема сверточной нейронной сети

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) − это тип нейронных сетей, широко используется для обработки последовательностей: тексты и речь. Рекуррентные нейронные сети состоят из набора повторяющихся блоков, каждый из которых имеет вход, выход и скрытое состояние (рисунок 7). Рекуррентные нейронные сети используются для обработки изображений, если изображение представлено в виде последовательности.

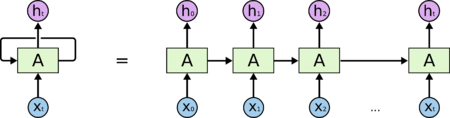


Рисунок 7 – Схема рекуррентной нейронной сети

Глубокие нейронные сети (Deep Neural Networks, DNN) − это тип нейронных сетей, имеющий множество скрытых слоев. Чем больше слоев, тем глубже и сложнее становится сеть. Глубокие нейронные сети используются для решения задач классификации и регрессии, а также для обработки изображений.

Сверточно-рекуррентные нейронные сети (Convolutional Recurrent Neural Networks, CRNN) − это комбинация сверточных и рекуррентных нейронных сетей. Широко используется для обработки изображений, где каждый пиксель рассматривается как последовательность.

Автоэнкодеры (Autoencoders) − это нейронные сети, используемые для изучения скрытого представления данных. Автоэнкодеры могут использоваться для обработки изображений путем сжатия их до более компактного представления. Состоит из двух частей: энкодера, который преобразует входные данные в скрытое представление, и декодера, который преобразует скрытое представление обратно в исходные данные. Таким образом, автоэнкодеры могут использоваться для изучения скрытых признаков изображений и для генерации новых изображений.

Генеративно-состязательные сети (Generative Adversarial Networks, GAN) − это нейронные сети, которые используются для генерации новых изображений. Они состоят из двух сетей: генеративной и дискриминативной. Генеративная сеть генерирует новые изображения, а дискриминативная сеть пытается отличить эти изображения от реальных. Задача генеративной сети - создавать изображения, которые дискриминативная сеть не сможет отличить от реальных. GAN могут использоваться для генерации новых изображений, а также для улучшения качества изображений.

Нейронные сети с вниманием (Attention Neural Networks) − это тип нейронных сетей, который используется для обработки последовательностей и для обработки изображений. Они позволяют моделировать важность каждой части входных данных при принятии решений. Таким образом, они позволяют моделям сосредоточиться на наиболее важных частях данных при решении задач [3].

Выбор конкретной архитектуры нейронной сети зависит от ряда факторов, включая:

Тип задачи: различные типы задач требуют разных архитектур нейронных сетей. Например, для распознавания объектов на изображениях обычно используют сверточные нейронные сети, в то время как для обработки последовательностей, таких как тексты или речь, обычно используют рекуррентные нейронные сети.

Количество данных: некоторые архитектуры требуют большого количества данных для обучения, в то время как другие могут обучаться на меньшем объеме данных. Например, глубокие нейронные сети требуют большого количества данных для обучения, тогда как нейронные сети с вниманием могут работать с меньшим объемом данных.

Вычислительная мощность: некоторые архитектуры требуют большого количества вычислительной мощности для обучения и работы, в то время как другие могут работать на более слабых устройствах. Например, сверточные нейронные сети требуют большого количества вычислительной мощности для обучения, тогда как нейронные сети с вниманием могут работать на более слабых устройствах.

Сложность модели: более сложные архитектуры могут иметь более высокую точность, но требуют большего времени и ресурсов для обучения и работы. Например, глубокие нейронные сети могут достигать высокой точности, но требуют большого количества времени и ресурсов для обучения.

Наличие предобученных моделей: некоторые архитектуры имеют предобученные модели, которые могут быть использованы для решения сходных задач. Это может ускорить процесс обучения и улучшить точность модели.

В целом, выбор архитектуры нейронной сети зависит от конкретной задачи, доступных ресурсов и требований к точности и скорости работы модели.

## **1.6 Переобучение**

Переобучение (overfitting) − это явление, когда нейронная сеть хорошо обучается на тренировочных данных, но показывает плохую производительность на новых, ранее не виданных данных. В этом случае сеть "запоминает" тренировочные данные, вместо того, чтобы изучать общие закономерности в данных, и не способна обобщать на новые данные.

Избежать переобучения можно с помощью следующих методов:

Регуляризация: добавление штрафных элементов в функцию потерь, которые уменьшают веса сети, чтобы уменьшить их абсолютные значения. Например, L1 и L2 регуляризация.

Dropout: случайное отключение нейронов на каждой итерации обучения. Это предотвращает "зависимость" нейронов друг от друга и помогает сети избежать переобучения.

Уменьшение размера сети: уменьшение количества слоев или нейронов может помочь предотвратить переобучение.

