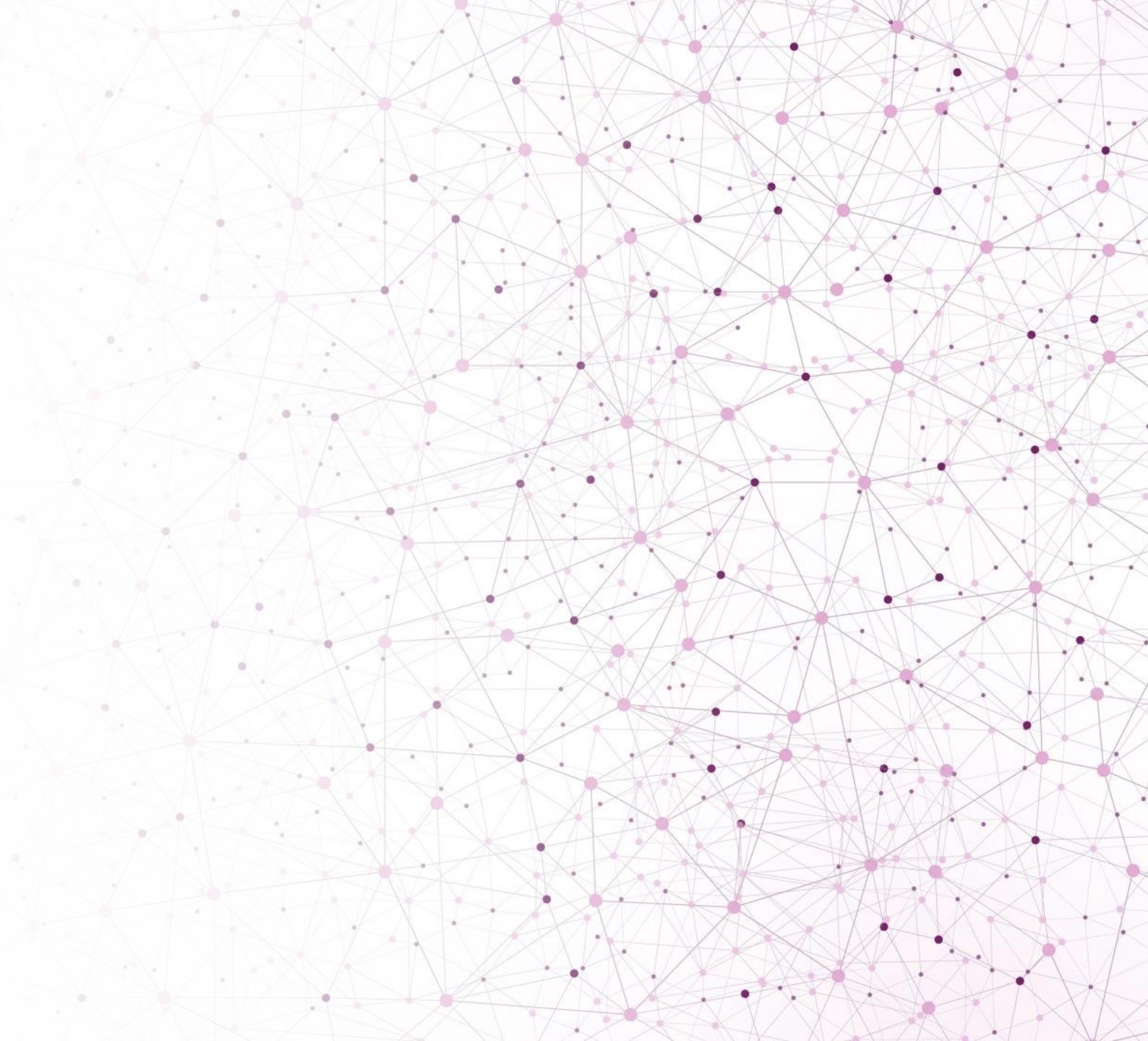




4. Глубокое обучение

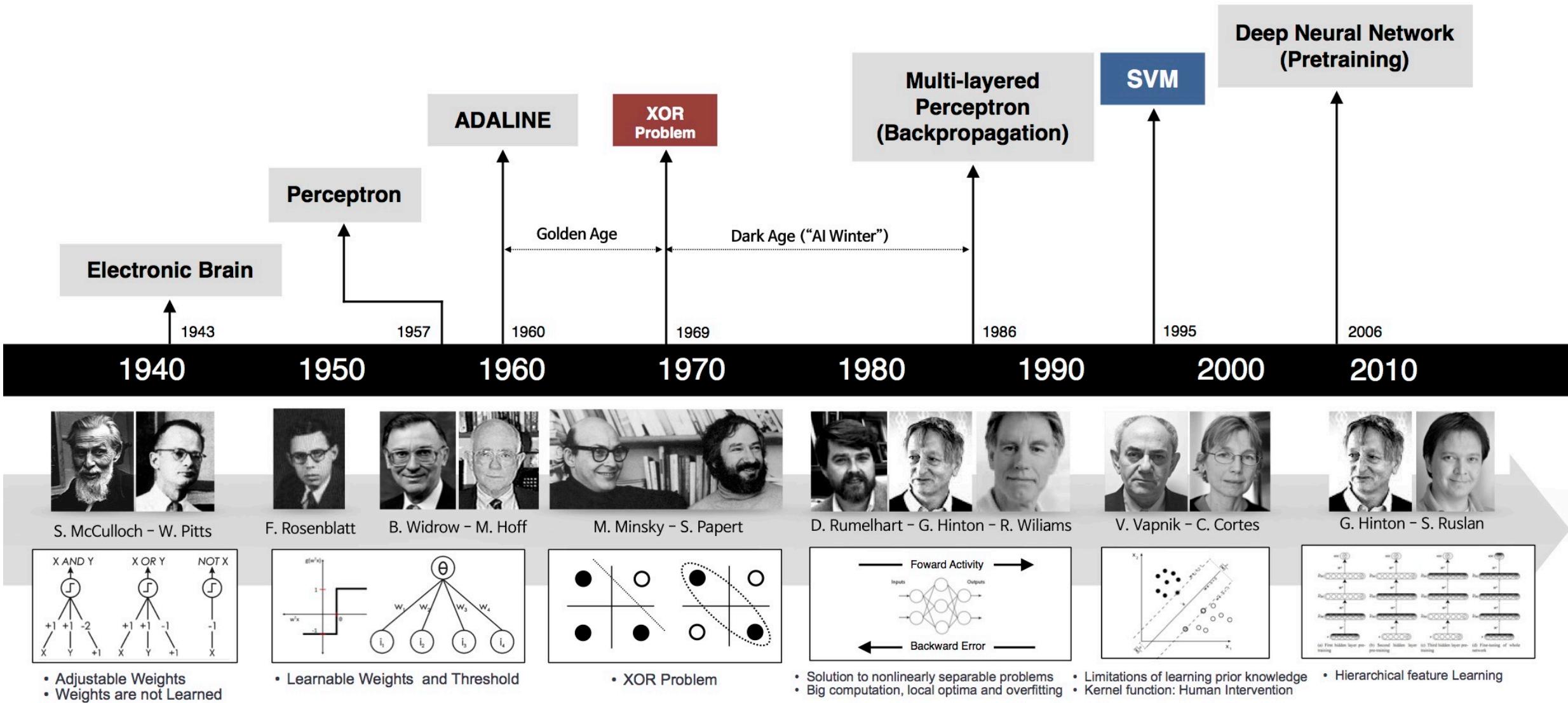
Зеленков Ю. А. (2024)



Содержание

- Краткая история глубокого обучения.
- Представление данных для нейронных сетей. Тензоры. Библиотеки Tensor Flow и Keras. GPU и TPU.
- Глубокое обучение в задачах компьютерного зрения. Сверточные нейронные сети.
- Глубокое обучение для последовательностей и текста.
- Генеративные модели.
- Эмбеддинги.
- Большие языковые модели.

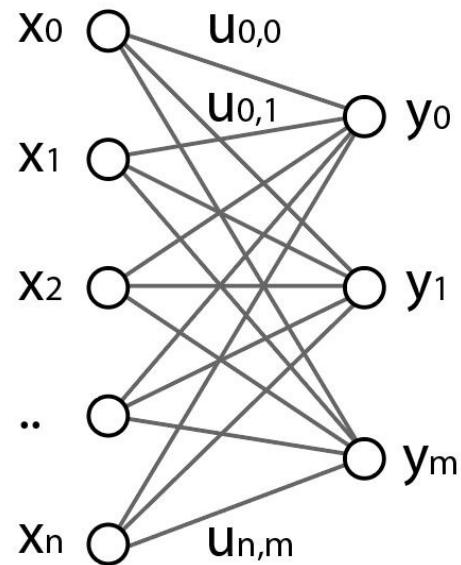




Source: http://beamlab.org/deeplearning/2017/02/23/deep_learning_101_part1.html

Представление данных для нейронных сетей

Нейронная сеть

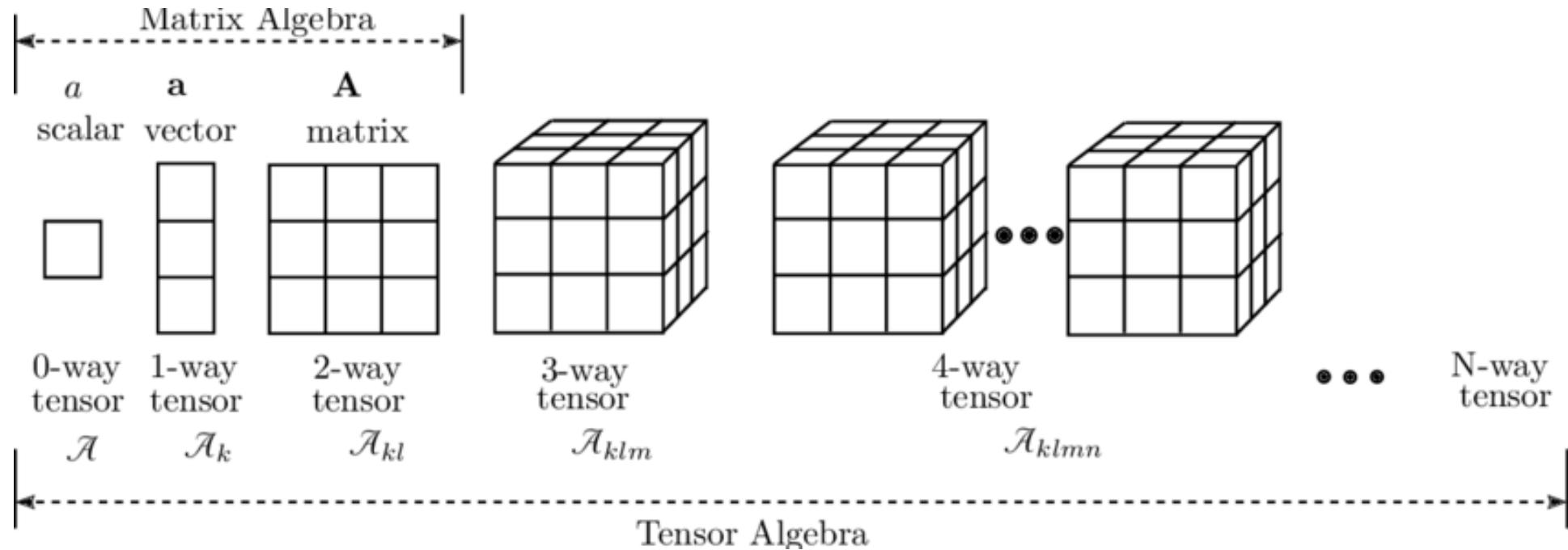


a) Fully connected layer

$$\begin{matrix} & 1 \times N & N \times M & M \times 1 \\ \begin{matrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{matrix}^T & \otimes & \begin{matrix} u_{0,0} & u_{0,1} & u_{0,2} \\ u_{1,0} & u_{1,1} & u_{1,2} \\ u_{2,0} & u_{2,1} & u_{2,2} \\ \vdots \\ u_{n,0} & & u_{n,m} \end{matrix} & = \begin{matrix} y_0 \\ y_1 \\ y_2 \end{matrix} \end{matrix}$$

b) Expressed as a tensor product

Тензоры



Атрибуты тензора:

- Ранг - количество осей.
- Форма – кортеж целых чисел, описывающих количество элементов по каждой оси.
- Тип данных.

Примеры тензоров

- Табличные (векторные) данные – 2-мерный тензор с формой `{instances, features}`
- Временные ряды (последовательности) – 3-мерный тензор `{instances, time, labels, features}`. Каждый образец - это последовательность признаков.
- Изображения – 4-мерный тензор `{instances, height, width, color}`. Каждый образец это матрица пикселей, а каждый пиксель кодируется цветом.
- Видео – 5 мерный тензор `{instances, frames, height, width, color}`. Каждый образец – это последовательность кадров / изображений.

Библиотеки TensorFlow и Keras



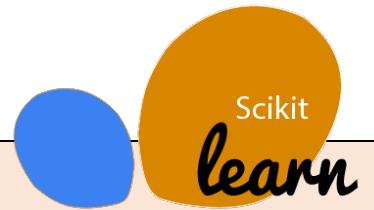
- Maintainer: Google
- Open Source
- API: Python, R, Java, C...
- CPU / GPU
- Automatic differentiation scheme "Define-by-Run"



