МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

09.04.01 «Информатика и вычислительная техника»

Обзор наиболее распространенных топологий нейронных сетей

КУРСОВАЯ РАБОТА

по дисциплине:
Современные платформы программирования

Исполнитель:

студент группы 8ВМ22 Ямкин Н.Н. 09.03.2023

Руководитель:

преподаватель Кривошеев Н.А.

Томск – 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Сегодня обработка больших объемов информации является критически важной в различных областях, включая бизнес, науку, технологии, медицину, образование и в многих других сферах деятельности. Существует огромное количество данных, создаваемых и собираемых каждый день, и умение эффективно обрабатывать эти данные может дать конкурентное преимущество в любой области.

Обработка больших объемов данных позволяет находить скрытые закономерности и тренды, выявлять причинно-следственные связи, строить прогнозы и принимать более осознанные решения. Например, в бизнесе анализ данных может помочь в принятии решений о маркетинге, оптимизации производства, улучшении качества продукции и т.д. В медицине анализ больших объемов данных может помочь выявить причины заболеваний и разработать новые методы лечения.

Таким образом, обработка больших объемов информации имеет большое значение в современном мире, и способность анализировать, обрабатывать и использовать данные становится все более важной.

Нейронные сети являются мощным инструментом для обработки широкого спектра данных. Они могут работать с текстовым, звуковым, графическим (изображения, видео) форматом, а также с временными рядами и многими другими типами данных.

В зависимости от информации, могут используются разные типы нейронных сетей.

Цель работы: рассмотреть наиболее популярные топологии нейронных сетей, принцип их обучения и сферы применения.

Задачи:

1. Рассмотреть понятие нейрона;
2. Краткий обзор полносвязной нейронной сети прямого распростарения;

3. Краткий обзор глубокой нейронной сети;
4. Краткий обзор свёрточной нейронной сети;
5. Краткий обзор рекуррентной нейронной сети;
6. Обзор алгоритма обратного распространения ошибки.

ПОНЯТИЕ НЕЙРОНА

Искусственные нейронные сети и их составляющие

Искусственные нейронные сети были построены по принципу биологических нейронных сетей, которые представляют собой сети нервных клеток, выполняющие определенные физиологические функции. Составным элементом нейронных сетей являются нейроны (представлены на рисунке 1).

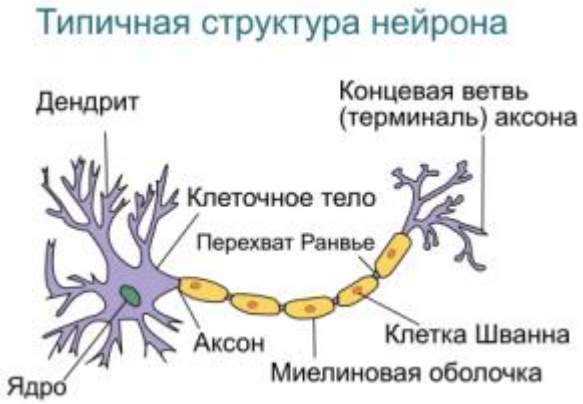

Рисунок 1 - Структура биологического нейрона

У нейрона есть несколько функций:

- Приёмная функция: синапсы получают информацию;
- Интегративная функция: на выходе нейрона сигнал, который несёт информацию обо всех суммированных в нейроне сигналах;
- Проводниковая функция: по аксону проходит информация к синапсу;
- Передающая функция: импульс, достигший окончания аксона, заставляет медиатор передавать возбуждение следующему нейрону.

Синапсами называют связи, по которым выходные сигналы одних нейронов поступают на входы других. Каждая связь характеризуется своим весом. Связи с положительным весом называются возбуждающими, а с отрицательным — тормозящими. Выход нейрона называется аксоном. В искусственной нейронной сети искусственный нейрон — это некоторая нелинейная функция, аргументом которой является линейная комбинация всех входных сигналов. Такая функция называется активационной. Затем

результат активационной функции посылается на выход нейрона. Объединяя такие нейроны с другими, получают искусственную нейронную сеть.

Активационная функция

Функция активации нейрона характеризует зависимость сигнала на выходе нейрона от суммы сигналов на его входах. Обычно функция является монотонно возрастающей и находится в области значений [-1,1] (гиперболический тангенс) и [0,1] (сигмоида). Для некоторых алгоритмов обучения необходимо, чтобы активационная функция была непрерывно дифференцируемой на всей числовой оси. Искусственный нейрон характеризуется своей активационной функцией (например, название «сигмоидальный нейрон»).

Основными активационными функциями являются: Пороговая активационная функция (функция Хевисайда). Нельзя использовать для алгоритма обратного распространения ошибки:

$$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \quad (1.1)$$

- Сигмоидальная активационная функция:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (1.2)$$

- Гиперболический тангенс:

$$tanh(x) = \frac{e^{Ax} - e^{-Ax}}{e^{Ax} + e^{-Ax}} \quad (1.3)$$

Перцептрон

Перцептрон — тип искусственного нейрона, разрабатываемый Фрэнком Розенблаттом в 1950-ых и 1960-ых годах. В современных работах чаще всего используют другую модель искусственного нейрона — сигмоидальный нейрон. Чтобы понять, как работает сигмоидальный нейрон, необходимо рассмотреть структуру и принцип работы перцептрона. Перцептрон принимает на вход значения x_1, x_2 … x_n и выдает бинарный результат.

