Глубинное обучение

Лекция 8: Transformer-архитектура

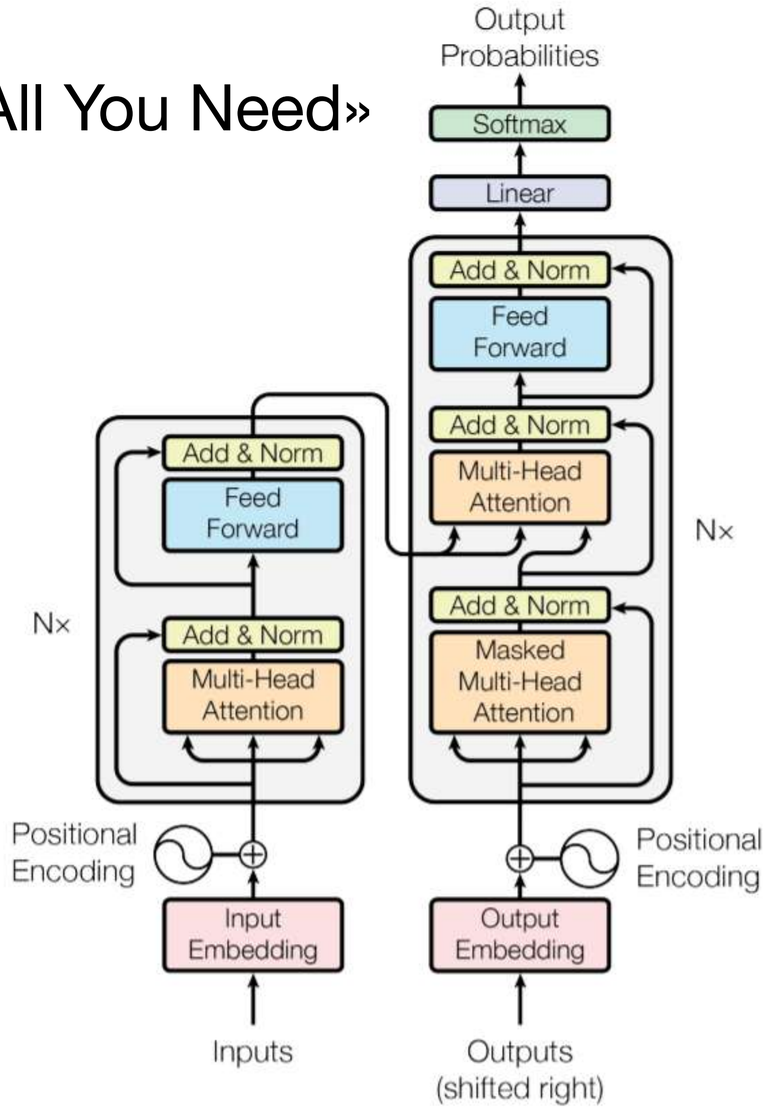
Лектор: Максим Рябинин
Исследователь, Yandex Research

Программа ML Residency: yandex.ru/yaintern/research_ml_residency

ФКН ВШЭ, 2022

Трансформер

- Статья «Attention Is All You Need»



[Vaswani et al., 2017]

Transformer

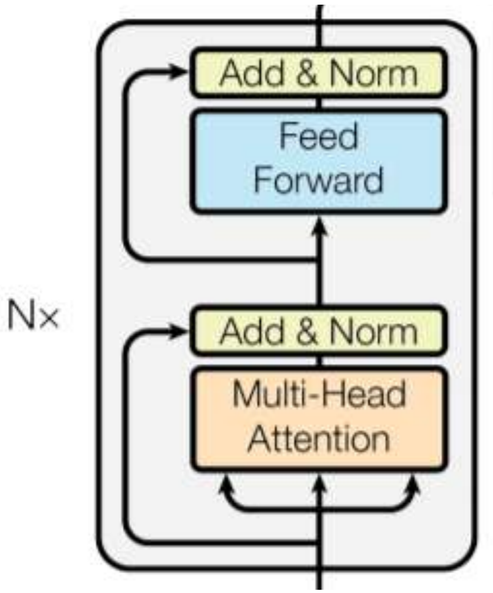
- Статья «Attention Is All You Need»
- Основной блок трансформера:
  - Self attention (Q = query, K = key, V = value, d_k = dimension)

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

  - Multi-head self-attention (N = num layers, h = num heads)

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

  - 2-layer NN with ReLU (сейчас чаще другие активации)
  - LayerNorm(x + Sublayer(x))



[Vaswani et al., 2017]

Transformer

- Статья «Attention Is All You Need»
- Основной блок трансформера
- Кодирование входа:
  - Представление токенов (обычные представления)
  - Представление позиции:
    - Словарь представлений позиций
    - Частотное кодирование (d_{model}=512):

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

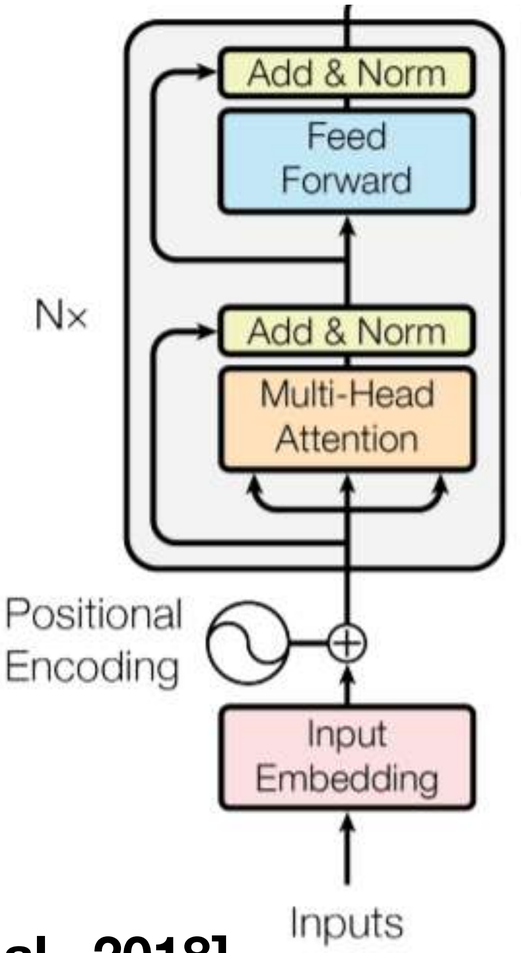
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

