DALL-E и мультимодальный поиск

Сергей Николенко

СПбГУ — Санкт-Петербург

13 ноября 2025 г.

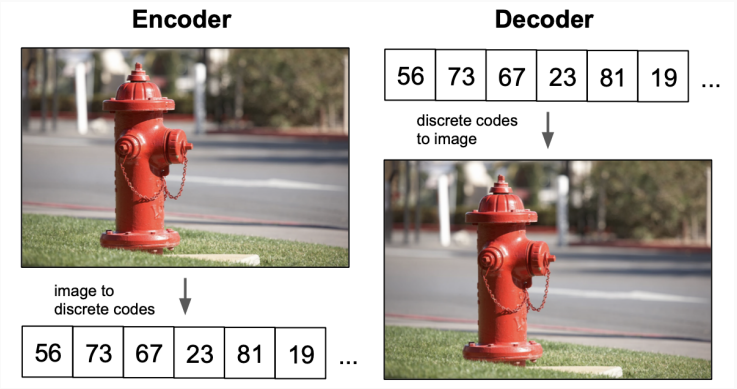
Random facts:

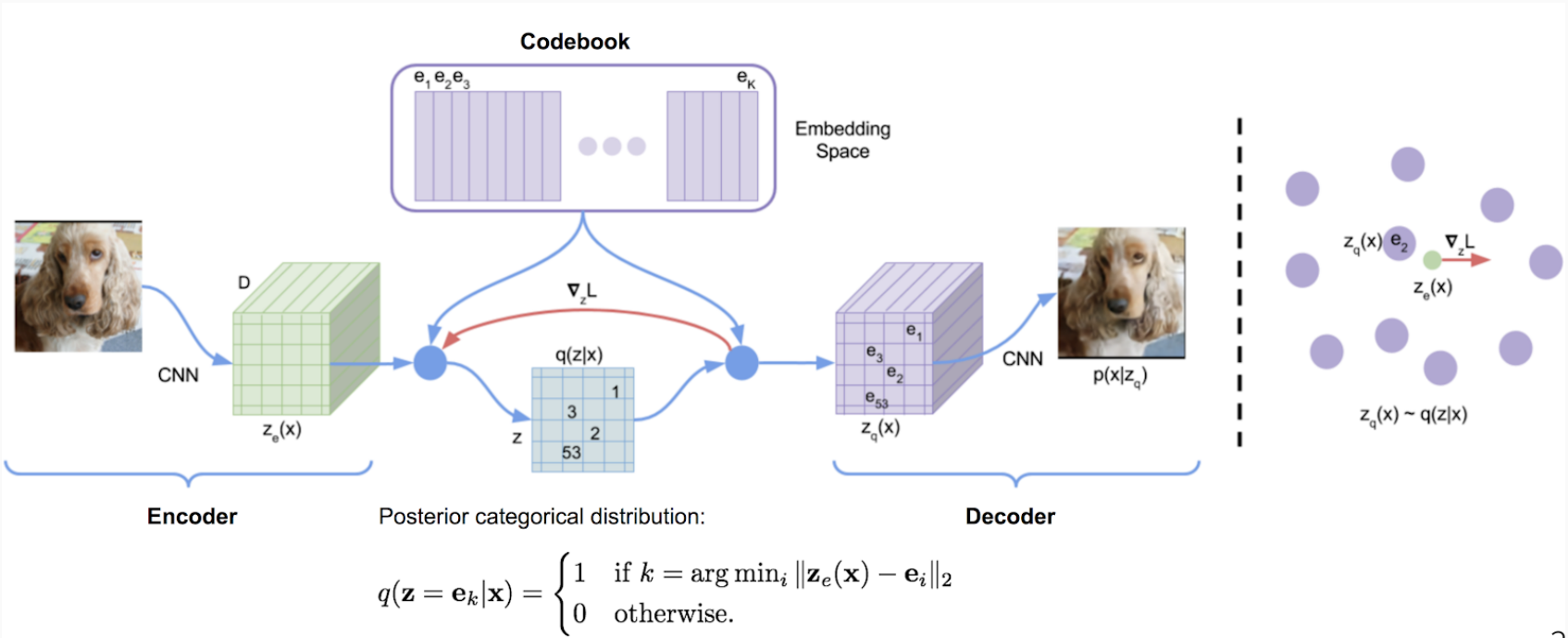
- 13 ноября 1002 г., в день Святого Брайса, по приказу короля Англии Этельреда II Неразумного были убиты практически все даны, жившие в Англии; эффект оказался логичным, но вряд ли ожидаемым: Свен Вилобородый приплыл в Англию с огромным флотом и жаждой мести и к 1013 году полностью господствовал в Англии
- 13 ноября 1841 г. Джеймс Брейд впервые посетил демонстрацию «животного магнетизма» Чарльза Лафонтена; Брейд отнёсся скептически, но впоследствии нашёл рациональное зерно и стал первым современным исследователем гипноза
- 13 ноября 1862 г. Чарльз Доджсон начал выполнять обещание, данное Алисе Лидделл; из дневника: «Начал писать сказку об Алисе, надеюсь закончить её к Рождеству»
- 13 ноября 1887 г. — «кровавое воскресенье», правда, не в России; 10000 человек вышли в Лондоне на марш против происходящего в Ирландии и были разогнаны, 75 получили тяжёлые травмы; пехота была на месте, но приказа стрелять так и не получила
- 13 ноября 1947 г. официально завершилась разработка автомата Калашникова (АК-47); к этому типу принадлежит около 1/5 всего автоматического стрелкового оружия в мире

DALL-E

dVAE + Transformer = DALL-E

- Мы уже обсуждали VQ-VAE:

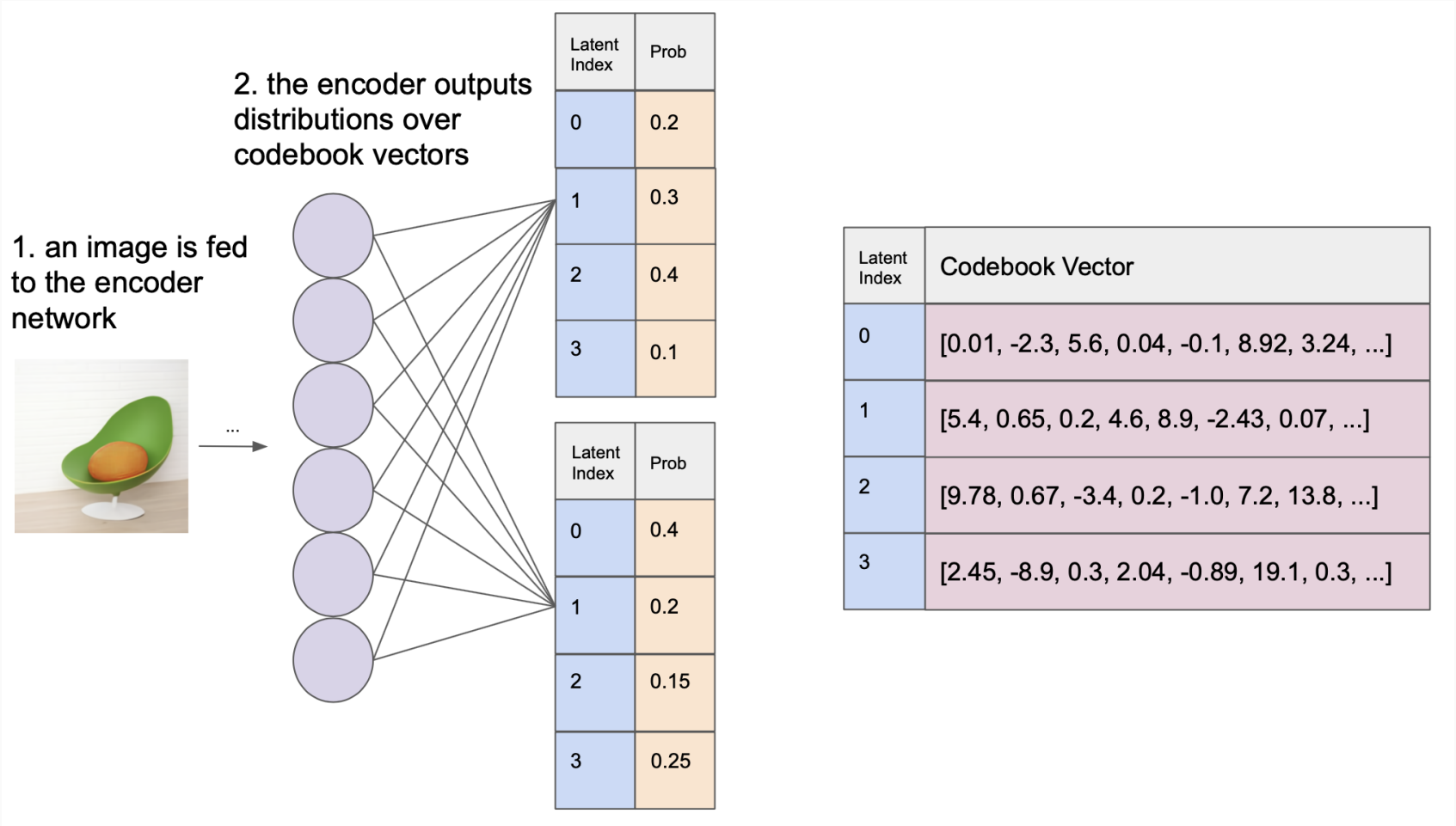




$$q(\mathbf{z} = \mathbf{e}_k|\mathbf{x}) = \begin{cases} 1 & \text{if } k = \arg\min_i \|\mathbf{z}_e(\mathbf{x}) - \mathbf{e}_i\|_2 \\ 0 & \text{otherwise.} \end{cases}$$

dVAE + Transformer = DALL-E

- dVAE – это VQ-VAE, который выдаёт не один вектор из словаря, а распределение на словаре:



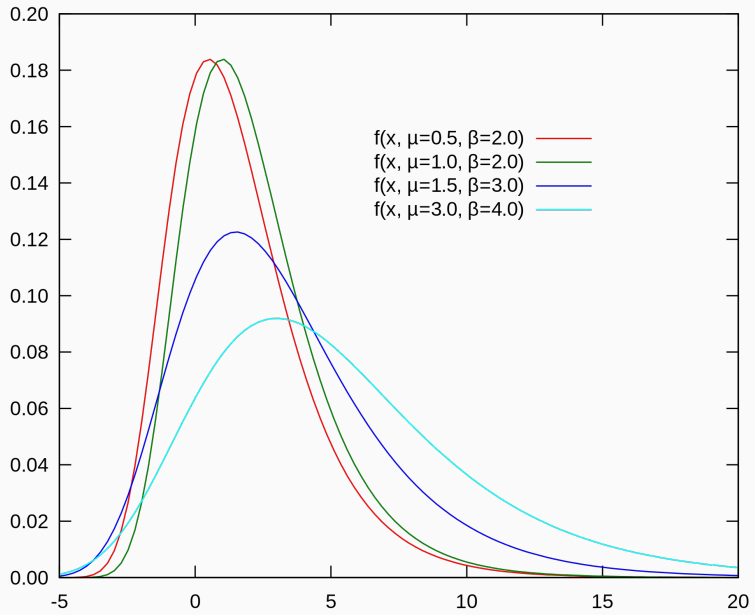
Latent Index	Codebook Vector
0	[0.01, -2.3, 5.6, 0.04, -0.1, 8.92, 3.24, ...]
1	[5.4, 0.65, 0.2, 4.6, 8.9, -2.43, 0.07, ...]
2	[9.78, 0.67, -3.4, 0.2, -1.0, 7.2, 13.8, ...]
3	[2.45, -8.9, 0.3, 2.04, -0.89, 19.1, 0.3, ...]