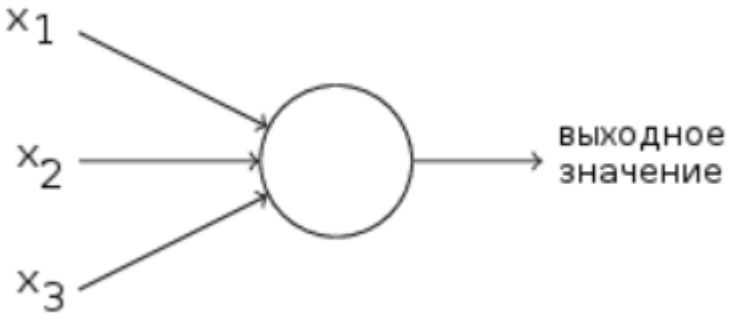

Рисунок 2 – Схема перцептрона

Розенблатт предложил использовать веса — числа, выражающие важность вклада каждого входа в конечный результат. Взвешенная сумма (или веса) сравнивается с пороговым значением (threshold), и по результатам определяется, будет ли выдан 0 или 1. Пороговое значение также является параметром нейрона.

$$f(x) = \begin{cases} 1, & f(\sum_{i=1}^{N} w_i \cdot x_i + b) \geq threshold \\ 0, & f(\sum_{i=1}^{N} w_i \cdot x_i + b) < threshold \end{cases} \quad (1.4)$$

Обучение перцептрона состоит в поиске оптимальной матрицы весов, для которой функция ошибки будет минимальной. Если бы небольшое изменение весов (или смещения) вызывало небольшое же изменение на выходе сети, то желаемое поведение нейронной сети можно было бы получить с помощью простых модификаций смещений и весов в процессе обучения. Однако обучение не так просто осуществить, если нейронная сеть состоит из перцептронов. Небольшое изменение весов или смещения одного из перцептронов сети может кардинально изменить выходное значение перцептрона, например, с 0 на 1. Поэтому самое незначительное изменение значений одного из элементов сети может создать значительные трудности в понимании изменения поведения сети. Поскольку задача обучения нейронной

сети является задачей поиска минимума функции ошибки в пространстве состояний обучения, то для ее решения могут применяться стандартные методы теории оптимизации.

Сигмоидальный нейрон

Сигмоидальные нейроны похожи на перцептроны, однако небольшие изменения в их весах и смещениях незначительно изменяют выход нейрона.

На вход сигмоидального нейрона подаются любые значения между 0 и 1. На выходе также выдаётся значение между 0 и 1, так как в качестве активационной функции используется сигмоида, являющаяся нелинейной.

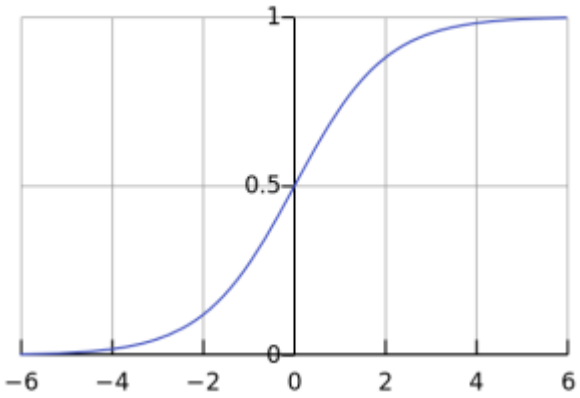

Рисунок 3 – Сигмоидальная функция

$$f(x) = \frac{1}{1+e^{-\beta x}} \tag{1.5}$$

Чем больше β (параметр наклона сигмоидальной функции активации), тем сильнее крутизна графика. При $\beta \to \infty$ сигмоида стремится к функции Хевисайда.

Важным свойством сигмоидальной функции является её дифференцируемость. Применение непрерывной функции активации позволяет использовать при обучении градиентные методы.

Нейроны можно разделить на группы в зависимости от их положения в сети:

- входные нейроны принимают исходный вектор данных;

- в промежуточных нейронах происходят основные вычислительные операции — обучение;
- выходные нейроны — выдают результат работы сети.

ТОПОЛОГИИ НЕЙРОННЫХ СЕТЕЙ

Существует несколько наиболее популярных топологий нейронных сетей, которые используются в различных областях машинного обучения и искусственного интеллекта. Ниже приведены некоторые из них:

- Полносвязная нейронная сеть прямого распространения – это наиболее простая и распространенная топология нейронной сети, которая состоит из одного или нескольких слоев нейронов, связанных между собой без обратной связи. Такие сети используются для задач классификации и регрессии.

- Глубокая (Deep) – это топология нейронной сети, которая имеет большое количество слоев, что позволяет ей обучаться сложным взаимодействиям входных данных и извлекать высокоуровневые признаки. Глубокие сети используются во многих областях машинного обучения, включая компьютерное зрение, естественный язык и обработку аудио.

- Сверточная (Convolutional) – это топология нейронной сети, которая имеет специальную структуру слоев, называемых сверточными слоями, которые способны выделять локальные признаки из входных данных, таких как изображения. Эта топология часто используется в компьютерном зрении и распознавании образов.

