ВНИМАНИЕ И САМОВНИМАНИЕ

Сергей Николенко

СПбГУ — Санкт-Петербург

16 октября 2026 г.





Random facts:

- 16 октября 1834 г. сгорел Вестминстерский дворец: в печах под палатой лордов жгли вышедшие из употребления казначейские бирки, и огонь перекинулся на обшивку
- 16 октября 1846 г. в Массачусетской больнице впервые публично провели операцию под эфирным наркозом
- 16 октября 1869 г. рабочие, копавшие колодец на ферме в Кардиффе (штат Нью-Йорк), нашли трёхметрового «окаменевшего великана»; это была мистификация из гипса
- 16 октября 1923 г. Уолт и Рой Дисней подписали первый контракт, и этот день считается днём основания студии
- 16 октября 1975 г. у трёхлетней Рахимы Бану в Бангладеш диагностировали натуральную оспу; она выздоровела, и это был последний в истории случай Variola major
- 16 октября 1978 г. Иоанн Павел II стал первым с 1523 года неитальянцем на папском престоле

Краткое резюме предыдущей серии

- В прошлый раз мы закончили с компьютерным зрением (сегментация, adversarial examples) и перешли к последовательностям: рекуррентные сети, затухающие градиенты, LSTM и GRU.
- Но у рекуррентной сети вся история сжата в одно скрытое состояние. Сегодня научим её *смотреть назад* — механизм внимания: сначала в зрительных моделях, потом в машинном переводе.
- А потом выкинем рекуррентность совсем: внимание последовательности к самой себе, *самовнимание*.

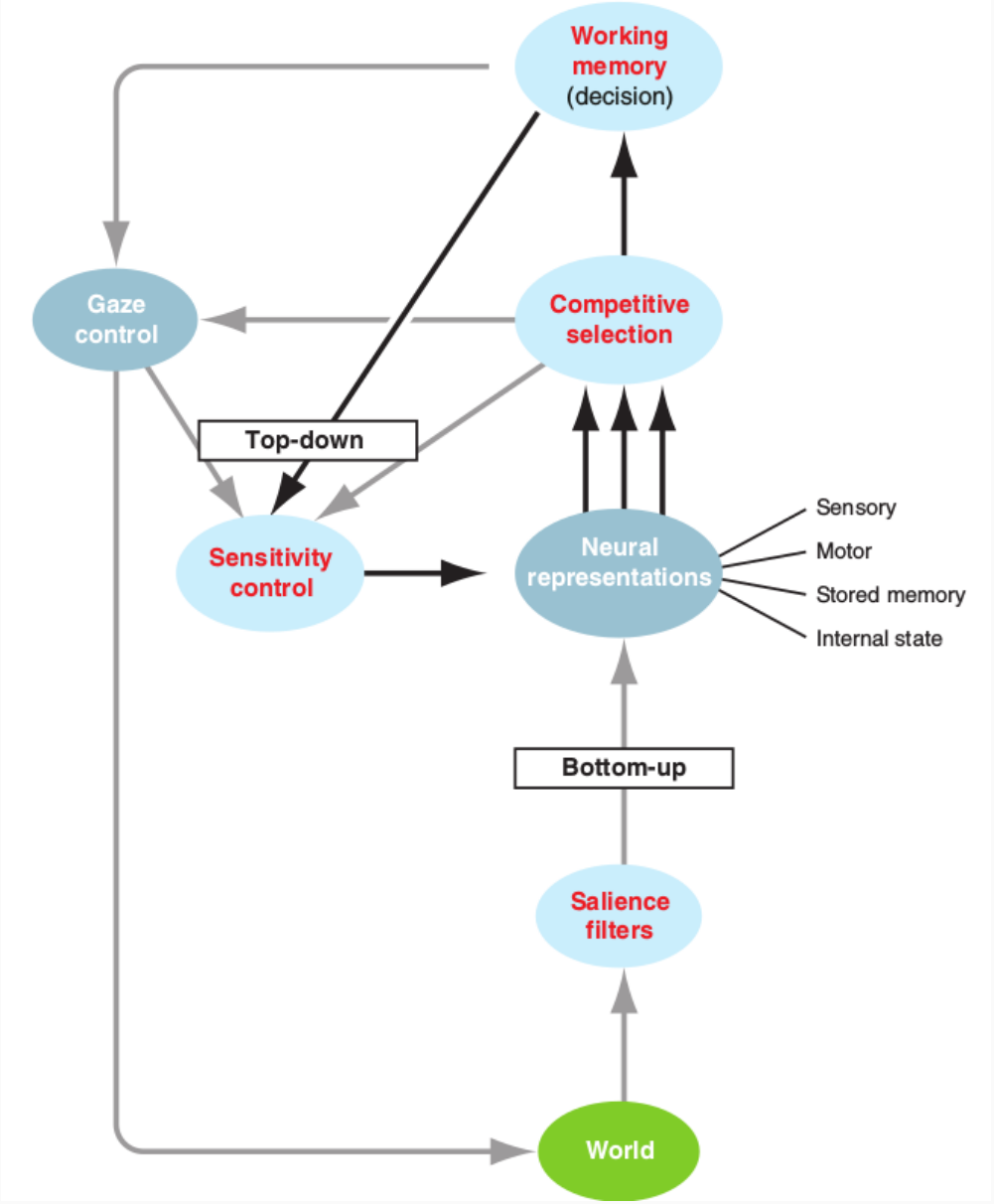
Что такое внимание?

Внимание

- Вы же сейчас *внимательно* меня слушаете, правильно?
- А что это значит?..
- Изображение с сетчатки тоже проходит через CNN, но потом мы часть его замечаем, а часть не особенно. Что это значит?
- Оказывается, что это довольно сложный вопрос.
- А.Р. Лурия: внимание, память и активация коры.