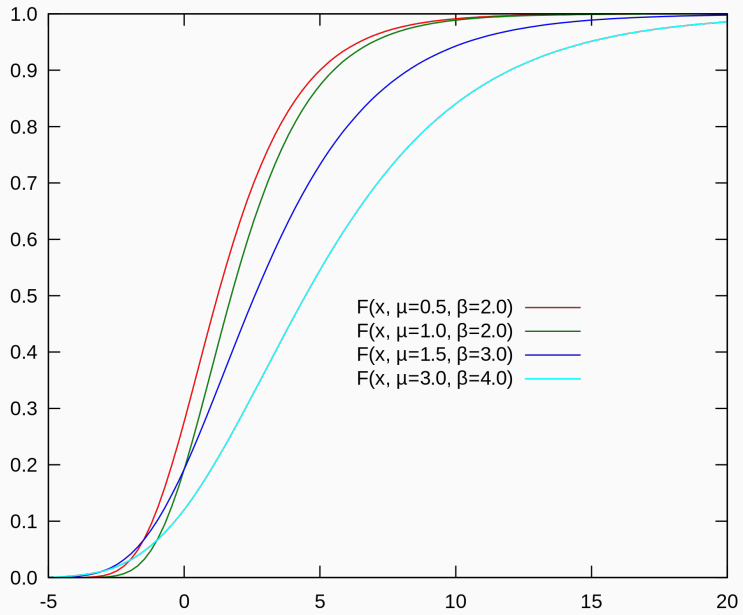
- Как протащить градиент через сэмплирование?

dVAE + Transformer = DALL-E

- dVAE считает градиенты не так, как VQ-VAE; для дискретизации служит *Gumbel-Softmax distribution*:
- Gumbel-Max trick: можно сэмплировать из дискретного распределения с вероятностями классов π_i как $z = \arg\max_i (g_i + \log \pi_i)$, где g_i берутся из распределения Гумбеля:

$$p(g_i) = e^{-(g_i + e^{-g_i})}, \quad F(g_i) = e^{-e^{-g_i}}.$$





dVAE + Transformer = DALL-E

- Объяснение трюка:

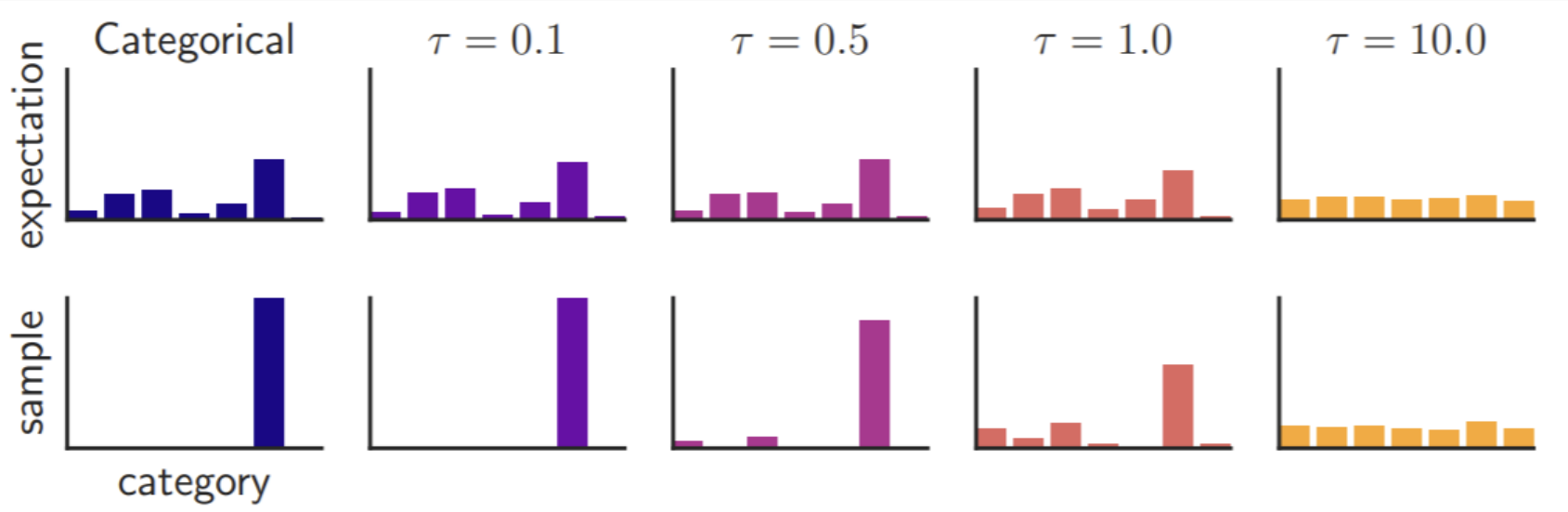
$$
\begin{aligned}
p(z=k) &= p\left(g_k + \log \pi_k \geq g_j + \log \pi_j \text{ для всех } j\right) = \\
&= \int_{-\infty}^{\infty} \prod_{j \neq k} p\left(g_k + \log \pi_k \geq g_j + \log \pi_j \middle| g_k\right) p(g_k)\, \mathrm{d}g_k = \\
&= \int_{-\infty}^{\infty} \prod_{j \neq k} e^{-e^{-g_k - \log \pi_k + \log \pi_j}} e^{-(g_k + e^{-g_k})} \mathrm{d}g_k = \\
&= \int_{-\infty}^{\infty} e^{-\sum_{j \neq k} \pi_j e^{-g_k - \log \pi_k}} \pi_k e^{-\left(g_k + \log \pi_k + \pi_k e^{-g_k - \log \pi_k}\right)} \mathrm{d}g_k = \\
&= \pi_k \int_{-\infty}^{\infty} e^{-g_k - \log \pi_k - \left(\pi_k + \sum_{j \neq k} \pi_j\right) e^{-g_k - \log \pi_k}} \mathrm{d}g_k = \pi_k.
\end{aligned}
$$

dVAE + Transformer = DALL-E

- A Gumbel-Softmax — это релаксация дискретного распределения:

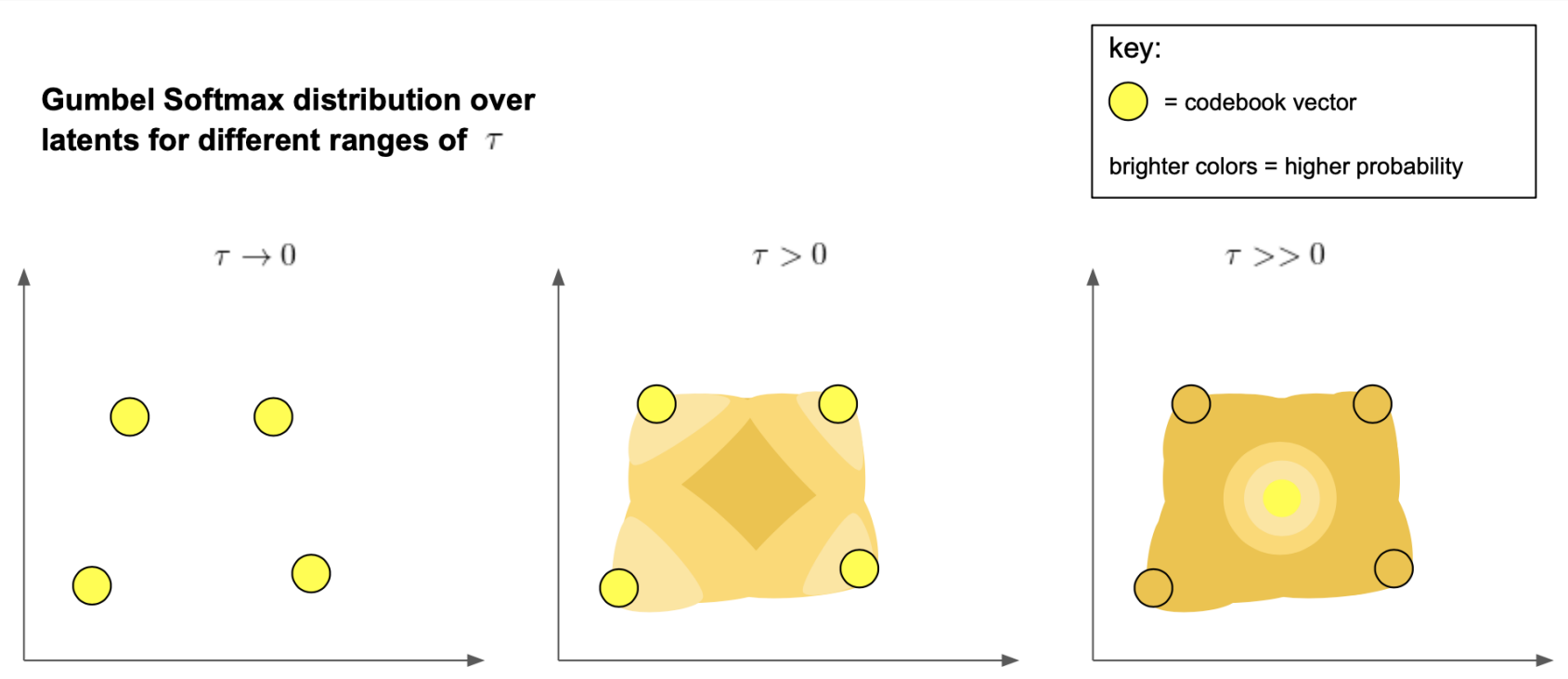
$$y_i = \text{softmax}\left(\frac{g_i + \log \pi_i}{\tau}\right) = \frac{e^{\frac{1}{\tau}(g_i + \log \pi_i)}}{\sum_j e^{\frac{1}{\tau}(g_j + \log \pi_j)}},$$

и при $\tau \to 0$ это стремится к дискретному распределению с вероятностями π_i.



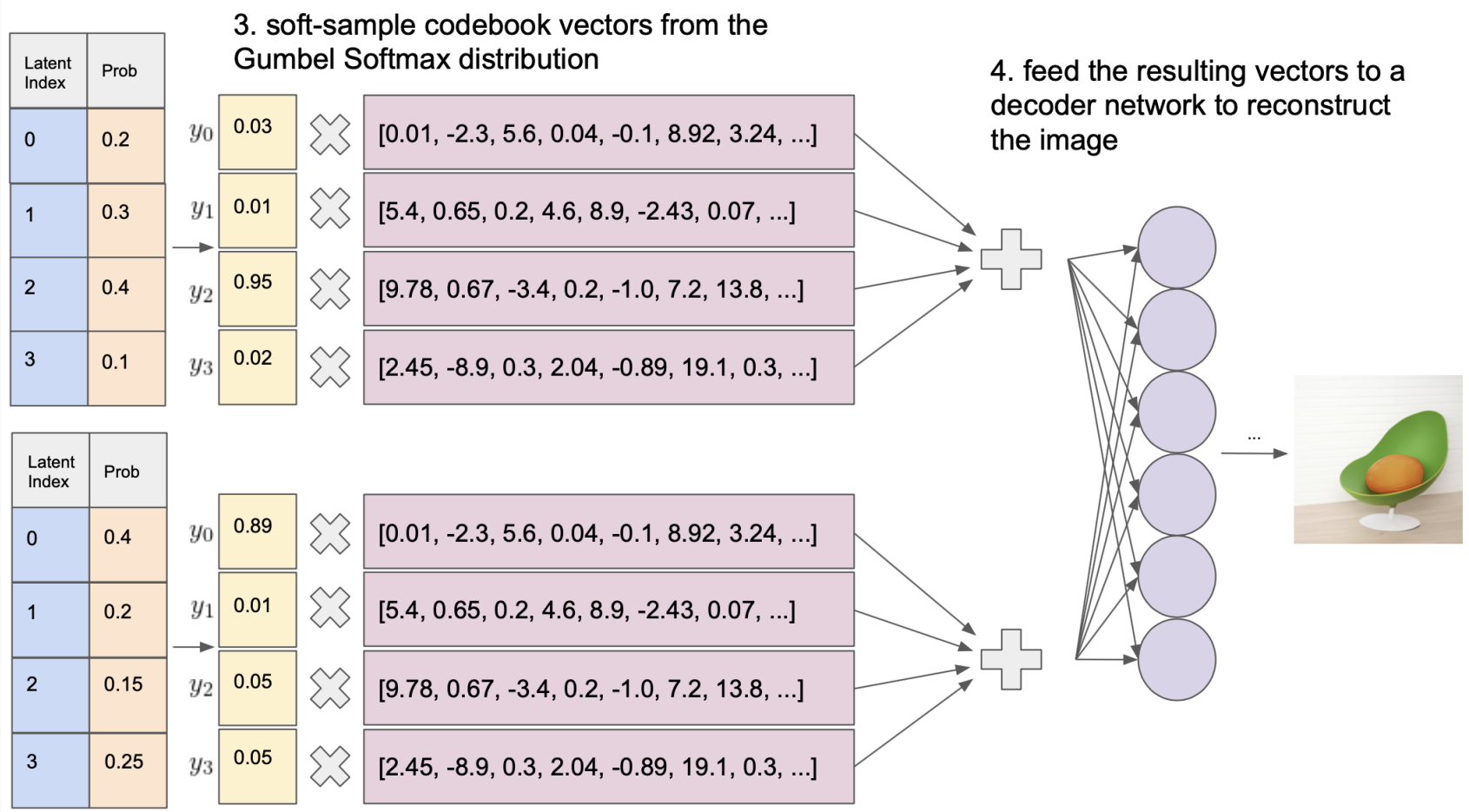
dVAE + Transformer = DALL-E

- Это то, что мы делаем с кодовыми векторами — сэмплируем выпуклую комбинацию $\mathbf{z} = \sum_{j=1}^{k} y_j \mathbf{e}_j$ через Gumbel-Softmax (это тоже, кстати, reparametrization trick: теперь g_i сэмплируются отдельно), и теперь все градиенты проходят, а во время обучения потихоньку устремляем $\tau \to 0$:



dVAE + Transformer = DALL-E

- Общая картина:

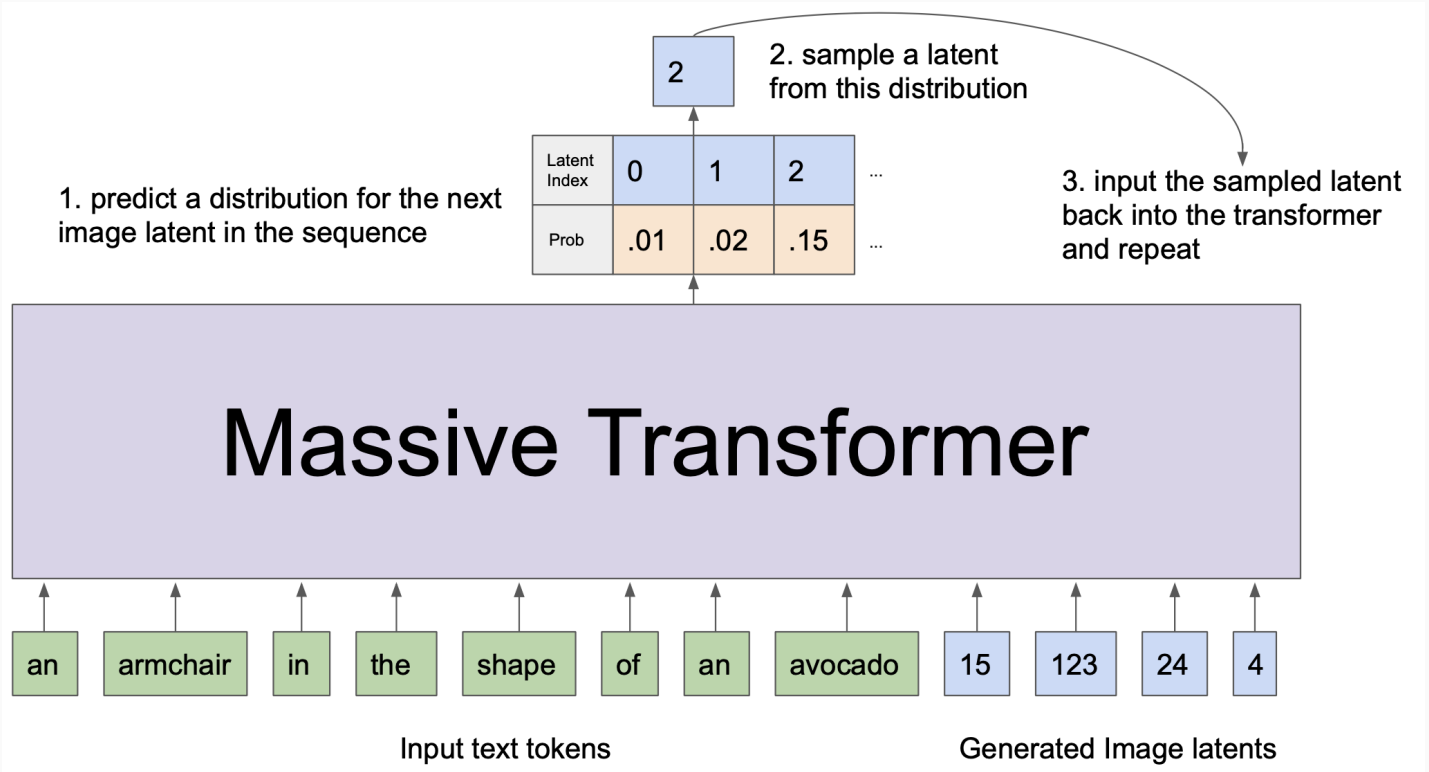


dVAE + Transformer = DALL-E

- И теперь DALL-E моделирует код $\mathbf{z}$ из текста $\mathbf{y}$, а потом использует его в dVAE, чтобы получить картинку $\mathbf{x}$:

$$p_{\theta,\psi}(\mathbf{x},\mathbf{y},\mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})\, p_\psi(\mathbf{y},\mathbf{z}).$$

- Трансформер порождает только $\mathbf{z}$, как последовательность:



dVAE + Transformer = DALL-E

- То есть фактически это одна большая вариационная оценка:

$$\ln p_{\theta,\psi}(\mathbf{x},\mathbf{y}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) - \beta\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\,\|p_\psi(\mathbf{y},\mathbf{z})\right)\right]:$$

  - $q_\phi(\mathbf{z}|\mathbf{x})$ — распределение над 32×32 токенами, порождённое энкодером dVAE по картинке $\mathbf{x}$;
  - $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$ — распределение на картинках, порождённое декодером dVAE по токенам $\mathbf{z}$; здесь мы предположим $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})$;
  - $p_\psi(\mathbf{y},\mathbf{z})$ — совместное распределение на текстах и латентных кодах, которое моделирует трансформер.

dVAE + Transformer = DALL-E

- То есть фактически это одна большая вариационная оценка:

$$\ln p_{\theta,\psi}(\mathbf{x},\mathbf{y}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) - \beta\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\,\|p_\psi(\mathbf{y},\mathbf{z})\right)\right].$$

- И обучение идёт в два приёма:
  - сначала максимизируем оценку по ϕ и θ, т.е. обучаем dVAE просто на картинках, без текстов; здесь считаем p_ψ равномерным, а q_ϕ релаксируем через Gumbel-Softmax;
  - потом фиксируем ϕ и θ и обучаем ψ, т.е. обучаем трансформер моделировать текст (BPE encoding) и картинки (их коды $\mathbf{z}$) совместно.
- И дальше очень, очень много трюков с распределённым обучением, потому что модели очень большие.

dVAE + Transformer = DALL-E

- Получается очень круто (для 2021), и с выдумкой:

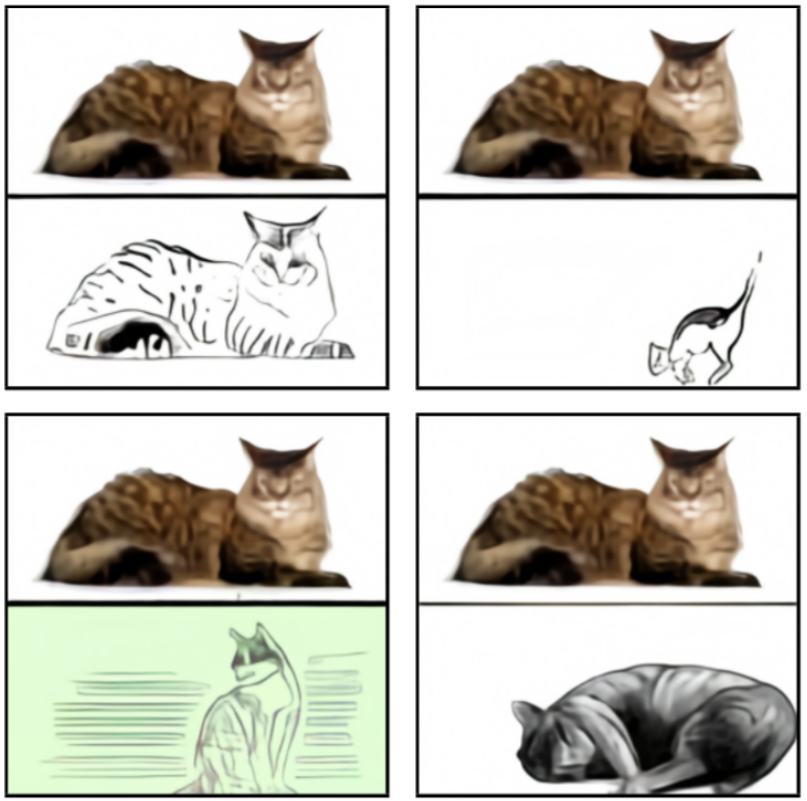
(a) a tapir made of accordion. a tapir with the texture of an accordion.

(b) an illustration of a baby hedgehog in a christmas sweater walking a dog

dVAE + Transformer = DALL-E

- Получается очень круто (для 2021), и с выдумкой:

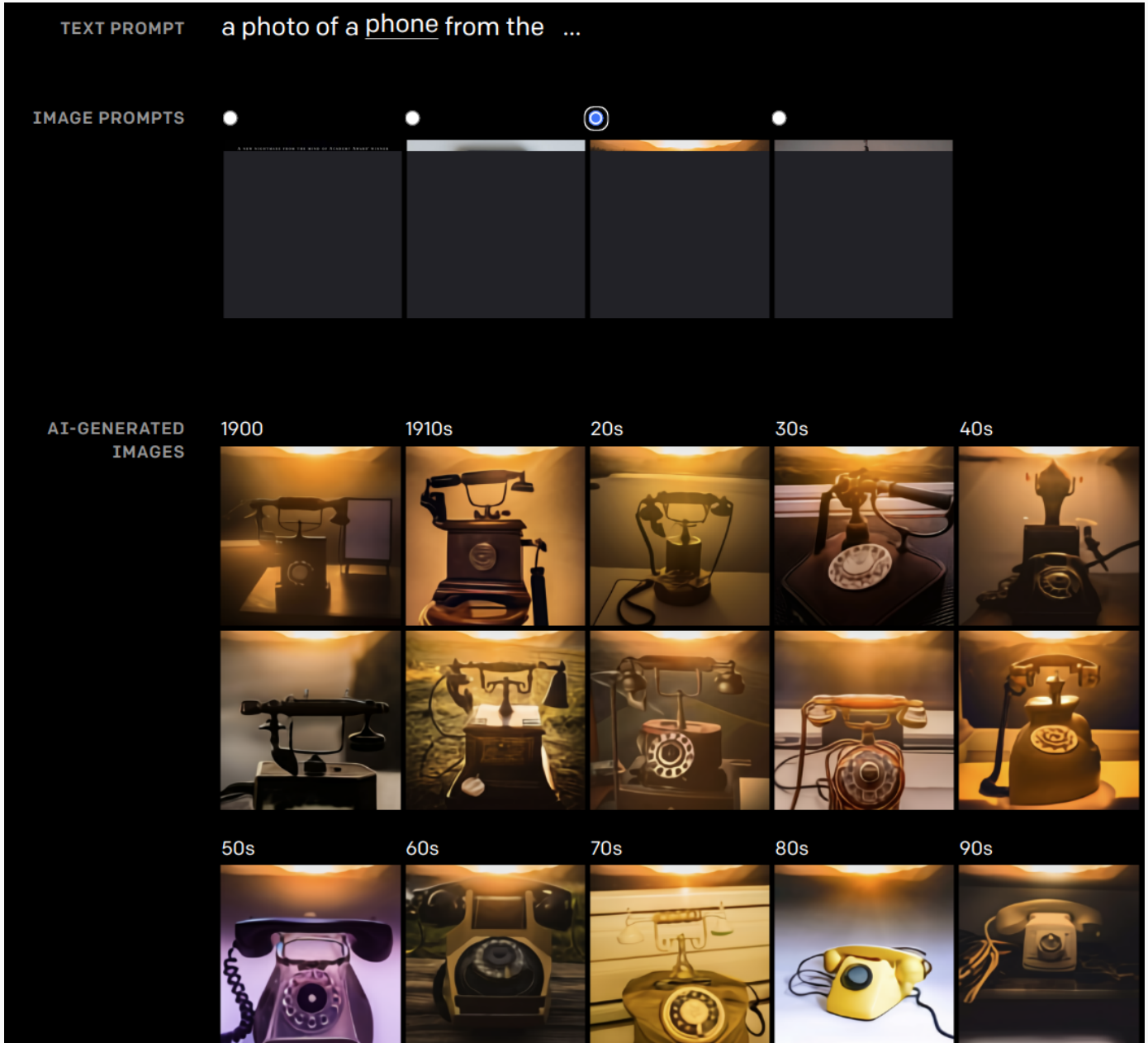
(c) a neon sign that reads “backprop”. a neon sign that reads “backprop”. backprop neon sign



(d) the exact same cat on the top as a sketch on the bottom

dVAE + Transformer = DALL-E

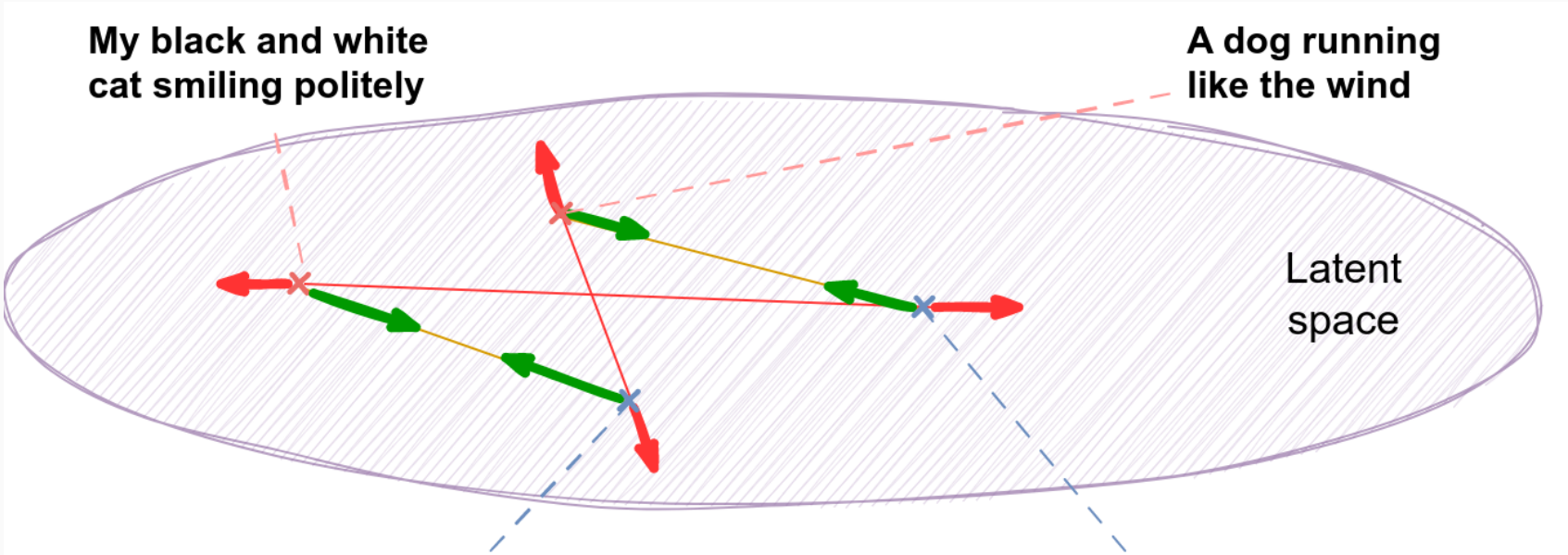
- В том числе сложные запросы (`https://openai.com/blog/dall-e/`):