- Рекуррентная (Recurrent) – это топология нейронной сети, которая имеет обратную связь между нейронами, что позволяет ей использовать контекст и последовательность входных данных, таких как тексты или временные ряды. Такие сети часто используются в задачах обработки естественного языка и прогнозирования.

Полносвязная нейронная сеть прямого распространения

Полносвязная нейронная сеть – это сеть, в которой каждый нейрон связан со всеми остальными нейронами, находящимися в соседних слоях, и в которой все связи направлены строго от входных нейронов к выходным.

Нейронная сеть составлена из множества простых нейронов так, что выход одного из нейронов будет входом другого.

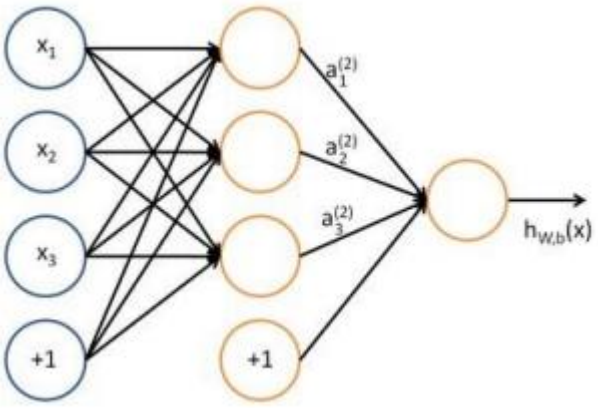

Рисунок 4 – Схема простейшей полносвязной нейронной сети прямого распространения [4]

Крайний левый слой называется входным, а крайний правый слой — выходным. Слой посередине является скрытым и называется так из-за того, что его значения не наблюдаются в тренировочных примерах. Таким образом в данной сети три элемента входа, три скрытых элемента и один выходной элемент. Пусть n_l – количество слоёв сети (в данном случае 3). W и b – векторы весов и смещений соответственно. Результат применения функции активации для i-го нейрона слоя l обозначается как a_i^l. Тогда выражение для $a_1^{(2)}$ примет следующий вид:

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^1) \quad (1.6)$$

Обозначив сумматорную функцию через z, получим выражение в векторной форме:

$$\begin{cases} z^{(2)} = W^{(1)}x + b^{(1)} \\ a^{(2)} = f(z^{(2)}) \end{cases}$$
$$\begin{cases} z^{(3)} = W^{(2)}x + b^{(2)} \\ a^{(2)} = f(z^{(3)}) \end{cases} \quad (1.7)$$

Общая формула будет выглядеть таким образом:

$$\begin{cases} z^{(l)} = W^{(l)} a^{(l)} + b^{(l)} \\ a^{(l)} = f(z^{(l)}) \end{cases} \quad (1.8)$$

Сетью прямого распространения называются нейронные сети, которые используют выход одного слоя в качестве входных данных для следующего слоя.

Обучение нейронных сетей

Общие понятия в обучении нейронных сетей:

Эпоха – один прямой проход по всем тренировочным примерам.

Размер серии (batch) – количество тренировочных примеров для одной итерации прямого прохода.

Количество итераций – количество проходов: каждый проход использует примеры (batch).

То есть имея 1000 примеров, batch = 500, нам потребуется две итерации, чтобы завершить одну эпоху.

С математической точки зрения, обучение нейронных сетей – многопараметрическая задача нелинейной оптимизации.

Алгоритм обратного распространения ошибки

Алгоритм обратного распространения ошибки определяет стратегию подбора весов многослойной сети с применением градиентных методов оптимизации. Поскольку целевая функция, обычно определяемая как сумма квадратичных разностей между фактическими и ожидаемыми выходными значениями, является непрерывной, градиентные методы оптимизации являются эффективными при обучении сети.

Градиент функции – это вектор, который указывает направление наибольшего возрастания функции в каждой точке. В математическом смысле, градиент функции в точке определяется как вектор ее частных производных по каждой координате.

Градиент используется в алгоритмах оптимизации, таких как обратное распространение ошибки в нейронных сетях, для нахождения оптимальных параметров модели. При обучении нейронной сети градиент функции потерь вычисляется относительно параметров модели, и затем параметры обновляются в направлении антиградиента, чтобы минимизировать функцию потерь [3].

Если значение градиента большое, то это означает, что функция имеет крутой склон и ее значение быстро изменяется в данной точке, а если значение градиента близко к нулю, то это означает, что функция практически не изменяется в данной точке. В обратном распространении ошибки градиент помогает нейронной сети определить, как изменить параметры модели, чтобы уменьшить ошибку прогнозирования на обучающем наборе данных.

Основная идея алгоритма заключается в том, чтобы находить градиент (производную) функции потерь по параметрам нейронной сети и использовать его для обновления весов во всех слоях сети.

Процесс обратного распространения ошибки начинается с подачи входных данных на входной слой нейронной сети, после чего они проходят через все слои, пока не достигнут выходной слой, который выдаст предсказание модели. Затем вычисляется значение функции потерь, которая описывает, насколько сильно модель ошибается в предсказании.

Далее, происходит обратное распространение ошибки, при котором градиент функции потерь по каждому весу в сети вычисляется с помощью цепного правила дифференцирования. Этот градиент затем используется для обновления весов во всех слоях нейронной сети с помощью алгоритма градиентного спуска, который позволяет минимизировать значение функции потерь и улучшить качество предсказаний модели.

