Изображение выглядит как эмблема, символ, герб, нашивка

Автоматически созданное описание

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

**Институт** Информационных Технологий

**Кафедра** Вычислительной Техники

**ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ**

**по дисциплине**

**«Проектирование и обучение нейронных сетей»**

Студент группы:ИКБО-04-22 \_\_Кликушин В.И.\_ *(Ф. И.О. студента)*

Преподаватель \_\_Семенов Р.Э.\_\_

*(Ф.И.О. преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2025

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc192611309)

[1 ТРАНСФОРМЕР 5](#_Toc192611310)

[1.1 Теоретический раздел 5](#_Toc192611311)

[1.2 Постановка задачи 5](#_Toc192611312)

[1.3 Программная реализация 5](#_Toc192611313)

[1.4 Выводы по разделу 5](#_Toc192611314)

[2 ГЕНЕРАТИВНАЯ СЕТЬ 6](#_Toc192611315)

[2.1 Теоретический раздел 6](#_Toc192611316)

[2.2 Постановка задачи 6](#_Toc192611317)

[2.3 Программная реализация 6](#_Toc192611318)

[2.4 Выводы по разделу 6](#_Toc192611319)

[3 ГРАФОВАЯ СЕТЬ 7](#_Toc192611320)

[3.1 Теоретический раздел 7](#_Toc192611321)

[3.2 Постановка задачи 7](#_Toc192611322)

[3.3 Программная реализация 7](#_Toc192611323)

[3.4 Выводы по разделу 7](#_Toc192611324)

[4 СВОЯ ТЕМА 8](#_Toc192611325)

[4.1 Теоретический раздел 8](#_Toc192611326)

[4.2 Постановка задачи 8](#_Toc192611327)

[4.3 Программная реализация 8](#_Toc192611328)

[4.4 Выводы по разделу 8](#_Toc192611329)

[ЗАКЛЮЧЕНИЕ 9](#_Toc192611330)

[ПРИЛОЖЕНИЯ 10](#_Toc192611331)

# ВВЕДЕНИЕ

# 1 ТРАНСФОРМЕР

## 1.1 Теоретический раздел

Модели трансформеров используются для решения всевозможных NLP задач, в числе которых:

* zero-shot classification – классификация неразмеченных данных;
* генерация текста;
* mask filling - задача предсказать правильное слово (точнее, лексему) в середине последовательности;
* named entity recognition — это задача в области обработки естественного языка (NLP), направленная на выделение и классификацию именованных сущностей в тексте, таких как имена людей, названия организаций, даты, местоположения, суммы денег и другие типы специфических объектов;
* ответы на вопросы (извлечение ответа из контекста);
* обобщение - задача сократить текст до более короткого, сохранив при этом все (или большинство) важных аспектов, упомянутых в тексте;
* машинный перевод.

Архитектура трансформера была представлена в июне 2017 года. В центре внимания первоначального исследования были задачи перевода. Затем было представлено несколько влиятельных моделей, в том числе:

* июнь 2018 года: GPT, первая предварительно обученная модель трансформера, использовалась для тонкой настройки на различных задачах NLP и получила лучшие результаты;
* октябрь 2018 года: BERT, еще одна большая предварительно обученная модель, предназначенная для создания лучших резюме текстов;
* февраль 2019 года: GPT-2, улучшенная (и более крупная) версия GPT, которая не была сразу опубликована по этическим соображениям;
* октябрь 2019 года: DistilBERT, дистиллированная версия BERT, которая на 60% быстрее, на 40% легче и сохраняет 97% производительности BERT;
* октябрь 2019 года: BART и T5, две большие предварительно обученные модели, использующие ту же архитектуру, что и оригинальная модель трансформера;
* май 2020 года: GPT-3, еще более крупная версия GPT-2, способная хорошо справляться с различными задачами без необходимости в дообучении.