CLIP и BLIP

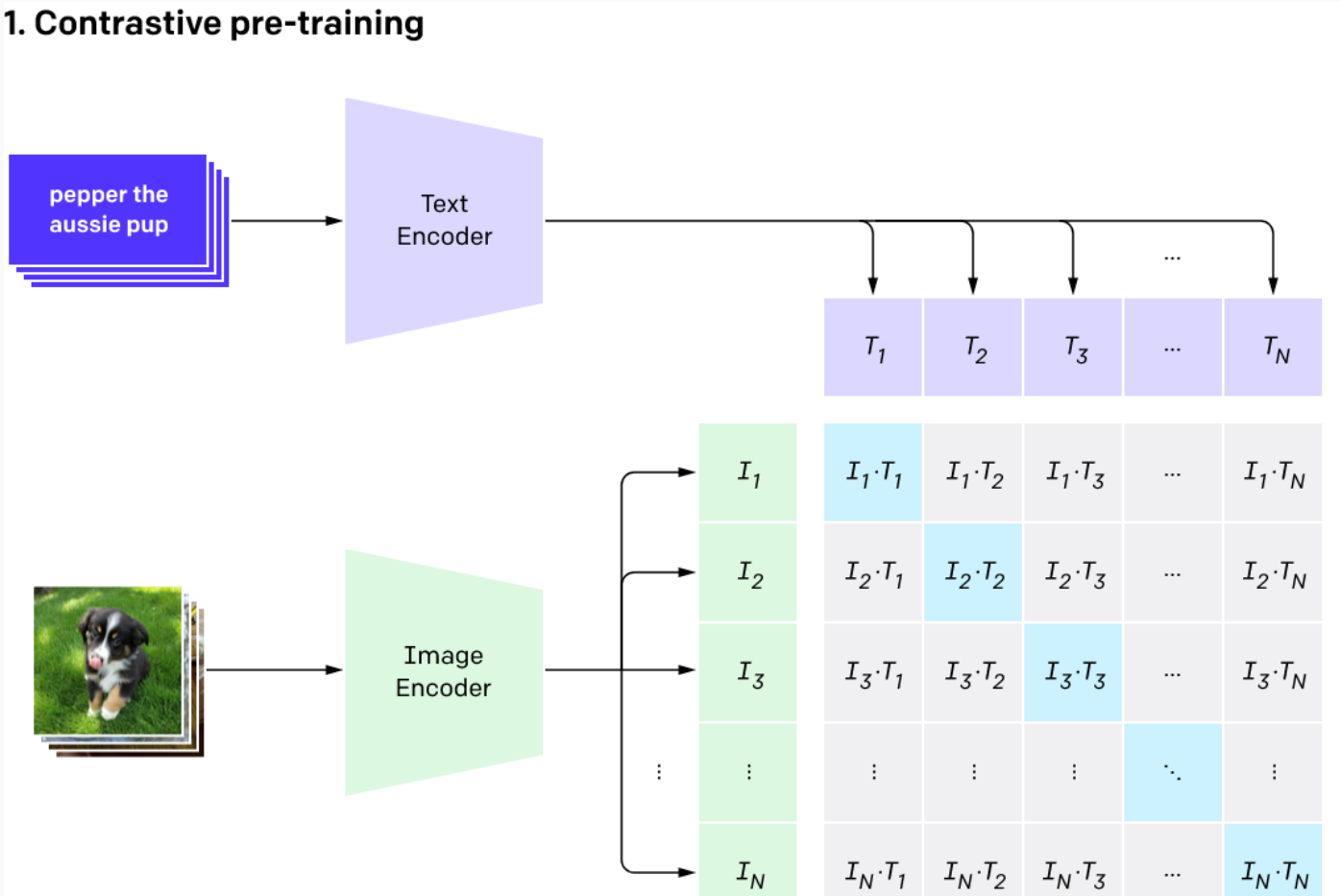
CLIP

- Contrastive pretraining: давайте тексты и картинки в латентном пространстве будем притягивать и отталкивать



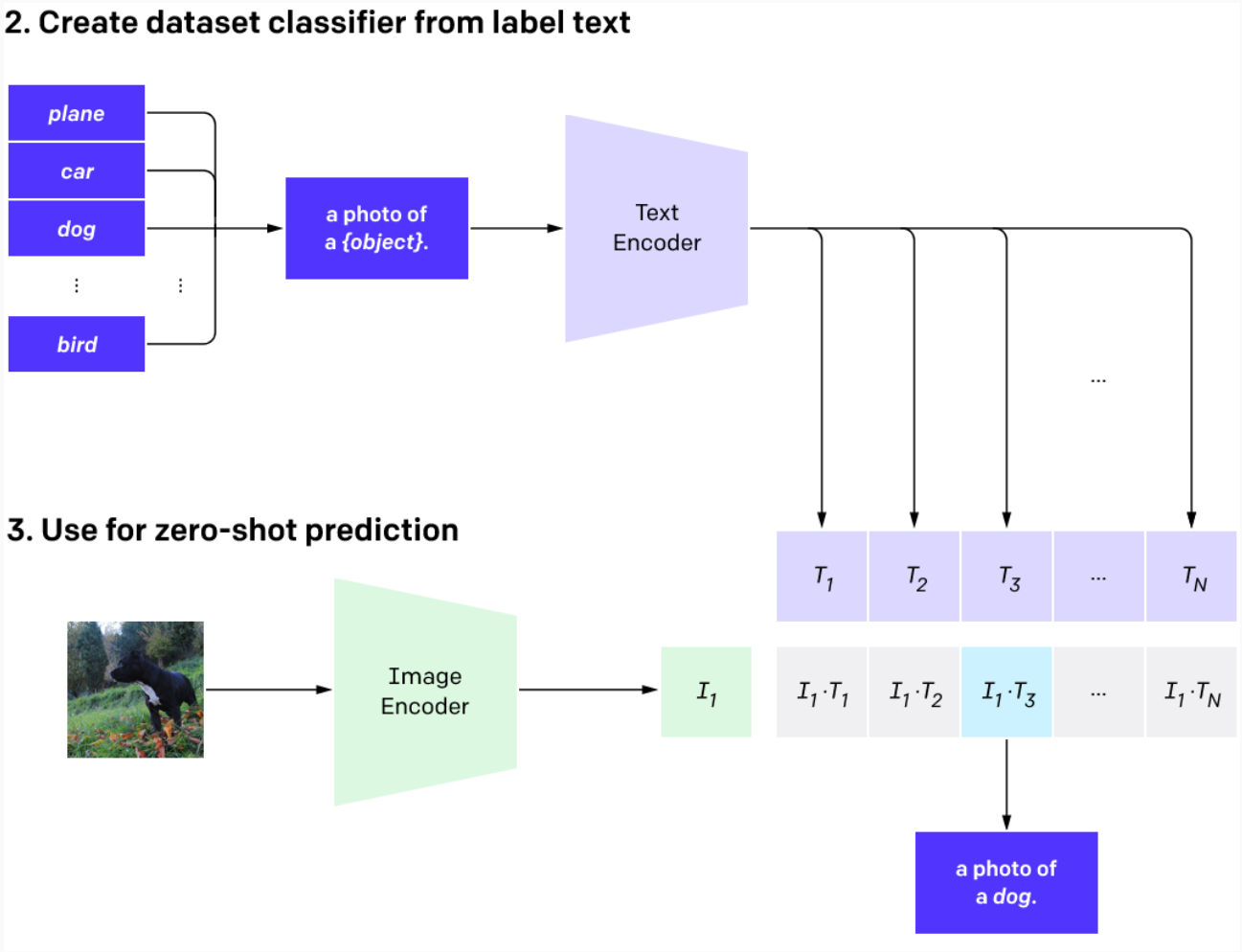
CLIP

- В 2021 году появилась мультимодальная модель CLIP (Contrastive Language–Image Pre-training; не путать… ох, ни с чем не путать) от OpenAI
- Используют для предобучения пары «картинка-текст», которых в интернете немало:



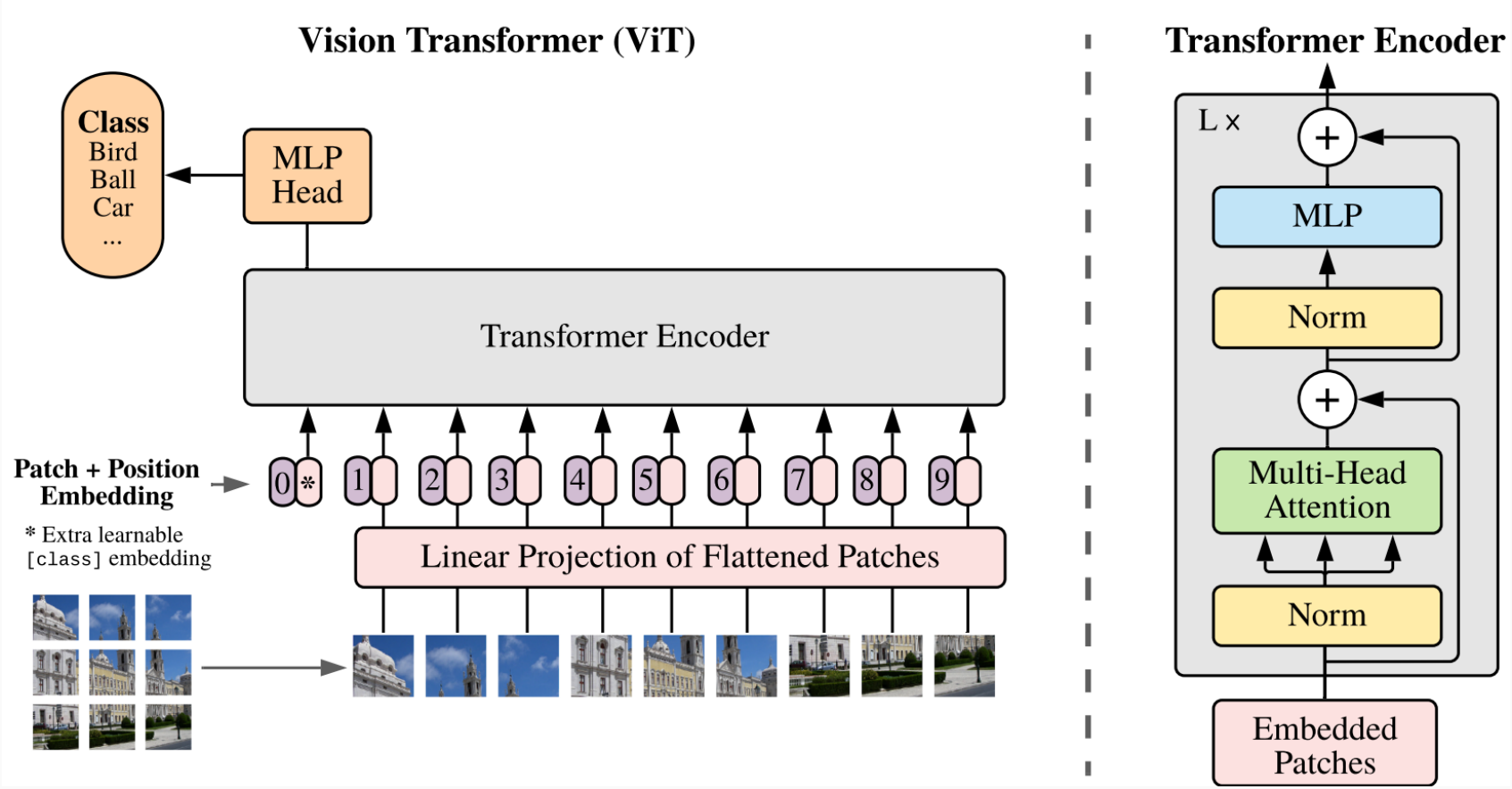
CLIP

- А потом, например, для zero-shot классификации можно просто из метки класса делать запрос:



CLIP

- Сама модель — ViT (Vision Transformer) (Dosovitsky et al., 2020)