Алгоритм обратного распространения ошибки является основой обучения многих типов нейронных сетей, включая сверточные и рекуррентные сети.

Пусть имеется конечный набор тренировочных данных (m примеров). Для обучения нейронной сети применяют пакетный градиентный спуск (batch gradient descent). Квадратичная ошибка целевой функции (squared-error loss function) для одного примера будет вычислена по формуле:

$$J(W,b,x,y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2 \tag{1.9}$$

Задача — минимизировать *J(W,b)*. Для обучения нейронной сети необходимо инициализировать каждый параметр $W_{ij}^{(l)}$ и $b_i^{(l)}$ малыми случайными величинами, близкими к нулю, а затем применить алгоритм оптимизации (упоминавшийся выше градиентный спуск).

Так как *J (W, b)* не является выпуклой функцией, то градиентный спуск восприимчив к локальным оптимумам. Каждая итерация градиентного спуска обновляет параметры таким образом:

$$W_{ij}^{+} = W_{ij} - \alpha \frac{\partial}{\partial W_{ij}} J(W,b)$$
$$b_i^{+} = b_i - \alpha \frac{\partial}{\partial b_i} J(W,b), \tag{1.10}$$

где α – скорость обучения (learning rate).

Шаги алгоритма обратного распространения ошибки:

1) Осуществляется прямой проход по сети, вычисляются активации слоёв L_2, L_3 и так далее до выходного слоя L_{nl};

2) Для каждого выходного элемента i в выходном слое n_l (the output layer) рассчитывается ошибка:

$$\delta_i^l = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2}\left\|h_{W,b}(x^i) - y^i\right\|^2 = -(y_i - a_i^{n_l}) \cdot f'(z_i^{n_l}) \tag{1.11}$$

3) Для слоя $l = n_l - 1, n_l - 2, n_l - 3 \ldots$:

$$\delta_i^l = (\sum_{j=1}^{s_{l+1}} W_{ij}^{(l)}\, \delta_i^{l+1}) \cdot f'(z_i^l) \tag{1.12}$$

4) Вычисляются частные производные:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b, x, y) = \alpha_j^l \delta_i^{l+1} \quad (1.13)$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b, x, y) = \delta_i^{l+1} \quad (1.14)$$

В матричной форме алгоритм будет записан таким образом (•) обозначено произведение Адамара — покомпонентное произведение двух матриц):

1) Осуществляется прямой проход по сети, вычисляются активации слоёв L_2, L_3 и так далее до выходного слоя L_{nl};

2) Матрица ошибок для выходного слоя n_l:

$$\delta^{n_l} = -(y_i - a^{n_i}) \cdot f'(z^{n_i}) \quad (1.15)$$

3) Для слоя l = n_l – 1, n_l – 2, n_l – 3 …:

$$\delta^l = ((W^l)^T \delta^{l+1}) \cdot f'(z^l) \quad (1.16)$$

4) Вычисление частных производных:

$$\nabla_{w^l} J(W, b, x, y) = \delta_i^{l+1} (a^l)^T \quad (1.17)$$

$$\nabla_{b^l} J(W, b, x, y) = \delta_i^{l+1} \quad (1.18)$$

Недостатки градиентного спуска

Основная трудность обучения нейронных сетей состоит в методах выхода из локальных минимумов. Недостатками градиентного спуска при обучении сети являются:

- Паралич сети

Значения весов сети в результате коррекции могут стать очень большими величинами. Поскольку ошибка, посылаемая обратно в процессе обучения, пропорциональна производной сжимающей функции, то процесс обучения может почти остановиться. Это можно предотвратить, уменьшая шаг η.

- Размер шага

Если значение шага не изменяется, и оно довольно мало, то метод сходится слишком медленно. Если же шаг слишком велик, то может возникнуть паралич сети. Необходимо изменять значение шага: увеличивать до тех пор, пока не прекратится улучшение оценки в направление антиградиента и уменьшать, если оценка не улучшается.

Глубокие нейронные сети

Глубокими нейронными сетями называются такие сети, в которых есть несколько скрытых слоев. Поскольку каждый скрытый слой вычисляет нелинейное преобразование предыдущего слоя, глубокая сеть может иметь значительно большую репрезентативную мощность (то есть может представлять значительно более сложные функции), чем малослойная.

Глубокие нейронные сети (Deep Neural Networks, DNN) являются расширением классических нейронных сетей, состоящих из большого количества слоев. Такие сети могут состоять из сотен или даже тысяч слоев, что позволяет им обрабатывать более сложные и абстрактные задачи, чем классические нейронные сети.

Глубокие нейронные сети (ГНС) имеют несколько преимуществ по сравнению с классическими нейронными сетями (НС):

1. Лучшая производительность: ГНС могут обрабатывать сложные и большие объемы данных с высокой точностью и скоростью. Они могут использоваться для обработки изображений, звука, текста и других видов данных, что делает их полезными в различных задачах машинного обучения, таких как распознавание речи, обработка естественного языка и компьютерное зрение.

2. Лучшая способность к автоматическому извлечению признаков: ГНС способны автоматически извлекать признаки из данных, что может улучшить точность и эффективность алгоритмов машинного обучения.

3. Большая гибкость: ГНС могут быть настроены для выполнения различных задач машинного обучения, их можно настраивать на определенный тип данных или задачи, что позволяет им достигать более высокой точности и эффективности в решении конкретных задач.