- Maintainer: François Chollet
- Open Source
- Python
- TensorFlow frontend



Код MLP в Keras

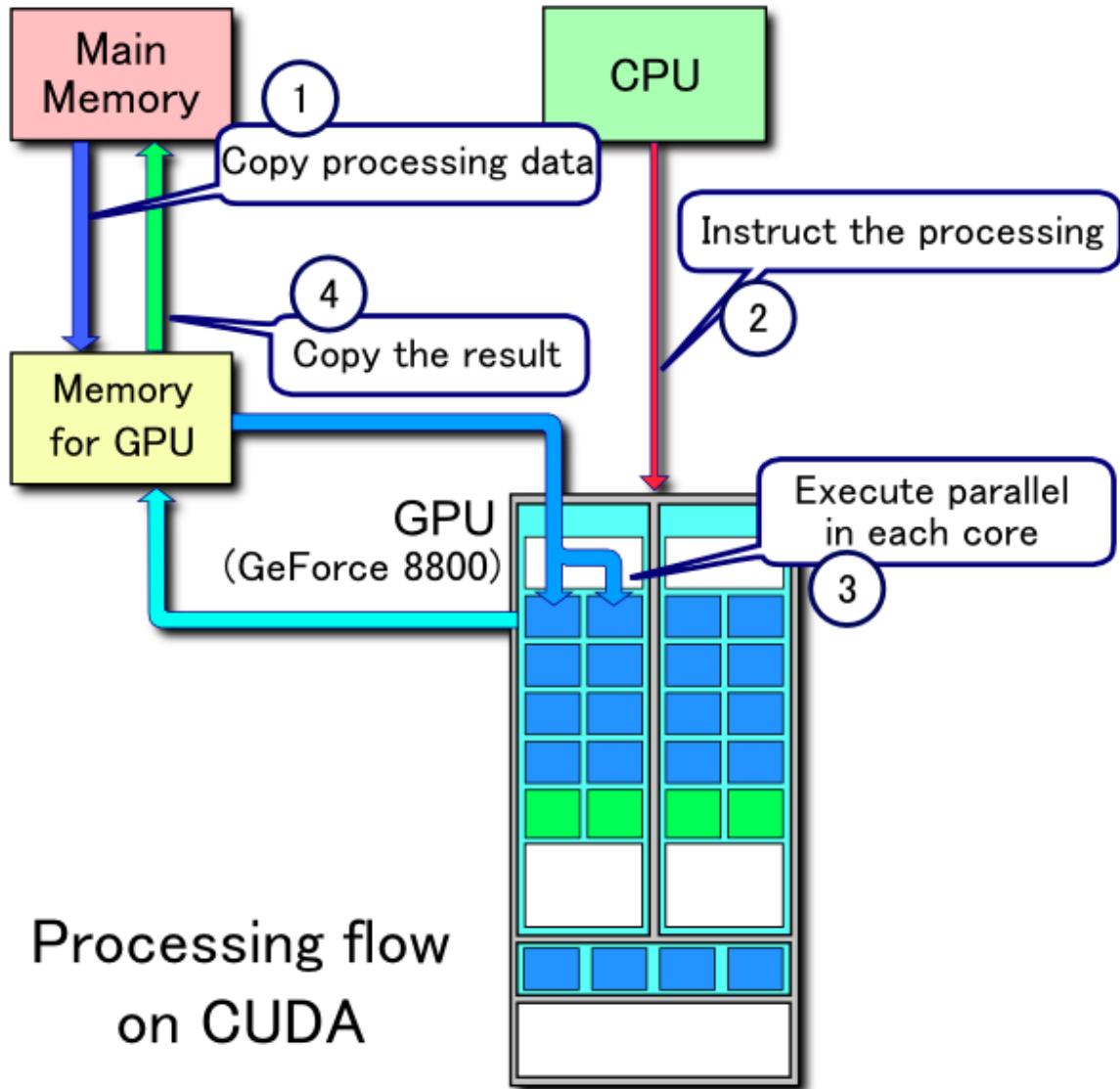


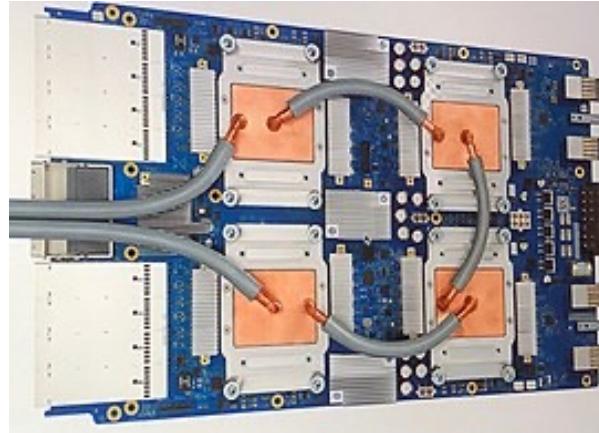
```
from sklearn.neural_network import MLPRegressor  
  
model = MLPRegressor(hidden_layer_sizes=(10,2), activation='relu', solver='adam')  
model.fit(X_train, y_train)
```



```
import tensorflow as tf  
  
model = tf.keras.models.Sequential([  
    tf.keras.layers.Dense(10, activation='relu', input_shape=(None, X_train.shape[1])),  
    tf.keras.layers.Dense(2, activation='relu'),  
    tf.keras.layers.Dense(1) ])  
  
model.compile( loss = 'mean_squared_error',  
               optimizer = 'SGD',  
               metrics = [ 'mean_squared_error' ] )  
  
history = model.fit(X_train, y_train, epochs=10, validation_data = (X_valid, y_valid), verbose=0)
```

Graphical Process Unit (GPU)





	NVIDIA Tesla A100	NVIDIA Hopper H100
On chip memory	80 GB	80 GB
Memory bandwidth	2 TB/s	2 TB/s
TFLOPS	312 TFLOPS	1 GFLOPS
Price	\$ 17 000	\$ 40 000

Tensor Processing Unit products^{[13][14][15]}

	TPUv1	TPUv2	TPUv3	TPUv4 ^{[14][16]}	TPUv5 ^[17]	Edge v1
Date introduced	2016	2017	2018	2021	2023	2018
Process node	28 nm	16 nm	16 nm	7 nm	Unstated	
Die size (mm ²)	331	< 625	< 700	< 400	Unstated	
On-chip memory (MiB)	28	32	32	32	48	
Clock speed (MHz)	700	700	940	1050	Unstated	
Memory	8 GiB DDR3	16 GiB HBM	32 GiB HBM	32 GiB HBM	16 GB HBM	
Memory bandwidth	34 GB/s	600 GB/s	900 GB/s	1200 GB/s	819 GB/s	
TDP (W)	75	280	220	170	Not Listed	2
TOPS (Tera Operations Per Second)	92	45	123	275	393	4
TOPS/W	0.31	0.16	0.56	1.62	Not Listed	2

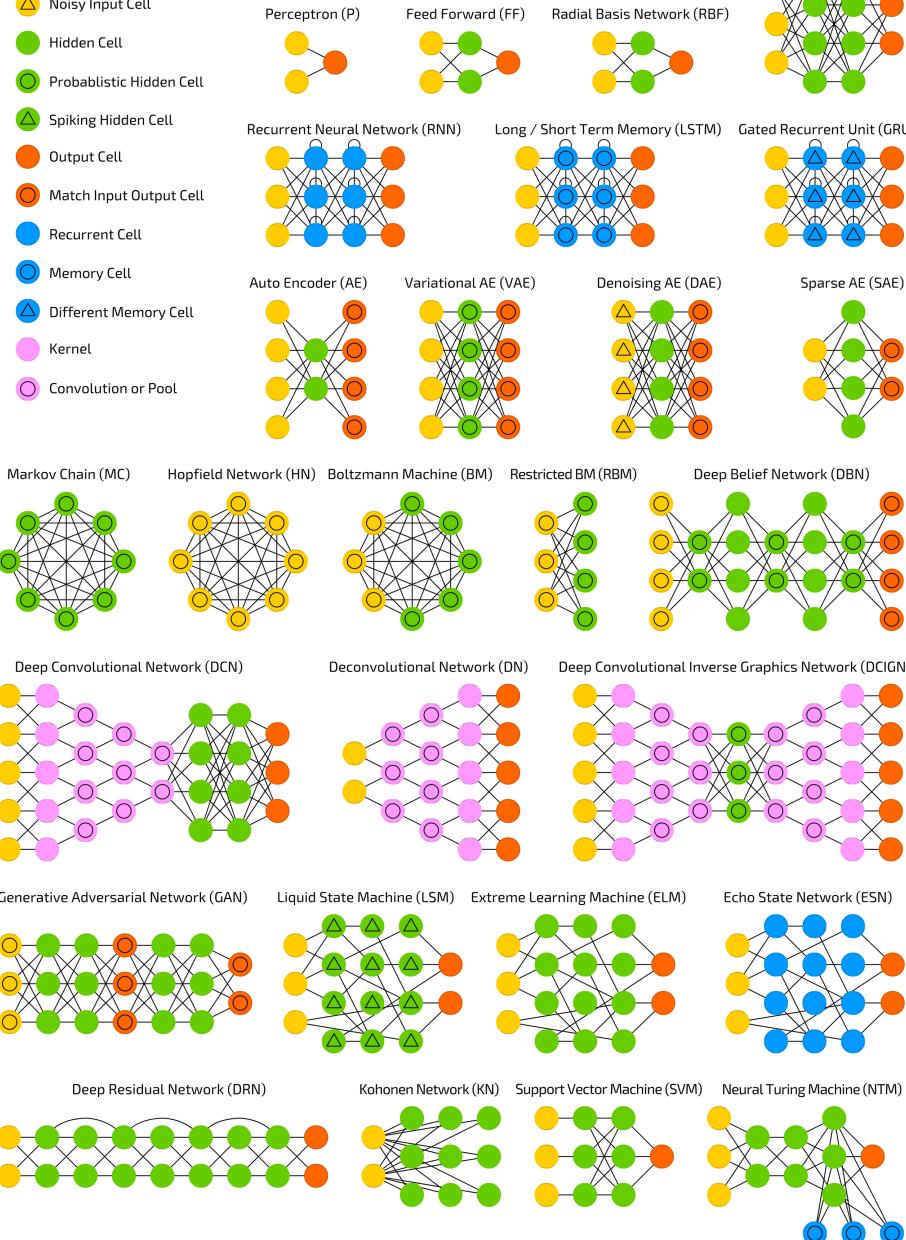
https://en.wikipedia.org/wiki/Tensor_Processing_Unit

According to Google researchers, the TPU v4 is 1.2x to 1.7x faster than the NVIDIA A100 in various ML workloads.

A mostly complete chart of Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool



Искусственные нейронные сети.

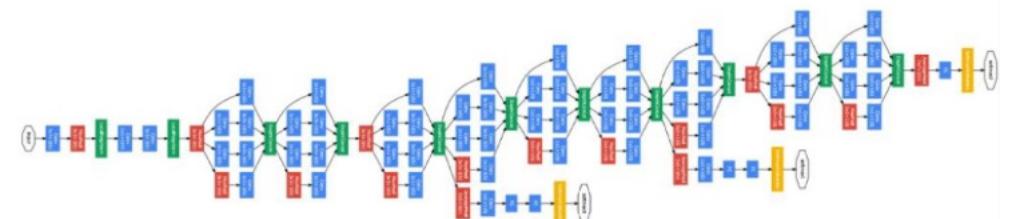
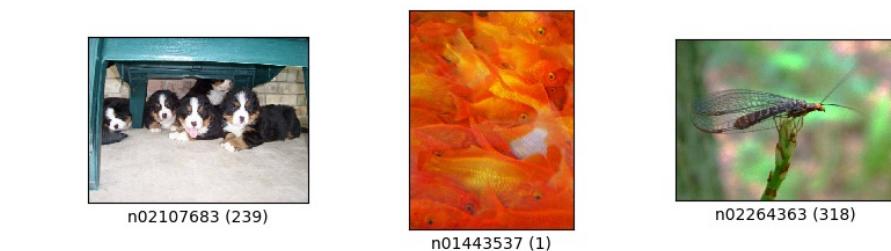
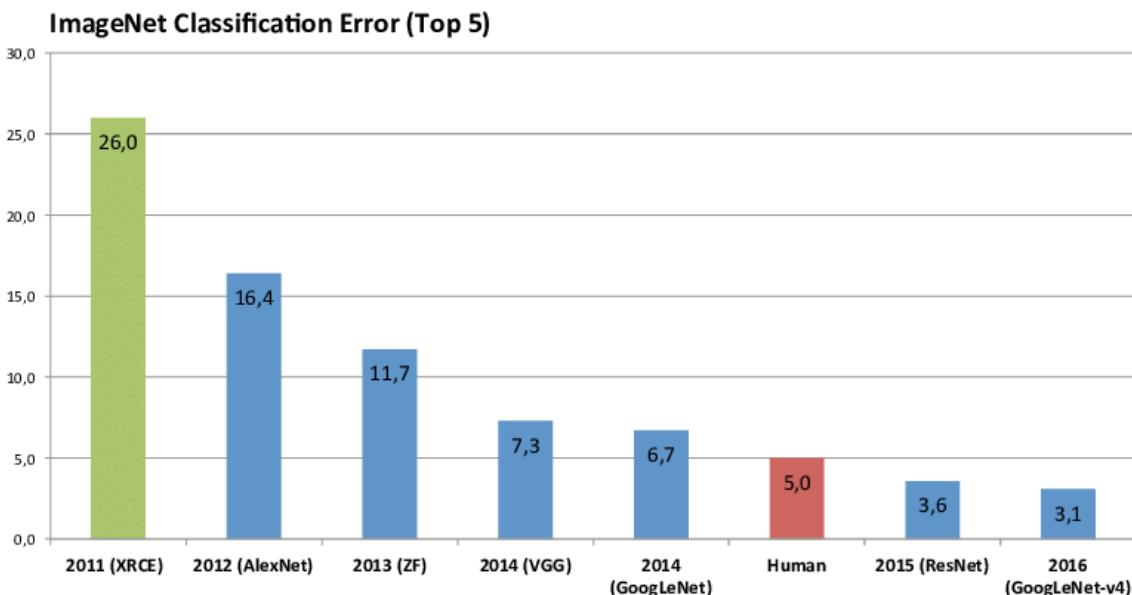
Для каждой задачи
своя архитектура.

Спойлер:
так было до 2019 года.

Компьютерное зрение.
Сверточные нейронные сети.

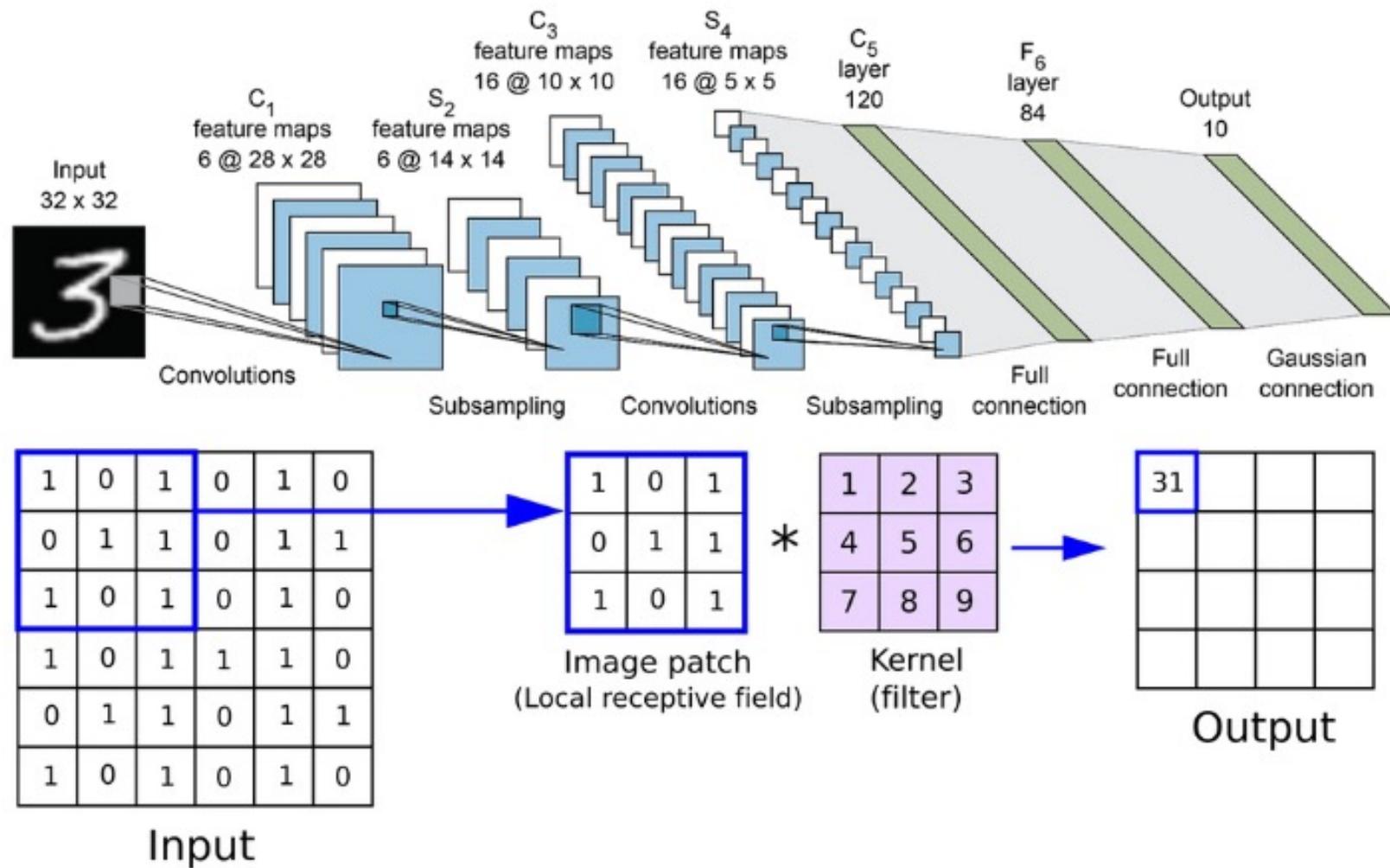
ImageNet

- 14+ млн. изображений
- 21+ тыс. категорий объектов
- В среднем 500 изображений на категорию
- <https://image-net.org>