Этот список далеко не полный и призван лишь выделить несколько видов моделей трансформеров. В целом их можно разделить на три категории:

* GPT-подобные (также называемые авторегрессионными моделями трансформаторов);
* BERT-подобные (также называемые автокодирующими моделями трансформеров);
* BART/T5-подобные (также называемые моделями трансформации последовательности в последовательность).

Все упомянутые выше модели трансформеров (GPT, BERT, BART, T5) были обучены как языковые модели. Это означает, что они обучались на больших объемах необработанного текста в режиме самоконтроля. Обучение с самоконтролем — это парадигма машинного обучения, при которой модель обучается решению задачи, используя сами данные для генерации управляющих сигналов, а не полагаясь на метки, предоставленные извне. Это означает, что люди не нужны для разметки данных.

За исключением нескольких моделей, общая стратегия достижения лучшей производительности заключается в увеличении размеров моделей и объема данных, на которых они предварительно обучаются. На Рисунке 1.1.1 представлен график, отображающий рост числа параметров моделей со временем.

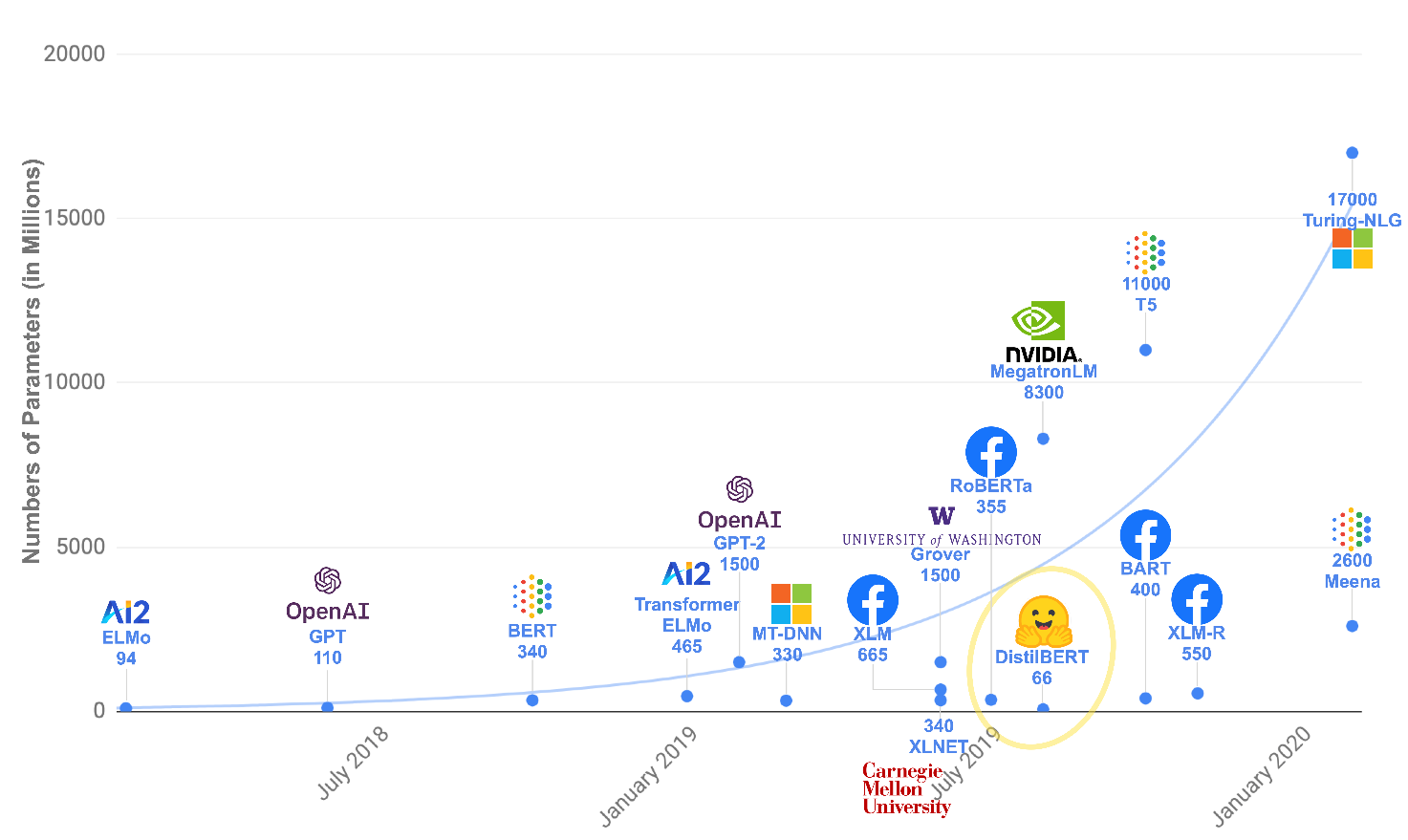


Рисунок 1.1.1 - Рост числа параметров моделей со временем

Идея архитектуры трансформера отображена на Рисунке 1.1.2.