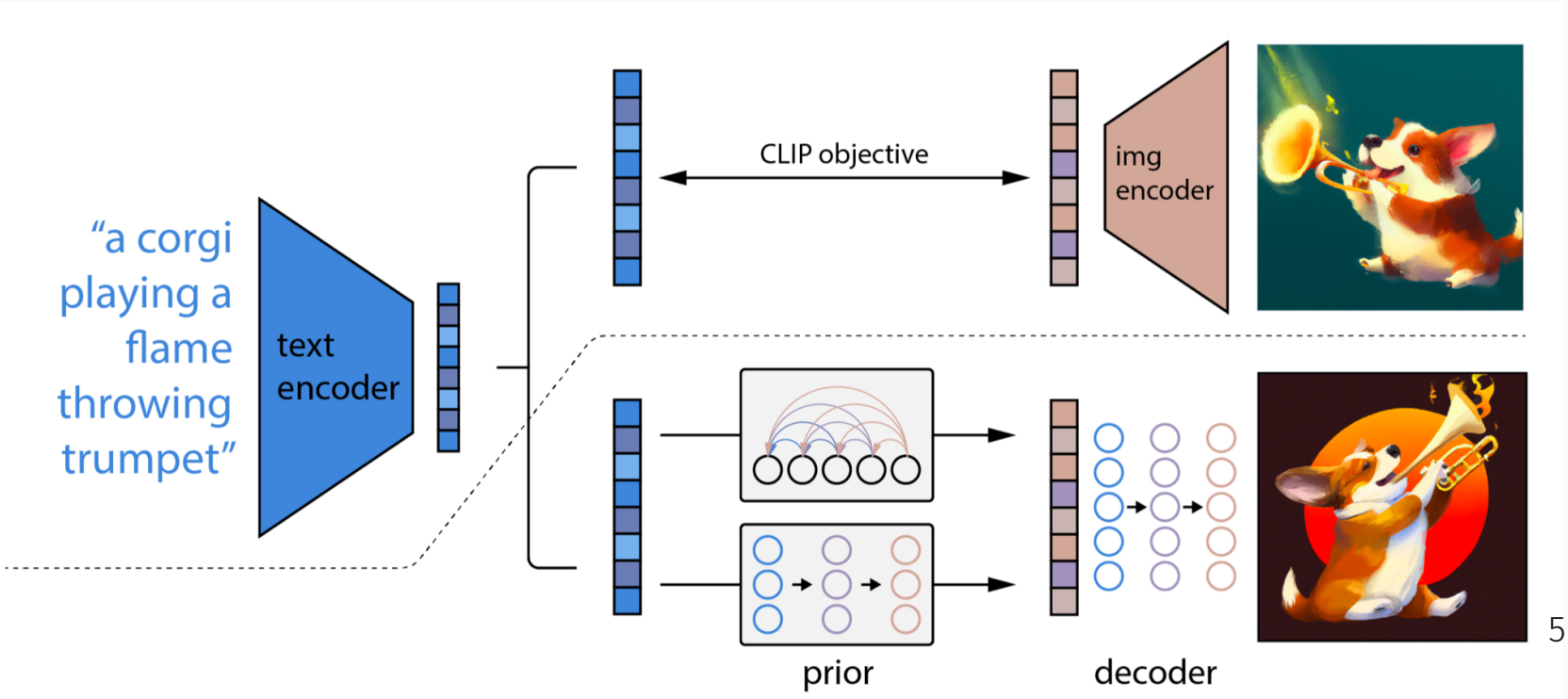


CLIP

- unCLIP, он же DALL-E 2 (Ramesh et al., 2022), обучает

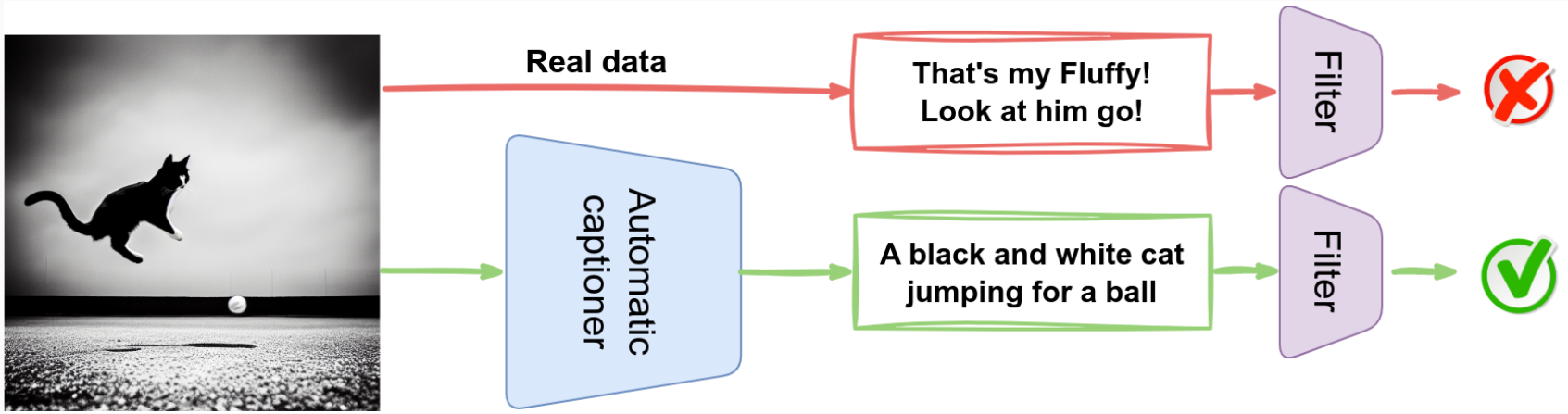
$$p\left(\mathbf{x}|y\right) = p\left(\mathbf{x}|\mathbf{c}, y\right) p\left(\mathbf{c}|y\right),$$

где prior model $p\left(\mathbf{c}|y\right)$ строит CLIP embedding по тексту, а в $p\left(\mathbf{x}|\mathbf{c}, y\right)$ сначала порождается «шум» (prior) для картинки, а потом по ней диффузионная модель рисует саму картинку



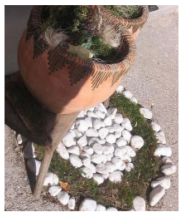
BLIP

- BLIP (Bootstrapping Language-Image Pre-training) (Li et al., 2022): порождают синтетические подписи, фильтруют их



T_w: "from bridge near my house"

T_s: "a flock of birds flying over a lake at sunset"



T_w: "in front of a house door in Reichenfels, Austria"

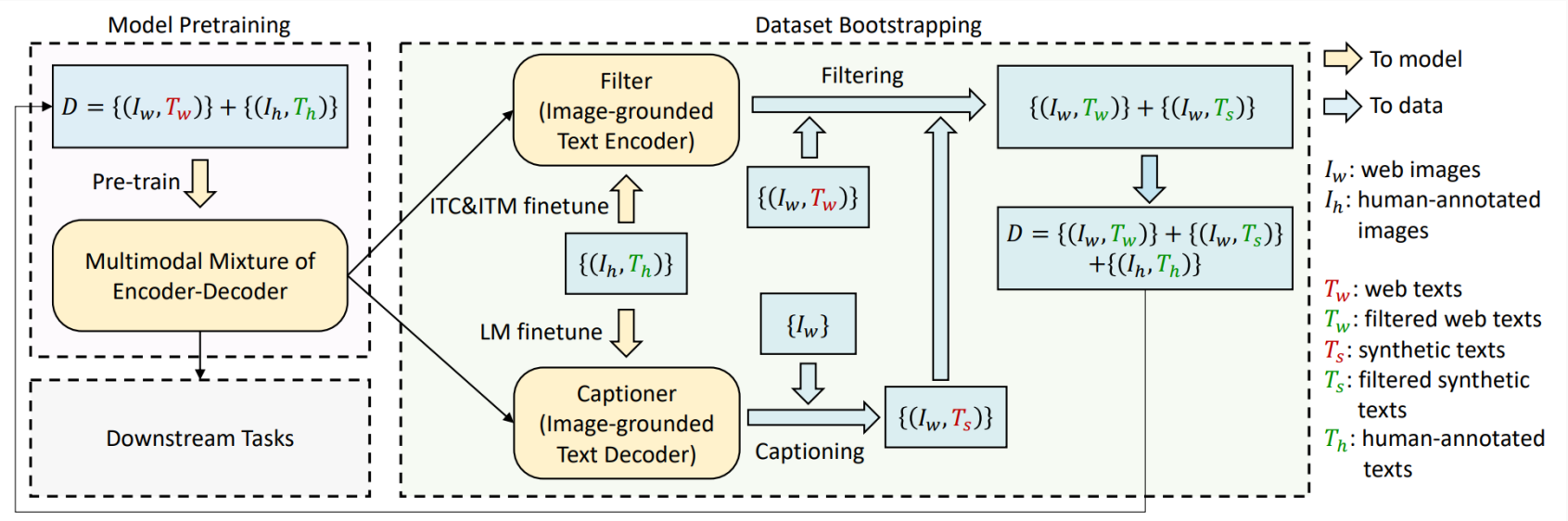
T_s: "a potted plant sitting on top of a pile of rocks"

T_w: "the current castle was built in 1180, replacing a 9th century wooden castle"

T_s: "a large building with a lot of windows on it"

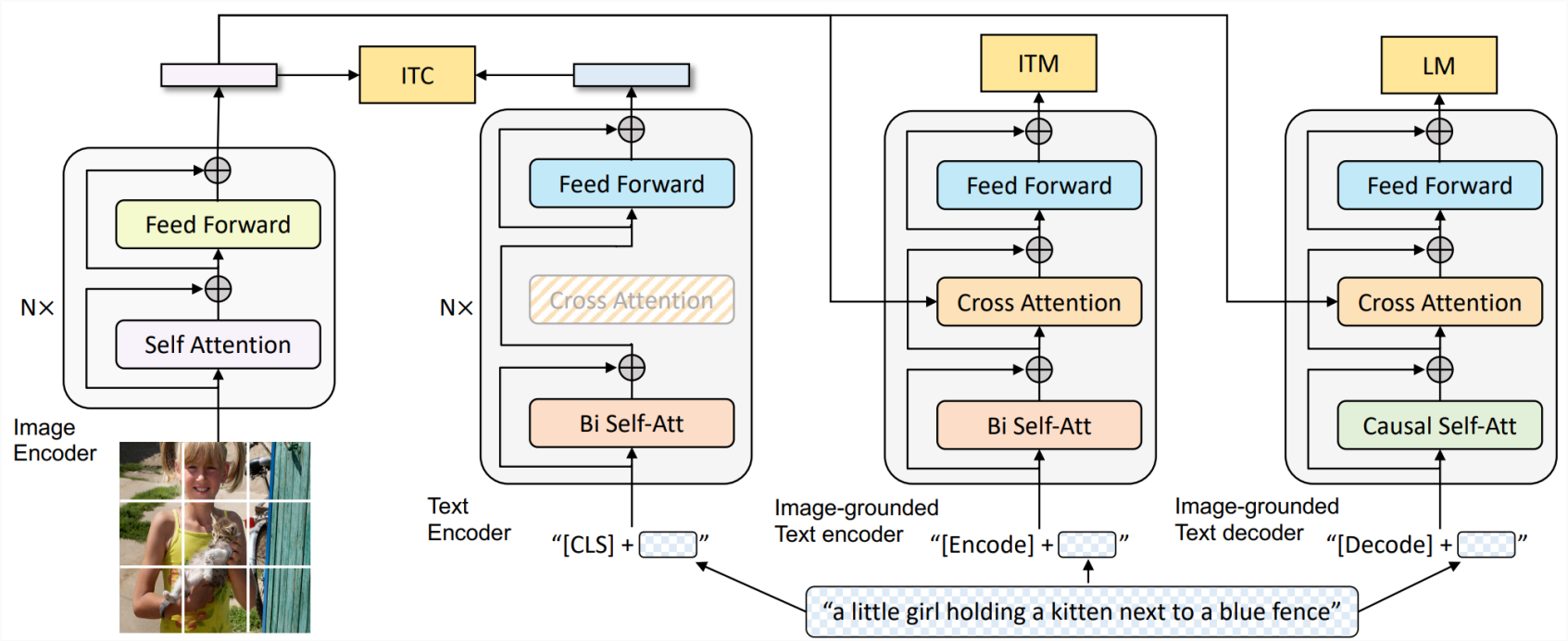
BLIP

- Структура обучения:



BLIP

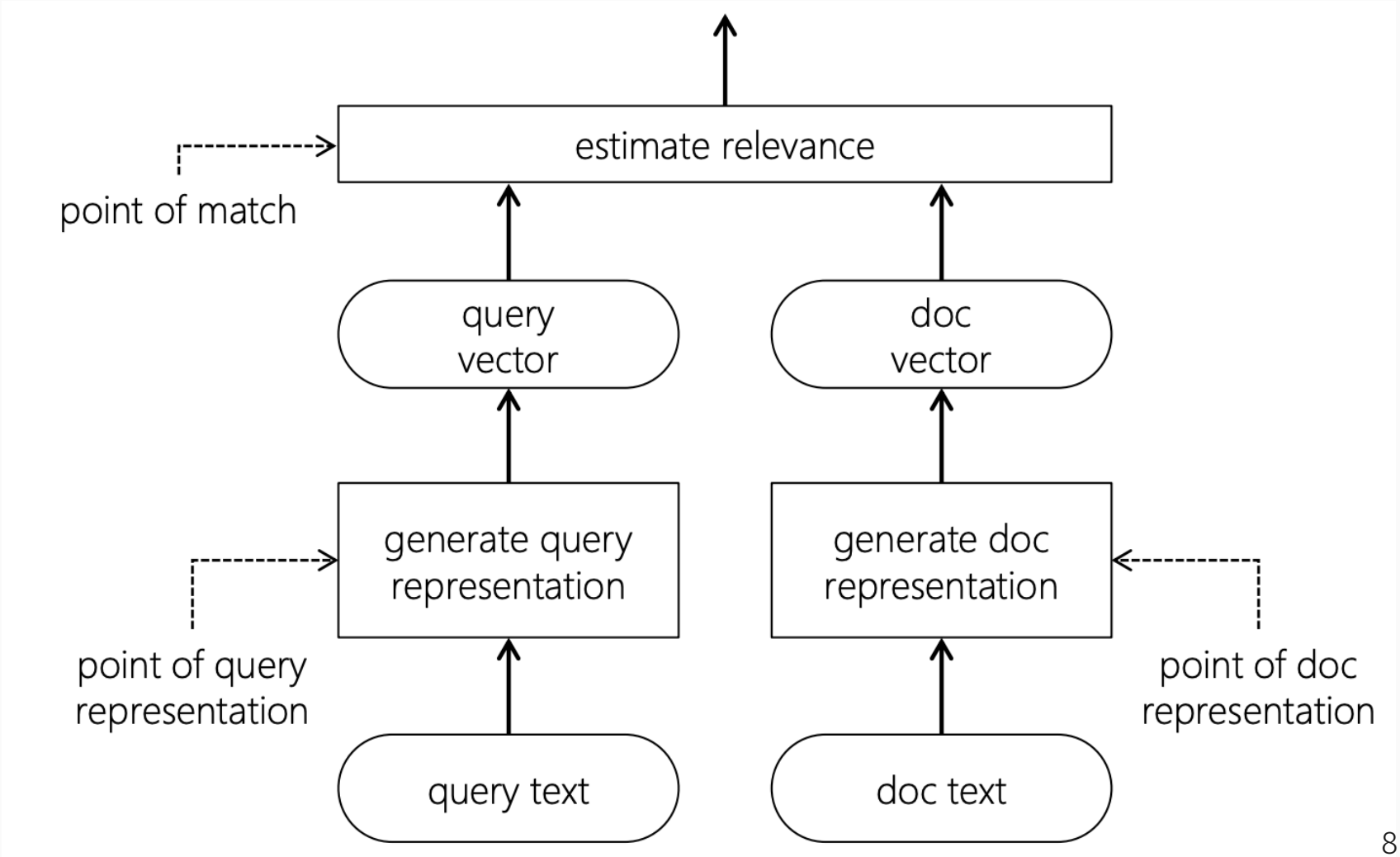
- Обучают три разных encoder'а и decoder: ITC – image-text contrastive loss (совмещаем представления текста и картинки), ITM – image-text matching loss (различаем положительные и отрицательные примеры пар), LM – language modeling (пишем подписи к картинкам)



Нейросетевой информационный поиск

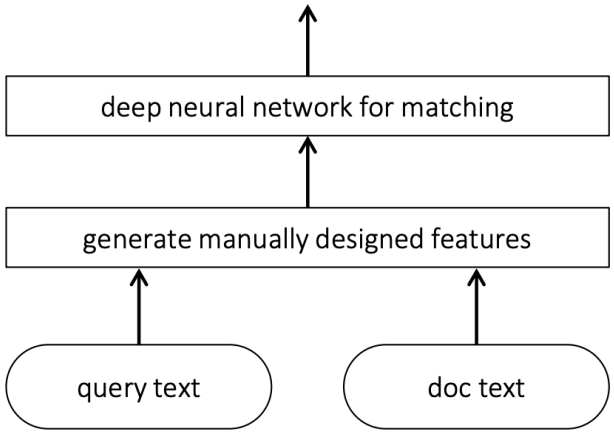
Neural information retrieval

- По идее, neural information retrieval работает так (Mitra, Craswell, 2018):

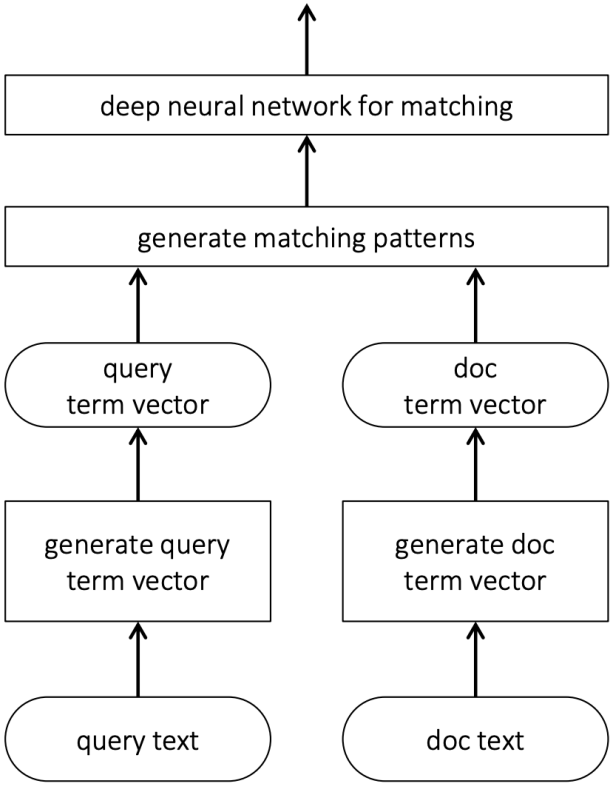


Neural information retrieval

- Здесь везде могут быть (и есть!) нейросети:



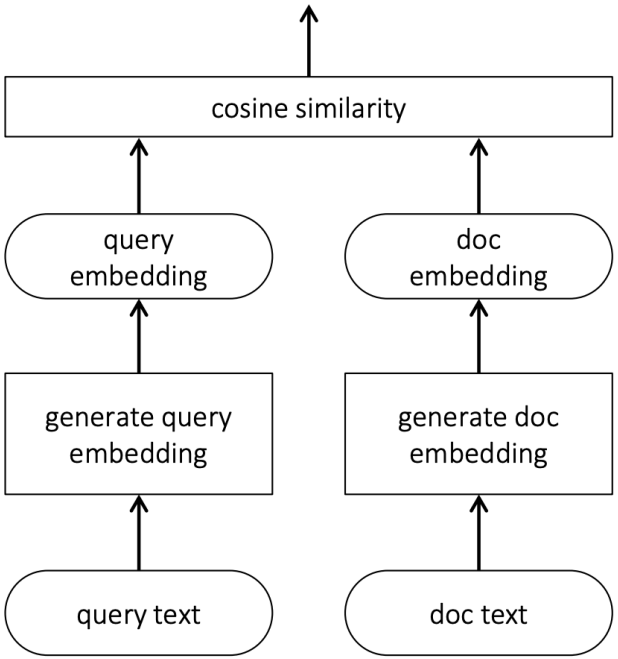
(a) Learning to rank using manually designed features (*e.g.*, Liu (2009))



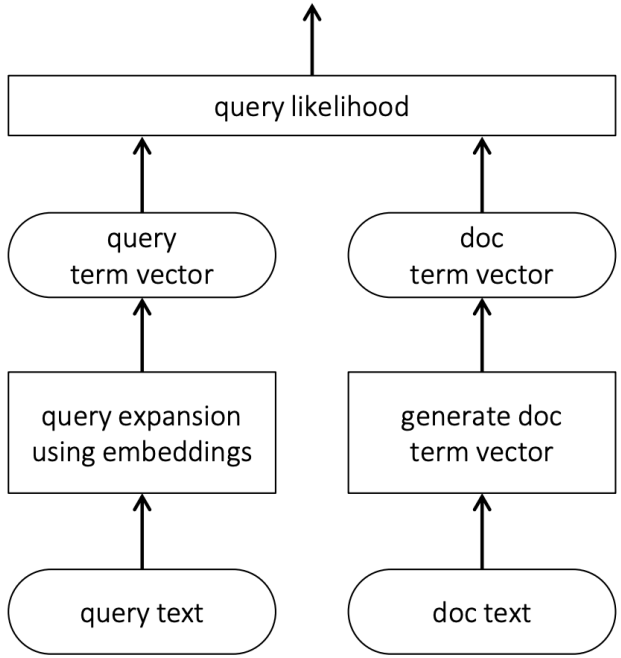
(b) Estimating relevance from patterns of exact matches (*e.g.*, (Guo *et al.*, 2016a; Mitra *et al.*, 2017a))

Neural information retrieval

- Здесь везде могут быть (и есть!) нейросети:



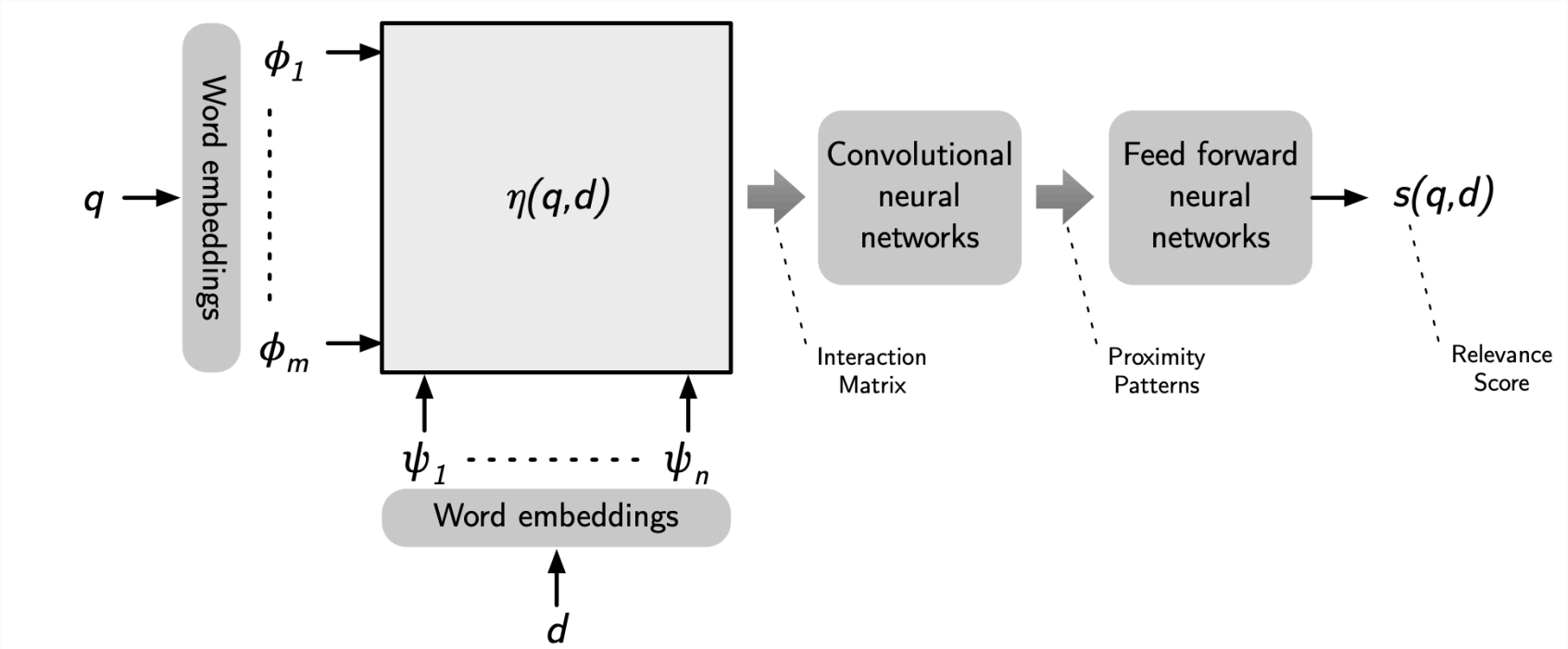
(c) Learning query and document representations for matching (*e.g.*, (Huang *et al.*, 2013; Mitra *et al.*, 2016a))



(d) Query expansion using neural embeddings (*e.g.*, (Roy *et al.*, 2016; Diaz *et al.*, 2016))