    - Relative Position Encoding – встраивается в self-attention

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}} \quad z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V)$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}}$$

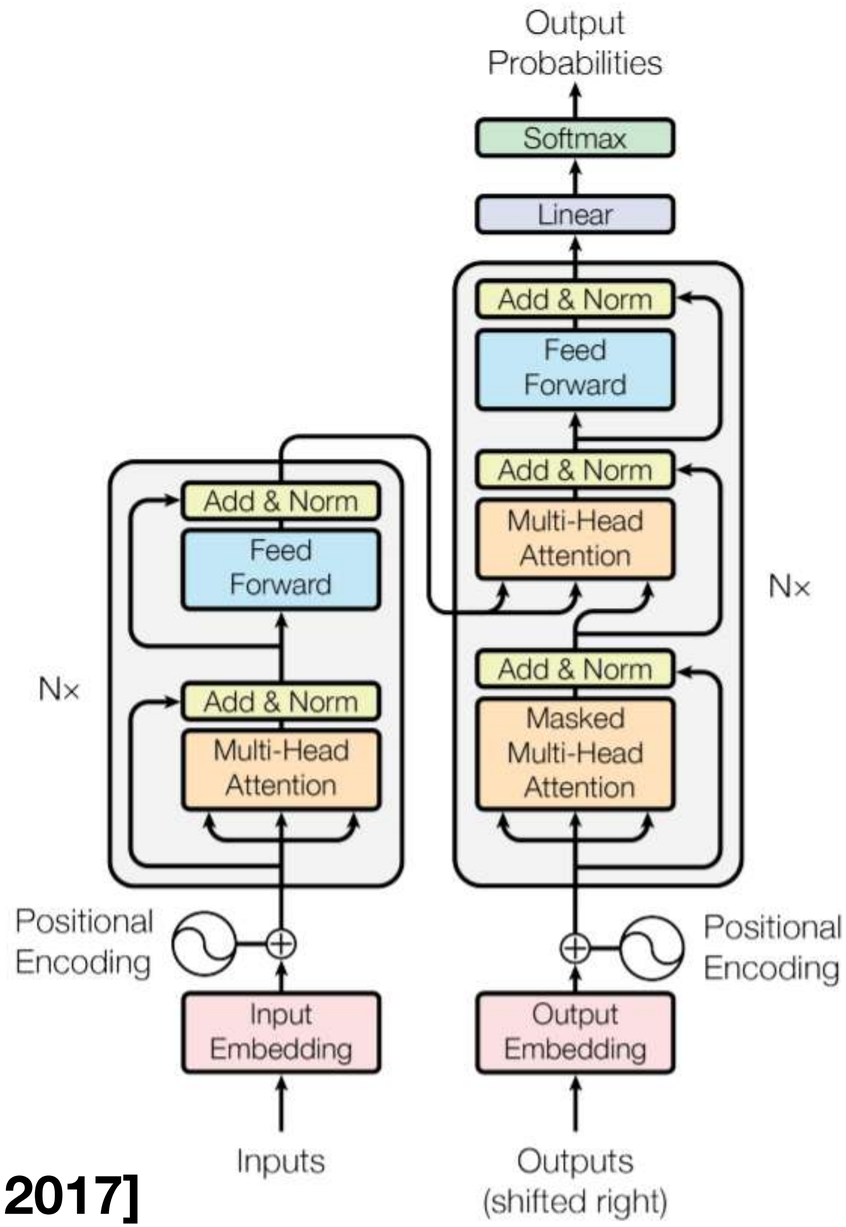
$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V)$$



[Shaw et al., 2018]

Transformer

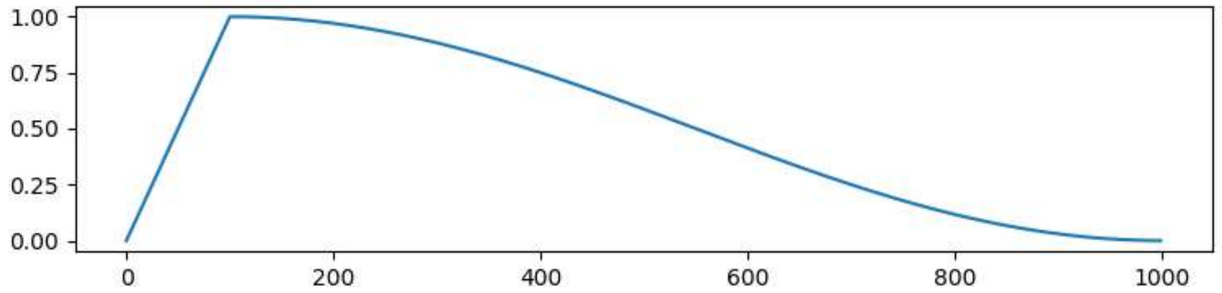
- Статья «Attention Is All You Need»
- Основной блок трансформера
- Кодирование входа
  - Представление входа и номера позиции
- Decoder трансформера
  - Авторегрессионное предсказание
  - Masked self-attention и attention на энкодер
  - На обучении маски в self-attention