4. Лучшая устойчивость к шуму: ГНС способны обрабатывать данные, которые содержат шум или ошибки, и продолжать работать даже в условиях неполной или неточной информации.

5. Способность к обучению без учителя: ГНС могут обучаться без учителя, что означает, что они могут извлекать скрытые закономерности из данных без предварительной разметки.

6. Улучшение воспроизводимости и обобщаемости: ГНС могут быть обучены на большом количестве данных и иметь возможность обобщаться на новые данные, что делает их полезными в задачах, связанных с распознаванием образов, классификацией и прогнозированием.

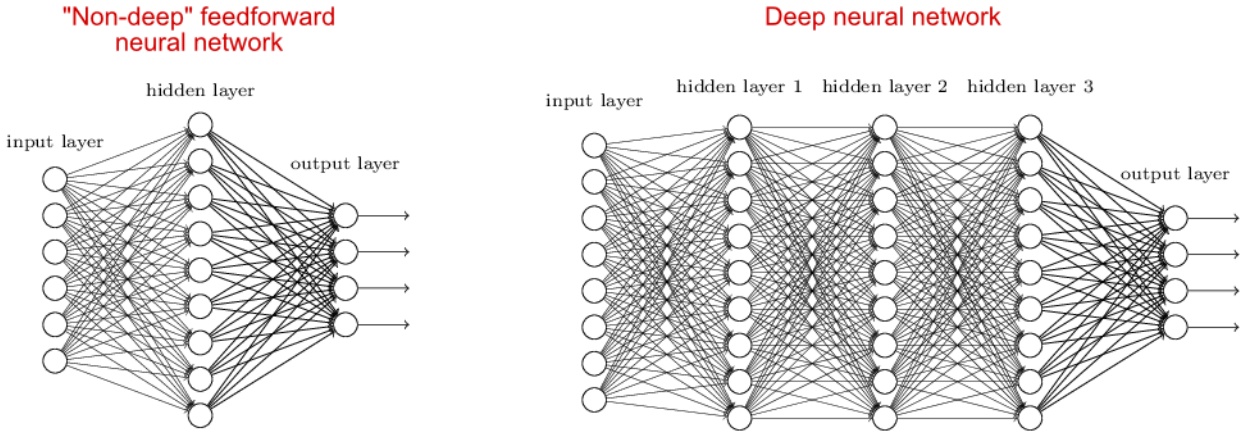

Рисунок 5 – Сравнение малослойной и глубокой нейронной сети [1]

Применение глубоких нейронных сетей

Глубокие нейронные сети применяются во многих современных сервисах. Некоторые из них:

1. Google Translate - сервис машинного перевода, который использует глубокие нейронные сети для перевода текстов на различные языки.

2. Siri и Alexa - голосовые помощники, использующие нейронные сети для распознавания голосовых команд и предоставления ответов на запросы.

3. Netflix - онлайн-кинотеатр, который использует нейронные сети для рекомендации фильмов и сериалов на основе предпочтений пользователя.

4. Facebook - социальная сеть, использующая глубокие нейронные сети для распознавания лиц на фотографиях и предложения тегов для них.

5. Tesla - автомобильный производитель, который использует глубокие нейронные сети для управления автомобилями с помощью системы автопилота.

6. Amazon - интернет-магазин, который использует глубокие нейронные сети для рекомендации товаров на основе истории покупок пользователя.

7. Google - поисковая система, использующая нейронные сети для обработки запросов и выдачи наиболее релевантных результатов.

8. Spotify - музыкальный сервис, использующий глубокие нейронные сети для рекомендации музыки на основе предпочтений пользователя и анализа музыкальных характеристик треков.

Проблемы обучения глубоких сетей

Хотя глубокие нейронные сети являются мощным инструментом для решения сложных задач, у них есть ряд проблем при обучении:

1. Проблема затухающего градиента: при обратном распространении ошибки градиент может становиться слишком маленьким на более ранних слоях ГНС, что затрудняет обучение этих слоев.

2. Проблема взрывающегося градиента: наоборот, градиент может стать слишком большим, что также может затруднить обучение.

3. Недостаток данных: глубокие нейронные сети требуют большого количества данных для обучения, особенно при использовании глубокого обучения с подкреплением.

4. Сложность выбора гиперпараметров: глубокие нейронные сети имеют множество гиперпараметров, таких как число слоев, размерность скрытых слоев, шаг обучения и т.д. Выбор этих параметров может существенно влиять на качество обучения, но определение оптимальных значений может быть трудным.

5. Вычислительная сложность: обучение глубоких нейронных сетей может быть очень вычислительно затратным и требовать мощных вычислительных ресурсов, таких как графические процессоры (GPU) и кластеры серверов.

6. Сигмоидальные активационные функции. Их применение может вызвать проблемы в обучении глубинных сетей: значения активаций в конечном слое будут близки к нулю на ранних этапах обучения, замедляя этот процесс. Были предложены альтернативные активационные функции, которые не так страдают от ограничения.

7. Выбор подходящих начальных весов в стохастическом градиентном спуске существенно влияет на способность обучать глубокие сети.

Свёрточные нейронные сети

Свёрточная нейронная сеть — архитектура нейронных сетей, изначально созданная для эффективного распознавания изображений: чередуются свёрточные слои (convolutions) с нелинейными активационными функциями (ReLU или гиперболический тангенс tanh) и слои объединения (pooling layers).