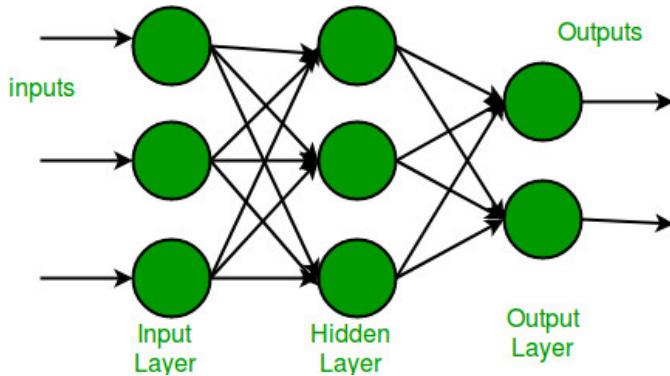
Реальная нейросеть: GoogLeNet (2014)

Convolution Neural Network



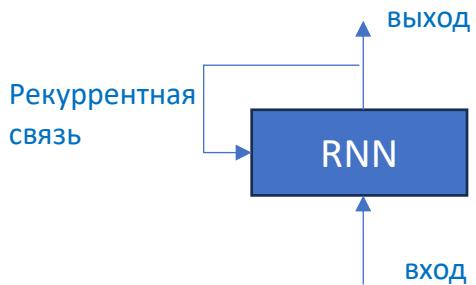
Последовательности. Рекуррентные сети

Multi-Layer Perceptron (MLP)

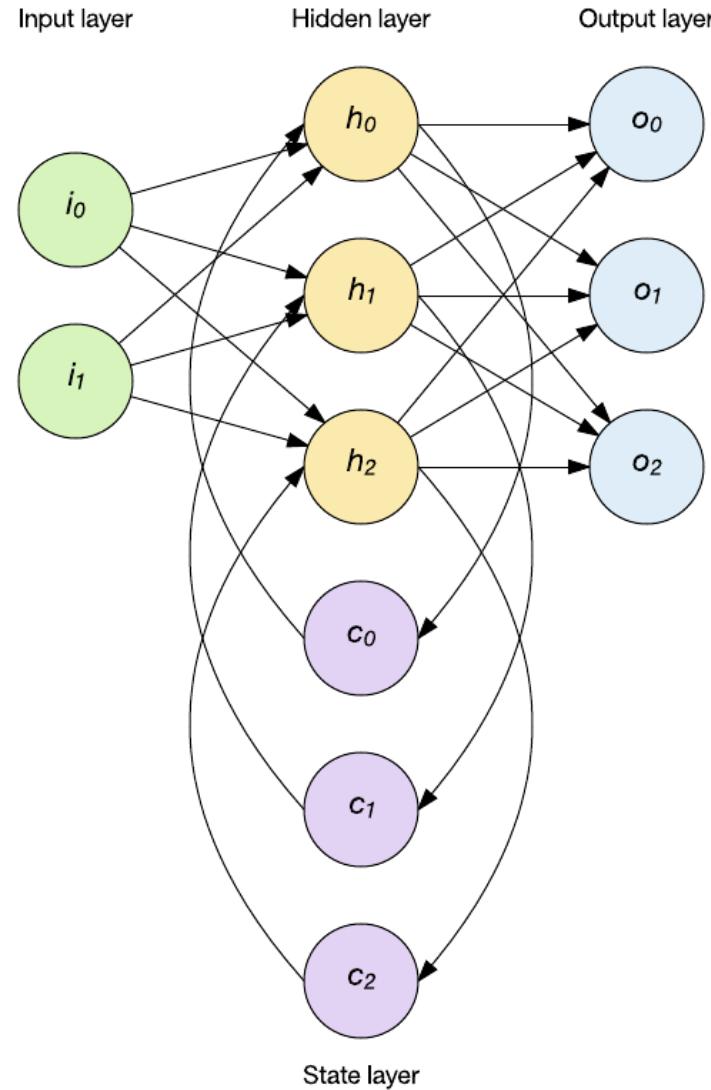


Проблемы с обработкой последовательностей, поскольку не запоминает состояние / контекст.

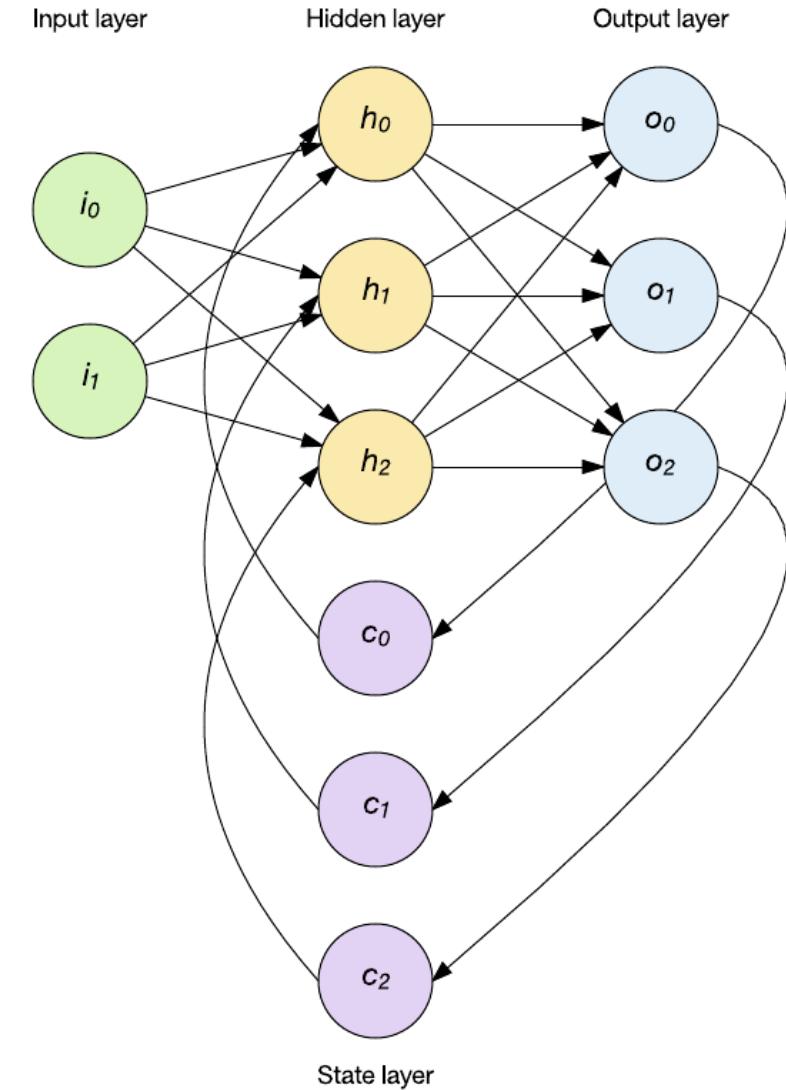
Биологический интеллект воспринимает информацию последовательно, сохраняя внутреннюю модель и постоянно дополняя ее новыми данными.



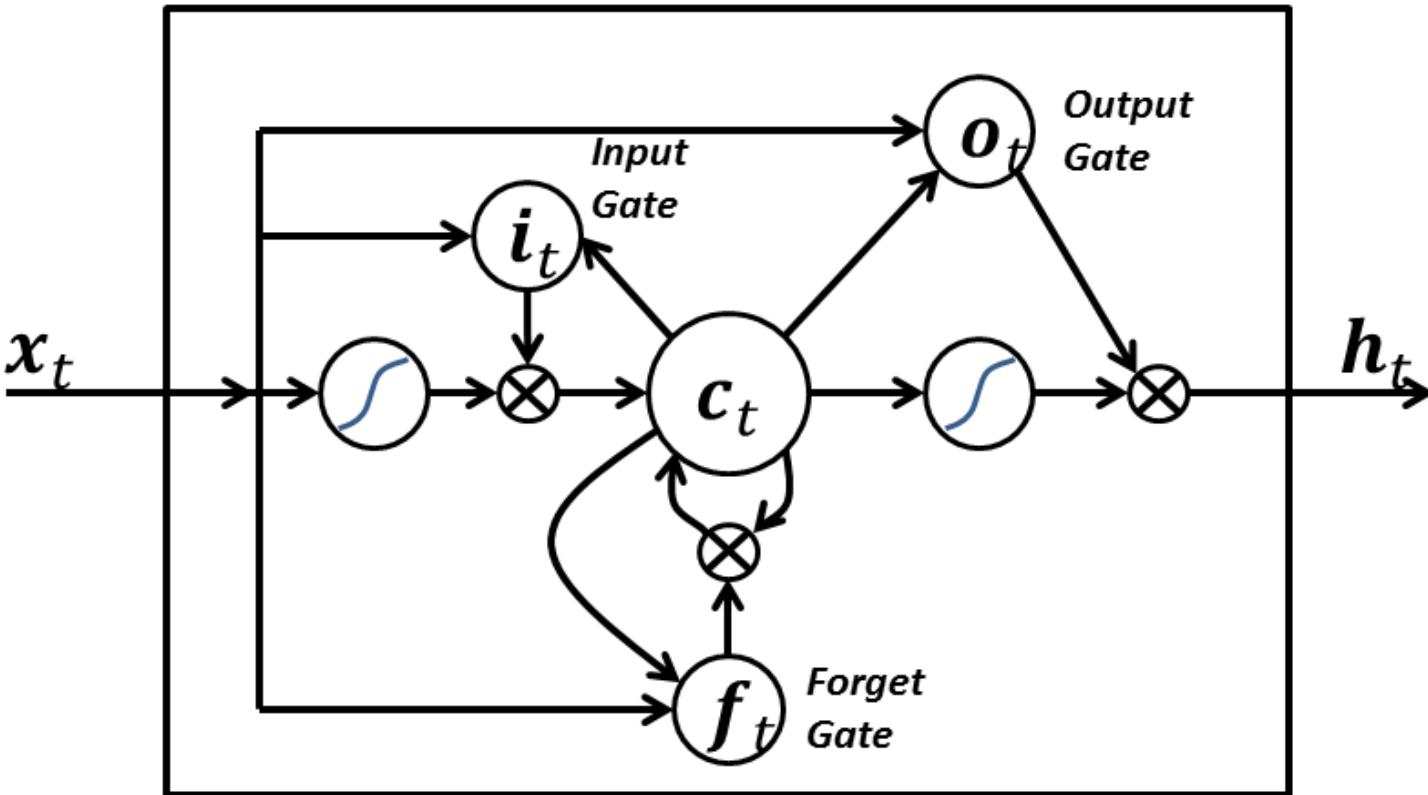
Elman network



Jordan network



Long Short-Term Memory (LSTM)



$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \circ \sigma_h(c_t) \end{aligned}$$

Переменные:

- x_t — входной вектор,
- h_t — выходной вектор,
- c_t — вектор состояний,
- W , U и b — матрицы параметров и вектор,
- f_t , i_t и o_t — векторы вентиляй,
- f_t — вектор вентиля забывания, вес запоминания старой информации,
- i_t — вектор входного вентиля, вес получения новой информации,
- o_t — вектор выходного вентиля, кандидат на выход.

Функции активации:

- σ_g : на основе [сигмоиды](#).
- σ_c : на основе [гиперболического тангенса](#).
- σ_h : на основе гиперболического тангенса,

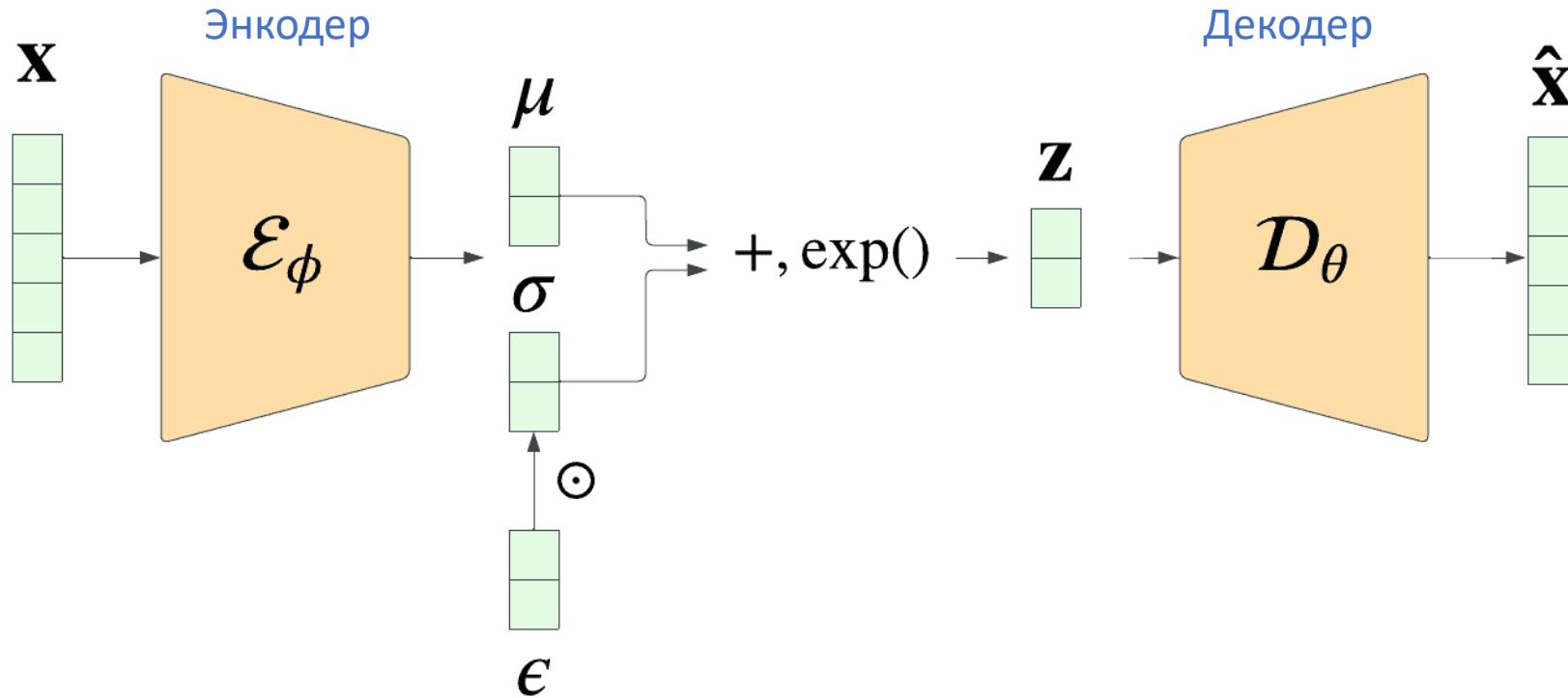
2009 - победа на соревнованиях по распознаванию рукописного текста (ICDAR)