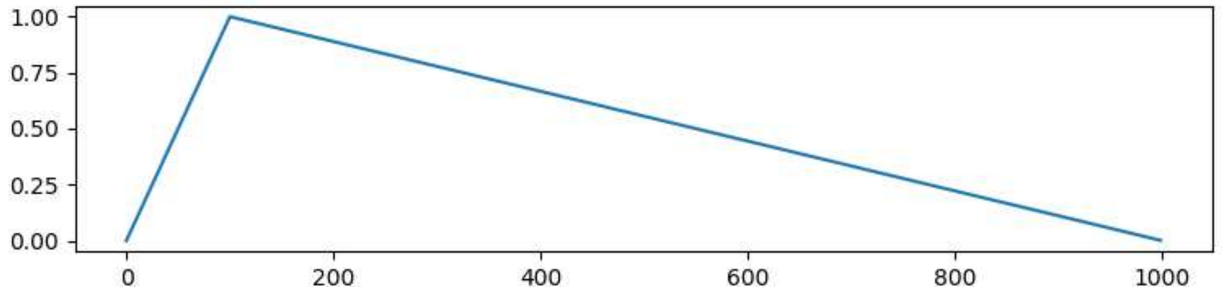


[Vaswani et al., 2017]

Обучение трансформеров

- Adam с расписанием размера шага
  - Линейный разогрев
  - Далее затухание





- Достаточно большой размер батча
  - Несколько GPU
  - Последовательное вычисление подбатчей (model parallelism/gradient accumulation)

- Label smoothing
  - Log loss с таргетом (1-α, $\frac{\alpha}{d-1}$,…, $\frac{\alpha}{d-1}$) вместо (1, 0, …, 0)

image credit:
huggingface.co

Предобученные представления слов из языковых моделей

Зачем нужны предобученные модели?

- Чтобы решать любую задачу!
- Способ по умолчанию: дообучаем на данные задачи
- Сравнение моделей:
  - Дообучать на набор разных задач и брать агрегированный результат (среднее ☹)
  - Бенчмарки: GLUE, SuperGLUE
  - Статические наборы данных приводят к переобучению сообщества и застою
  - Динамические во времени бенчмарки: GEM , DynaBench

- На разных задачах нужны разные модели
- Отличная библиотека:
  - https://github.com/huggingface/transformers

Виды языковых моделей

- Только encoder (ELMo, BERT, RoBERTa, etc.)
- Encoder-decoder (T5, BART, etc.)
- Только decoder – генеративные модели
  (GPT-X, Turing-NLG, Megatron LM)

BERT

- BERT = Bidirectional Encoder Representations from Transformers

- Очень большой трансформер:
  - L – глубина, H – размерность внутри, A – число голов multi-head attention
  - $BERT_{BASE}$: L=12, H=768, A=12, Total Parameters=110M
  - $BERT_{LARGE}$: L=24, H=1024, A=16, Total Parameters=340M

- Обучение:
  - Masked LM: random 15% output tokens
  - Inputs:
    - 80% - [MASK]
    - 10% - исходное слово
    - 10% - случайное слово
  - Next sentence prediction

Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
Label = IsNext
Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

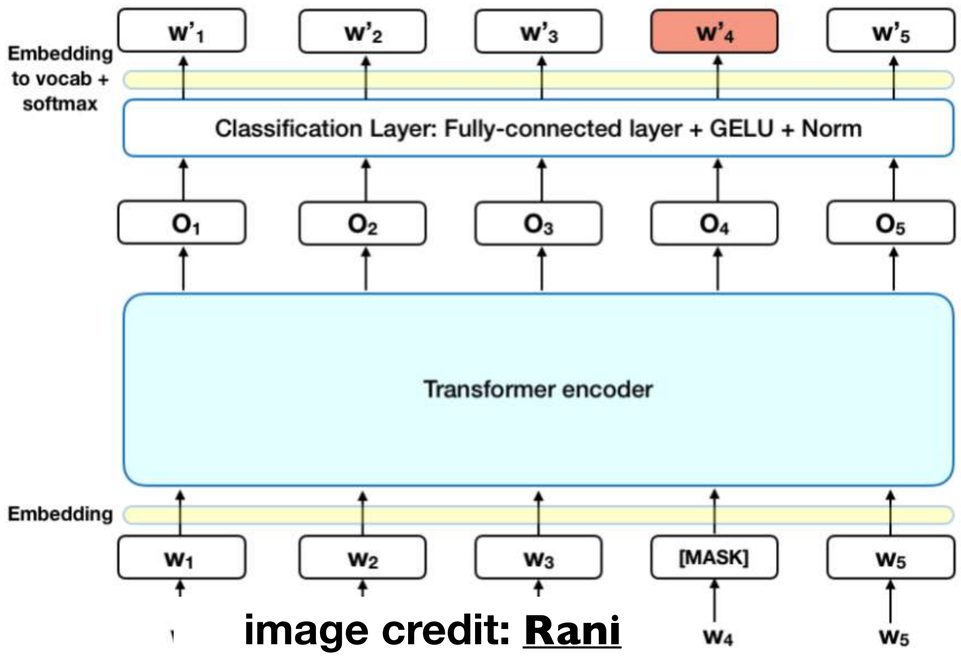


image credit: **Rani Horev**

[Devlin et al., 2018]

RoBERTa – как BERT, но лучше

- BERT был существенно недообучен
  - RoBERTa: подбираем гиперпараметры и обучаем дольше
  - Большой батч
  - byte-level (а не character-level, unicode!) BPE
  - генерация масок на лету

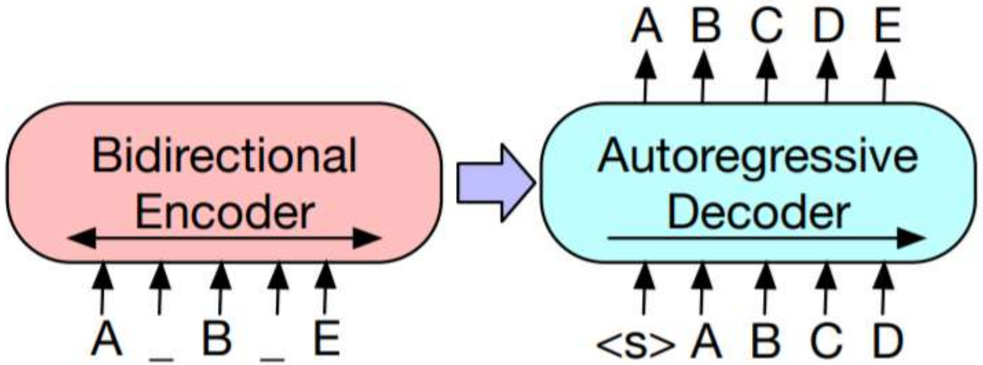
[Liu et al., 2019]