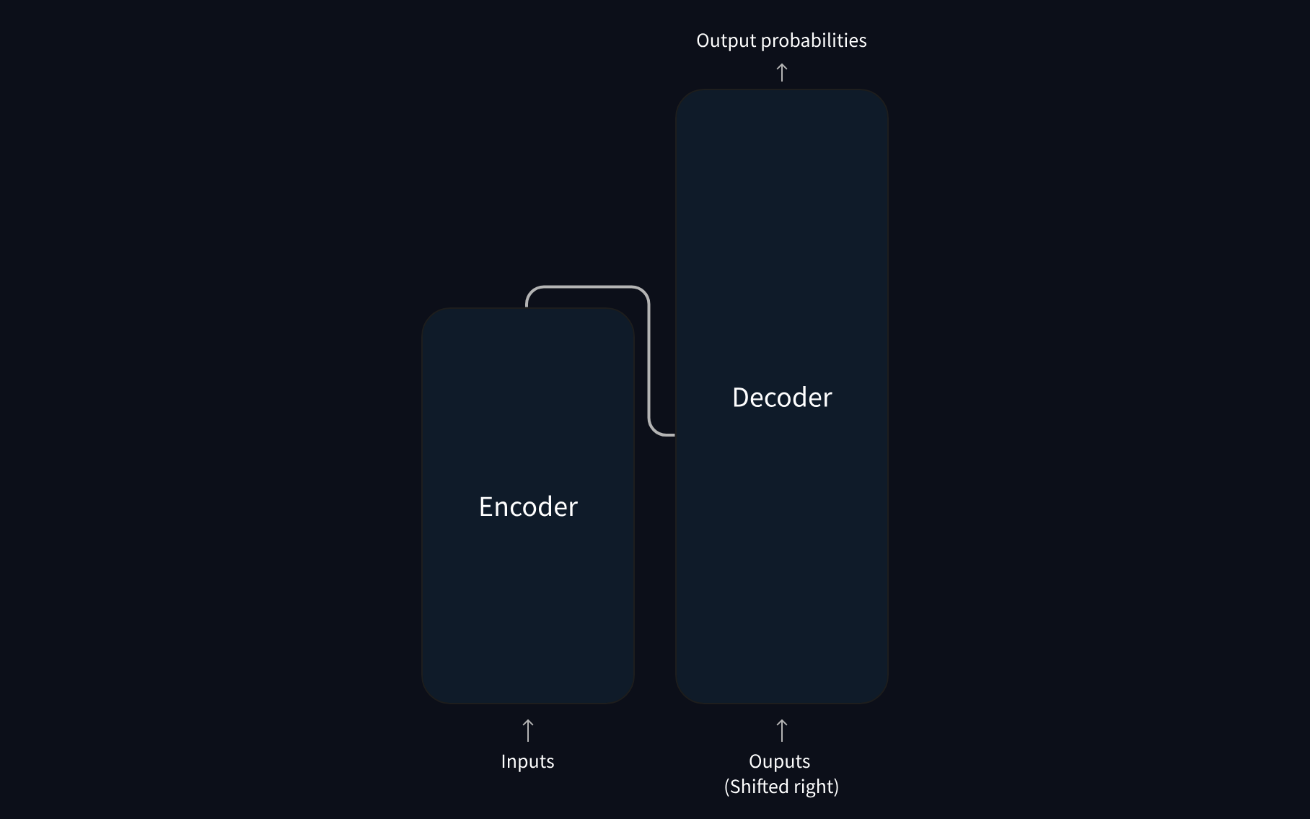


Рисунок 1.1.2 - Идея архитектуры трансформера

Архитектура трансформера состоит из двух блоков:

* кодировщик: кодировщик получает входной сигнал и строит его представление (его признаки). Это означает, что модель оптимизирована для получения понимания от входных данных;
* декодер: декодер использует представление (признаки) кодера вместе с другими входными данными для создания целевой последовательности. Это означает, что модель оптимизирована для генерации выходных данных.

Каждая из этих частей может использоваться независимо, в зависимости от задачи:

* модели, включающие только кодировщик: хорошо подходят для задач, требующих понимания входных данных, таких как классификация предложений и распознавание именованных сущностей;
* модели, использующие только декодер: хорошо подходят для генеративных задач, таких как генерация текста;
* модели кодировщика-декодировщика или модели «последовательность-последовательность»: хорошо подходят для генеративных задач, требующих входных данных, таких как перевод или обобщение.

Ключевой особенностью моделей трансформеров является то, что они строятся с помощью специальных слоев, называемых слоями внимания. Фактически, название статьи, представляющей архитектуру трансформера, было «Attention Is All You Need»!