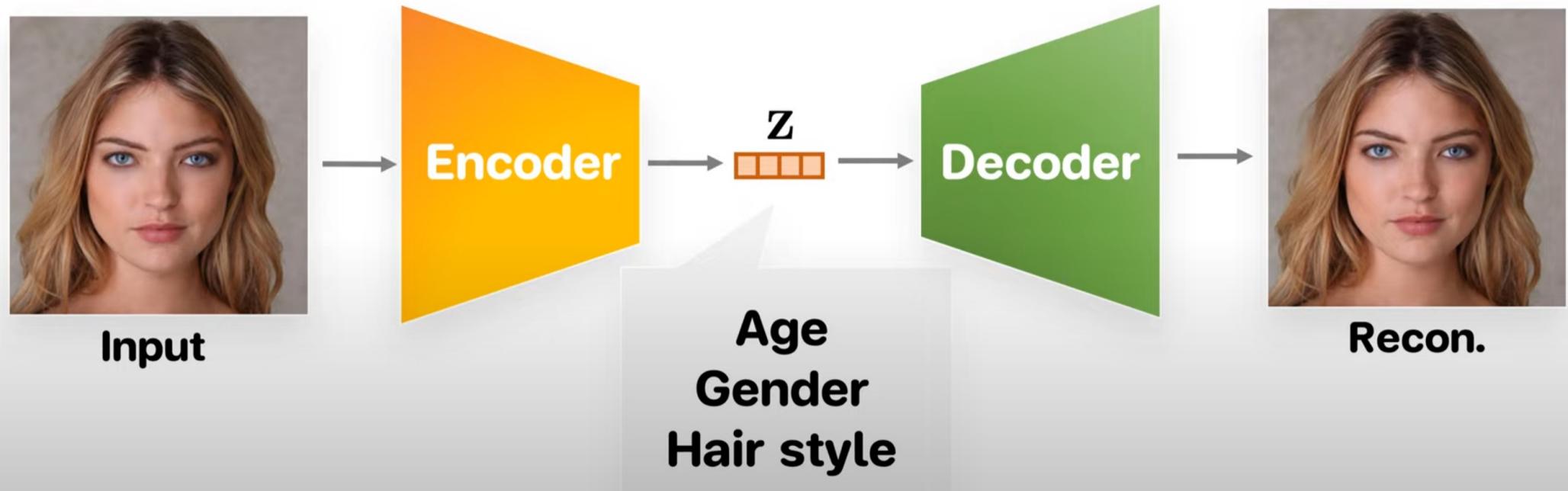
2013 - ошибка в 17,7 % в задаче распознавания фонем на корпусе естественной речи TIMIT

Генеративные модели

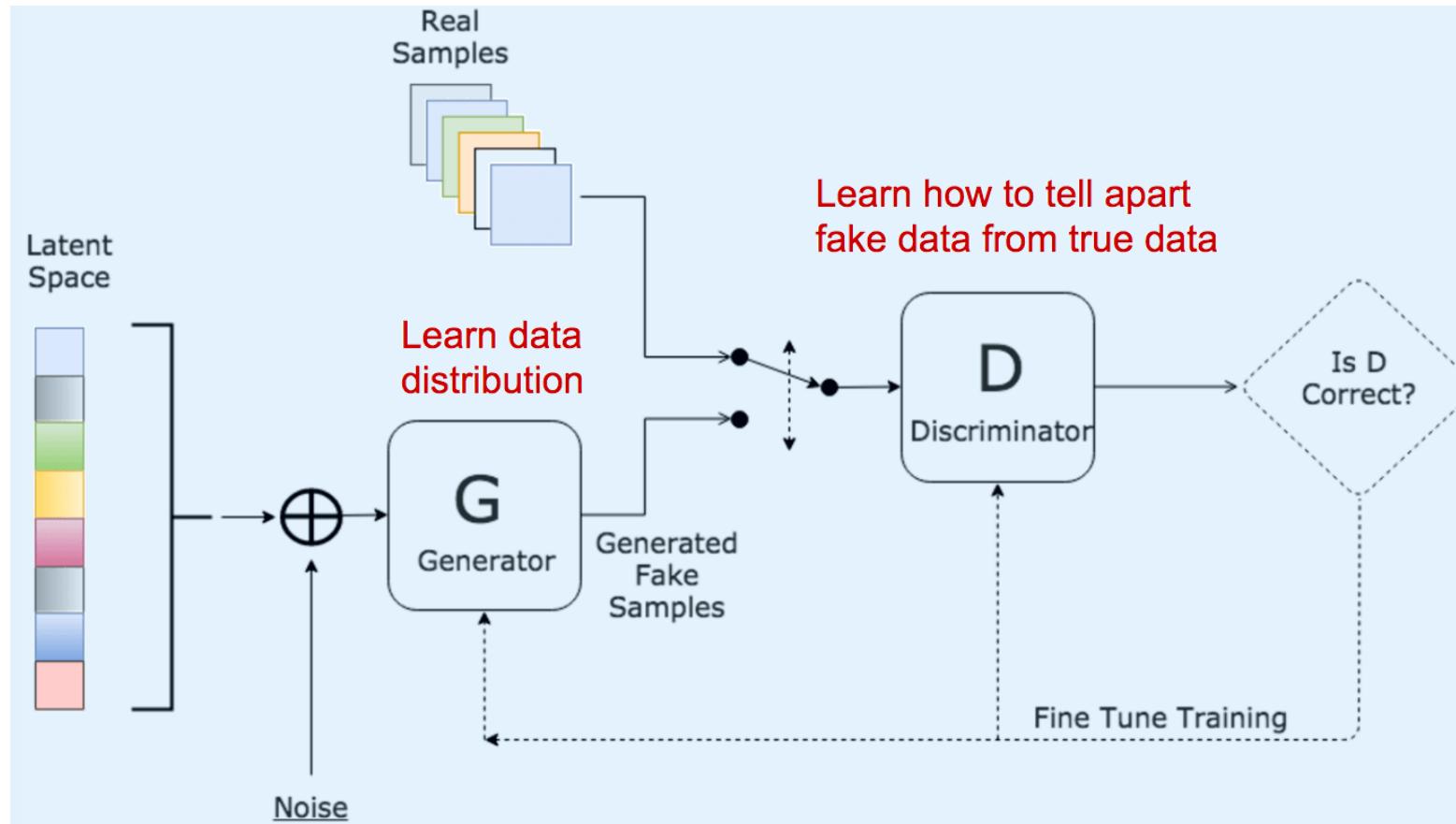
Variational Autoencoder (VAE)

Вариационный автокодировщик





Генеративная состязательная сеть Generative Adversarial Network (GAN)



Input



Output



Horse → Zebra

Input



Output



Summer → Winter

Input



Output



Monet → Photo



Zebra → Horse



Winter → Summer



Photo → Monet

Векторное представление слов. Embedding

Вектор вещественных чисел, кодирующий значение слова таким образом, что слова, которые находятся ближе в пространстве векторов, ближе по значению.

Word2Vec

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasut@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

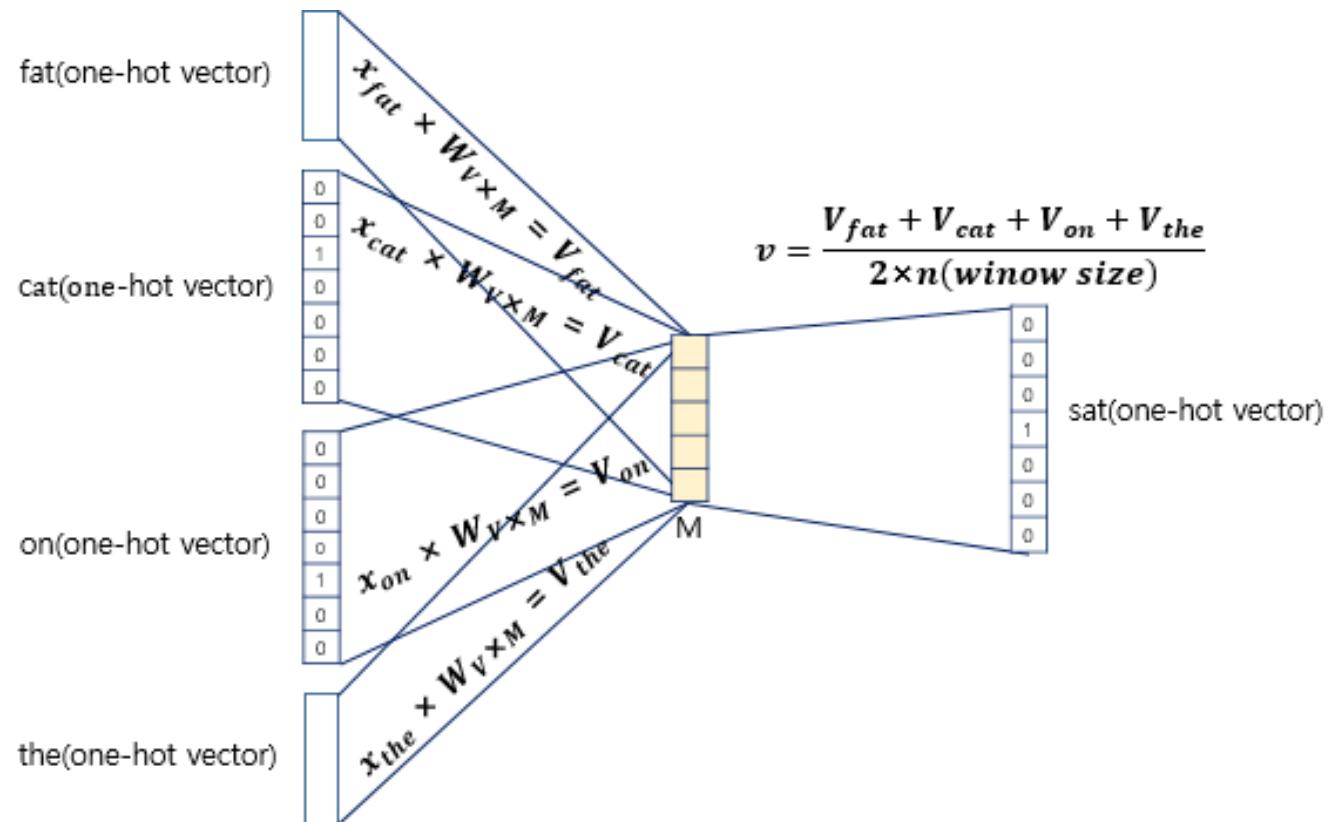
Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

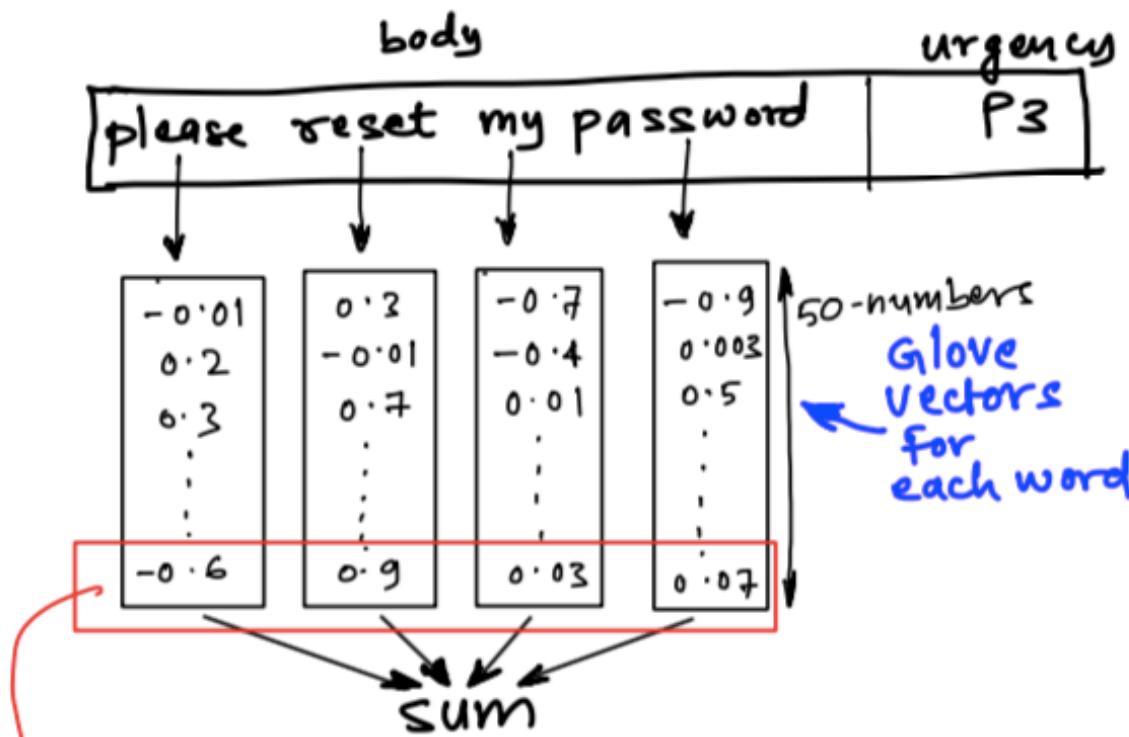
An inherent limitation of word representations is their indifference to word order and their inability to represent idiomatic phrases. For example, the meanings of “Canada” and “Air” cannot be easily combined to obtain “Air Canada”. Motivated by this example, we present a simple method for finding phrases in text, and show that learning good vector representations for millions of phrases is possible.



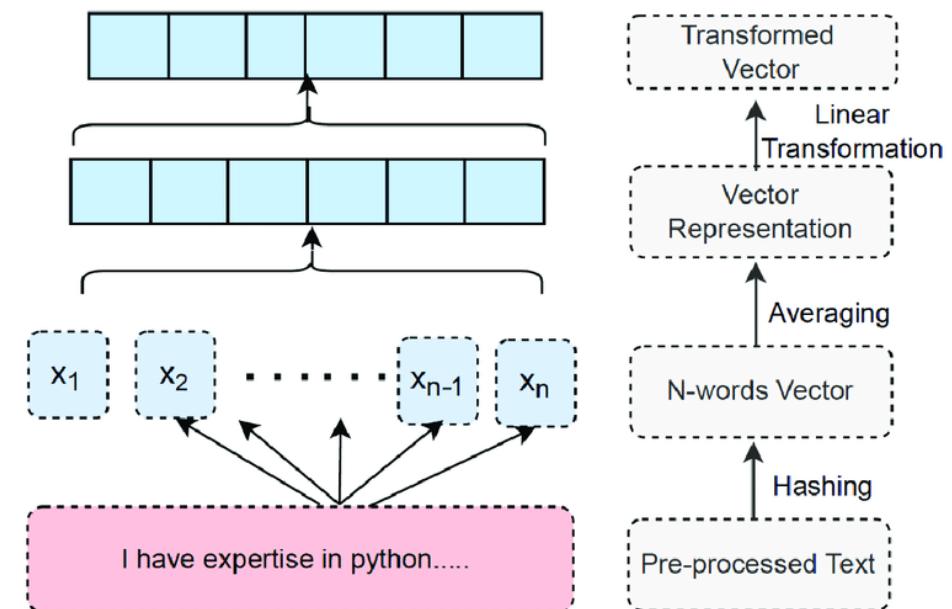
king – man + woman = queen

Другие подходы

GloVe – учет глобального контекста
(Stanford University)



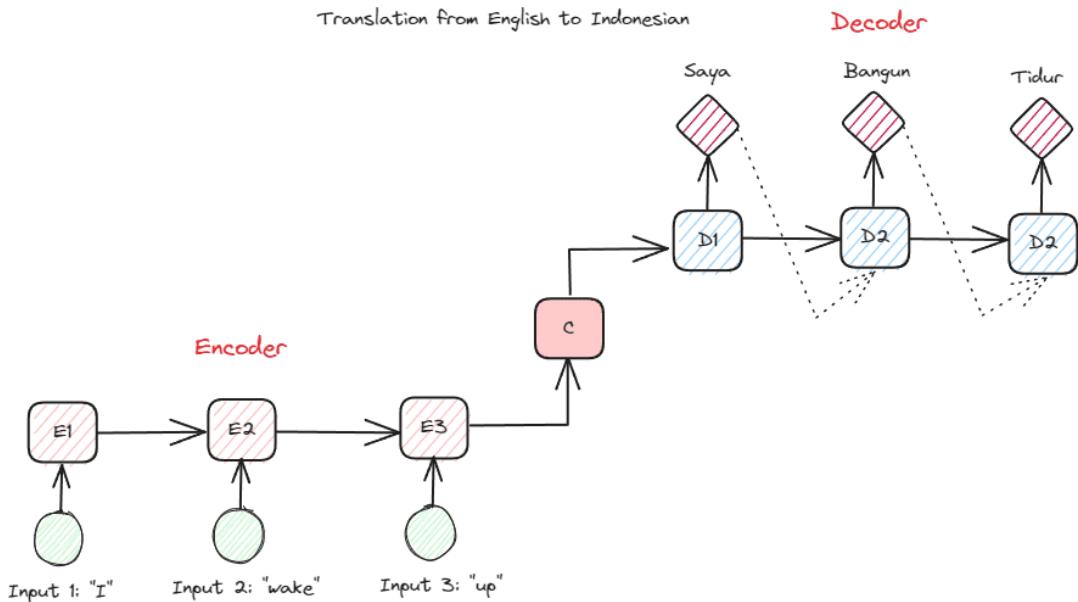
fastText – учет глобального контекста
(Facebook)



Transformer

Задача машинного перевода

Две рекуррентных сети



Модель принимает последовательность слов на одном языке на входе, и должна сгенерировать последовательность слов на другом языке.