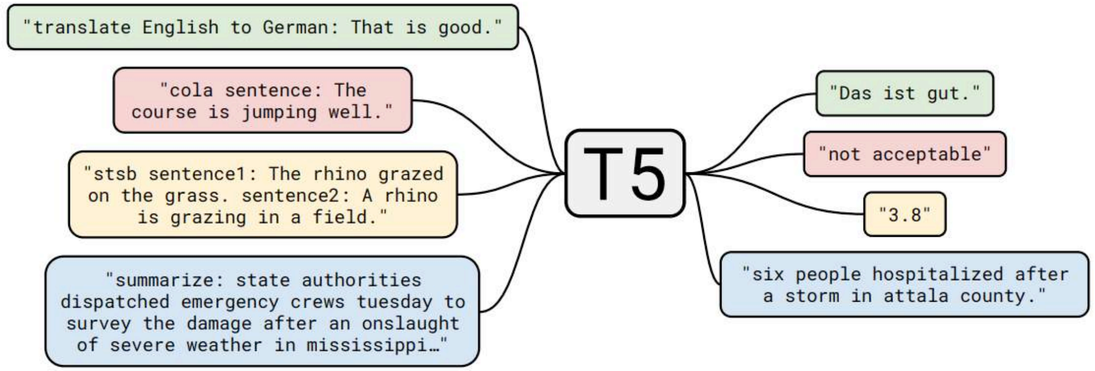
Модели меньшего размера

- ALBERT [Lan et al., 2019]: представления меньшей размерности, одни и те же параметры на разных слоях
- DistilBERT [Sanh et al., 2019]: дистилляция BERT в меньшую модель
- Chinchilla [Hoffmann et al., 2022]: обучаем меньшие модели дольше, получаем результаты сравнимые или лучше аналогов 2x размера

Модели encoder-decoder

- BART: Bidirectional and Auto-Regressive Transformers [Lewis et al.; 2019]
  - Обучается на восстановлении шума
  - Шум со структурой





- T5: Text-to-Text Transfer Transformer [Raffel et al.; 2019]
  - Попробовали много разных вариантов BERT-like функций потерь
  - Формулируем любую задачу как text-to-text
  - Очень много данных

Только decoder – генеративные модели

- Популяризованы семейством моделей GPT, GPT-2, GPT-3 (OpenAI)
  - Внедрялись архитектурные трюки
  - Больше данных
  - Больше размер моделей
  - Больше вычислений
  - GPT-3 – 175B параметров, обучение стоит миллионы $
  - GPT-3 не выложена, используется в продуктах и API

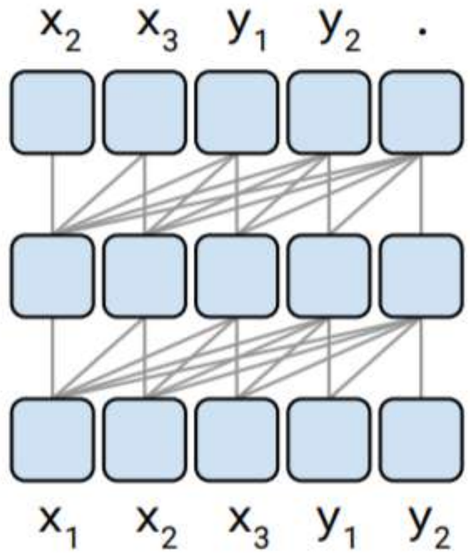


image credit: [Raffel et al.; 2019]

Mega Language Models

- Текущий чемпион (из публично известных):
  - PaLM (Google)
  - 540 миллиардов параметров
  - Hardware and software

- Модели перестают быть для всех.
  Теперь это IP компаний (но не всегда, см. OPT)

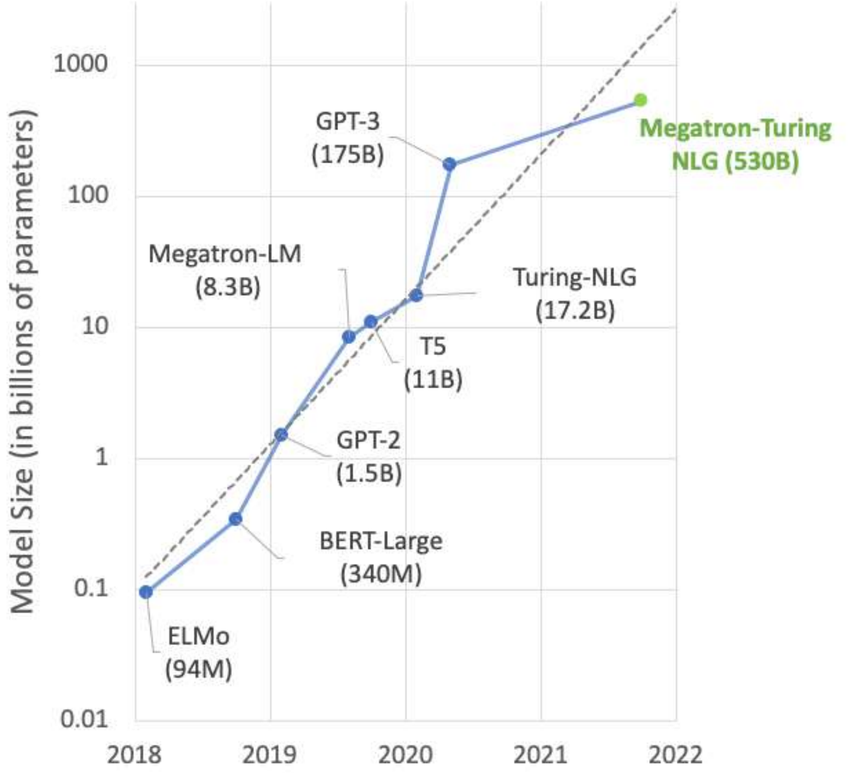


image credit: **NVIDIA blog**

GPT-3: использование без дообучения

- GPT «понимает» задачу по нескольким примерам
- Prompt engineering как новая парадигма

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.
Poor English input: I'd be more than happy to work with you in another project.
Good English output: I'd be more than happy to work with you on another project.