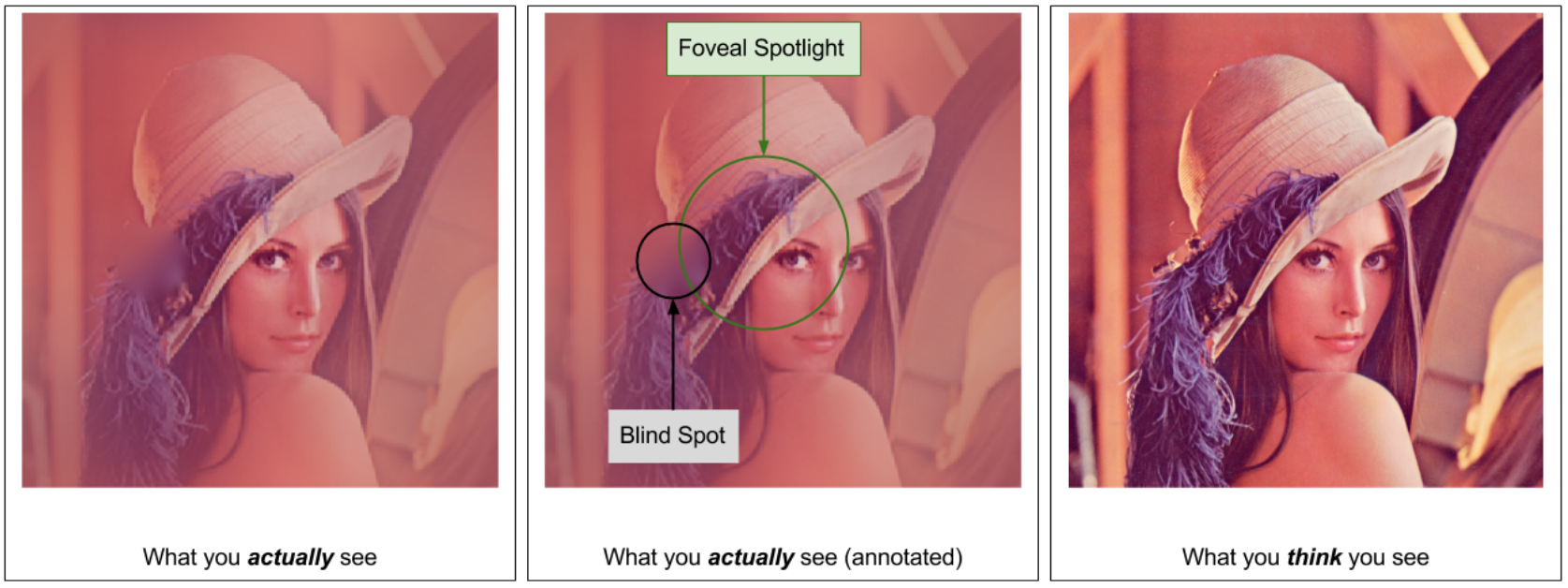
Внимание

- Дело во взаимодействии с рабочей памятью (Knudsen, 2007):



Внимание

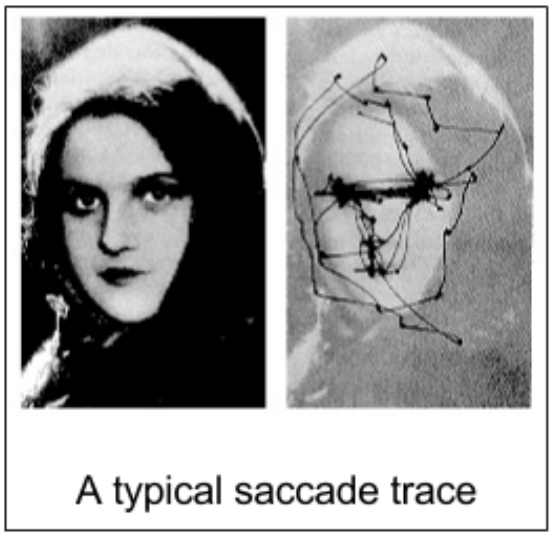
- Как это реализовать в нейронной сети? Особенно «сознательное» внимание.
- Но и «бессознательное» тоже; например, с картинками: мы же на самом деле мало чего видим в каждый момент времени.
- Центральная ямка сетчатки (fovea):



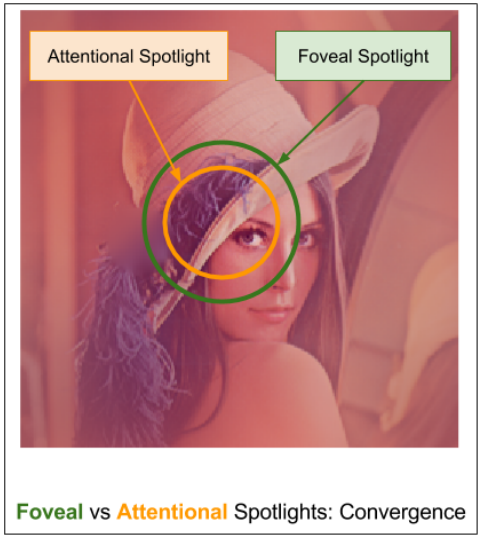
- Simons, Chabris, 1999: `https://www.youtube.com/watch?v=vJG698U2Mvo`

Внимание

- Мы делаем *саккады*, причём это тоже не всегда помогает:



A typical saccade trace

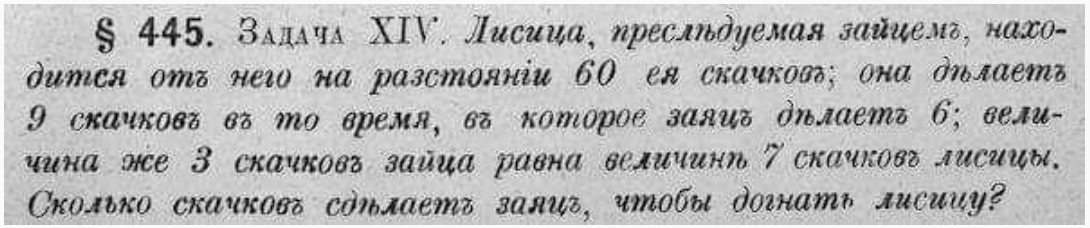


Foveal vs Attentional Spotlights: Convergence



Foveal vs Attentional Spotlights: Divergence

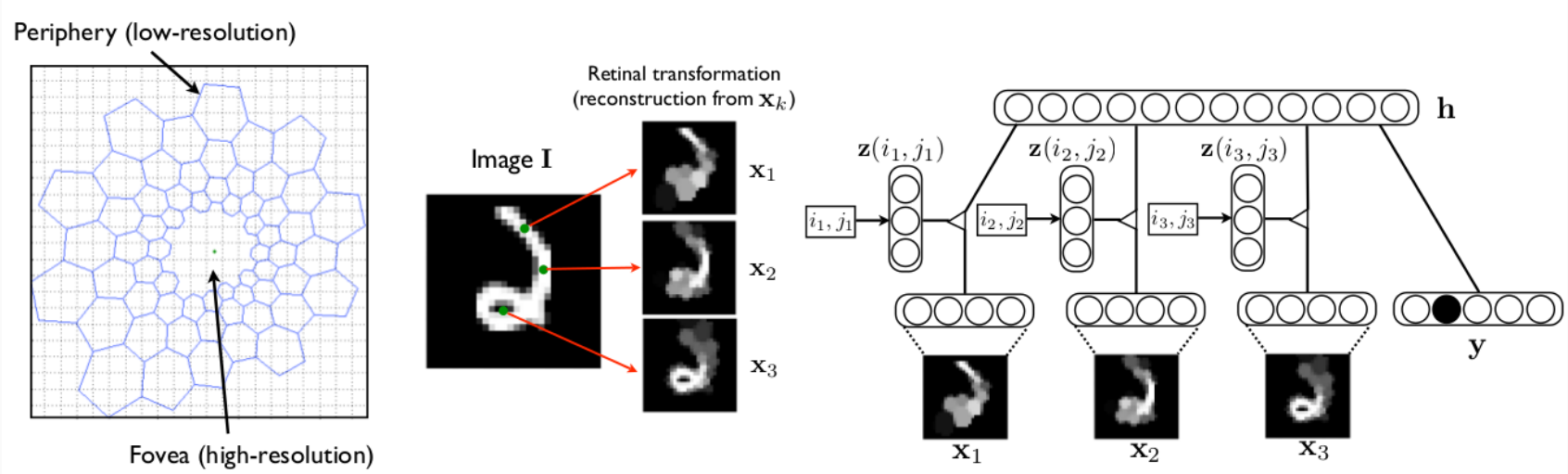
- Или вот не совсем зрительный пример:



§ 445. Задача XIV. *Лисица, преслѣдуемая зайцемъ, находится отъ него на разстояніи 60 ея скачковъ; она дѣлаетъ 9 скачковъ въ то время, въ которое заяцъ дѣлаетъ 6; величина же 3 скачковъ зайца равна величинѣ 7 скачковъ лисицы. Сколько скачковъ сдѣлаетъ заяцъ, чтобы догнать лисицу?*

Foveal glimpses

- В нейросети мы тоже хотим осознанно понимать, «на что» смотреть.
- Одна из первых работ (Larochelle, Hinton, 2010):

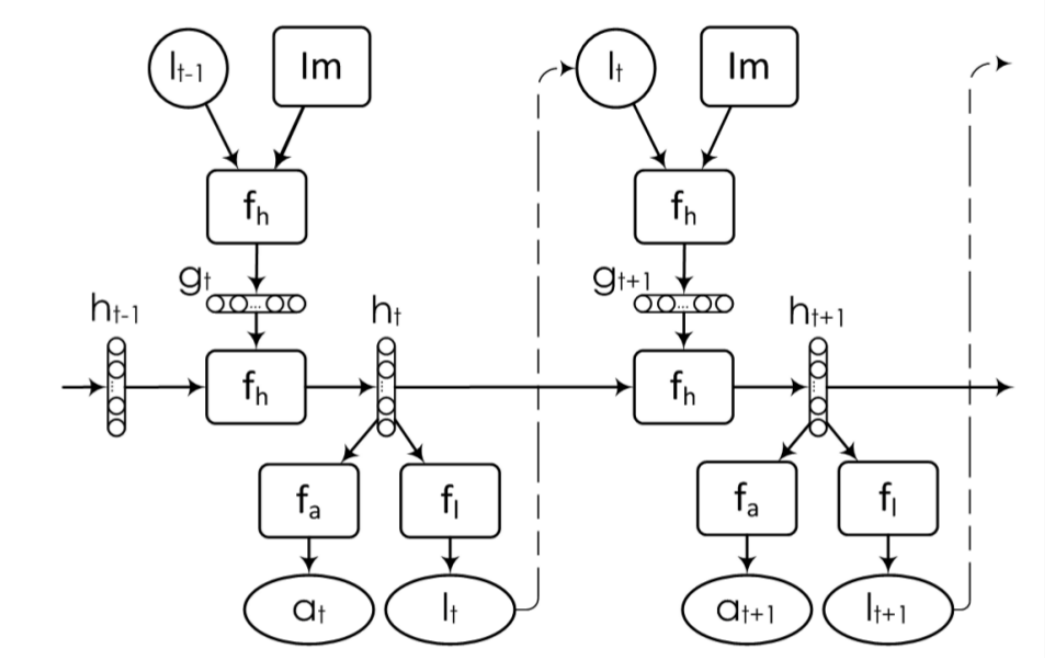


- Пытаются моделировать положения фиксаций и строить последовательность при помощи RBM.
- Последовательность – значит...

Рекуррентные модели зрительного внимания

Recurrent visual attention

- По-настоящему всё появилось в (Mnih et al., 2014), «Recurrent Models of Visual Attention»:
  - из предыдущего $\mathbf{h}_{t-1}$ и положения $\mathcal{L}_t$ для нового «взгляда» f_g делает $\mathbf{g}_t$, вход для шага t;
  - из $\mathbf{h}_{t-1}$ и $\mathbf{g}_t$ функцией f_h получается $\mathbf{h}_t$;
  - из него – «действие» $a_t = f_a(\mathbf{h}_t)$ и положение следующего «взгляда» $\mathcal{L}_{t+1} = f_l(\mathbf{h}_t)$.



Recurrent visual attention

- Давайте разберёмся в модели формально:

$$\begin{aligned}
\mathbf{g}_t &= f_g(\mathbf{x}_t, \mathbf{l}_{t-1}; \theta_g), \\
\mathbf{h}_t &= f_h(\mathbf{h}_{t-1}, \mathbf{g}_t; \theta_h), \\
\mathbf{l}_t &\sim p(\cdot \mid f_l(\mathbf{h}_t; \theta_l)), \\
a_t &\sim p(\cdot \mid f_a(\mathbf{h}_t; \theta_a)).
\end{aligned}$$

- После очередного действия получается новое наблюдение $\mathbf{x}_{t+1}$ и награда r_t, которая будет скорее всего в конце, после всех шагов, за правильную классификацию.
- Что это напоминает?..