Ранняя остановка (early stopping): остановка процесса обучения, когда производительность на валидационном наборе данных перестает улучшаться.

Увеличение размера обучающей выборки: более разнообразные данные могут помочь сети обобщать на новые данные и избежать переобучения.

Data augmentation: модификация существующих обучающих данных для создания новых, более разнообразных данных.

Transfer learning: использование предварительно обученных моделей, которые обучались на схожих задачах, и дообучение их на новых данных.

Выбор конкретного метода зависит от конкретной задачи и данных. Обычно комбинация нескольких методов может дать лучший результат.

# **2 Методы обучения нейронных сетей для распознавания изображений**

## **2.1 Методы обучения**

Методы обучения нейронных сетей используются для улучшения точности и эффективности распознавания изображений. Существует несколько методов обучения:

Обучение с учителем (supervised learning) − это метод обучения нейронной сети, при котором используется набор обучающих данных, состоящий из входных данных и соответствующих им правильных ответов. Эти правильные ответы называются метками (labels).

В процессе обучения нейронная сеть использует эти метки   
для корректировки своих весов и биасов, чтобы минимизировать функцию потерь (loss function) и улучшить точность предсказаний на новых данных.

В случае распознавания изображений, входные данные представляют собой изображения, а метки − соответствующие классы, которым принадлежат эти изображения. Например, если речь идет об обучении нейронной сети для классификации изображений кошек и собак, то каждому изображению кошки будет соответствовать метка "кошка", а каждому изображению собаки - метка "собака".

Одним из наиболее популярных алгоритмов обучения с учителем для распознавания изображений является обратное распространение ошибки (backpropagation). Он заключается в пересчете весов и биасов нейронной сети на основе ошибок, допущенных в процессе предсказания меток для каждого изображения в обучающем наборе.

Обучение с учителем требует большого количества размеченных данных для достижения высокой точности предсказаний на новых данных.

Обучение без учителя (unsupervised learning) − это метод машинного обучения, в котором алгоритмы пытаются извлечь скрытые структуры или закономерности из не размеченных данных. В отличие от обучения с учителем, в котором модель обучается на основе размеченных данных с явными метками классов, в обучении без учителя модель не имеет явных меток классов и должна самостоятельно выявлять скрытые структуры и закономерности в данных.

Основным методом обучения без учителя является кластеризация, которая позволяет группировать объекты на основе их сходства. Другими методами обучения без учителя являются снижение размерности, поиск аномалий, генеративные модели и др.

Примеры алгоритмов обучения без учителя в задачах распознавания изображений включают в себя метод главных компонент (PCA) для снижения размерности данных, метод k-средних для кластеризации изображений и алгоритмы генеративных моделей, такие как автокодировщики (autoencoders) и генеративные состязательные сети (GANs) для генерации изображений.

Обучение без учителя может быть полезным для задач, в которых размеченных данных недостаточно или их получение является трудоемким и дорогостоящим процессом. Однако, поскольку модели в обучении без учителя не получают явных меток классов, оценка их точности и качества может быть более сложной задачей, чем в обучении с учителем.

Обучение с подкреплением (reinforcement learning) − это метод обучения машинного обучения, при котором агент принимает действия в некоторой среде, и получает положительную или отрицательную награду в зависимости от результата своих действий. Цель агента − максимизировать награду, и он может использовать свой опыт для улучшения своих действий в будущем.

Обучение с подкреплением применяется в различных областях, включая игры, робототехнику, управление процессами и т.д. Для обучения с подкреплением используется модель марковского процесса принятия решений (Markov decision process, MDP), которая определяет правила, по которым агент выбирает действия в каждом состоянии.

Одним из основных компонентов обучения с подкреплением является функция оценки ценности (value function), которая оценивает, насколько хороши действия агента в каждом состоянии. Эта функция может быть вычислена с помощью алгоритмов, таких как Q-обучение или SARSA. Кроме того, для выбора действий агентом может быть использована стратегия, которая определяет, каким образом выбираются действия в каждом состоянии.

Обучение с подкреплением может быть сложным процессом, поскольку агент может столкнуться с проблемой исследования-использования (exploration-exploitation dilemma), когда он должен выбирать между изучением новых действий и использованием известных действий для максимизации награды. Для решения этой проблемы могут использоваться различные методы, такие как жадная стратегия, эпсилон-жадная стратегия, стратегия мягкой максимизации и т.д.

## **2.2 Предварительная обработка изображения, используемых для улучшения качества и точности распознавания изображений**

Предварительная обработка изображений − это важный шаг при работе с изображениями, который может существенно улучшить качество распознавания объектов на изображениях. Ниже приведен обзор методов предварительной обработки изображений.