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.
Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.

Poor English input: The patient was died.
Good English output: The patient died.

[Brown et al., 2020]

GPT-3: использование без дообучения

- GPT «понимает» задачу по нескольким примерам
- Prompt engineering как новая парадигма

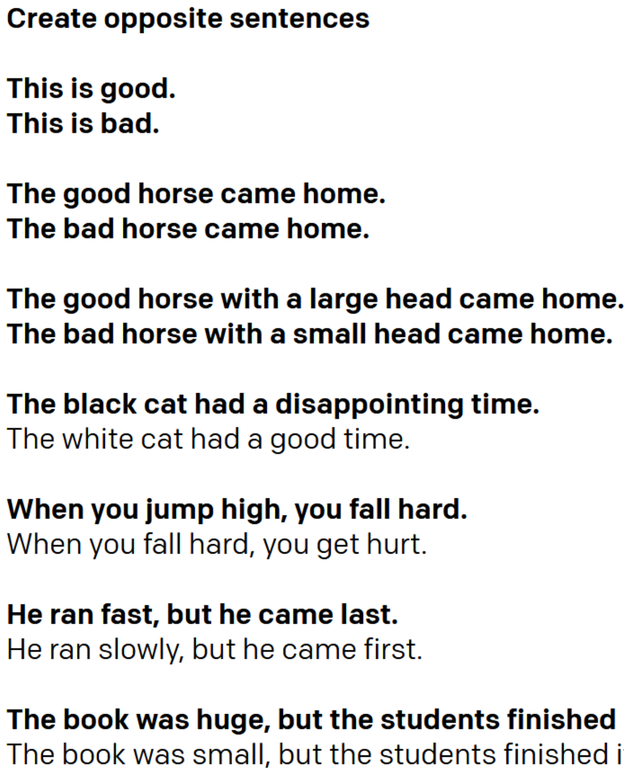
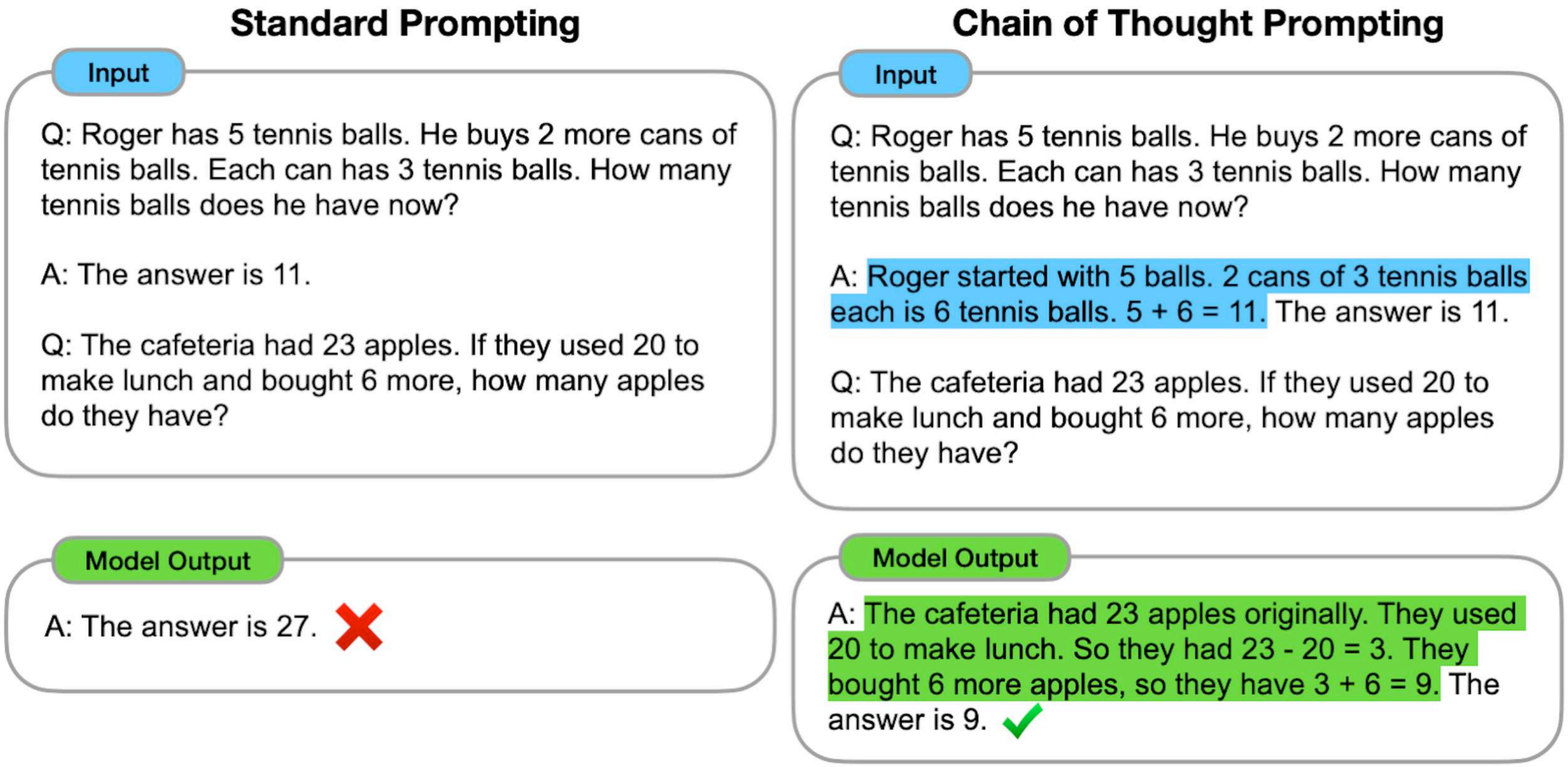


image credit and more examples:
gpt3experiments.substack.com/p/is-gpt-3-really-doing-few-shot-learning