Архитектура трансформера изначально была разработана для перевода. В процессе обучения кодер получает входные данные (предложения) на определенном языке, а декодер - те же предложения на нужном целевом языке. В кодировщике слои внимания могут использовать все слова в предложении (поскольку, перевод данного слова может зависеть от того, что находится как после, так и до него в предложении). Декодер, однако, работает последовательно и может обращать внимание только на те слова в предложении, которые он уже перевел (то есть только на те, которые предшествуют слову, генерируемому в данный момент). Например, когда мы предсказали первые три слова переведенной цели, мы передаем их декодеру, который затем использует все входные данные кодера, чтобы попытаться предсказать четвертое слово. Оригинальная архитектура трансформера представлена на Рисунке 1.1.3.

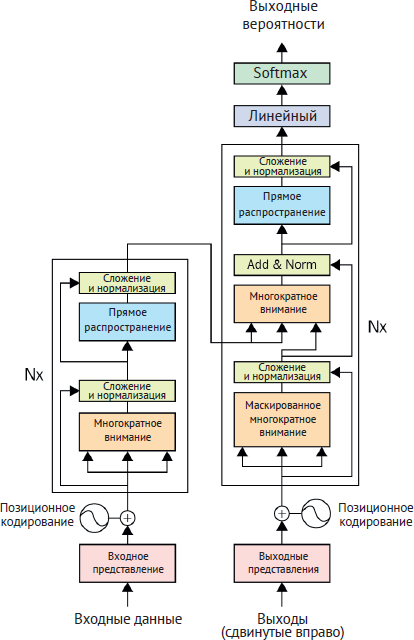


Рисунок 1.1.3 – Оригинальная архитектура трансформера

Модели кодировщиков используют только кодировщик модели трансформера. На каждом этапе уровни внимания могут обращаться ко всем словам в исходном предложении. Эти модели часто характеризуются как обладающие «двунаправленным» вниманием, и их часто называют моделями автокодирования. Предварительное обучение таких моделей обычно сводится к тому, чтобы каким-то образом испортить данное предложение (например, замаскировать в нем случайные слова) и поставить перед моделью задачу найти или восстановить исходное предложение. Модели-кодировщики лучше всего подходят для задач, требующих понимания полного предложения, таких как классификация предложений, распознавание именованных сущностей (и в более общем случае классификация слов) и экстрактивные ответы на вопросы.