Recurrent visual attention

- ...о да, это reinforcement learning!
- Выучить надо стохастическую стратегию $\pi((\mathbf{l}_t, a_t) \mid \mathbf{s}_{1:t}; \theta)$, которая по истории будет выдавать следующее действие.
- У нас π задаётся через RNN, а оптимизировать надо

$$J(\theta) = \mathbb{E}_{p(\mathbf{s}_{1:T};\theta)}[R] = \mathbb{E}_{p(\mathbf{s}_{1:T};\theta)}\left[\sum_{t=1}^{T} r_t\right].$$

- Выглядит очень сложно – ожидание по последовательностям действий, т.е. по пространству большой размерности.
- Но есть методы – давайте сделаем preview тому, что потом будет в reinforcement learning.

Recurrent visual attention

- (Williams, 1992): алгоритм REINFORCE, в котором доказывается и используется выборочная оценка этого ожидания. Давайте выведем, а потом применим к нашему случаю...
- Нам надо оптимизировать

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{T} r_t(s_t, a_t) \right] \approx \frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} r(s_t^{(i)}, a_t^{(i)}),$$

  где мы взяли M примеров траекторий τ.
- Определим $r(\tau) = \sum_t r(s_t, a_t)$. Тогда

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [r(\tau)] = \int \pi_\theta(\tau) r(\tau) \mathrm{d}\tau,$$

  т.е. тот самый страшный интеграл по траекториям.
- Но оказывается, что можно продифференцировать по θ...

Recurrent visual attention

- Продифференцируем по θ:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta \pi_\theta(\tau) r(\tau) \mathrm{d}\tau \\
&= \int \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} r(\tau) \mathrm{d}\tau \\
&= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) \mathrm{d}\tau \\
&= \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) r(\tau) \right] \\
&\approx \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta \log \pi_\theta(\tau^{(i)}) r^{(i)}(\tau),
\end{aligned}$$

если приблизить выборкой; но сначала давайте ещё посмотрим на $\pi_\theta(\tau)$...

Recurrent visual attention

- Вероятность определяется как

$$\pi_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t).$$

- Берём логарифм, а потом заметим, что от θ зависят только действия:

$$\nabla_\theta \log \pi_\theta(\tau) =$$
$$= \nabla_\theta \left(\log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t \mid s_t) + \sum_{t=1}^{T} \log p(s_{t+1} \mid s_t, a_t) \right) =$$
$$= \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t).$$

Recurrent visual attention

- Итого получается вполне tractable градиент:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[r(\tau) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$
$$\approx \frac{1}{M} \sum_{i=1}^{M} r(\tau^{(i)}) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)}).$$

- У нас тоже можно считать, что награда R даётся целиком:

$$\nabla_\theta J(\theta) \approx \frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} \nabla_\theta \log \pi \left(\mathbf{l}_t^{(i)}, a_t^{(i)} \mid \mathbf{s}_{1:t}^{(i)}; \theta \right) R^{(i)}.$$

- Т.е. надо уметь считать $\log \pi \left(\mathbf{l}_t^{(i)}, a_t^{(i)} \mid \mathbf{s}_{1:t}^{(i)}; \theta \right)$, но в случае RNN это просто градиент сети, который можно посчитать через backpropagation.
- Ещё можно сделать частично supervised loss на последнем шаге, где мы знаем классификацию.

Recurrent visual attention

- Результаты:



(a) Translated MNIST inputs.



(b) Cluttered Translated MNIST inputs.

(a) 28x28 MNIST

Model	Error
FC, 2 layers (256 hiddens each)	1.69%
Convolutional, 2 layers	1.21%
RAM, 2 glimpses, 8×8, 1 scale	3.79%
RAM, 3 glimpses, 8×8, 1 scale	1.51%
RAM, 4 glimpses, 8×8, 1 scale	1.54%
RAM, 5 glimpses, 8×8, 1 scale	1.34%
RAM, 6 glimpses, 8×8, 1 scale	1.12%
RAM, 7 glimpses, 8×8, 1 scale	**1.07%**

(b) 60x60 Translated MNIST

Model	Error
FC, 2 layers (64 hiddens each)	6.42%
FC, 2 layers (256 hiddens each)	2.63%
Convolutional, 2 layers	1.62%
RAM, 4 glimpses, 12×12, 3 scales	1.54%
RAM, 6 glimpses, 12×12, 3 scales	**1.22%**
RAM, 8 glimpses, 12×12, 3 scales	**1.2%**

Recurrent visual attention

- Результаты:

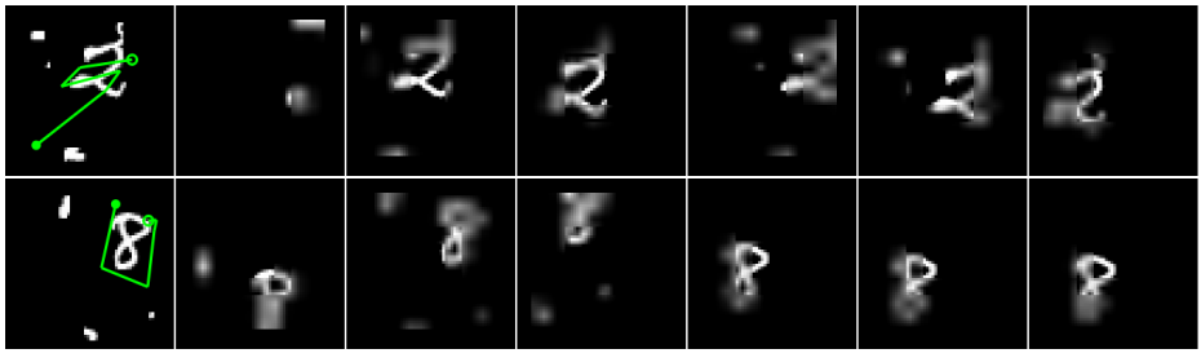
(a) 60x60 Cluttered Translated MNIST

Model	Error
FC, 2 layers (64 hiddens each)	28.58%
FC, 2 layers (256 hiddens each)	11.96%
Convolutional, 2 layers	8.09%
RAM, 4 glimpses, 12 × 12, 3 scales	4.96%
RAM, 6 glimpses, 12 × 12, 3 scales	4.08%
RAM, 8 glimpses, 12 × 12, 3 scales	4.04%
RAM, 8 random glimpses	14.4%