[Brown et al., 2020]

Chain-of-thought prompting

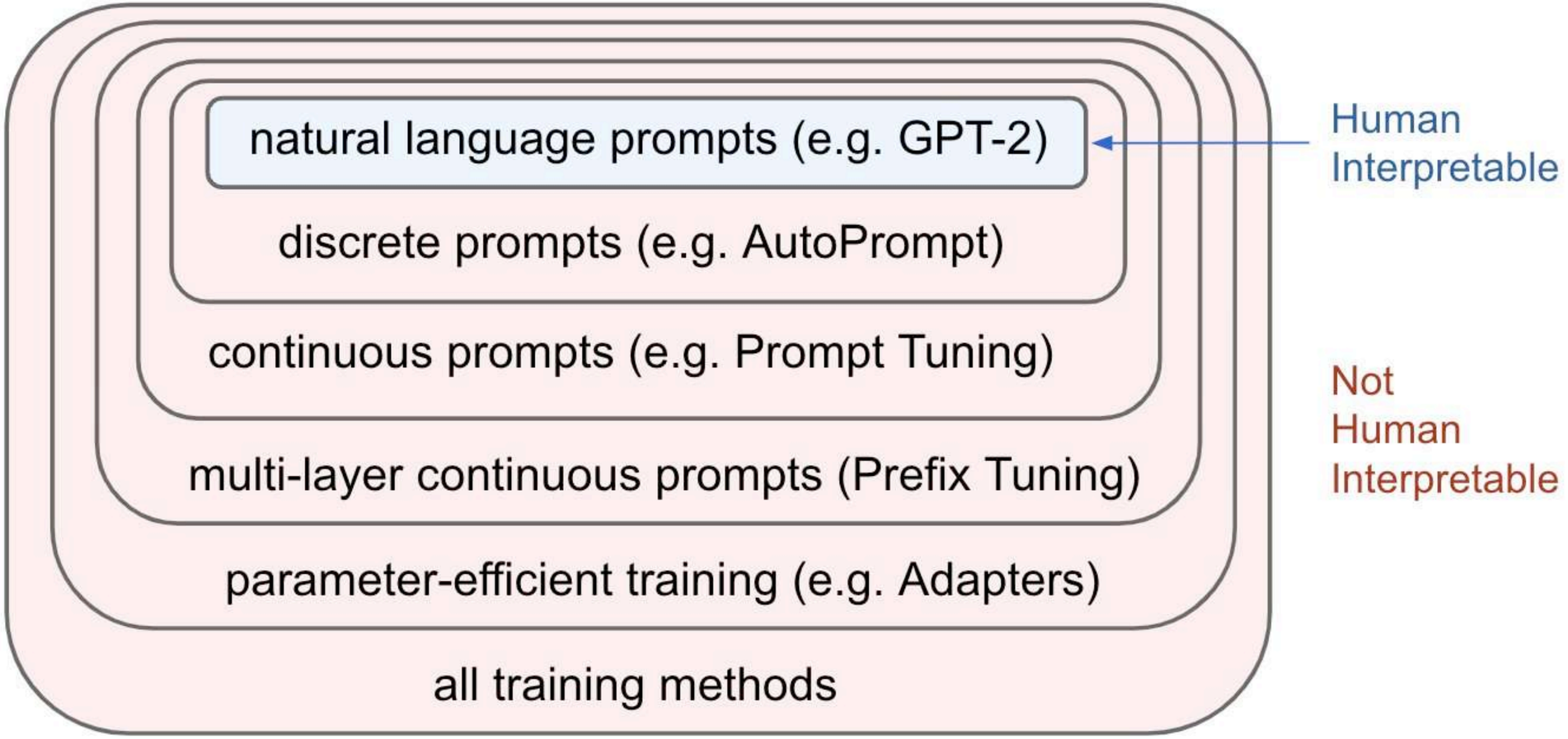


ai.googleblog.com/2022/05/language-models-perform-reasoning-via.html

A Taxonomy of Prompting Methods

By Graham Neubig (10/15/2022)

See CMU ANLP Prompting Lecture, A Unified View of Parameter-Efficient Transfer Learning



GPT-2: https://openai.com/blog/better-language-models/

AutoPrompt: https://arxiv.org/abs/2010.15980

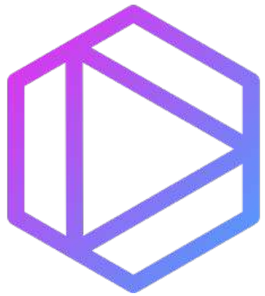
Prompt Tuning: https://arxiv.org/abs/2104.08691