При этом необходимо сохранять внутреннее состояние модели, поскольку слова вводятся и генерируются по одному, но перевод зависит от всего контекста.

Используются две сети:

- энкодер – формирование обновление внутреннего представления контекста
- декодер – генерация слов из входной последовательности с учетом контекста.

Проблемы с длинными последовательностями данных.

<https://www.kdnuggets.com/large-language-models-explained-in-3-levels-of-difficulty>

Transformer

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Lukasz Kaiser*
Google Brain
lukaszkaiser@google.com

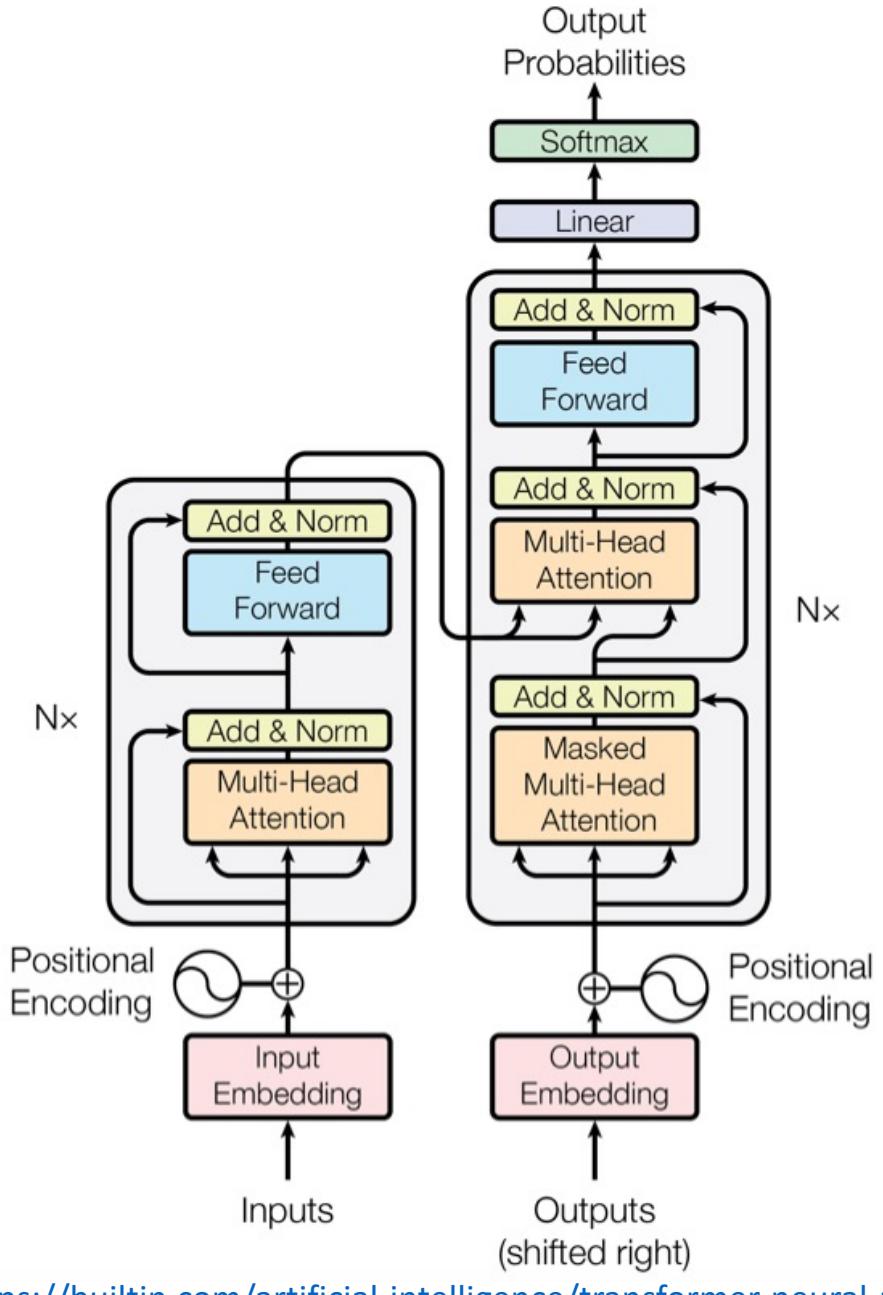
Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Vaswani, A. et al. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

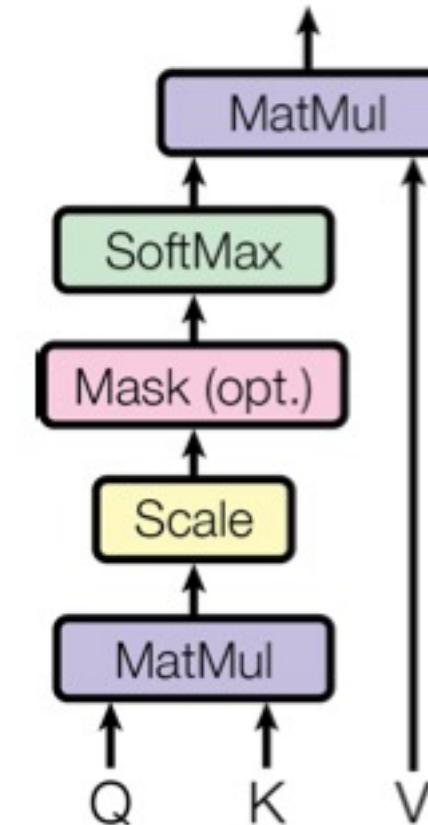
Преобразование последовательности (Input) в последовательность (Output)



<https://builtin.com/artificial-intelligence/transformer-neural-network>

Transformer

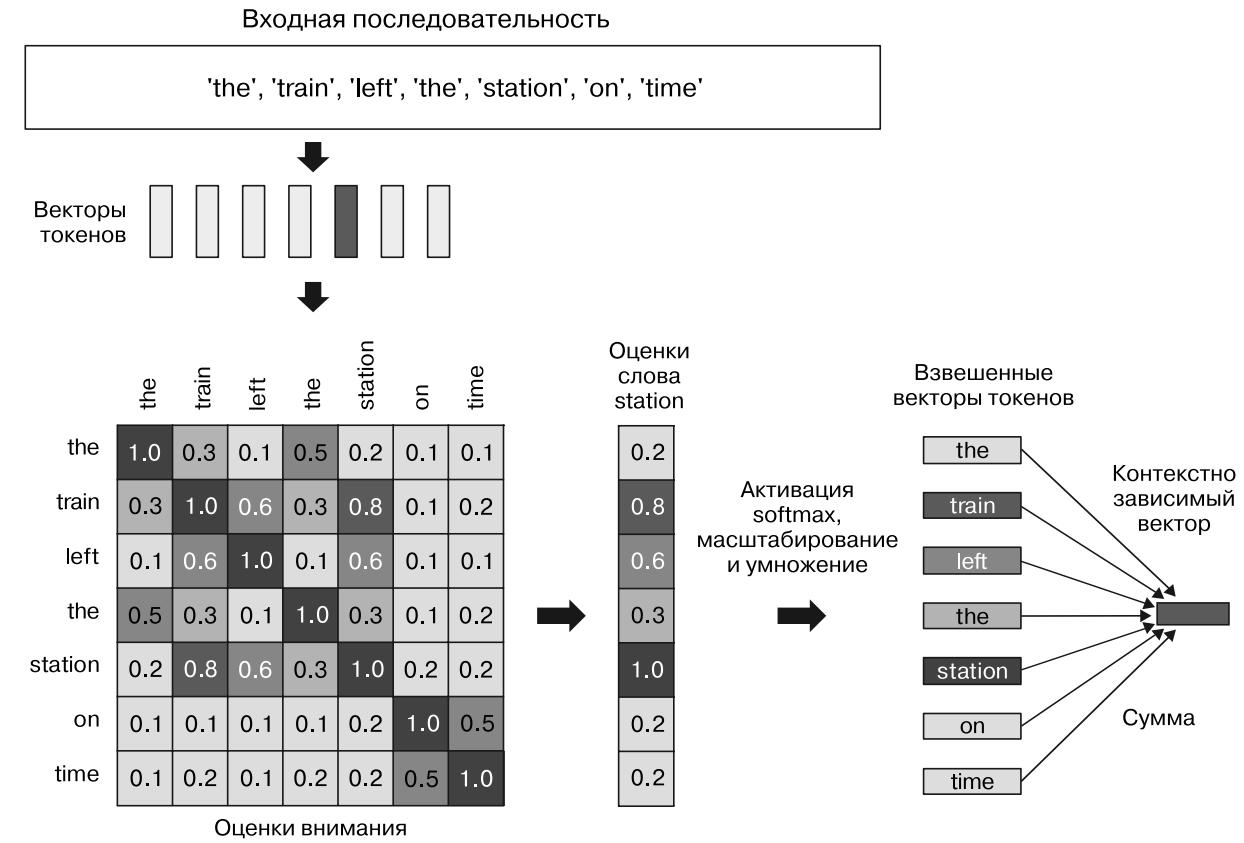
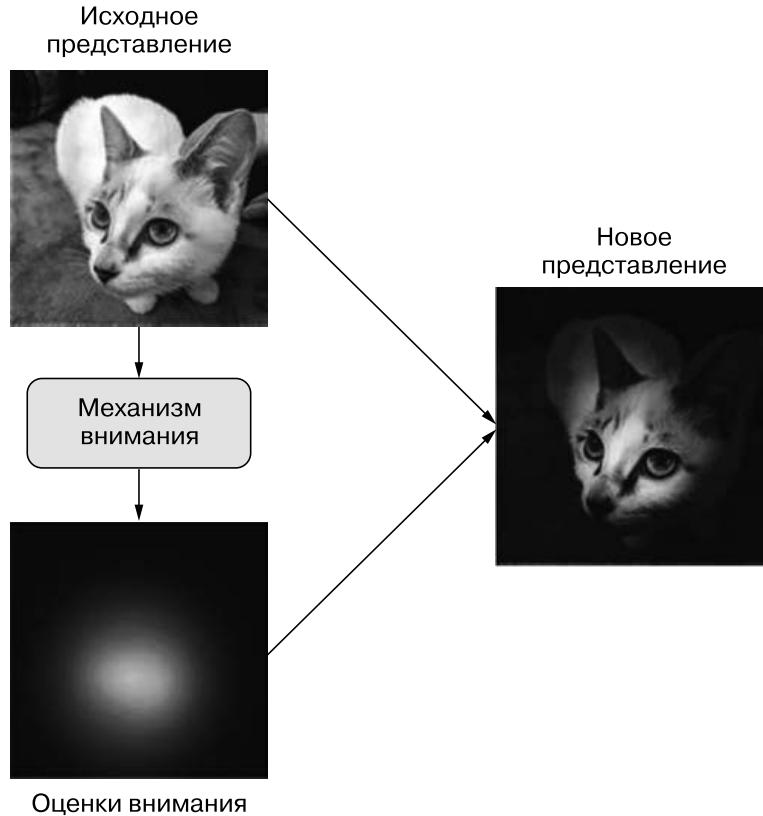
- Рекуррентные модели генерируют последовательность скрытых состояний h_t , как функцию предыдущего скрытого состояния h_{t-1} и входа для позиций t . Эта последовательная природа исключает распараллеливание в обучающих примерах, что становится критичным при больших длинах последовательностей.
- Использование механизма внимания позволяет использовать информацию полученную не только из последнего скрытого состояния, но и любого скрытого состояния h_t для любого t .
- В Трансформере нет рекуррентных связей, все основано только на механизме внимания, связывающем различные позиции одной последовательности.
- Transformer динамически создает эмбеддинги в процессе обучения.



Queries Keys Values
 d_k - dimension

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Механизм внимания

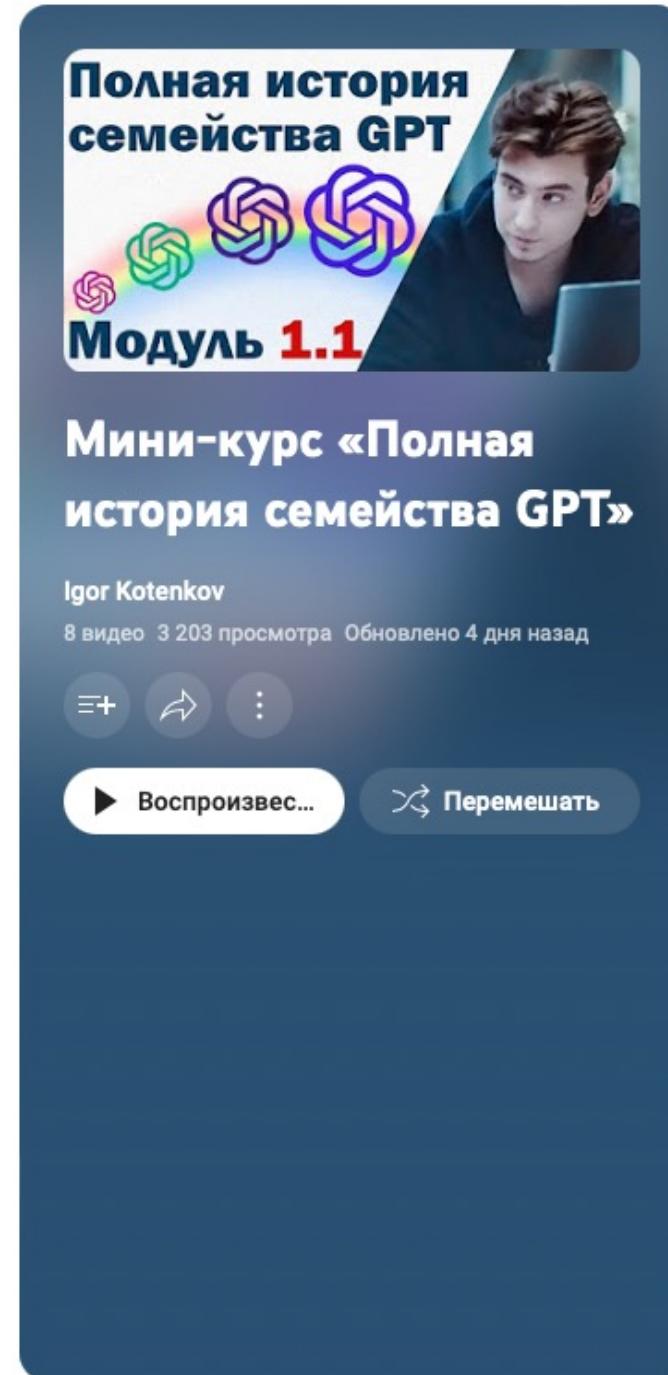


Большие языковые модели

Large Language Models (LLM)