Фильтрация изображений: Один из наиболее распространенных методов предварительной обработки изображений, который позволяет убрать шумы на изображениях и улучшить их качество. Для этого используются различные фильтры, такие как фильтр Гаусса, медианный фильтр и др.

Увеличение разрешения изображений: Этот метод предварительной обработки изображений позволяет улучшить качество изображений, увеличивая разрешение изображения без потери качества. Для этого используются различные методы, такие как интерполяция изображений и Super-Resolution.

Цветовая коррекция: Метод предварительной обработки, который позволяет корректировать цвет изображений. Цветовая коррекция может быть использована для согласования цвета между несколькими изображениями, улучшения контраста и убирания эффекта смещения цветов.

Конвертация цветных изображений в оттенки серого: Метод предварительной обработки, который может быть использован для упрощения обработки изображений и уменьшения размера данных для обучения нейронных сетей.

Нормализация изображений: Метод предварительной обработки, который позволяет нормализовать яркость и контрастность изображений, что может улучшить качество и точность распознавания объектов на изображениях.

Обрезка изображений: Метод предварительной обработки, который может быть использован для удаления ненужных элементов на изображениях и улучшения качества распознавания объектов на изображениях.

Нормализация данных: Метод предварительной обработки, который может быть использован для нормализации данных, например, нормализация данных обучения, чтобы сбалансировать вклад каждой обучающей выборки в обучение нейронной сети.

## **2.3 Оценка и сравнение производительности нейронных сетей**

Оценка и сравнение производительности нейронных сетей является важной задачей в области машинного обучения. В частности, для задач распознавания изображений, таких как классификация или сегментация, необходимо оценить точность, скорость и эффективность различных архитектур нейронных сетей.

Метрики точности являются одним из наиболее распространенных методов оценки производительности нейронных сетей для задач распознавания изображений. Они измеряют, насколько хорошо модель классифицирует изображения на правильные категории.

Одной из основных метрик точности является accuracy (точность), которая определяет долю правильно классифицированных объектов в наборе данных. Формула для вычисления accuracy:

accuracy = (количество правильных предсказаний) / (общее количество предсказаний)

Другой популярной метрикой является F1 score, которая учитывает не только количество правильных предсказаний, но и количество ложно-положительных и ложно-отрицательных результатов. F1 score можно вычислить по формуле (10):

где precision (точность) − это доля правильно предсказанных положительных результатов относительно всех положительных результатов;

recall (полнота) − это доля правильно предсказанных положительных результатов относительно всех истинных положительных результатов.

Скорость также является важной метрикой производительности нейронных сетей. Скорость работы нейронной сети зависит от размера входных данных, архитектуры сети и используемого оборудования. Существуют различные методы ускорения работы нейронных сетей, такие как параллелизация вычислений и использование аппаратного ускорения (например, графических процессоров).

Эффективность является метрикой, которая оценивает соотношение между точностью и скоростью работы нейронной сети. Часто используемой метрикой эффективности является FLOPS (floating point operations per second), которая измеряет количество операций с плавающей запятой, которые может выполнить нейронная сеть в секунду.

Некоторые из методов сравнения архитектур нейронных сетей включают в себя сравнение количества параметров и сложности модели. Количество параметров − это количество настраиваемых весов и смещений, которые требуется оптимизировать в процессе обучения модели. Более простые модели имеют меньшее количество параметров и обычно более быстро обучаются, но могут иметь низкую точность. Сложность модели - это количество операций, которые требуются для выполнения вычислений в нейронной сети. Более сложные модели могут иметь более высокую точность, но могут быть более медленными и требовательными к вычислительным ресурсам.

Другой метод сравнения архитектур нейронных сетей − это сравнение результатов на стандартных наборах данных, таких как CIFAR-10, CIFAR-100, MNIST и ImageNet. Эти наборы данных широко используются для сравнения производительности различных архитектур нейронных сетей. Некоторые соревнования, такие как ImageNet Large Scale Visual Recognition Challenge (ILSVRC), используются для тестирования различных архитектур и оценки их производительности.

## **2.4 Применение нейронных сетей для распознавания изображений в реальных приложениях**

Нейронные сети широко применяются для распознавания изображений в реальных приложениях, так как они могут достичь высокой точности и обеспечивать автоматизацию задач, которые раньше требовали большого количества ручной работы. Рассмотрим некоторые из наиболее распространенных приложений распознавания изображений, использующих нейронные сети:

1. Распознавание объектов − это одна из наиболее распространенных задач в области компьютерного зрения, которая заключается в определении, какие объекты находятся на изображении и где они находятся. Примером может служить система автоматического распознавания лиц на фотографиях или в видеопотоке.
2. Автоматическое распознавание лиц − это приложение распознавания изображений, которое использует нейронные сети для идентификации лиц на фотографиях или в видеопотоке. Такие системы используются для различных целей, от безопасности до облегчения процесса идентификации на фотографиях в социальных сетях.
3. Распознавание жестов − это приложение распознавания изображений, которое используется для распознавания жестов, сделанных руками, и их преобразования в соответствующие команды. Это приложение может использоваться в различных областях, например, в контроллерах игр, в системах управления презентациями и т.д.
4. Распознавание паттернов − это приложение, которое использует нейронные сети для распознавания паттернов на изображениях, таких как текстуры, шаблоны и прочее. Оно может использоваться в различных областях, включая медицину, промышленность и т.д.
5. Оптическое распознавание символов (OCR) − это приложение распознавания изображений, которое используется для распознавания текста на изображении и его преобразования в текстовый формат. Оно может использоваться для автоматического распознавания номеров паспортов, банковских карт и т.д.
6. Распознавание медицинских изображений − это приложение, которое использует нейронные сети для распознавания различных медицинских изображений, таких как рент графии, МРТ, УЗИ и т.д.   
   Оно может использоваться для автоматической диагностики и классификации различных заболеваний на основе медицинских изображений.
7. Распознавание эмоций − это приложение, которое использует нейронные сети для распознавания эмоций на лицах людей на фотографиях или в видеопотоке. Оно может использоваться для различных целей, включая оценку эмоциональной реакции на рекламу, изучение поведения покупателей в магазинах и т.д.

В целом, нейронные сети для распознавания изображений имеют широкий спектр применений в различных областях

## **2.5 Библиотеки открытого исходного кода для машинного обучения и глубокого обучения на языке программирования Python**

С развитием машинного обучения и искусственного интеллекта, фреймворки для глубокого обучения, такие как TensorFlow и Keras, стали широко используемыми инструментами для создания и обучения нейронных сетей. TensorFlow и Keras предоставляют высокоуровневые API для создания, обучения и применения нейронных сетей с минимальными затратами на программирование. В этой главе мы рассмотрим основные функции TensorFlow и Keras и их применение в машинном обучении.

TensorFlow TensorFlow − это открытый программный фреймворк для машинного обучения, разработанный компанией Google. Он был создан для обработки больших объемов данных и построения масштабируемых вычислительных моделей глубокого обучения. Основным объектом в TensorFlow является граф вычислений, который состоит из узлов (операций) и ребер (тензоров), где каждый узел представляет математическую операцию, а каждое ребро - поток данных.

Основные функции TensorFlow:

1. Создание графа вычислений: TensorFlow предоставляет API для создания графа вычислений, который позволяет определить математические операции и их зависимости друг от друга. Этот граф может быть оптимизирован и распределен между несколькими устройствами для ускорения обучения и применения моделей.
2. Автоматическое дифференцирование: TensorFlow позволяет автоматически вычислять производные функций, что делает возможным использование алгоритмов градиентного спуска для обучения нейронных сетей.
3. Работа с различными типами данных: TensorFlow поддерживает работу с различными типами данных, такими как числа с плавающей точкой, целые числа, строки и т.д.
4. Распределенное обучение: TensorFlow позволяет распределять обучение между несколькими устройствами, такими как CPU и GPU, а также между несколькими узлами в сети.
5. Визуализация графа вычислений: TensorFlow предоставляет инструменты для визуализации графа вычислений, что позволяет легко отслеживать и отлаживать модели.

Keras Keras − это высокоуровневый API для машинного обучения, разработанный с учетом простоты и быстроты создания и обучения нейронных сетей. Keras работает поверх TensorFlow, что позволяет использовать все преимущества TensorFlow вместе с удобством работы в Keras. Он поддерживает большое количество типов нейронных сетей, таких как сверточные нейронные сети, рекуррентные нейронные сети, автоэнкодеры и т.д.

Основные функции Keras:

1. Простота использования: Keras предоставляет удобный интерфейс для создания и обучения нейронных сетей с минимальным количеством кода.
2. Модульность: Keras позволяет легко создавать сложные модели, состоящие из нескольких слоев, каждый из которых выполняет свою специфическую функцию.
3. Поддержка множества типов нейронных сетей: Keras поддерживает множество типов нейронных сетей, что позволяет выбрать наилучший тип для конкретной задачи.
4. Автоматическое дифференцирование: Keras использует TensorFlow для автоматического вычисления производных функций, что позволяет использовать алгоритмы градиентного спуска для обучения нейронных сетей.
5. Работа с различными типами данных: Keras поддерживает работу с различными типами данных, такими как числа с плавающей точкой, целые числа, строки и т.д.
6. Обучение на GPU: Keras позволяет обучать нейронные сети на GPU, что ускоряет процесс обучения.
7. Применение TensorFlow и Keras в машинном обучении: TensorFlow и Keras используются для решения множества задач машинного обучения, таких как:
8. Классификация изображений: TensorFlow и Keras широко используются для классификации изображений, таких как фотографии, рентгеновские снимки и т.д. Нейронные сети, обученные на большом количестве изображений, могут точно определять объекты на изображении и классифицировать их.
9. Распознавание речи: TensorFlow и Keras могут использоваться для обучения нейронных сетей, которые распознают речь и переводят ее в текст. Это может быть полезно, например, для создания голосовых помощников.
10. Рекомендательные системы: TensorFlow и Keras могут использоваться для обучения нейронных сетей, которые рекомендуют пользователю товары или услуги на основе его предпочтений и поведения в интернете.
11. Обработка естественного языка: TensorFlow и Keras могут использоваться для обучения нейронных сетей, которые обрабатывают текст и выполняют задачи, такие как классификация текста, машинный перевод и т.д.
12. Анализ временных рядов: TensorFlow и Keras могут использоваться для обучения нейронных сетей, которые анализируют временные ряды, такие как данные о температуре, финансовые данные и т.д [2].