В отличие от сети прямого распространения, где каждый входной нейрон соединяется с выходным нейроном в следующем слое, в свёрточных сетях для получения выходных значений используются свёртки. Свёрточная нейронная сеть (СНС) обрабатывает изображения путём применения операции свёртки к входным данным, которая заключается в перемещении ядра (или фильтра) по изображению и вычислении скалярного произведения между

ядром и областью изображения, попадающей под него. Это действие повторяется для всех возможных положений ядра на изображении.

В результате свёртки получается новое изображение, в котором каждый пиксель содержит информацию о том, насколько ярким был фрагмент изображения, попадающий под ядро. Затем обработанное изображение проходит через слой активации, который применяет к каждому пикселю некоторую нелинейную функцию, например, функцию ReLU.

Далее обработанное изображение может проходить через другие свёрточные слои, пулинг слои (которые уменьшают размерность изображения, например, путём выбора максимального значения в каждом окне), а также полносвязные слои (которые соединяют каждый пиксель изображения с каждым нейроном в слое).

В конечном итоге свёрточная нейронная сеть может выдавать на выходе предсказания для задач обработки изображений, таких как классификация, детекция объектов, сегментация изображений и др.

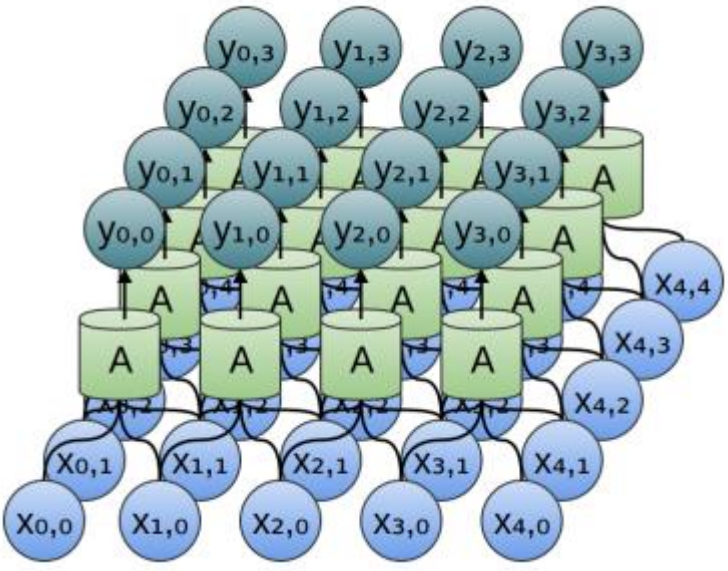

Рисунок 6 – Двумерный свёрточный слой [5]

Сверточные нейронные сети имеют несколько преимуществ по сравнению с другими видами нейронных сетей:

- Эффективная обработка изображений: сверточные нейронные сети специально разработаны для обработки изображений и достигают высокой точности в задачах компьютерного зрения, таких как распознавание объектов на изображениях, сегментация изображений, классификация изображений и т.д.
- Уменьшение количества параметров: сверточные нейронные сети используют параметры, которые разделяются между всеми сверточными слоями, что позволяет значительно уменьшить количество параметров и ускорить процесс обучения.
- Инвариантность к сдвигу: сверточные нейронные сети инвариантны к сдвигу входных данных, что означает, что они могут распознавать объекты на изображении, независимо от того, где они находятся.
- Масштабируемость: сверточные нейронные сети легко масштабируются и могут быть использованы для обработки изображений разного размера.

Применение свёрточных нейронных сетей

Сверточные нейронные сети (CNN) широко используются в различных сервисах, требующих обработки изображений и видео. Ниже приведены некоторые примеры сервисов, которые используют сверточные нейронные сети:

1. Распознавание объектов и классификация изображений: сервисы, такие как Google Images, Bing Images и Flickr, используют сверточные нейронные сети для распознавания объектов и классификации изображений.
2. Распознавание лиц: сервисы, такие как Facebook и Snapchat, используют сверточные нейронные сети для распознавания лиц на фотографиях и видео.
3. Автоматическая обработка изображений: сервисы, такие как Instagram и Photoshop, используют сверточные нейронные сети для автоматической обработки изображений.

4. Обработка видео: сервисы, такие как YouTube и Netflix, используют сверточные нейронные сети для обработки видео, например, для автоматической классификации видео по жанру или для рекомендации контента.

5. Автоматическая диагностика медицинских изображений: в медицине сверточные нейронные сети используются для автоматической диагностики медицинских изображений, таких как рентгеновские снимки, томограммы и МРТ.

6. Автоматическое управление производственными процессами: сверточные нейронные сети используются в автоматическом управлении производственными процессами, например, для контроля качества продукции на конвейере или для определения дефектов на поверхности деталей.

Проблемы обучения свёрточных нейронных сетей

Обучение сверточных нейронных сетей может столкнуться с рядом проблем, включая:

1. Оверфиттинг: это происходит, когда модель слишком хорошо запоминает тренировочные данные, в результате чего она не обобщается на новые данные. Одним из способов решения этой проблемы является использование регуляризации, такой как Dropout.

2. Недообучение: это происходит, когда модель недостаточно обучена, что приводит к плохой производительности на тренировочных и тестовых данных. Решение этой проблемы может включать увеличение количества данных для обучения или изменение гиперпараметров модели.