Для тех, кто хочет погрузиться в тему глубже

[Смотреть курс](#)



Как работает трансформер

Supervised learning

Нужна огромная выборка на двух языках

Target Model Outputs

Output Vocabulary: a am I thanks student <eos>

position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0

Trained Model Outputs

Output Vocabulary: a am I thanks student <eos>

position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001

INPUT

je suis étudiant

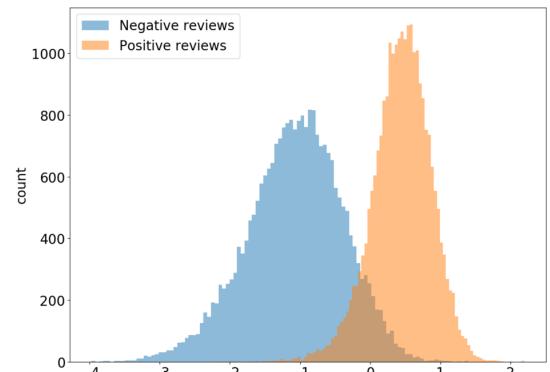
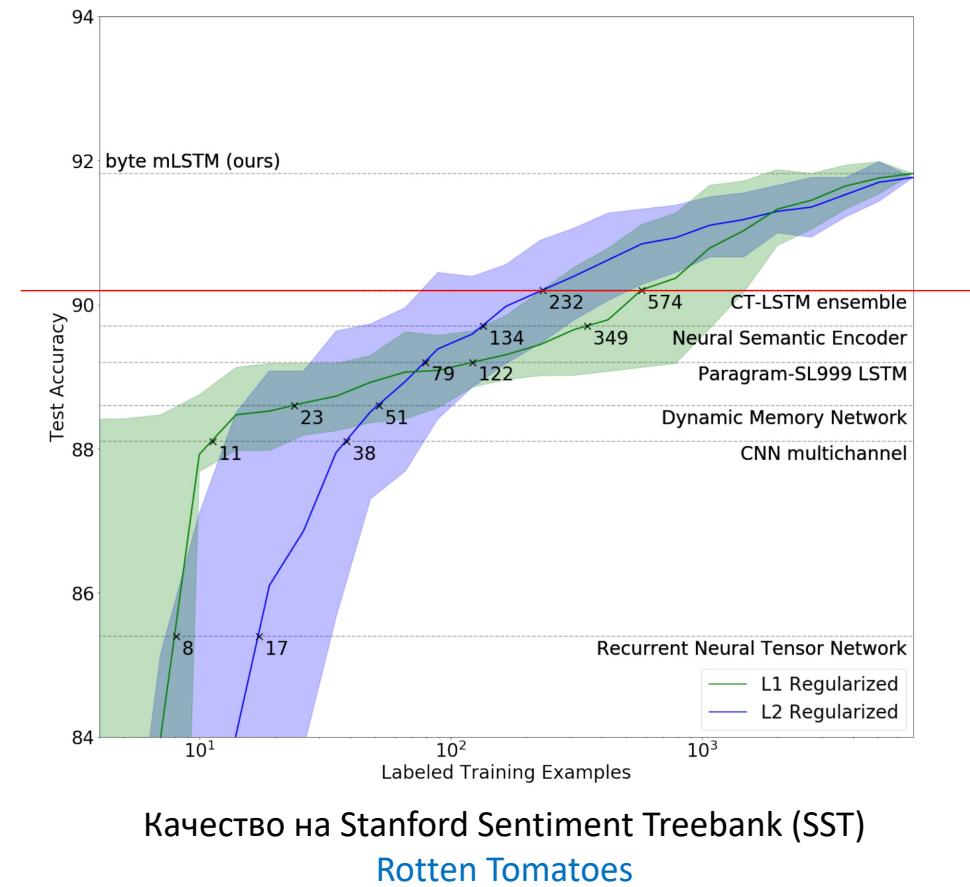
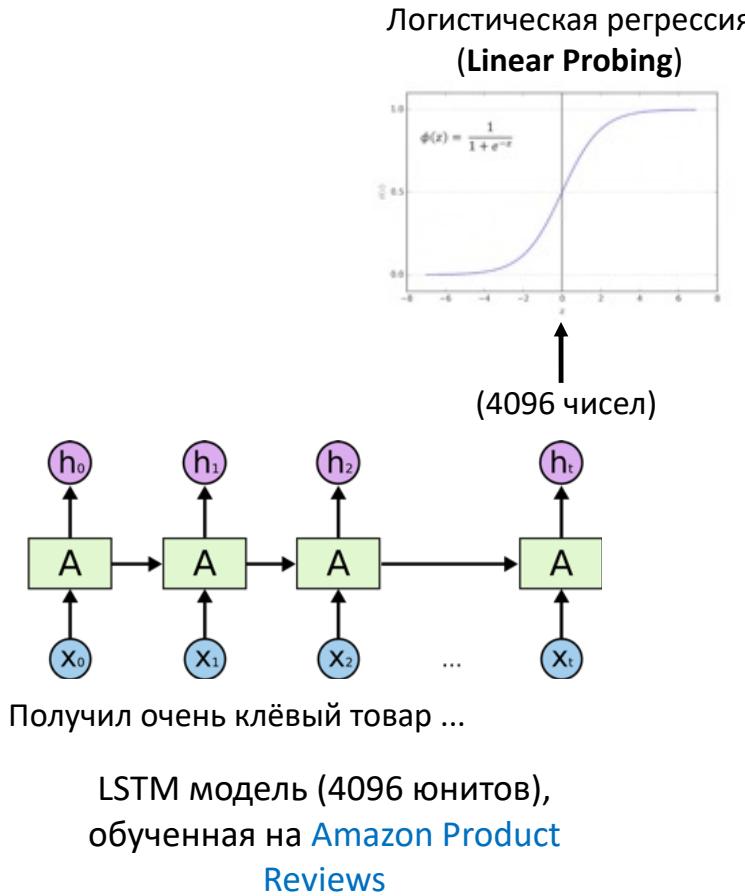
Предсказание следующего слова в последовательности – обучение без учителя.

- Не нужна разметка, поэтому можно собрать много данных
- Больше данных – более робастная (обобщающая) модель
- Вместо предсказания чего-то конкретного будем предсказывать все, что угодно (== предсказание следующего слова).
- Правильное предсказание означает, что модель понимает «реальность», которая создала предсказываемую сущность.
- Внутри модели формируется **модель реальности (?)**

Пример: предсказать последнюю главу книги после прочтения всех предыдущих.

Для этого надо понять содержание предыдущих глав.

Первый эксперимент OpenAI – определение эмоциональной окраски

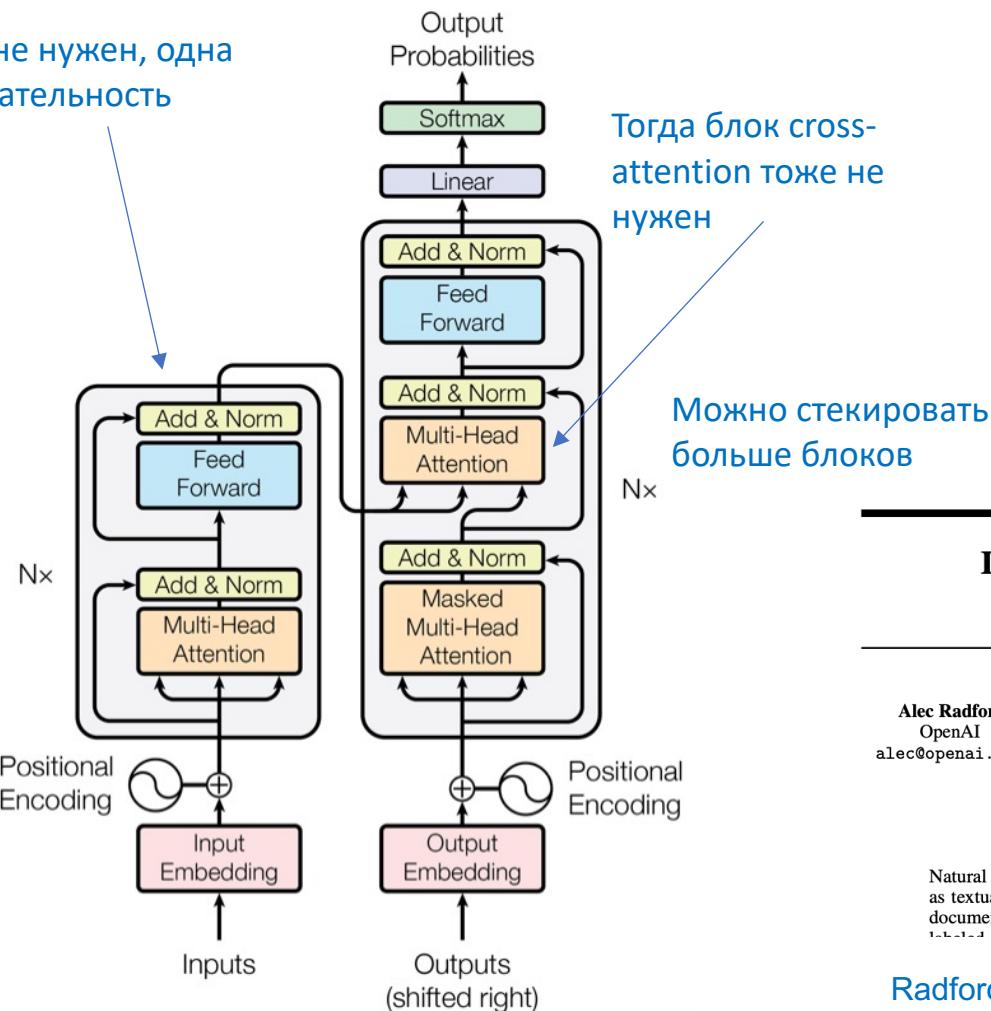


Распределение активации
одного нейрона на датасете
[IMDB reviews](#)

Это поведение на проявляется на
малых сетях (512 юнитов)

Трансформация трансформера

Энкодер не нужен, одна последовательность



Тогда блок cross-attention тоже не нужен

Можно стекировать больше блоков



Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

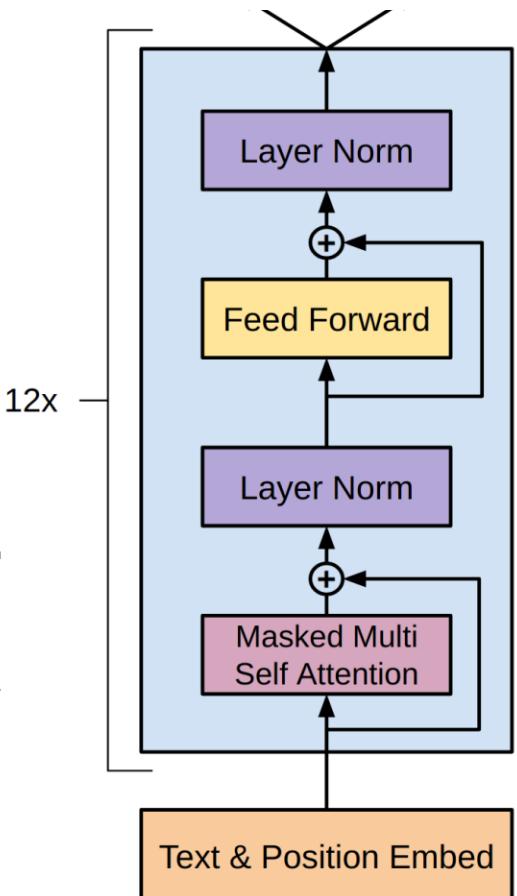
Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant,

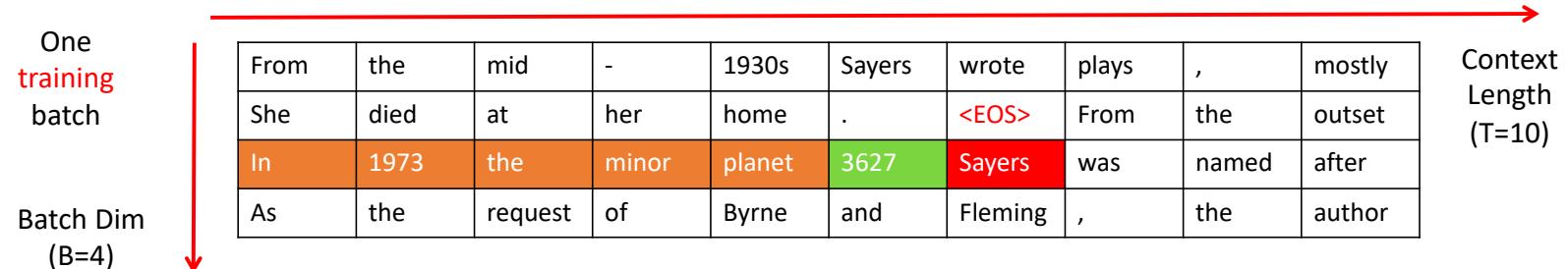
Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.



Семейство GPT

	GPT-1 (2018)	GPT-2 (2019)	GPT-3 (2020)	GPT-4 (2023)
Кол-во параметров	117М	1.5B	175B	1.76 Т
Кол-во слоев	12	48	96	
Размер контекста	512	1024	2048	
Размерность вектора эмбеддинга	768	1 600	12 288	~ 25 000
n_{heads}	12 ($d_{\text{head}} = 64$)	25 ($d_{\text{head}} = 64$)	96 ($d_{\text{head}} = 128$)	

Контекст:



Чему и как учили GPT-3

Данные

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Вычислительные ресурсы

Model	Total train compute (PF-days)	Total train compute (flops)	\$10M
T5-3B	1.04E+02	9.00E+21	
T5-11B	3.82E+02	3.30E+22	
GPT-3 13B	2.68E+02	2.31E+22	
GPT-3 175B	3.64E+03	3.14E+23	

TOP10 System - November 2023 <https://top500.org>

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	Aurora - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	4,742,808	585.34	1,059.33	24,687
3	Eagle - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Microsoft Azure United States	1,123,200	561.20	846.84	

Оценка человеком

PA can you list the common attributes in that?

Which response do you prefer?
Your choice will help make ChatGPT better.

 Response 1
Sure, based on the list of numbers and their square roots, here are some observations or common attributes:

- Perfect Squares:** Some of the numbers are perfect squares, meaning their square roots are integers. These numbers are:
 - 16 (square root is 4)
 - 100 (square root is 10)
 - 144 (square root is 12)
 - 9 (square root is 3)
 - 256 (square root is 16) ■
- Irrational Numbers:** Some of the square roots are irrational numbers, which means they cannot be expressed as exact fractions, and

 Response 2
Certainly! When examining the list of numbers and their square roots, several attributes or characteristics become apparent:

- Integer Roots:** Some numbers (16, 100, 144, 9, 256) are perfect squares, meaning their square roots are integers.
- Irrational Roots:** The other numbers (2, 11, 61, 18, 72) are not perfect squares, and their square roots are irrational numbers. This means their decimal representations go on forever without repeating.
- Range:** The square roots span a wide range of values, from approximately 1.4142 to 16. Stop generating
- Even and Odd:** The original list contains both even and odd

even or odd
Send a message

Instead of training the model to accurately predict the next word, we will train it to generate responses that are better (according to human perception).