Кроме того, TensorFlow и Keras используются во многих реальных приложениях, таких как автономные автомобили, распознавание рукописного текста, голосовые помощники и т.д. Они являются ключевыми инструментами для решения сложных задач машинного обучения и продолжают развиваться и улучшаться с каждым годом.

# **3 Реализация GAN и её обучение**

## **3.1 Реализация дискриминатора**

Сверточную нейронную сеть, используемую в задаче генеративно-состязательных сетей (GANs), и представляет собой дискриминатор.

Основная цель сверточной нейронной сети − отличать подлинные данные от сгенерированных.

Архитектура модели состоит из нескольких слоев (рисунок 8):

1. Входной слой (InputLayer): Принимает входные изображения размером 64x64 пикселя и 3 цветовых канала.
2. Сверточный слой (Conv2D): Сверточный слой с 64 фильтрами размером 3×3 и функцией активации нелинейности. Он уменьшает размерность изображения вдвое (32×32) и создаёт 64 канала.
3. LeakyReLU слой: Применяет функцию активации "leaky ReLU" для введения нелинейности.
4. Dropout слой: Слой, применяющий случайное отключение некоторых нейронов для борьбы с переобучением.
5. Дополнительные сверточные слои (Conv2D): Аналогичные сверточному слою 1, но с увеличенным количеством фильтров (128, 256 и 512 соответственно). Каждый слой уменьшает размерность изображения вдвое.
6. Flatten слой: Преобразует выходы последнего сверточного слоя в плоский вектор.
7. Полносвязный слой (Dense): Слой с 1 нейроном, который выдает оценку подлинности изображения.



Рисунок 8 – Реализация дискриминатора на языке python

Общее количество параметров модели составляет 4,314,753, и все они являются обучаемыми. Это означает, что модель будет обновлять эти параметры в процессе обучения, чтобы научиться отличать подлинные изображения от сгенерированных.

## **3.2 Реализация генератора**

Генератор представляет обратную сверточную нейронную сеть, которая используется в генеративно-состязательных сетях (GANs). Она отвечает за генерацию фальшивых изображений, которые в идеале должны быть достаточно реалистичными, чтобы обмануть дискриминатор.

Архитектура модели "generator" включает следующие слои (рисунок 9):

1. Входной слой (InputLayer): Принимает вектор шума   
   размером 128.
2. Полносвязный слой (Dense): Преобразует входной шум в вектор размером 8192.
3. BatchNormalization слой: Нормализует данные по батчу для стабилизации обучения.
4. LeakyReLU слой: Применяет функцию активации "leaky ReLU" для введения нелинейности.
5. Reshape слой: Преобразует выходы предыдущего слоя в 4-мерный тензор размером 4×4×512.
6. Транспонированный сверточный слой (Conv2DTranspose): Использует операцию обратной свертки для увеличения размера тензора в два раза. Начинает с 4×4×512 и преобразует его в 8×8×256.
7. Дополнительные слои BatchNormalization и LeakyReLU: Применяются для стабилизации и введения нелинейности.
8. Дополнительные транспонированные сверточные слои (Conv2DTranspose): Продолжают увеличивать размер тензора в два раза на каждом слое (16×16×128, 32×32×64 и 64×64×32 соответственно).
9. Последний сверточный слой (Conv2D): Преобразует тензор в финальное изображение размером 64×64 с 3 цветовыми каналами.
10. Activation слой: Применяет активацию на последнем слое для получения окончательного сгенерированного изображения.



Рисунок 9 – Реализация генератора на языке python

Общее количество параметров модели составляет 5,437,667, из которых 5,420,323 являются обучаемыми. Модель обучается путем обновления этих параметров с целью генерации высококачественных и реалистичных изображений.