(b) 100x100 Cluttered Translated MNIST

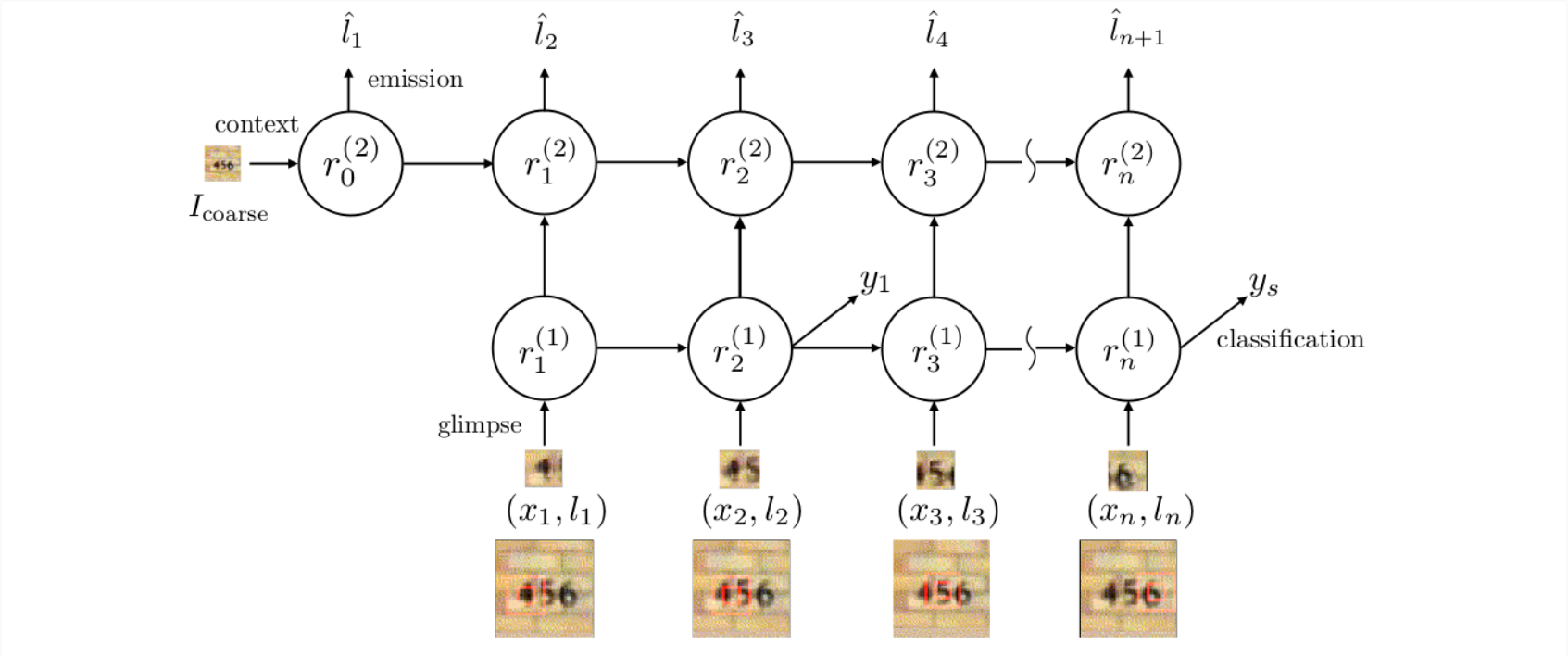
Model	Error
Convolutional, 2 layers	14.35%
RAM, 4 glimpses, 12 × 12, 4 scales	9.41%
RAM, 6 glimpses, 12 × 12, 4 scales	8.31%
RAM, 8 glimpses, 12 × 12, 4 scales	8.11%
RAM, 8 random glimpses	28.4%

- А вот как внимание гуляет по картинке:



Recurrent visual attention

- В следующей работе (Ba et al., 2015) сделали глубокую модель:



- Кстати, обучали по-другому, вариационными методами. Как это?..

Recurrent visual attention

- Нам нужно классифицировать, т.е. $p(y \mid \mathbf{x}, \theta)$ максимизировать.
- Маргинализуем по положениям glimpses:

$$\log p(y \mid \mathbf{x}, \theta) = \log \sum_l p(l \mid \mathbf{x}, \theta) p(y \mid l, \mathbf{x}, \theta).$$

- Запишем вариационную нижнюю оценку свободной энергии (как её получить?):

$$\begin{aligned} \log \sum_l p(l \mid \mathbf{x}, \theta) p(y \mid l, \mathbf{x}, \theta) &\geq \sum_l p(l \mid \mathbf{x}, \theta) \log p(y, l \mid \mathbf{x}, \theta) + H[l] \\ &= \sum_l p(l \mid \mathbf{x}, \theta) \log p(y \mid l, \mathbf{x}, \theta). \end{aligned}$$

Recurrent visual attention

- И теперь можно брать производные:

$$\frac{\partial J}{\partial \theta} = \sum_l p(l \mid \mathbf{x}, \theta) \frac{\partial \log p(y \mid l, \mathbf{x}, \theta)}{\partial \theta} + \sum_l \log p(y \mid l, \mathbf{x}, \theta) \frac{\partial p(l \mid \mathbf{x}, \theta)}{\partial \theta}$$

$$= \sum_l p(l \mid \mathbf{x}, \theta) \left[\frac{\partial \log p(y \mid l, \mathbf{x}, \theta)}{\partial \theta} + \log p(y \mid l, \mathbf{x}, \theta) \frac{\partial \log p(l \mid \mathbf{x}, \theta)}{\partial \theta} \right].$$

- А эту сумму уже будем приближать выборкой:

$$\frac{\partial J}{\partial \theta} \approx \frac{1}{M} \sum_{i=1}^{M} \left[\frac{\partial \log p(y \mid l^{(i)}, \mathbf{x}, \theta)}{\partial \theta} + \log p(y \mid l^{(i)}, \mathbf{x}, \theta) \frac{\partial \log p(l^{(i)} \mid \mathbf{x}, \theta)}{\partial \theta} \right],$$

$$\text{где} \quad l^{(i)} \sim p(l_n \mid \mathbf{x}, \theta) = N(l_n \mid \hat{l}_n, \Sigma).$$