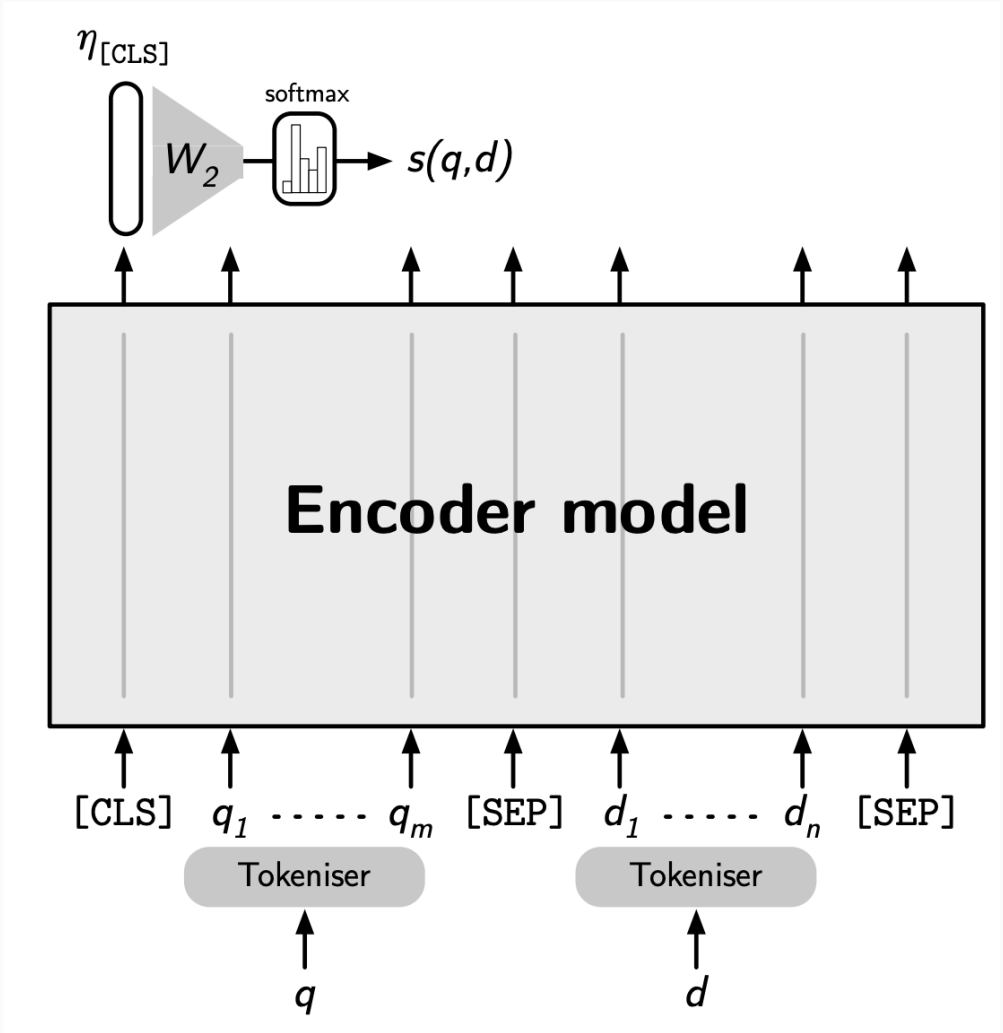
Neural information retrieval

- Можно предсказывать оценку релевантности (Tonellotto, 2022):

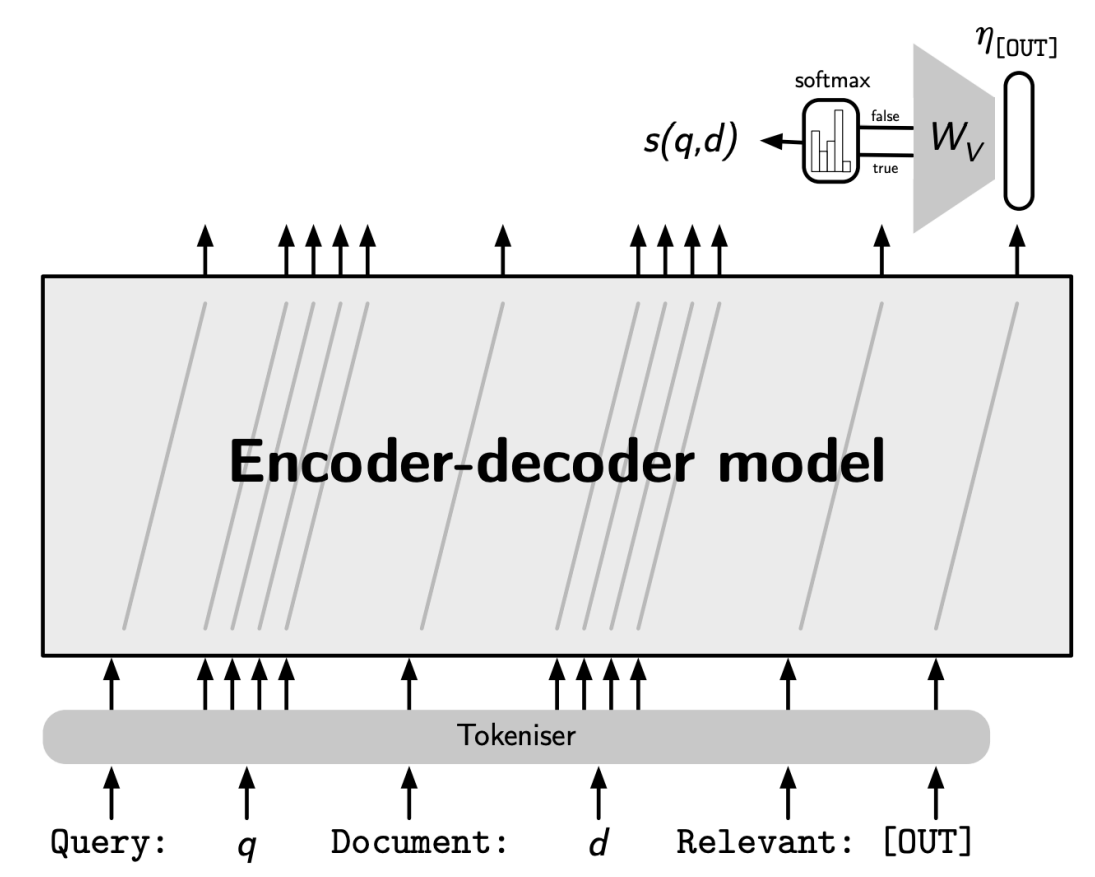


Neural information retrieval

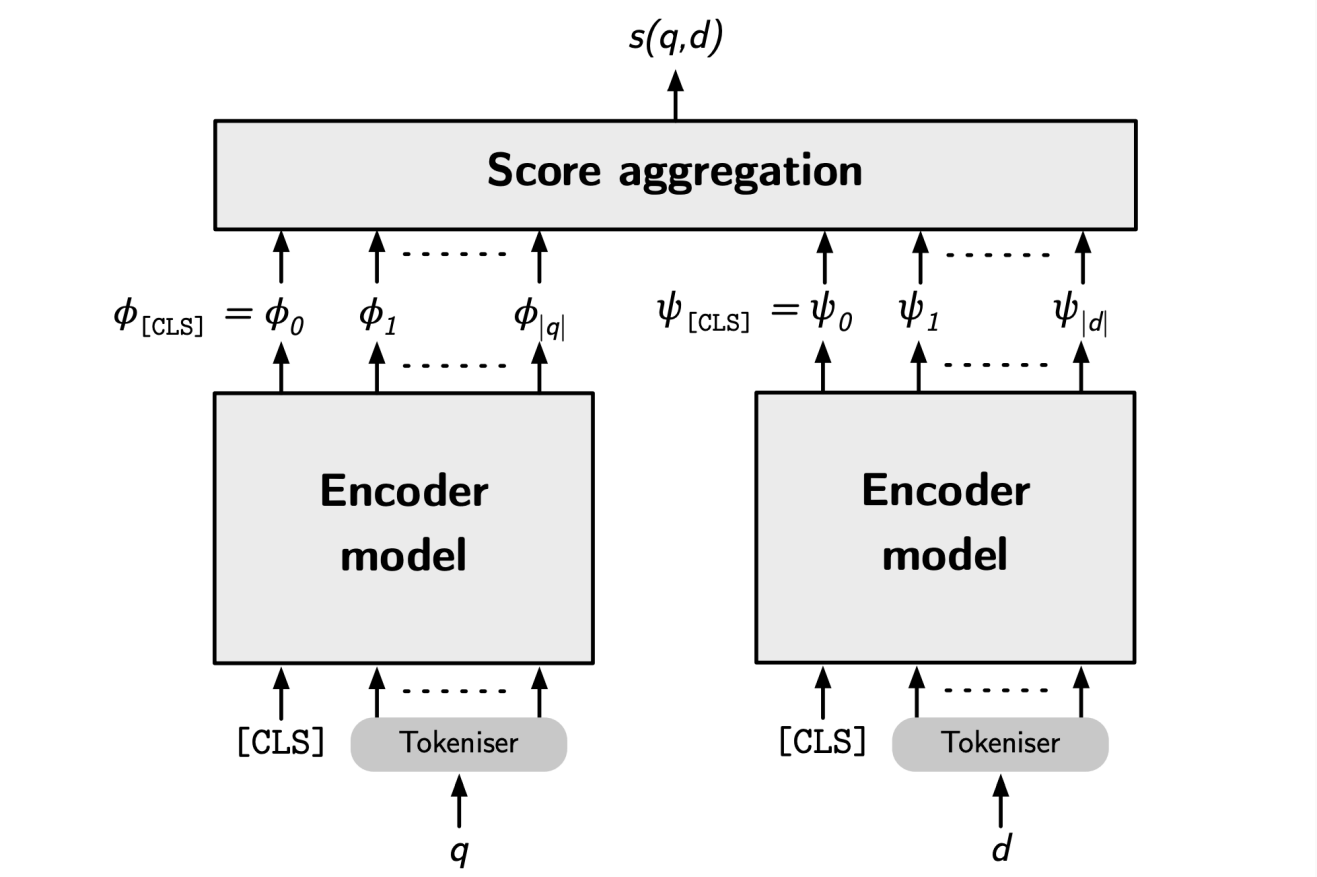
- В том числе трансформерами:



- Или дообучить языковую модель оценивать релевантность:



- А можно сконцентрироваться на представлениях, так, чтобы потом их можно было считать отдельно и переиспользовать:



- Но здесь интересно, какие выбирать функции ошибки: pointwise vs. pairwise vs. listwise
- Пока всё это была обычная классификация: релевантно или нет; функции ошибки вроде перекрёстной энтропии

$$p(y_{q,d} \mid q, d) = p(y_{q,d} \mid \mathbf{x}_{q,d}) = \frac{\exp(\gamma \cdot s(\mathbf{x}_{q,d}, y_{q,d}))}{\sum_{y \in \mathcal{Y}} \exp(\gamma \cdot s(\mathbf{x}_{q,d}, y))},$$

$$L_{\text{classification}} = -\log\left(p(y_{q,d} \mid q, d)\right) = -\log\left(\frac{\exp(\gamma \cdot s(\mathbf{x}_{q,d}, y_{q,d}))}{\sum_{y \in \mathcal{Y}} \exp(\gamma \cdot s(\mathbf{x}_{q,d}, y))}\right).$$

- Но это же не то, что нужно в ранжировании!

Neural information retrieval

- *Contrastive loss* (контрастивная ошибка) — предположим, что релевантные документы должны быть ближе к запросу, чем нерелевантные:

$$L_{\text{Contrastive}}(q, d, y_{q,d}) = y_{q,d} \cdot L_{\text{pos}}(\text{dist}_{q,d}) + (1 - y_{q,d}) \cdot L_{\text{neg}}(\text{dist}_{q,d})$$

- Но если использовать просто расстояние $\text{dist}_{q,d}$, с этой ошибкой документы будут разлетаться на бесконечность; лучше добавить сюда margin m:

$$L_{\text{Contrastive}}(q, d, y_{q,d}) = y_{q,d} \cdot \frac{1}{2} \left(\max(0,\, m - \text{dist}_{q,d})\right)^2 + (1 - y_{q,d}) \cdot (\text{dist}_{q,d})^2$$

- А ещё лучше – triplet loss:

$$L_{\text{Triplet}}(q, d^+, d^-) = \max\left(0,\, \text{dist}_{q,d^+} - \text{dist}_{q,d^-} + m\right)$$

Neural information retrieval

- С другой стороны, можно оптимизировать перекрёстную энтропию на документах; вероятность того, что d^+ будет ранжирован выше всех других, равна

$$p(d^+ \mid q) = \frac{\exp(\gamma \cdot s(q, d^+))}{\sum_{d \in D} \exp(\gamma \cdot s(q, d))}$$

- А значит, можно максимизировать разницу между оценкой для d^+ и для других:

$$L_{\text{CE}}(q, d^+, D) = -\log\left(p(d^+ \mid q)\right) = -\log\left(\frac{\exp(\gamma \cdot s(q, d^+))}{\sum_{d \in D} \exp(\gamma \cdot s(q, d))}\right)$$

- В чём здесь проблема?

Neural information retrieval

- Знаменатель фиг подсчитаешь, там сумма по всем документам. Что делать?
- Иногда пробовали иерархический softmax
- Но обычно делают importance sampling. Начнём с логарифма:

$$L_{\text{CE}}(q, d^+, D) = -\log\left(\frac{\exp(\gamma \cdot s(q, d^+))}{\sum_{d \in D}\exp(\gamma \cdot s(q, d))}\right) =$$
$$= -\gamma \cdot s(q, d^+) + \log\sum_{d \in D}\exp(\gamma \cdot s(q, d)).$$

- И для обучения нейросети нужно научиться брать от этого градиент.