Модели "generator" и "discriminator" работают вместе в процессе обучения GAN. Генератор пытается создать изображения, которые дискриминатор не сможет отличить от реальных данных, а дискриминатор стремится правильно классифицировать подлинные и сгенерированные изображения. Этот процесс обучения позволяет достигнуть улучшения качества генерации изображений со временем.

## **3.3 Создание интерфейса**

Для создания интерфейса будем использовать библиотеку Tkinter, которая позволяет создает интерфейс с заданными элементами:

Окно размером 400×400 в котором есть (рисунок 10):

Кнопка "Select Dataset Folder", которая позволяет выбрать папку в которой храниться датасет на котором планируется обучить нейронную сеть.

Поле "Enter the number of epochs" в которое вводится количество эпох.

Кнопка "Start Training", которая запускает обучение.

Прогресс обучения представлен в виде шкалы прогресса, которая заполняется по мере обучения нейронной сети.

Кнопка "Generate Image", при нажатии на которую в новом окне будет появляться изображение, сгенерированное нейронной сетью.

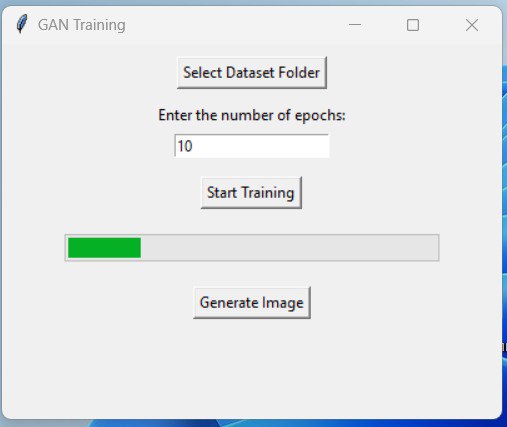


Рисунок 10 – Интерфейс нейронной сети

## **3.4 Выбор данных для обучения нейронной сети и их подготовка**

Попробуем обучить нейронную сеть на двух наборах данных.

Первый набор данных:

Первый датасет, который мы будем использовать содержит изображения лиц персонажей мультфильмом.

Рассмотрим некоторые особенности этого датасета (рисунок 11):

1. Содержание: Датасет содержит порядка 21551 изображений, представляющих лица различных персонажей. Все изображения имеют формат JPEG или PNG.
2. Разнообразие персонажей: В датасете представлены различные стили анимации и типы персонажей. А также представлены лица с различными выражениями эмоций, такими как счастье, грусть, удивление, злость и др. А также персонажи могут быть в различных позах и ракурсах.
3. Разрешение изображений: Разрешение изображений варьируется от 64×64 до 128×128.



Рисунок 11 – Примеры изображений из первого датасета

Перед загрузкой изображений в нейронную сеть для обучения нужно их подготовить.

Второй набор данных:

Второй датасет, который мы будем использовать содержит изображения котов.

Рассмотрим некоторые особенности этого датасета (рисунок 12):

1. Содержание: Датасет содержит порядка 15067 изображений, представляющих различные породы котов. Все изображения имеют формат JPEG или PNG.
2. Разрешение изображений: Разрешение изображений варьируется начинается от 256×256.