Prefix Tuning: https://arxiv.org/abs/2101.00190

Adapters: https://arxiv.org/abs/2010.15980

Генеративные модели для кода

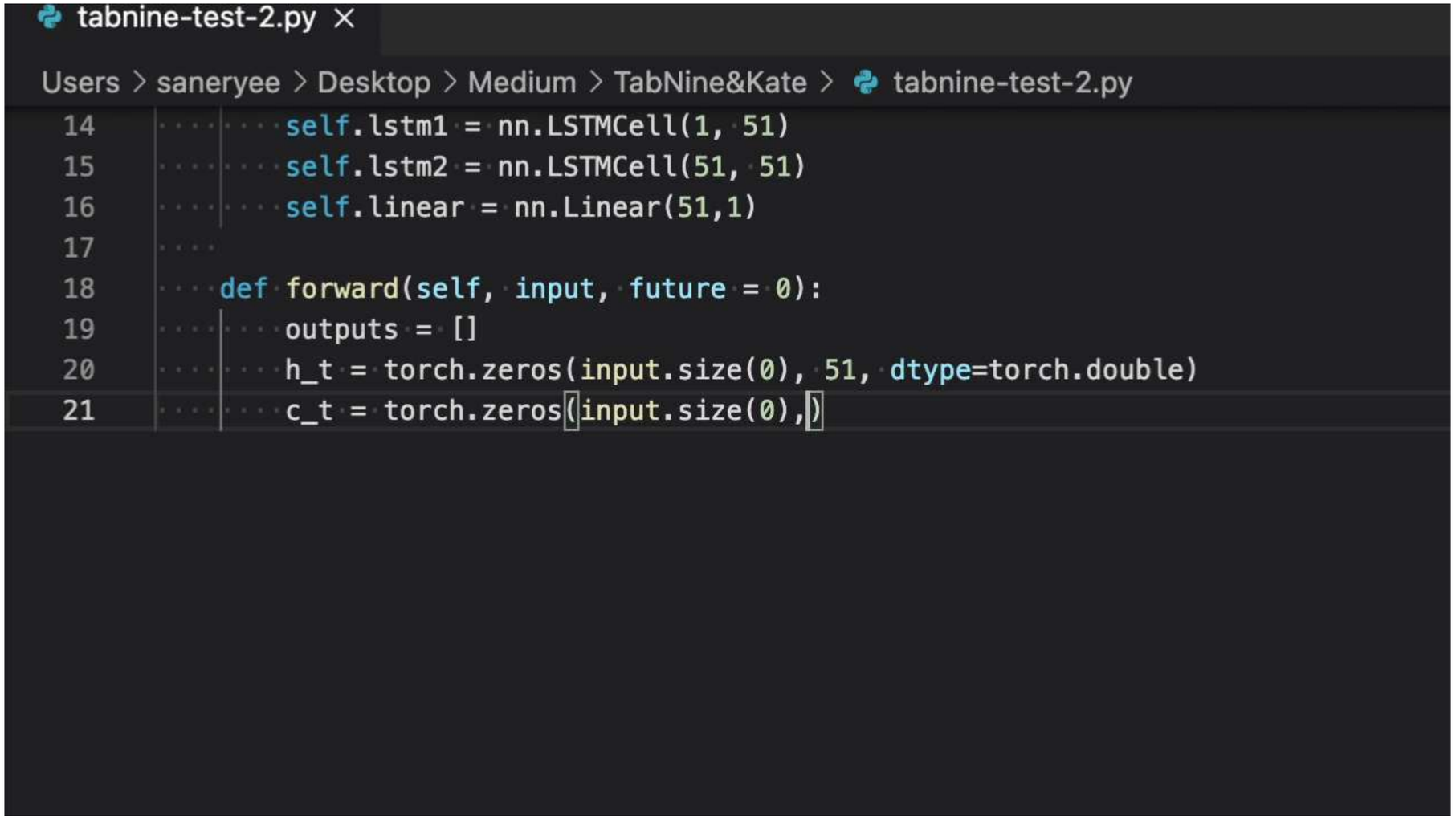
- TabNine



- GitHub Copilot / Open AI Codex [Chen et al., 2021]

- JetBrains Full Line Code Completion

TabNine



https://medium.com/swlh/kite-vs-tabnine-which-ai-code-autocomplete-should-you-choose-eb6eba85c3a6

GitHub Copilot / Open AI Codex

```python
import torch
from torchvision.datasets import ImageFolder
from torchvision import transforms,models
from torch import nn,optim

def finetune(folder, model):
    """fine tune pytorch model using images from folder and report results on validation set"""

    # build model
    model = getattr(models,model)()
    model.load_state_dict(torch.load(model))
    model = model.to(device)

    # load data
    train_set = ImageFolder(folder,'train',transform=data_transforms['train'])
    val_set = ImageFolder(folder,'val',transform=data_transforms['val'])
    train_loader = torch.utils.data.DataLoader(train_set,batch_size=4,shuffle=True,num_workers=2)
    val_loader = torch.utils.data.DataLoader(val_set,batch_size=4,shuffle=True,num_workers=2)
```

File context

Function description

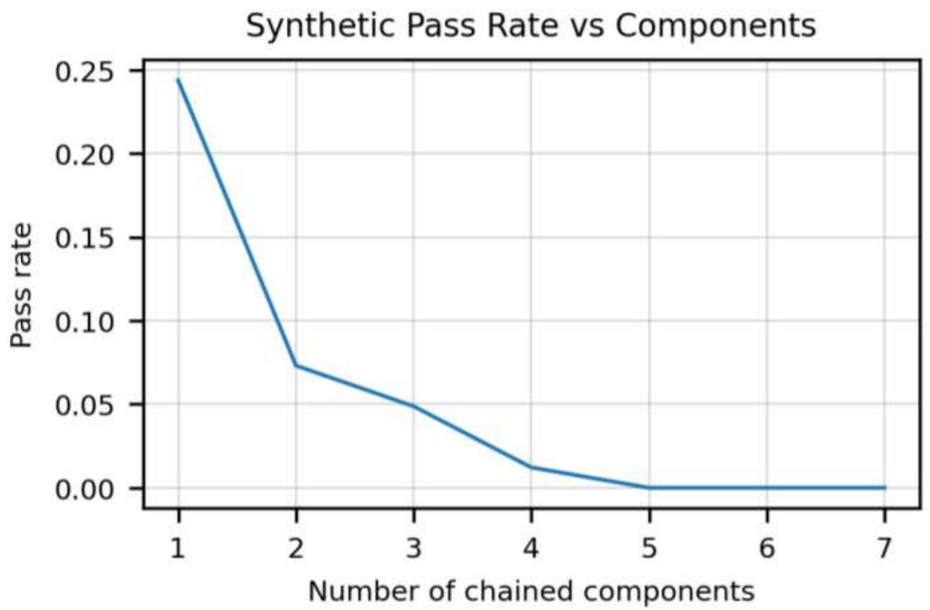
Generated code (89 lines)

Example from https://www.fast.ai/2021/07/19/copilot/

Codex работает?