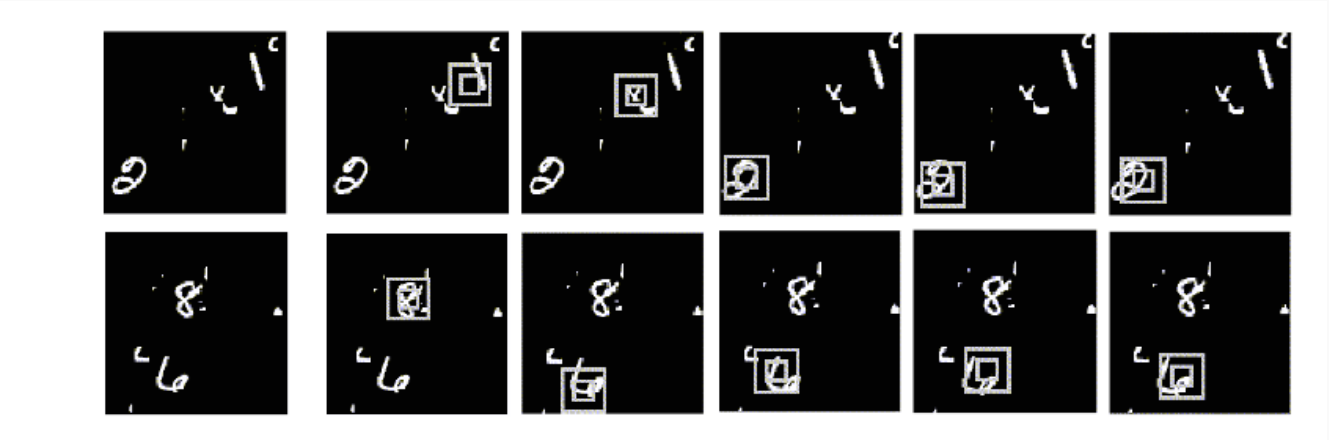
- И это уже алгоритм: сэмплируем glimpses, потом используем их в backpropagation.
- Как и в (Mnih et al., 2014), надо бы уменьшить дисперсию; для этого вычитают baseline, пока не будем углубляться.

Recurrent visual attention

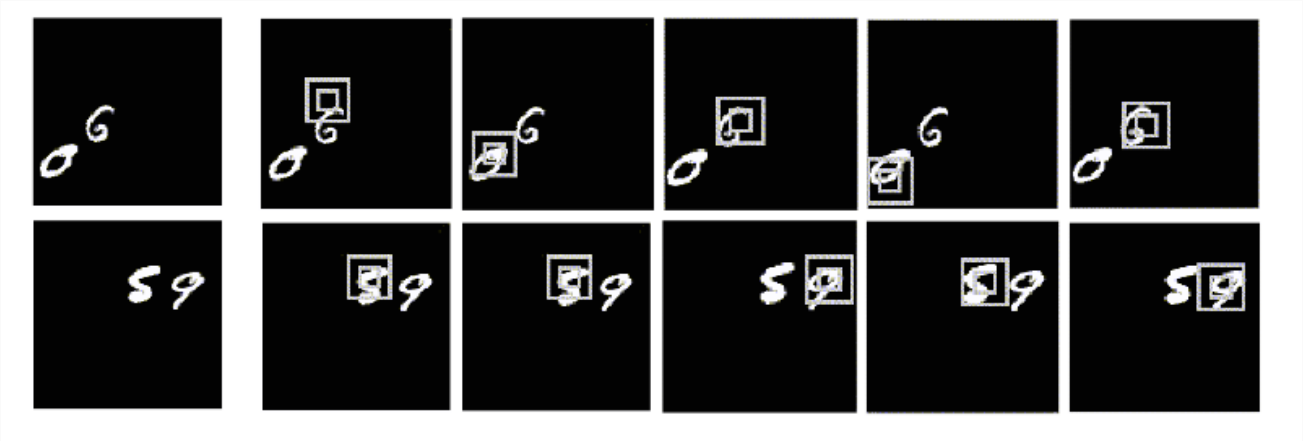
- Получился интересный результат – мы увидели, что примерно один и тот же алгоритм может получиться с двух разных сторон:
  - из обучения с подкреплением через REINFORCE;
  - из вариационной оценки собственно целевой функции.
- Важный гиперпараметр – размер glimpse, т.е. как переводить единицы измерений в системе координат glimpses в пиксели.
- То же самое легко расширить на последовательную классификацию нескольких объектов – просто фиксированное число glimpses на объект, потом классифицируем, плюс терминальное действие в конце всего.

Recurrent visual attention

- Вот как это работает на распознавании двух цифр:



- Любопытно, что на сложении по-другому:



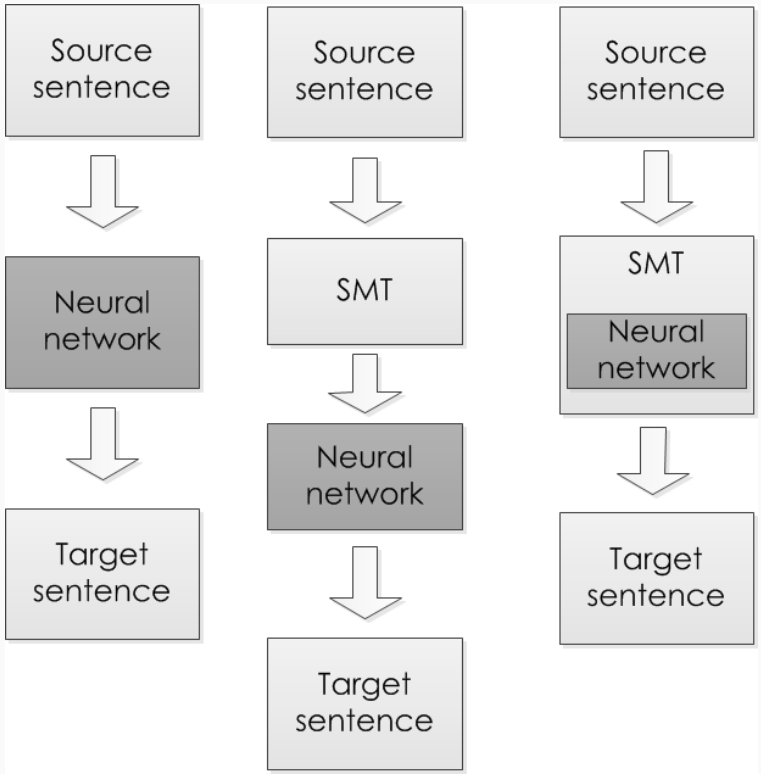
Машинный перевод: encoder-decoder и внимание

Машинный перевод

- Перевод – очень хорошая задача:
  - очевидно очень практическая;
  - очевидно очень высокоуровневая, требует понимания;
  - считается довольно неплохо квантифицируемой (BLEU, TER – хотя см. выше);
  - имеет большие доступные датасеты параллельных переводов.