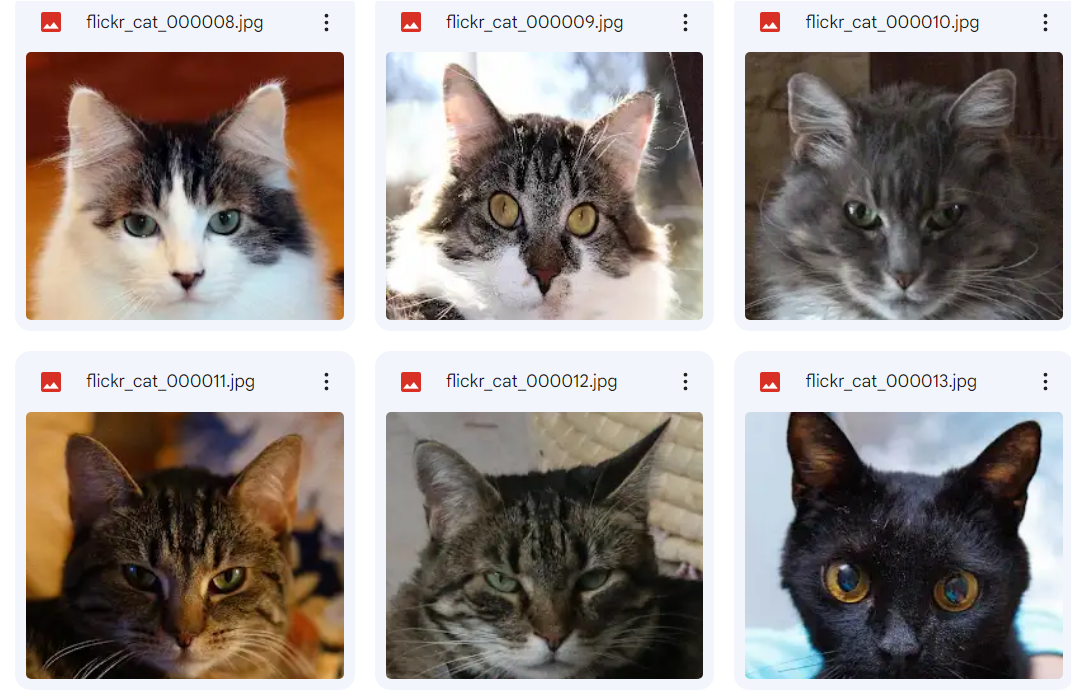


Рисунок 12 – Примеры изображений из второго датасета

Шаги необходимые для подготовки:

1. Приводим все изображения к единому формату (jpg).
2. Изменяем размер всех изображений до 64×64
3. Тип данных изображения приводится к tf.float32. Нейронные сети обычно ожидают входные данные в формате чисел с плавающей точкой.
4. Нормализуем значения пикселей изображения. Значения пикселей сдвигаются на 127,5 и затем делятся на 127,5, что приводит к значениям пикселей в диапазоне от -1 до 1. Это нужно для улучшение сходимости обучения: Нейронные сети лучше работают с входными данными, которые имеют небольшие значения и близкие к нулю.

## **3.5 Функция потерь и оптимизатор**

Выбор функции активации и оптимизатора являются важными аспектами при проектировании и обучении нейронных сетей. Функция активации играет решающую роль в обработке сигналов нейронной сети. Она определяет выходной сигнал каждого нейрона в сети и вносит нелинейность в модель, что позволяет сети обрабатывать сложные зависимости в данных.

В предоставленном коде выбрана функция потерь Binary Crossentropy (бинарная перекрестная энтропия). Она реализована с помощью функции "tf.keras.losses.BinaryCrossentropy". Эта функция потерь применяется для обучения дискриминатора и генератора в модели GAN.

Бинарная перекрестная энтропия используется для оценки различия между реальными и сгенерированными изображениями. Она помогает модели GAN стремиться к созданию более реалистичных изображений. В данном случае, функция потерь "BinaryCrossentropy" принимает в качестве аргумента "from\_logits=True", что означает, что выходные значения модели не подвергаются функции активации.

Помимо выбора функции активации, также необходимо уделить внимание выбору оптимизатора. Оптимизаторы используются для обновления весов сети в процессе обратного распространения ошибки. Они определяют, каким образом и с какой скоростью веса будут изменяться, чтобы минимизировать ошибку и улучшить производительность сети.

Оптимизатор, выбранный в предоставленном коде, − это Adam. Оптимизаторы отвечают за обновление параметров модели на основе градиентов функции потерь. Adam (Adaptive Moment Estimation) является одним из популярных методов оптимизации. Он адаптивно регулирует скорость обучения для каждого параметра, основываясь на истории градиентов и квадратов градиентов. В коде оптимизатор Adam инициализируется с параметрами "learning\_rate=0.0002" и "beta\_1=0.5". Параметр "learning\_rate" управляет скоростью обучения модели, а параметр "beta\_1" управляет экспоненциальным сглаживанием градиентов.

Процесс выбора случайного шумового вектора, передачи его через генератор и получение выходного изображения.

Процесс генерации новых изображений:

1. Случайный шумовой вектор: Генератор начинает процесс, принимая на вход случайный шумовой вектор. Этот вектор обычно является многомерным и содержит случайные значения, которые представляют собой информацию для генерации изображения. В предоставленном коде размерность шумового вектора определена как "latent\_dim", равная 128.
2. Генерация изображения: Случайный шумовой вектор передается через генератор. Генератор состоит из слоев, таких как плотный (Dense) слой, деконволюционные (deconvolution) блоки и сверточные (convolution) блоки. В процессе прохождения через эти слои случайный шумовой вектор преобразуется в изображение. В предоставленном коде генератор представлен функцией "build\_generator()", которая принимает шумовой вектор в качестве входа и возвращает сгенерированное изображение.
3. Получение выходного изображения: Выходной слой генератора использует функцию активации "tanh", чтобы ограничить значения пикселей в диапазоне от -1 до 1, что является обычным для изображений. Таким образом, генератор возвращает изображение, которое может быть использовано как выходная точка для обучения дискриминатора и дальнейшего улучшения генерируемых изображений.

**3.6 Процесс обучения**

Во время обучения GAN на первом датасете было выбрано 100 эпох, что означает, что весь обучающий набор данных будет пройден 100 раз.