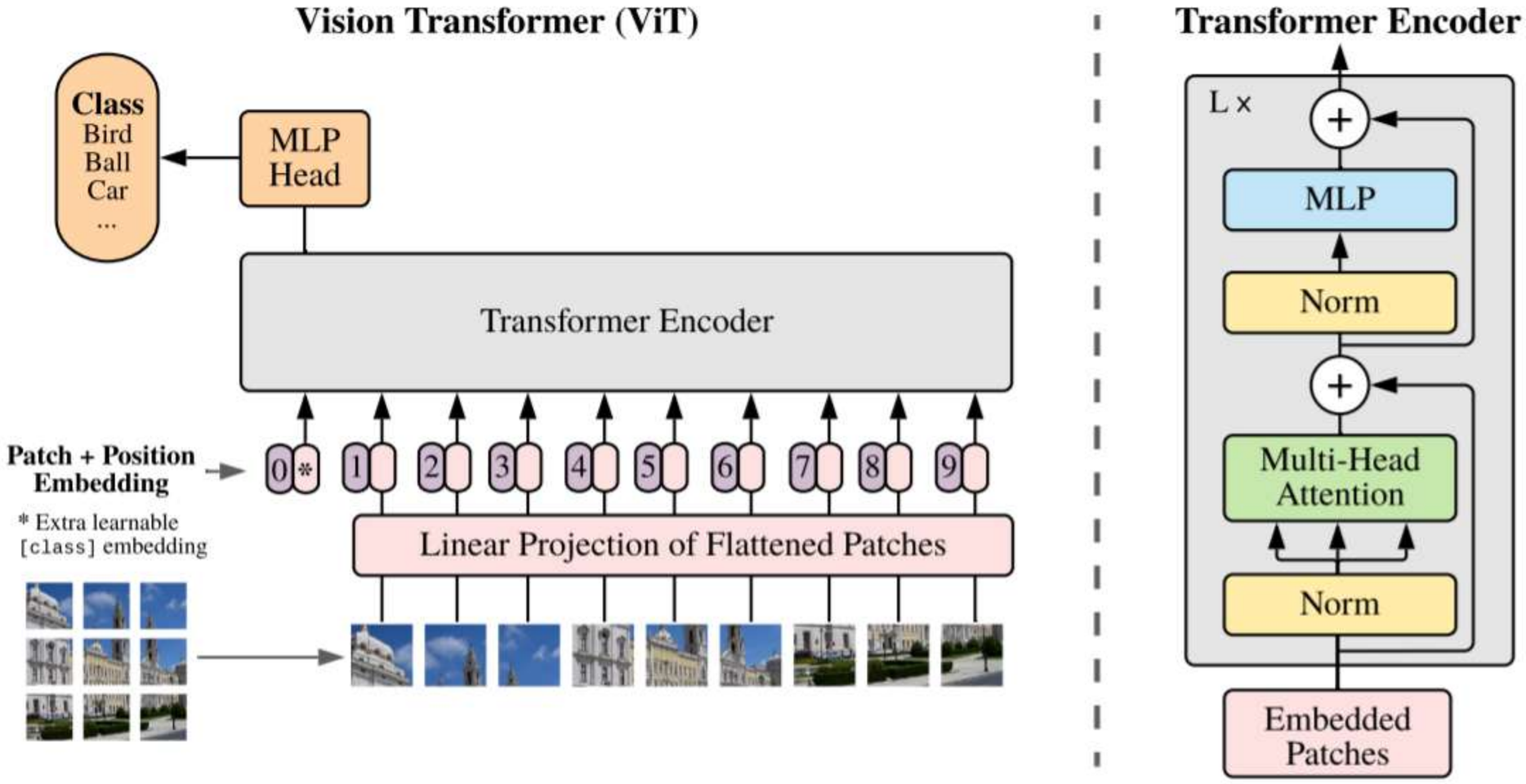
- Генерирует
- Результат похож на код и иногда работает
- Нет гарантий, могут быть сложные баги

```
def do_work(x, y, z, w):
    """ Add 3 to y, then subtract 4
    from both x and w. Return the
    product of the four numbers. """
    t = y + 3
    u = x - 4
    v = z * w
    return v
```



Figures from Chen et al. [2021; arXiv:2107.03374]

Трансформеры для изображений: ViT



[Dosovitskiy et al., 2020]

Трансформеры для изображений

- Горячая тема
- Пытаются применять во всех задачах, конкуренты сверток
  - Классификация: Visual Transformer [Wu et al., 2020], DEIT [Touvron et al., 2021]
  - Детекторы: DETR [Carion et al, 2020]; Deformable DETR [Zhu et al., 2020]
  - Сегментация: DPT [Ranftl et al., 2021]; SWIN [Liu et al. 2021]
  - Поза: PoseFormer [Zheng et al., 2021]; TransPose [Yang et al., 2021]
  - Действия на видео: Video Transformer [Neimark et al., 2021]

Заключение

- Трансформеры оккупировали NLP
- Предобученные представления очень важны
- Отличная библиотека-“клей”: github.com/huggingface/transformers 

- Трансформеры приходят и в другие области
  - Обработка кода
  - Компьютерное зрение
  - Обучение с подкреплением