Машинный перевод

- Статистический машинный перевод (statistical machine translation, SMT): моделируем условную вероятность $p(y \mid x)$ перевода y при условии исходного текста x.
- Классический SMT: моделируем $\log p(y \mid x)$ линейной комбинацией признаков, строим признаки.



Машинный перевод

- Нам больше интересно моделирование sequence-to-sequence:
  - RNN естественным образом моделирует последовательность $X = (x_1, x_2, \dots, x_T)$ как $p(x_1)$, $p(x_2 \mid x_1)$, ..., $p(x_T \mid x_{<T}) = p(x_T \mid x_{T-1}, \dots, x_1)$, и теперь $p(X)$ – это просто

$$p(X) = p(x_1)p(x_2 \mid x_1) \dots p(x_k \mid x_{<k}) \dots p(x_T \mid x_{<T});$$

  - так RNN и в языковых моделях используются;
  - предсказываем следующее слово на основе скрытого состояния и предыдущего слова;
- Как применить эту идею к переводу?

Метрики качества для sequence-to-sequence моделей

- Дальше будет самое интересное: машинный перевод, диалоговые модели, ответ на вопросы.
- Но как мы будем оценивать NLP-модели, которые генерируют текст?
- Есть метрики качества, которые сравнивают результат с правильными ответами:
  - BLEU (Bilingual Evaluation Understudy): перевзвешенная precision (в т.ч. для нескольких правильных ответов);
  - METEOR: гармоническое среднее precision и recall по униграммам;
  - TER (Translation Edit Rate): число исправлений между выходом и правильным ответом, делённое на среднее число слов;
  - LEPOR: комбинируем базовые факторы и метрики с настраиваемыми параметрами.
- Есть ещё куча метрик, связанных с представлениями слов и предложений (хотим поближе к правильному ответу).
- Одна только проблема...

Метрики качества для sequence-to-sequence моделей

- ...всё это вообще не работает.

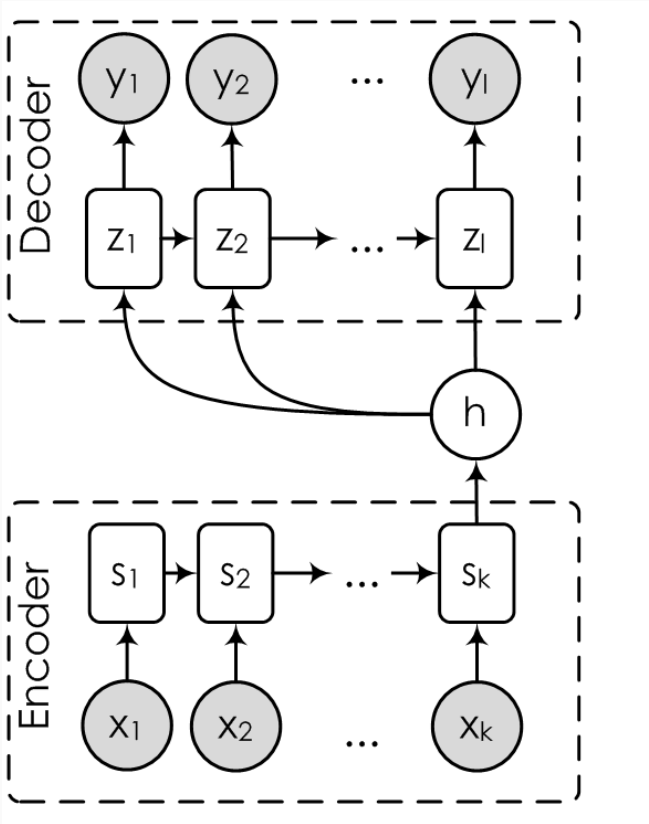
	Twitter				Ubuntu			
Metric	**Spearman**	p-value	**Pearson**	p-value	**Spearman**	p-value	**Pearson**	p-value
Greedy	0.2119	0.034	0.1994	0.047	0.05276	0.6	0.02049	0.84
Average	0.2259	0.024	0.1971	0.049	-0.1387	0.17	-0.1631	0.10
Extrema	0.2103	0.036	0.1842	0.067	0.09243	0.36	-0.002903	0.98
METEOR	0.1887	0.06	0.1927	0.055	0.06314	0.53	0.1419	0.16
BLEU-1	0.1665	0.098	0.1288	0.2	-0.02552	0.8	0.01929	0.85
BLEU-2	0.3576	< 0.01	0.3874	< 0.01	0.03819	0.71	0.0586	0.56
BLEU-3	0.3423	< 0.01	0.1443	0.15	0.0878	0.38	0.1116	0.27
BLEU-4	0.3417	< 0.01	0.1392	0.17	0.1218	0.23	0.1132	0.26
ROUGE	0.1235	0.22	0.09714	0.34	0.05405	0.5933	0.06401	0.53
Human	0.9476	< 0.01	1.0	0.0	0.9550	< 0.01	1.0	0.0

Table 3: Correlation between each metric and human judgements for each response. Correlations shown in the human row result from randomly dividing human judges into two groups.

- Тут нужно что-то новое. И пока не совсем ясно, что именно.