После обучения получаем (рисунок 13):



Рисунок 13 – Результат работы GAN на первом наборе

В результате этого относительно простого датасета, GAN достаточно быстро смог достичь хороших результатов. Это может быть связано с тем, что исходные данные имели небольшой уровень сложности. GAN смог легко улавливать основные закономерности и создавать реалистичные сэмплы.

Во время обучения GAN на втором датасете я выбрал 2000 эпох, что означает, что весь обучающий набор данных будет пройден 2000 раз.

После обучения получаем (рисунок 14):



Рисунок 14 – Результат работы GAN на втором наборе

Результаты, полученные после обучения на втором датасете, оказались хуже по сравнению с первым случаем. Это может быть обусловлено тем, что исходные данные были более сложными и содержали больше вариаций и деталей. GAN, требуя больше времени на изучение такого сложного датасета, мог не справиться с улавливанием всех нюансов и особенностей данных.

Таким образом, сложность и разнообразие исходных данных может значительно влиять на результаты обучения GAN. Более сложные данные требуют более длительного и глубокого обучения для достижения высокого качества сэмплов.

В данной работе была представлена реализация генеративно-состязательной сети (GAN), состоящей из генератора и дискриминатора. Дискриминатор обучается различать настоящие изображения (из обучающего набора) от сгенерированных генератором. Генератор обучается создавать изображения таким образом, чтобы они максимально походили на реальные изображения и "обманывали" дискриминатор.

Для обучения GAN в данной работе была выбрана бинарная кросс-энтропия (BinaryCrossentropy) в качестве функции потерь для дискриминатора и генератора. Эта функция позволяет измерять разницу между предсказаниями дискриминатора и истинными метками данных. Цель состоит в том, чтобы минимизировать эту разницу и достичь оптимальной работы обеих сетей.

Для активации в данной реализации GAN была использована функция "LeakyReLU" (линейная ректификация с утечкой) с параметром альфа равным 0.2. Эта функция активации позволяет избежать проблемы "мертвых нейронов" в нейронных сетях и способствует более стабильному обучению GAN.

Данный проект нельзя назвать конечным, так как функционал и его возможности требуют расширения:

1. Улучшение качества генерируемых изображений: Нужно продолжать исследования и эксперименты, чтобы улучшить качество генерируемых изображений.
2. Расширение функциональности: Добавить новые возможности модели, чтобы она стала более полезной и универсальной.
3. Интеграция в веб-сервис: Реализовать модель в веб-сервисе, чтобы пользователи могли легко получить к ней доступ с различных устройств пользоваться ею.

# **ЗАКЛЮЧЕНИЕ**

Нейронные сети являются важным инструментом в области распознавания объектов на изображениях. Они позволяют обрабатывать большие объемы данных и обнаруживать неизвестные объекты, а также способны адаптироваться к различным контекстам изображений. Несмотря на ограничения, такие как необходимость большого количества данных и вычислительную сложность, нейронные сети остаются одним из наиболее эффективных методов для решения задач распознавания объектов на изображениях. Благодаря новым алгоритмам и подходам к обучению, нейронные сети продолжают развиваться и находить новые применения во многих областях.

В данной работе мы исследовали и реализовали GAN (Generative Adversarial Network) − одну из передовых нейросетевых архитектур, способных генерировать реалистичные изображения. Наша цель заключалась в изучении возможностей DCGAN в области распознавания изображений и создании новых, высококачественных изображений.

Во время работы мы подробно рассмотрели архитектуру GAN и его основные компоненты, включая генератор и дискриминатор. Мы изучили процесс обучения GAN, включая выбор оптимизатора и функционала потерь, и применили его для обучения нашей модели с использованием доступного набора данных.

Проведя эксперименты и анализ результатов, мы выяснили, что наша реализация DCGAN демонстрирует впечатляющую способность генерировать высококачественные изображения, которые визуально соответствуют тем, на которых модель обучалась.

DCGAN является мощным инструментом в области распознавания изображений и генерации новых, реалистичных визуальных данных. Приложения GAN могут быть широкими, включая области медицинского изображения, создания искусственных сцен и многих других. Однако, как и у любой технологии, есть некоторые ограничения и возможности для улучшения, которые требуют дальнейших исследований и разработок.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Архитектура нейронной сети [Электронный ресурс] – // URL: https://habr.com/ru/articles/312450 (дата обращения: 13.03.2023)
2. Библиотеки для глубокого обучения [Электронный ресурс] – // URL: https://habr.com/ru/companies/ods/articles/325432 (дата обращения: 21.04.2023)
3. Генеративно-состязательная нейросеть (GAN) [Электронный ресурс] – // URL: https://neurohive.io/ru/osnovy-data-science/gan-rukovodstvo-dlja-novichkov (дата обращения: 13.03.2023)
4. Cвёрточные нейросети [Электронный ресурс] – // URL: https://medium.com/nuances-of-programming (дата обращения: 17.03.2023)