3. Сходство изображений: сверточные нейронные сети могут иметь проблемы с различением сходных изображений, таких как различные варианты одной и той же фотографии. Для решения этой проблемы можно использовать аугментацию данных, что помогает модели различать изображения с различными вариантами освещения, угла и т.д.

4. Потеря пространственной информации: сверточные нейронные сети могут потерять некоторую пространственную информацию при сжатии изображений. Эту проблему можно решить, используя пулинг-слои, такие как слой Max Pooling, который помогает уменьшить размерность, сохраняя при этом важные особенности изображения.

5. Сложность вычислений: сверточные нейронные сети могут быть очень сложными и требовательными к вычислениям, особенно при работе с большими изображениями. Для решения этой проблемы можно использовать техники оптимизации, такие как стохастический градиентный спуск, и использовать графические процессоры для ускорения вычислений.

Типовая структура

Структура сети является однонаправленной, а для обучения обычно используется метод обратного распространения ошибки. Сеть состоит из большого количества слоёв.

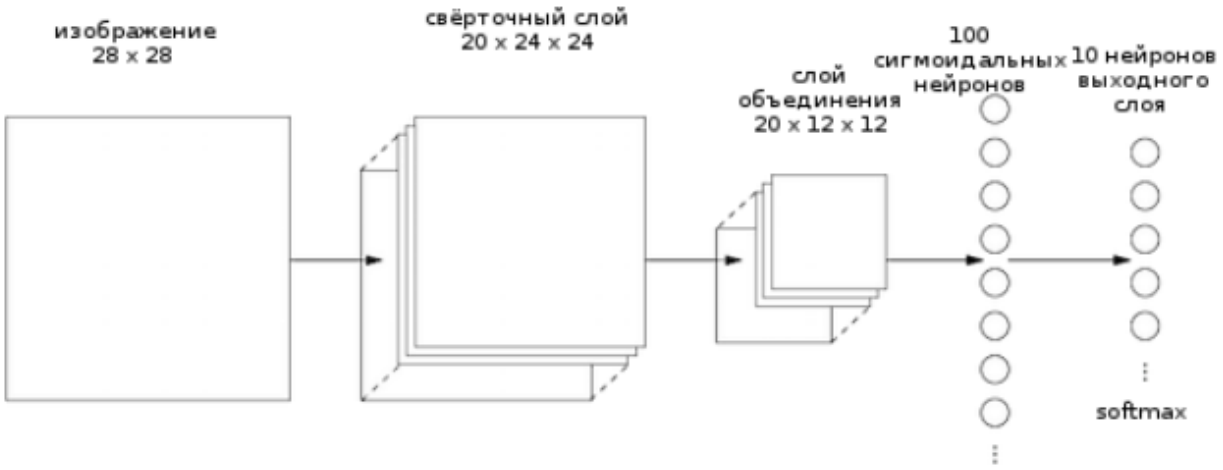

Рисунок 7 – Пример архитектуры свёрточной нейронной сети для распознавания объекта на изображении размерности 28x28 px

Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN) – это класс нейронных сетей, способных работать с последовательными данными, где каждый элемент последовательности имеет связь с предыдущим.

В отличие от прямого распространения сигнала, где каждый входной сигнал проходит через слой и напрямую влияет на выходной сигнал, в рекуррентных сетях каждый входной сигнал передается через слой и сохраняется внутреннее состояние, которое влияет на выходной сигнал и на следующий входной сигнал в последовательности. Это позволяет рекуррентным сетям обрабатывать последовательности переменной длины и использовать контекст прошлых входных сигналов для предсказания будущих.

В сетях прямого распространения используется единственный вход, который полностью определяет активации всех нейронов в оставшихся слоях. Такую сеть невозможно обучить предсказывать события, например, в сюжете фильма — неясно, как бы могла быть использована информация о предыдущих событиях в фильме. Рекуррентные нейронные сети призваны решить эту проблему.

Принцип работы RNN заключается в передаче информации от предыдущего шага обработки к следующему шагу обработки. Для этого в RNN присутствует рекуррентный слой, который обеспечивает обратную связь между выходом текущего шага и входом следующего шага. Таким образом, информация о предыдущих шагах сохраняется в скрытом состоянии RNN и используется для обработки текущего входа.

На каждом шаге обработки RNN принимает на вход новый элемент последовательности и скрытое состояние, которое было получено на предыдущем шаге. Затем RNN вычисляет новое скрытое состояние и выход текущего шага, которые используются на следующем шаге.

Таким образом, RNN обрабатывает последовательные данные, сохраняя информацию о предыдущих шагах и используя ее для обработки текущего шага. Это позволяет RNN быть эффективными в задачах, связанных с анализом текста, речи, временных рядов и т.д.