ЧТО ВЫУЧИВАЮТ МОДЕЛИ?

INTERPRETABILITY IN THE WILD: A CIRCUIT FOR INDIRECT OBJECT IDENTIFICATION IN GPT-2 SMALL

Kevin Wang¹, Alexandre Variengien¹, Arthur Conmy¹, Buck Shlegeris¹ & Jacob Steinhardt^{1,2}

¹Redwood Research

²UC Berkeley

kevin@rdwrs.com, alexandre@rdwrs.com,
arthur@rdwrs.com, buck@rdwrs.com, jsteinhardt@berkeley.edu

ABSTRACT

Research in mechanistic interpretability seeks to explain behaviors of machine learning (ML) models in terms of their internal components. However, most previous work either focuses on simple behaviors in small models or describes complicated behaviors in larger models with broad strokes. In this work, we bridge this gap by presenting an explanation for how GPT-2 small performs a natural language task called indirect object identification (IOI). Our explanation encompasses 26 attention heads grouped into 7 main classes, which we discovered using a combination of interpretability approaches relying on causal interventions. To our knowledge, this investigation is the largest end-to-end attempt at reverse-engineering a natural behavior “in the wild” in a language model. We evaluate the reliability of our explanation using three quantitative criteria—*faithfulness*, *completeness* and *minimality*. Though these criteria support our explanation, they also point to remaining gaps in our understanding. Our work provides evidence that a mechanistic understanding of large ML models is feasible, pointing toward opportunities to scale our understanding to both larger models and more complex tasks. Code for all experiments is available at <https://github.com/redwoodresearch/Easy-Transformer>.

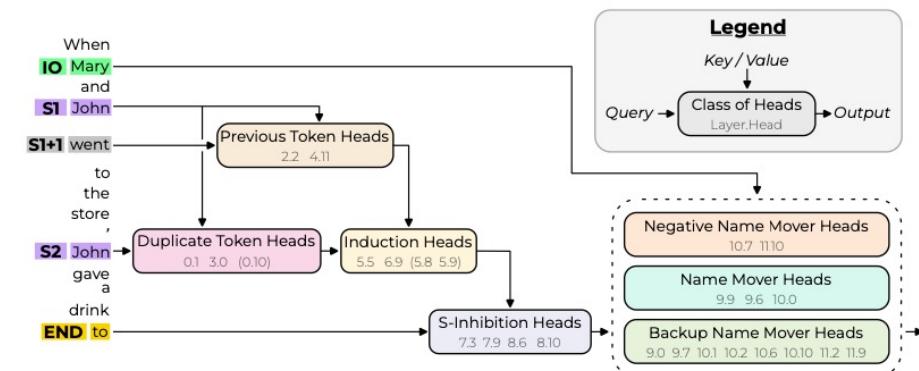
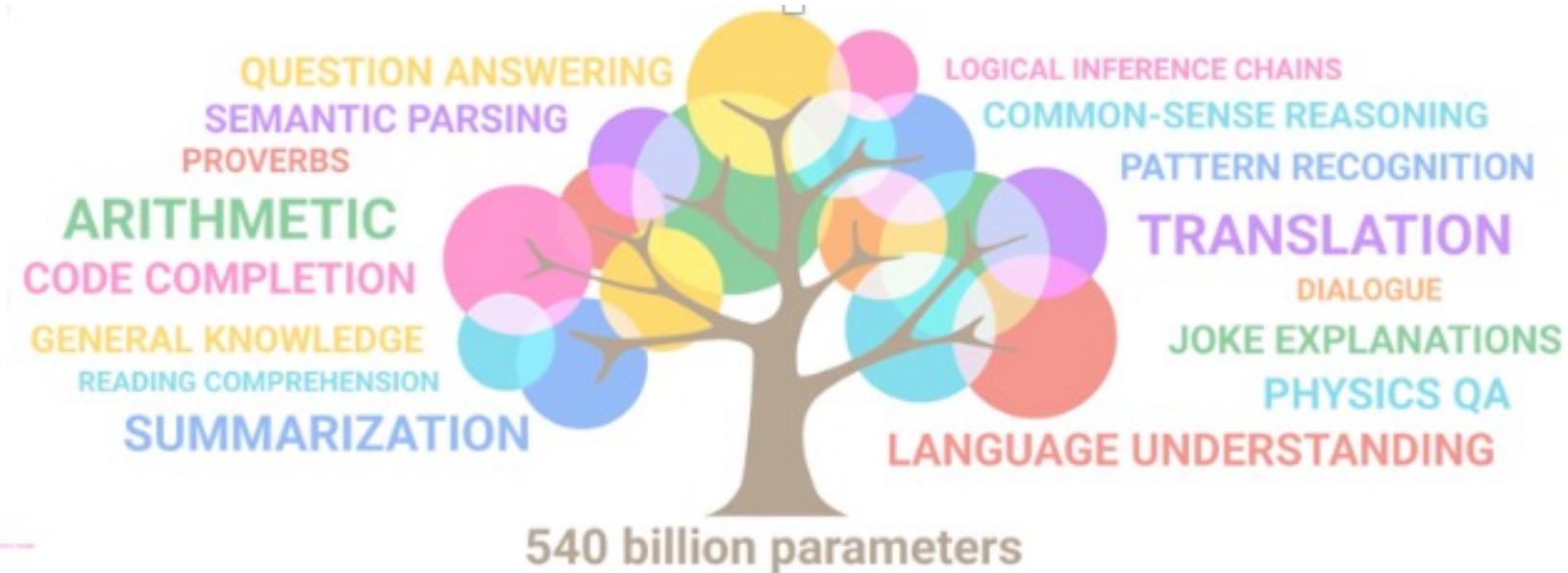
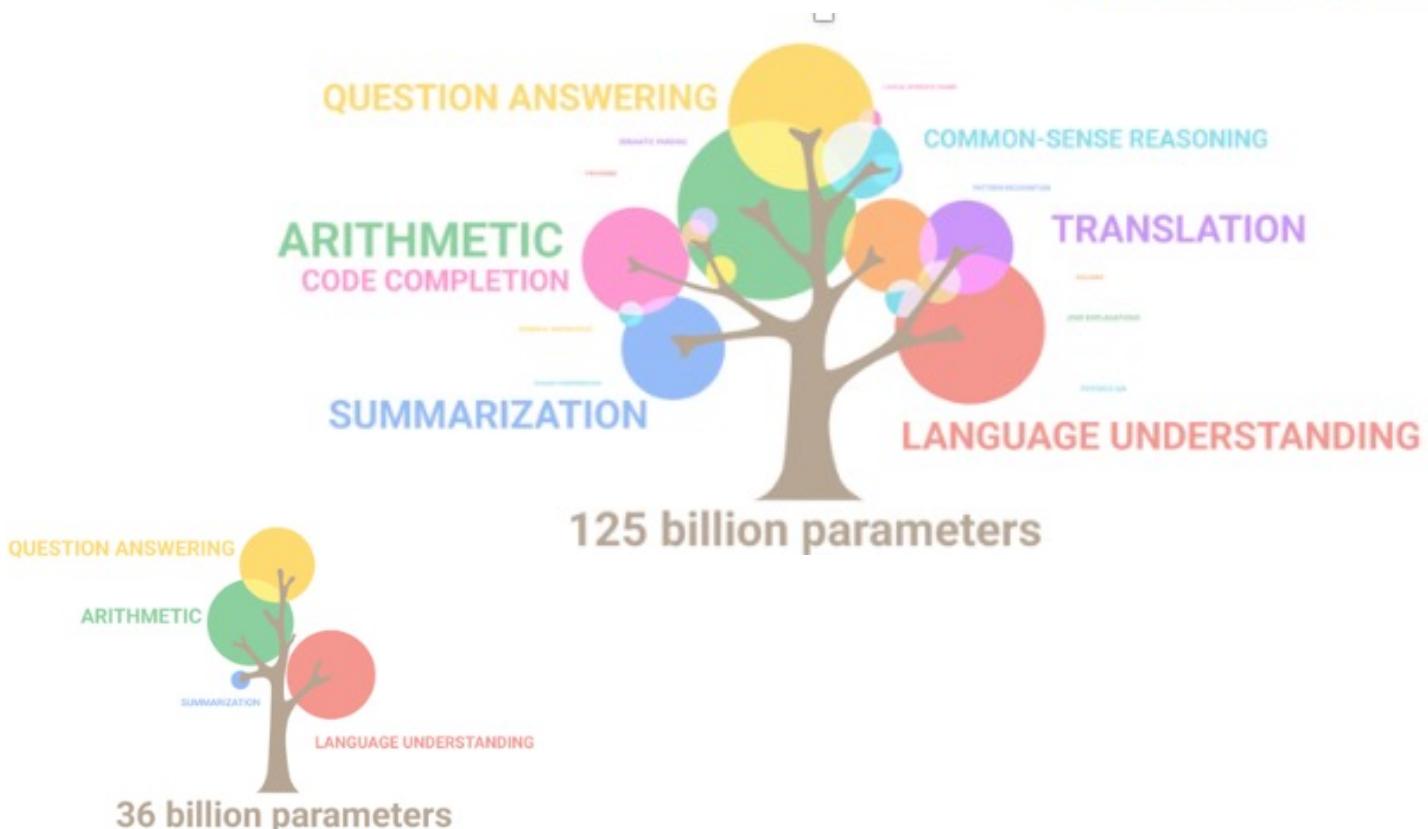


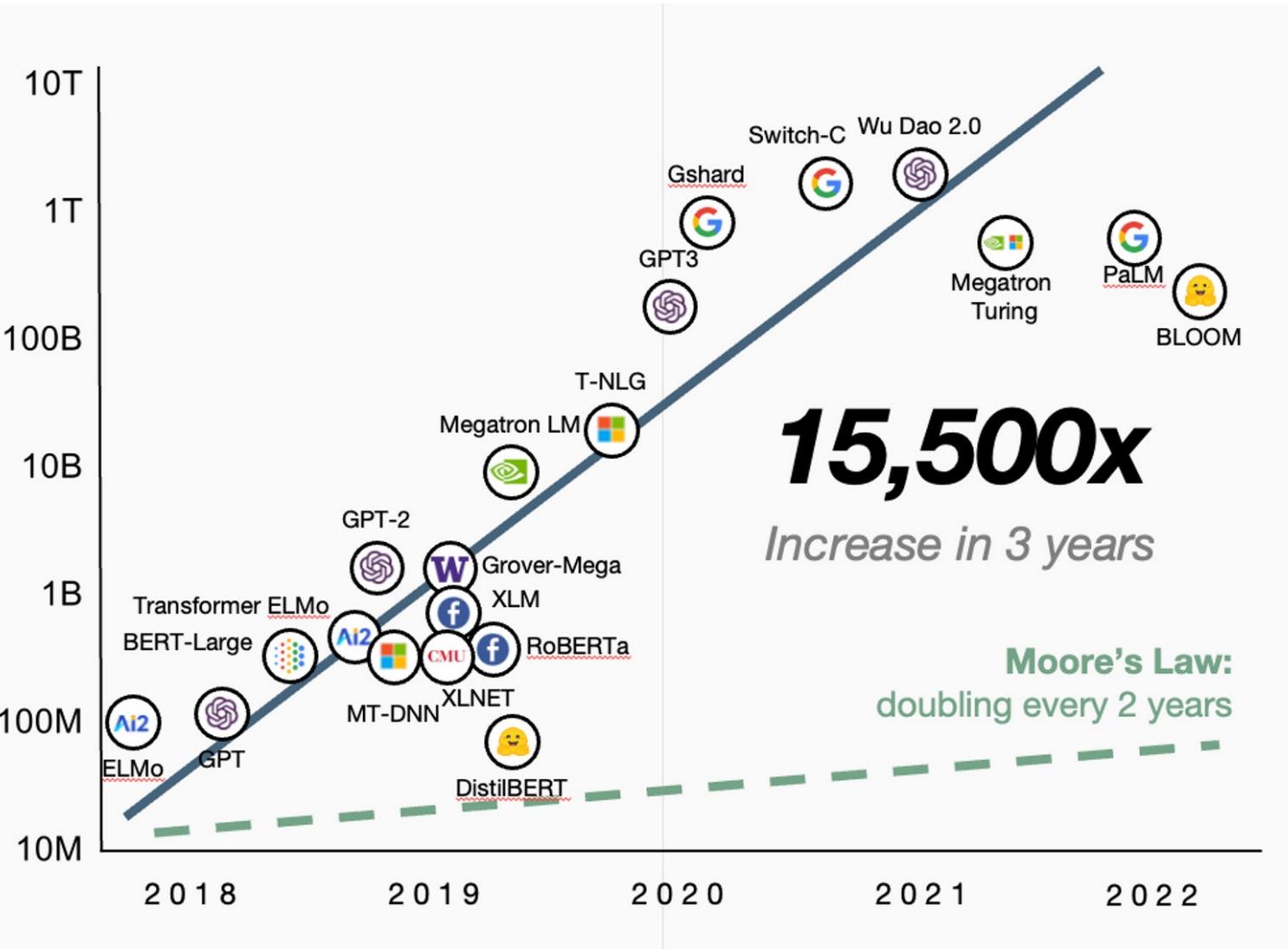
Figure 2: We discover a circuit in GPT-2 small that implements IOI. The input tokens on the left are passed into the residual stream. Attention heads move information between residual streams: the query and output arrows show which residual streams they write to, and the key/value arrows show which residual streams they read from.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., & Steinhardt, J. (2022). Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