Neural information retrieval

- Градиент:

$$\begin{aligned}
\nabla_\theta L_{\text{CE}}(q, d^+, D) &= -\gamma \nabla_\theta s(q, d^+) + \nabla_\theta \log \sum_{d \in D} \exp(\gamma \cdot s(q, d)) \\
&= -\gamma \nabla_\theta s(q, d^+) + \frac{\nabla_\theta \sum_{d \in D} \exp(\gamma \cdot s(q, d))}{\sum_{d \in D} \exp(\gamma \cdot s(q, d))} \\
&= -\gamma \nabla_\theta s(q, d^+) + \frac{\sum_{d \in D} \nabla_\theta \exp(\gamma \cdot s(q, d))}{\sum_{d \in D} \exp(\gamma \cdot s(q, d))} \\
&= -\gamma \nabla_\theta s(q, d^+) + \frac{\sum_{d \in D} \gamma \cdot \exp(\gamma \cdot s(q, d)) \nabla_\theta s(q, d)}{\sum_{d \in D} \exp(\gamma \cdot s(q, d))} \\
&= -\gamma \nabla_\theta s(q, d^+) + \gamma \sum_{d \in D} \frac{\exp(\gamma \cdot s(q, d))}{\sum_{d' \in D} \exp(\gamma \cdot s(q, d'))} \nabla_\theta s(q, d \\
&= -\gamma \nabla_\theta s(q, d^+) + \gamma \sum_{d \in D} p(d \mid q) \nabla_\theta s(q, d).
\end{aligned}$$

- Идея в том, чтобы второе слагаемое оценить через сэмплирование.

Neural information retrieval

- Noise Contrastive Estimation (NCE): рассмотрим это как бинарную классификацию
- Модель обучается различать сэмпл из истинного распределения $p(d|q)$ от сэмпла из зашумленного распределения $\tilde{p}(d)$, и в данных по k шумных примеров на каждый настоящий
- Тогда, если E значит истинное распределение,

$$p(E \mid q, d) = \frac{p(d \mid q)}{p(d \mid q) + k \times \tilde{p}(d)},$$
$$p(\bar{E} \mid q, d) = \frac{k \times \tilde{p}(d)}{p(d \mid q) + k \times \tilde{p}(d)},$$

а согласно нашей модели,

$$p(d \mid q) = \frac{\exp(\gamma \cdot s(q, d))}{\sum_{\bar{d} \in D} \exp(\gamma \cdot s(q, \bar{d}))} = \frac{\exp(\gamma \cdot s(q, d))}{z(q)}.$$

- Главный трюк – положить $z(q) = 1$:

$$p(d \mid q) = \exp(\gamma \cdot s(q, d))$$

- Тогда получается NCE loss:

$$L_{\text{NCE}} = -\sum_{\langle x, d^+ \rangle} \left[\log p(E \mid x, d^+) + \sum_{i=1}^{k} \log p(\bar{E} \mid x, y_i^-) \right] =$$

$$= -\sum_{\langle x, d^+ \rangle} \left[\log \frac{\exp(\gamma \cdot s(q, d^+))}{\exp(\gamma \cdot s(q, d^+)) + k \times \tilde{p}(d^+)} + \right.$$

$$\left. + \sum_{i=1}^{k} \log \frac{k \times \tilde{p}(y_i^-)}{\exp(\gamma \cdot s(q, d_i^-)) + k \times \tilde{p}(y_i^-)} \right].$$

- Negative sampling – давайте ещё упростим, заменив $k \times \tilde{p}(d)$ на единицу (но сохранив k отрицательных примеров):

$$p(E \mid q, d) = \frac{\exp(\gamma \cdot s(q,d))}{\exp(\gamma \cdot s(q,d)) + 1} = \frac{1}{1 + \exp(-\gamma \cdot s(q,d))},$$
$$p(\bar{E} \mid q, d) = \frac{1}{1 + \exp(\gamma \cdot s(q,d))},$$

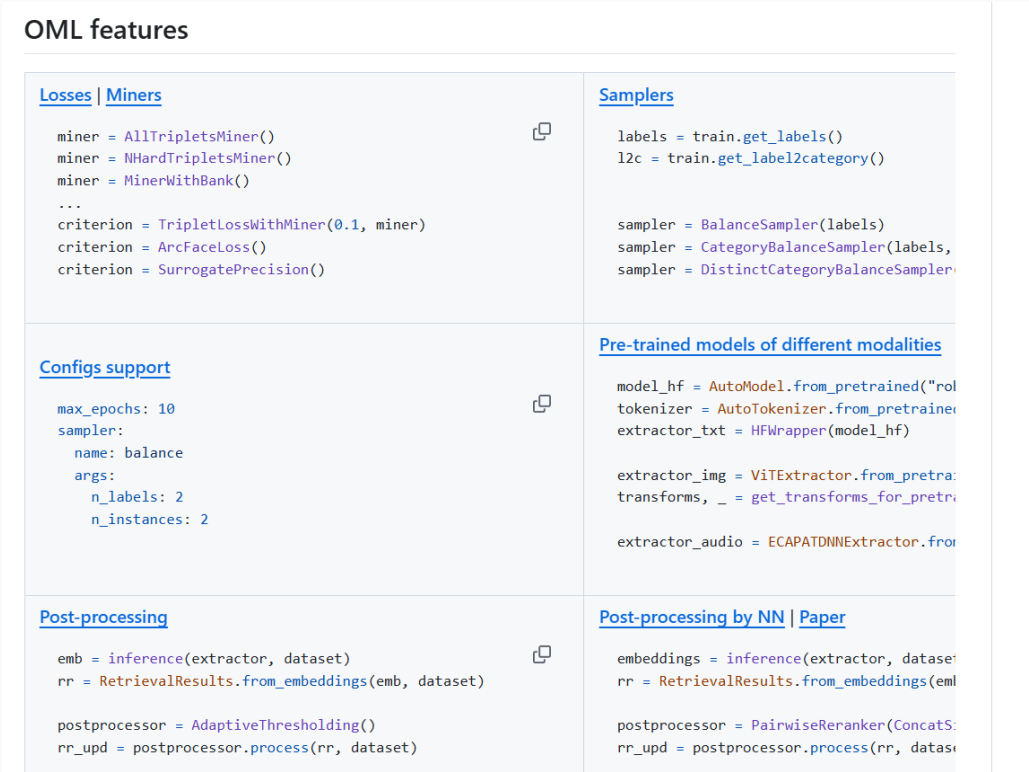
  то есть

$$L_{\text{NEG}} = -\sum_{\langle x, d^+ \rangle} \left[\log \frac{1}{1 + e^{-\gamma \cdot s(q,d^+)}} + \sum_{i=1}^{k} \log \frac{1}{1 + e^{\gamma \cdot s(q,d_i^-)}} \right].$$

- Дальше можно переходить к RankNet и LambdaRank, как мы обсуждали раньше.

STIR

- Пример: Siamese Transformer for Image Retrieval Postprocessing (STIR; Shabanov et al., 2024)
- Кстати, у Алексея хорошая библиотека для metric learning: `https://github.com/OML-Team/open-metric-learning`



OML features

Losses | Miners

```
miner = AllTripletsMiner()
miner = NHardTripletsMiner()
miner = MinerWithBank()
...
criterion = TripletLossWithMiner(0.1, miner)
criterion = ArcFaceLoss()
criterion = SurrogatePrecision()
```

Samplers

```
labels = train.get_labels()
l2c = train.get_label2category()

sampler = BalanceSampler(labels)
sampler = CategoryBalanceSampler(labels,
sampler = DistinctCategoryBalanceSampler
```

Configs support

```
max_epochs: 10
sampler:
  name: balance
  args:
    n_labels: 2
    n_instances: 2
```

Pre-trained models of different modalities

```
model_hf = AutoModel.from_pretrained("rol
tokenizer = AutoTokenizer.from_pretraine
extractor_txt = HFWrapper(model_hf)

extractor_img = ViTExtractor.from_pretra
transforms, _ = get_transforms_for_pretr

extractor_audio = ECAPATDNNExtractor.fro
```

Post-processing

```
emb = inference(extractor, dataset)
rr = RetrievalResults.from_embeddings(emb, dataset)

postprocessor = AdaptiveThresholding()
rr_upd = postprocessor.process(rr, dataset)
```

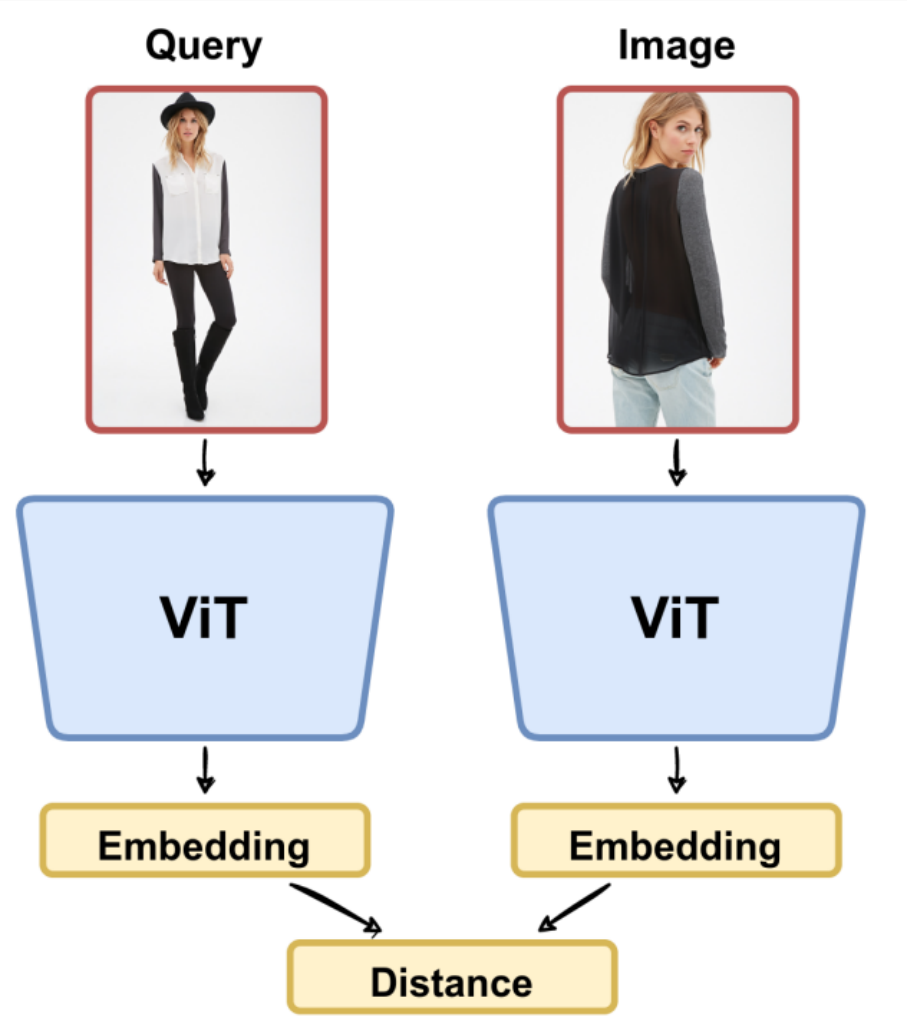
Post-processing by NN | Paper

```
embeddings = inference(extractor, dataset
rr = RetrievalResults.from_embeddings(emb

postprocessor = PairwiseReranker(ConcatS
rr_upd = postprocessor.process(rr, datas
```

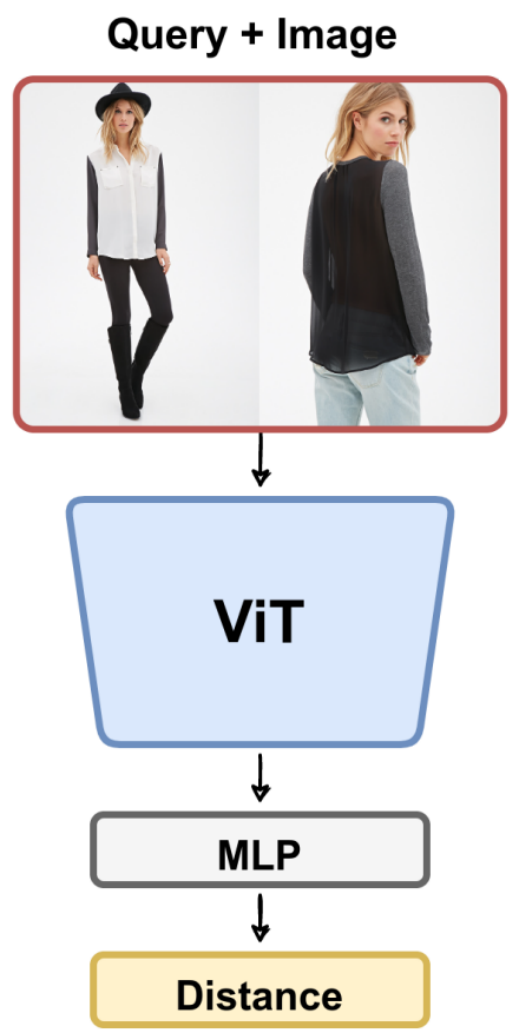
STIR

- ViT-Triplet: давайте просто возьмём стандартную сиамскую архитектуру с triplet loss и подставим туда ViT; это уже хорошо:



STIR

- STIR – а теперь давайте переранжируем топ выдачи, обучив маленький MLP поверх ViT для этого:



STIR

- И уже всё становится лучше, бьёт SotA в некоторых случаях, а статью вообще в итоге приняли на CIKM...

Спасибо!

Спасибо за внимание!