Декодерные модели используют только декодер модели трансформера. На каждом этапе для данного слова слои внимания могут обращаться только к словам, расположенным перед ним в предложении. Такие модели часто называют авторегрессионными. Предварительное обучение моделей-декодеров обычно сводится к предсказанию следующего слова в предложении. Эти модели лучше всего подходят для задач, связанных с генерацией текста.

Модели кодировщика-декодировщика (также называемые моделями последовательности) используют обе части архитектуры трансформера. На каждом этапе слои внимания кодера получают доступ ко всем словам в исходном предложении, в то время как слои внимания декодера получают доступ только к словам, расположенным перед данным словом во входном сообщении. Предварительное обучение этих моделей может быть выполнено с использованием задач моделей кодировщика или декодировщика, но обычно для этого требуется нечто более сложное. Например, модель T5 предварительно обучается путем замены случайных участков текста (которые могут содержать несколько слов) одним специальным словом-маской, а цель состоит в том, чтобы предсказать текст, который заменяет это слово-маска. Модели «последовательность-последовательность» лучше всего подходят для задач, связанных с генерацией новых предложений в зависимости от заданного исходного текста, таких как обобщение, перевод или генеративный ответ на вопрос.

## 1.2 Постановка задачи

Цель: реализовать обучение нейронной сети с одним нейронам по правилу Хебба для задачи классификации.

Задачи: изучить нейрон Хебба и правило его обучения.

## 1.3 Подход к решению

## 1.4 Выводы по разделу

# 2 ГЕНЕРАТИВНАЯ СЕТЬ

## 2.1 Теоретический раздел

## 2.2 Постановка задачи

Цель: научить перцептрон распознавать цифры, представленные в виде бинарных векторов, с заданной точностью и проверить его способность к обобщению на тестовой выборке.

Задачи: инициализировать перцептрон с произвольными начальными весами и параметрами, обучить модель, используя алгоритм обучения перцептрона на основе дельта-правила, проверить точность модели на тестовой выборке.

## 2.3 Подход к решению

## 2.4 Выводы по разделу

# 3 ГРАФОВАЯ СЕТЬ

## 3.1 Теоретический раздел

## 3.2 Постановка задачи

## 3.3 Подход к решению

## 3.4 Выводы по разделу

Алгоритм обратного распространения ошибки является основой для обучения многослойных нейронных сетей. Он позволяет эффективно корректировать веса сети на основе градиента функции ошибки.

Важные аспекты алгоритма:

* Использование функции активации для нелинейного преобразования данных.
* Итеративное обновление весов с помощью метода градиентного спуска.
* Возможность обучения сети с несколькими скрытыми слоями для решения сложных задач.

Таким образом, сеть обратного распространения ошибки успешно применяется для задач классификации и регрессии, позволяя моделировать сложные зависимости между входами и выходами.

# 4 СВОЯ ТЕМА

## 4.1 Теоретический раздел

## 4.2 Постановка задачи

Цель: разработать и обучить сверточную нейронную сеть для решения задачи классификации изображений рукописных цифр из набора данных MNIST.

Задачи: загрузить и подготовить данные MNIST, спроектировать архитектуру сверточной сети, обучить модель на обучающей выборке, оценить точность модели на тестовой выборке, визуализировать процесс обучения и результаты.

## 4.3 Программная реализация

## 4.4 Выводы по разделу

# ЗАКЛЮЧЕНИЕ

# ПРИЛОЖЕНИЯ

Приложение А.1 — Код реализации нейронной сети Хебба для моделирования логических функций.

Приложение А.2 — Код реализации нейронной сети Хебба для задачи классификации.

Приложение Б — Код реализации обучения перцептрона по дельта правилу.

Приложение В — Код реализации нейронной сети обратного распространения ошибки.

### Приложение А.1

Код реализации нейронной сети Хебба для моделирования логических функций

Листинг А.1 – Реализация нейронной сети Хебба для моделирования логических функций