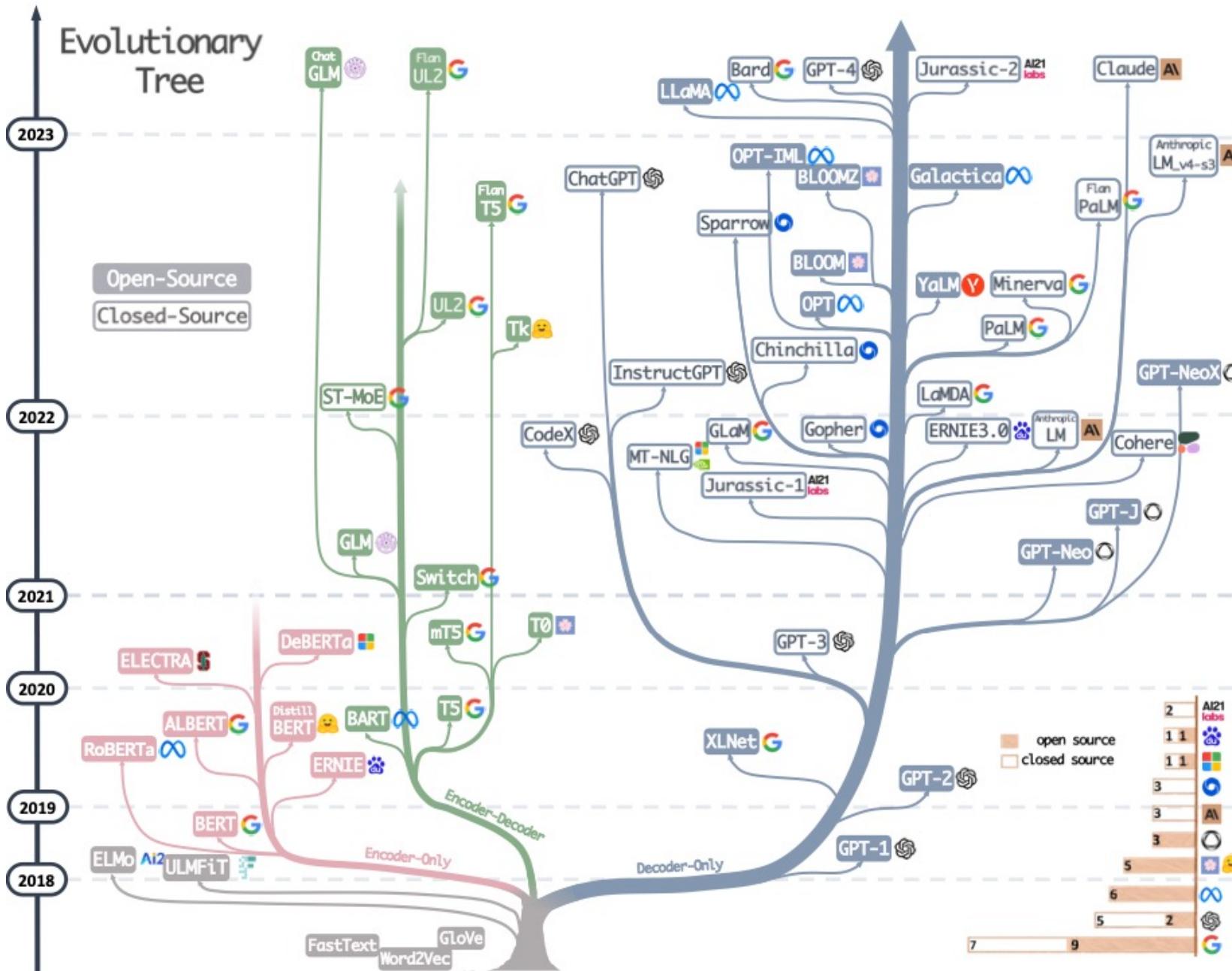
Эмерджентные свойства LLM



36 billion parameters

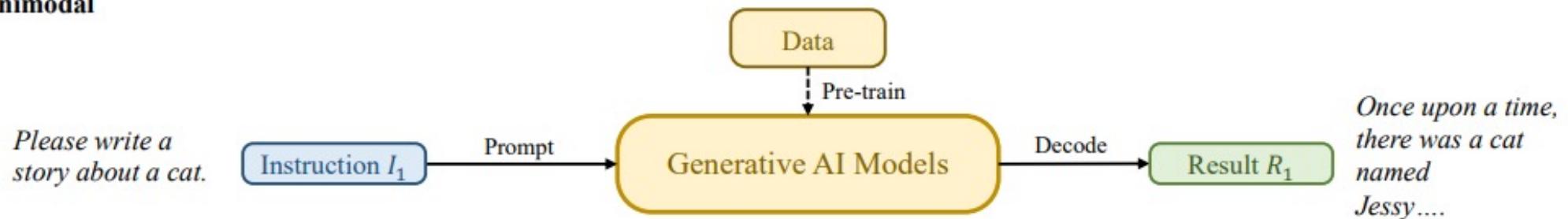


Усложнение
моделей

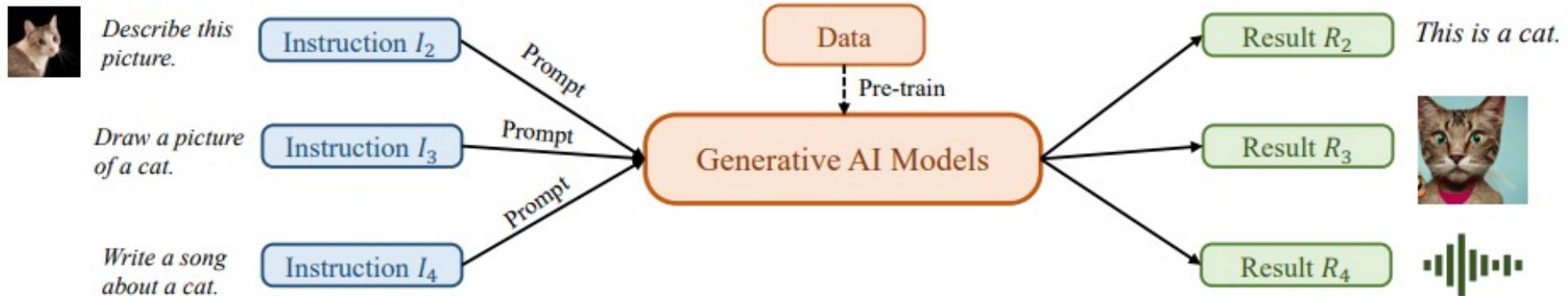


Мультимодальность

Unimodal



Multimodal



А что у нас?

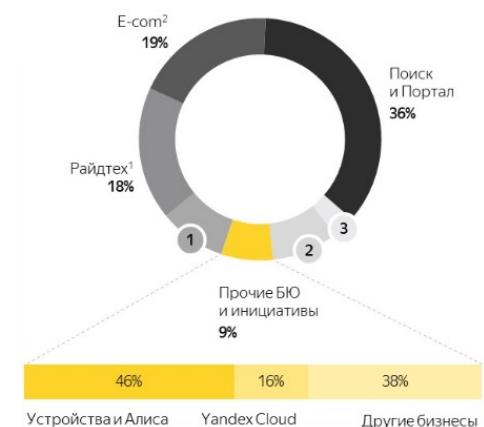
\$10M

\$100M

	GPT-1 (2018)	GPT-2 (2019)	GPT-3 (2020)	GPT-4 (2023)
Кол-во параметров	117M	1.5B	175B	1.76 T
Кол-во слоев	12	48	96	
Размер контекста	512	1024	2048	~ 32 000
Размерность вектора эмбеддинга	768	1 600	12 288	
n_{heads}	12 ($d_{\text{head}} = 64$)	25 ($d_{\text{head}} = 64$)	96 ($d_{\text{head}} = 128$)	

Кол-во параметров
Кол-во слоев
Размер контекста
Размерность вектора эмбеддинга
n_{heads}

YaGPT-2 (2023)
1.3B
24
8 000
12 800
24



Кейсы LLM: Yandex

Support

Подсказки

Переформулирование
ответа

Суммаризация истории
обращений

Оценка сервиса

Классификация

Тренажер оператора

HR

Анализ CV кандидатов

Персонализированные
коммуникации с
кандидатом

Мониторинг
производительности

HR self-service

Sales

Чат-бот / FAQ на сайте

Индивидуальные
письма от сейлзов

Вычитка / редактура

Автоматизация: follow-
ups, заполнение CRM

Аналитика продаж

Marketing

Персонализированный
маркетинг

Статьи, новости, посты в
блоге

Рекламные креативы

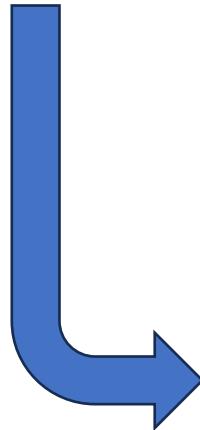
Аналитика упоминаний

Общение в SM

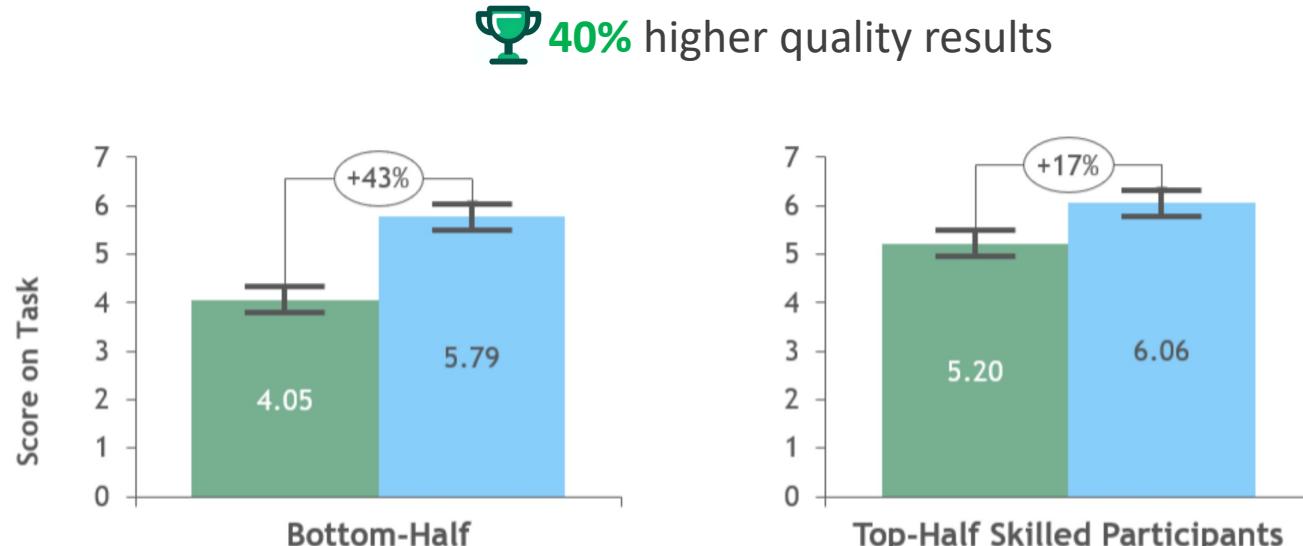
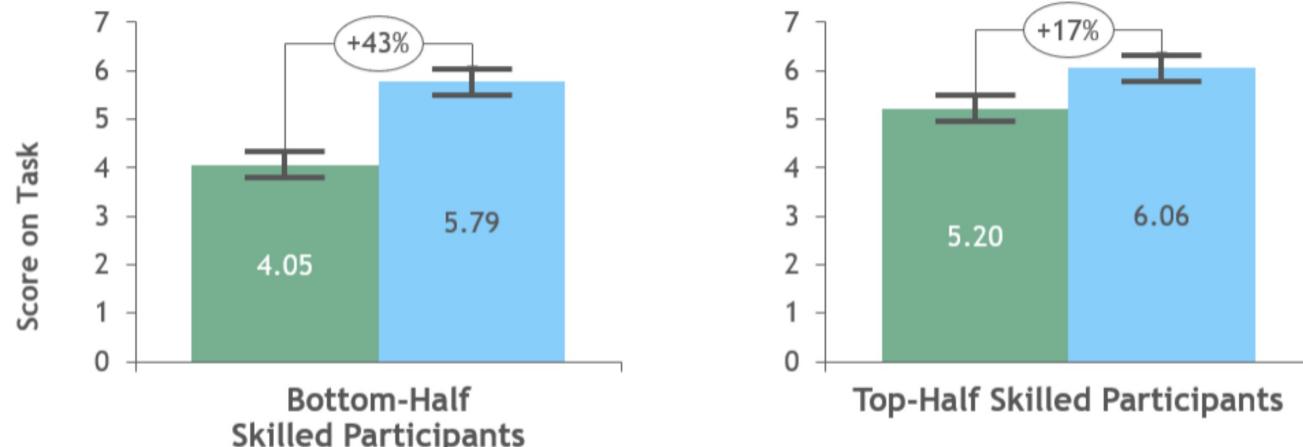
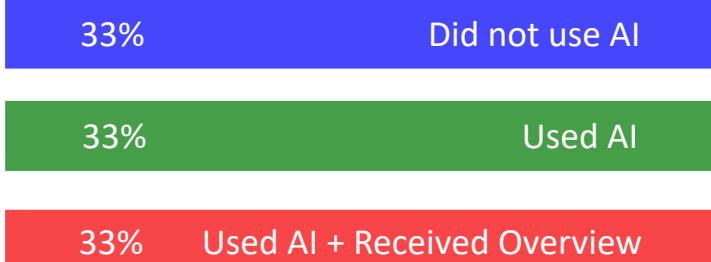
Кейс в консалтинге



- ❖ creative tasks ❖ writing tasks
- ❖ analytical tasks ❖ marketing tasks
- ❖ persuasiveness tasks

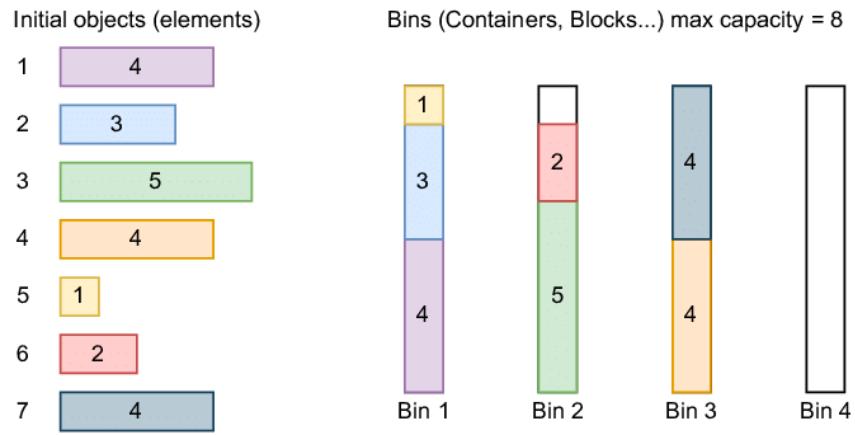


750+ consultants



Кейс в науке

Задача об упаковке рюкзака



эвристики

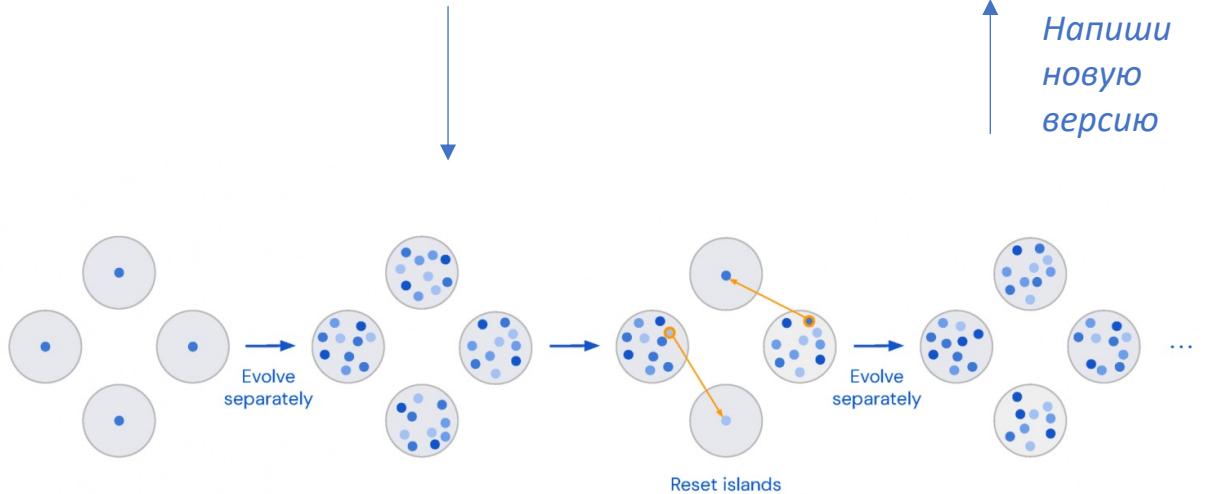
python code

упаковка

эвристика

Это пишет LLM

Напиши
новую
версию



	OR1	OR2	OR3	OR4	Weibull 5k	Weibull 10k	Weibull 100k
First Fit	6.42%	6.45%	5.74%	5.23%	4.23%	4.20%	4.00%
Best Fit	5.81%	6.06%	5.37%	4.94%	3.98%	3.90%	3.79%
FunSearch	5.30%	4.19%	3.11%	2.47%	0.68%	0.32%	0.03%

Table 1: Fraction of excess bins (lower is better) for various bin packing heuristics on the OR and Weibull datasets. *FunSearch* outperforms first fit and best fit across problems and instance sizes.

Вопросы?

Generative AI is a hammer, and no one knows what is and isn't a nail.