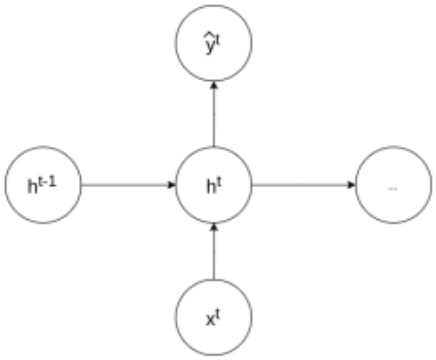

Рисунок 8 – Нейрон рекуррентной сети

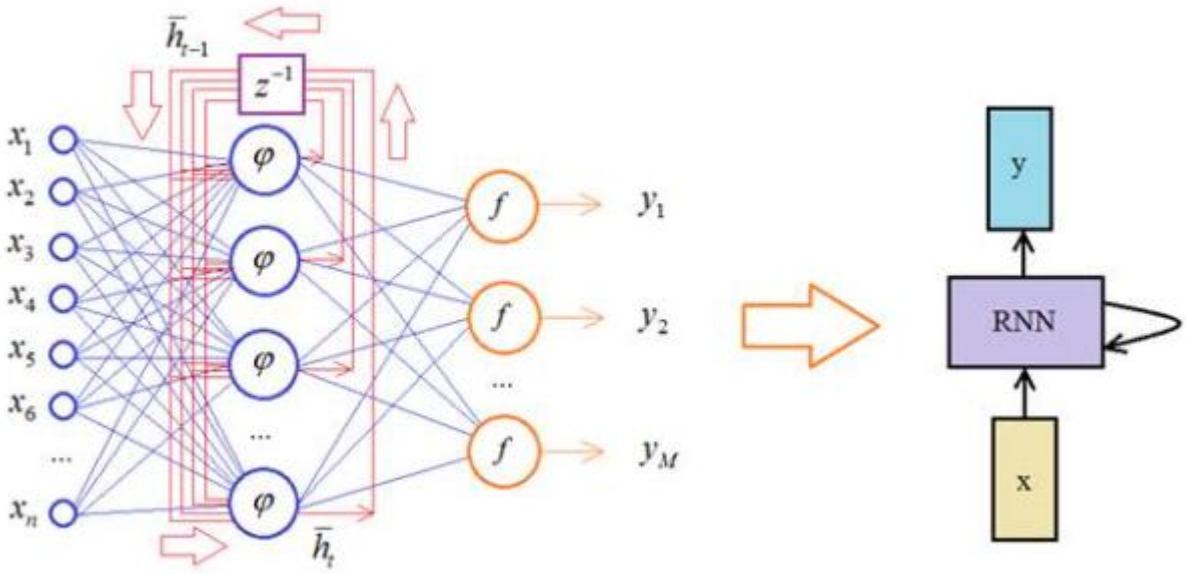

Рисунок 9 – Рекуррентная нейронная сеть [2]

Примерами задач, которые могут быть решены рекуррентными нейронными сетями, являются:

- Распознавание речи и голосовой поиск
- Машинный перевод
- Обработка естественного языка (NLP)

- Генерация текста и изображений
- Анализ временных рядов

Проблемы обучения рекуррентных нейронных сетей

Одной из основных проблем обучения рекуррентных нейронных сетей является проблема затухающих и взрывающихся градиентов. Когда градиент передается обратно через много временных шагов, он может становиться очень маленьким или очень большим, что затрудняет обучение сети. Для решения этой проблемы используются различные техники, такие как LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit), которые имеют специальные механизмы для управления потоком градиентов.

Другой проблемой является проблема переобучения, которая проявляется в том случае, когда модель слишком хорошо запоминает тренировочные данные и не может обобщать полученные знания на новые данные. Для решения этой проблемы можно использовать методы регуляризации, такие как dropout и L2-регуляризация, которые помогают снизить переобучение. Также можно использовать методы аугментации данных, которые позволяют создавать новые тренировочные данные на основе существующих.

Еще одной проблемой является проблема выбора оптимальной архитектуры модели и настройки ее гиперпараметров. Для решения этой проблемы применяются методы оптимизации гиперпараметров, такие как случайный поиск, сеточный поиск и оптимизация на основе градиентов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы был проведен обзор самых популярных архитектур нейронных сетей. Был описан принцип работы глубоких, свёрточных и рекуррентных нейронных сетей, рассмотрены проблемы, возникающие при обучении каждого из этих типов сетей, а также указаны сервисы, в которых они применяются. Дополнительно был рассмотрен алгоритм обратного распространения ошибки.

СПИСОК ИСТОЧНИКОВ

1. Глубокое обучение // ИТМО. URL: https://neerc.ifmo.ru/wiki/index.php?title=Глубокое_обучение (дата обращения: 07.03.2023).

2. Введение в рекуррентные нейронные сети // youtube.com. URL: https://www.youtube.com/watch?v=P6Zjr_dBCKE&list=PLA0M1Bcd0w8yv0XGiF1wjerjSZVSrYbjh (дата обращения: 08.03.2023).

3. Градиент // Википедия. Свободная энциклопедия URL: https://ru.wikipedia.org/wiki/Градиент (дата обращения: 08.03.2023).

4. Что такое нейронные сети, что они могут, и как написать нейронную сеть на Python? // University of Artificial Intelligence. URL: https://neural-university.ru/neural-networks basics#:~:text=%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D0%B0%D1%8F%20%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F%20%D1%81%D0%B5%D1%82%D1%8C%20%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B3%D0%BE%20%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F,%D0%BE%D1%82%20%D0%B2%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D1%85%20%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BE%D0%B2%20%D0%BA%20%D0%B2%D1%8B%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%BC. (дата обращения: 08.03.2023).

5. Принцип работы свёрточных нейронных сетей // DATA4. URL: http://data4.ru/convnn. (дата обращения: 08.03.2023).