Encoder-decoder архитектуры

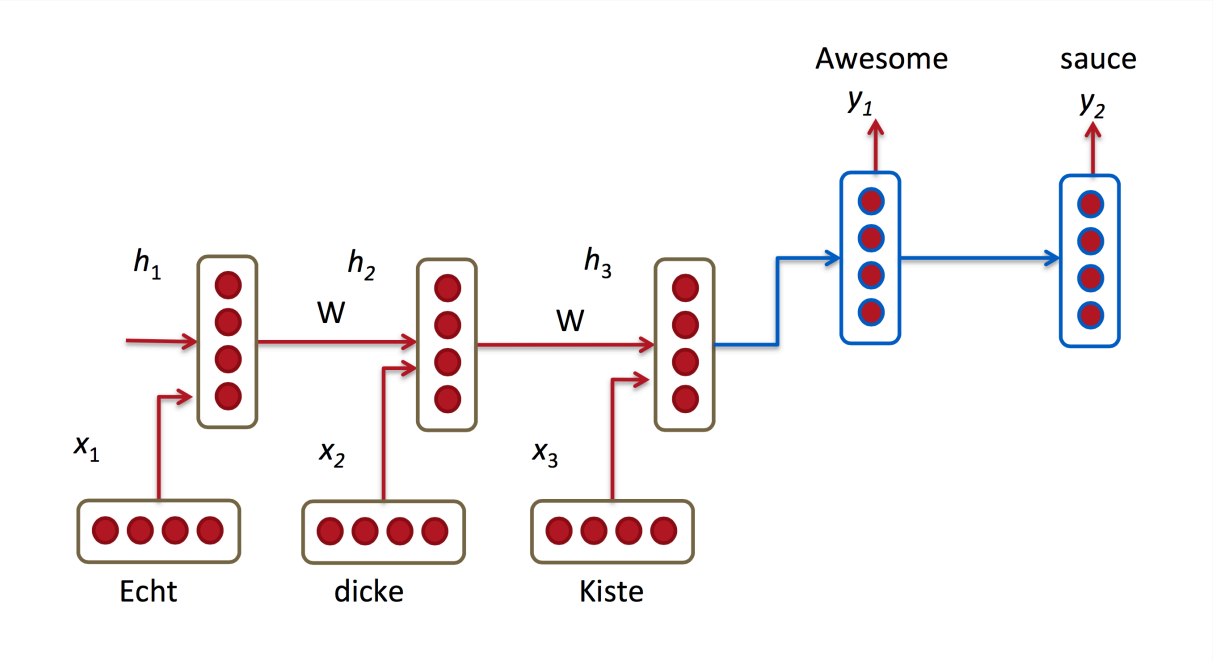
- Encoder-decoder архитектуры (Sutskever et al., 2014; Cho et al., 2014):



- Сначала кодируем, потом декодируем обратно.

Encoder-decoder архитектуры

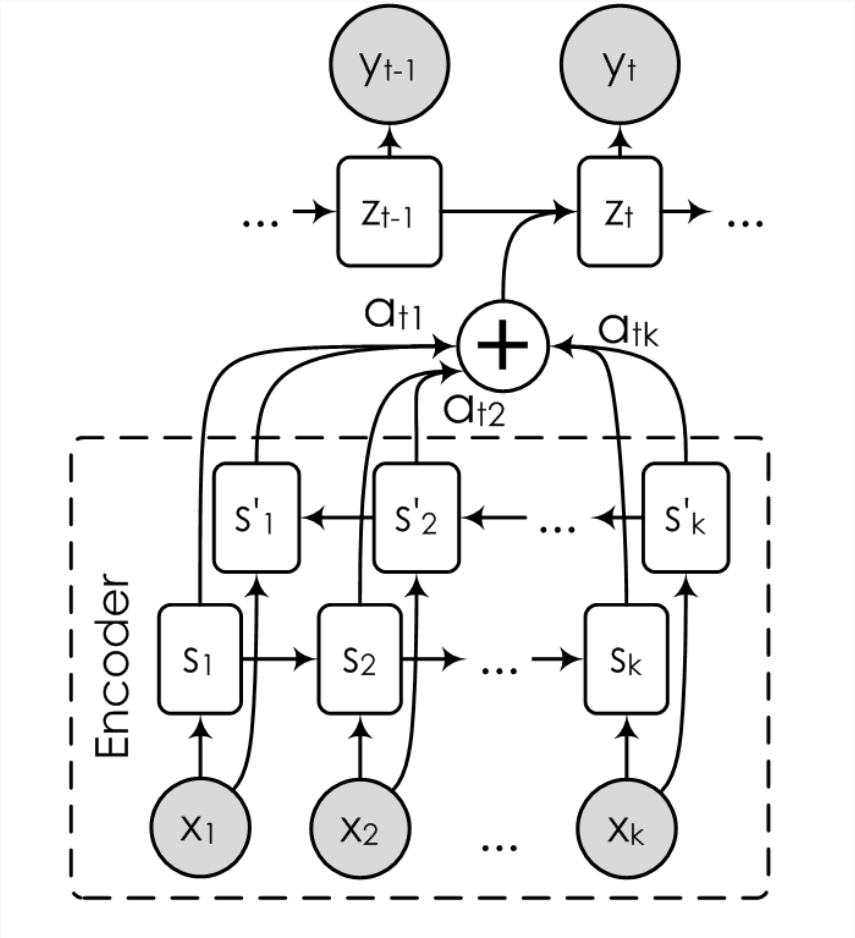
- Так же может работать и с переводом.



- Проблема: надо сжимать всё предложение в один вектор.
- С длинными участками текста это вообще перестаёт работать.

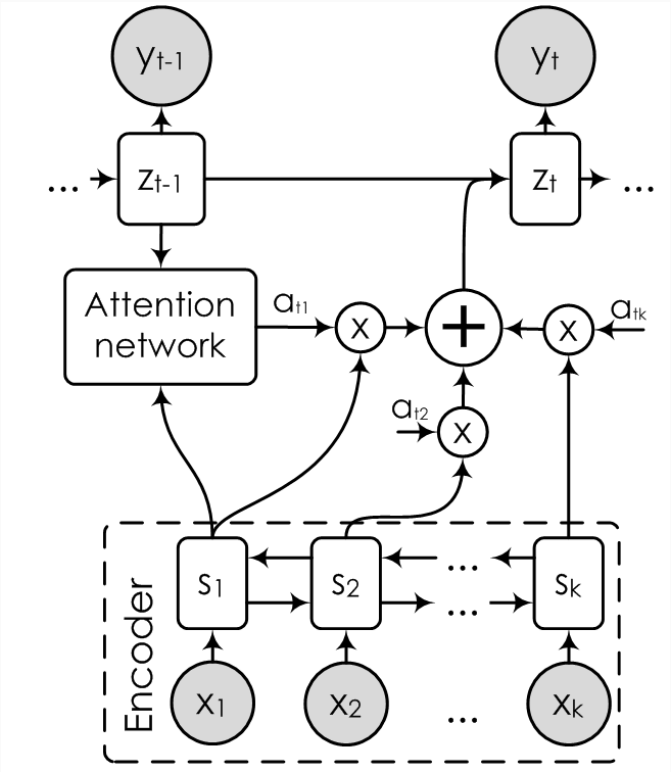
Внимание в нейронных сетях

- Решение: давайте обучим специальные веса, показывающие, насколько та или иная часть входа важна для текущего выхода.
- Прямое применение – двунаправленный LSTM плюс внимание (Bahdanau et al. 2014):



Внимание в нейронных сетях

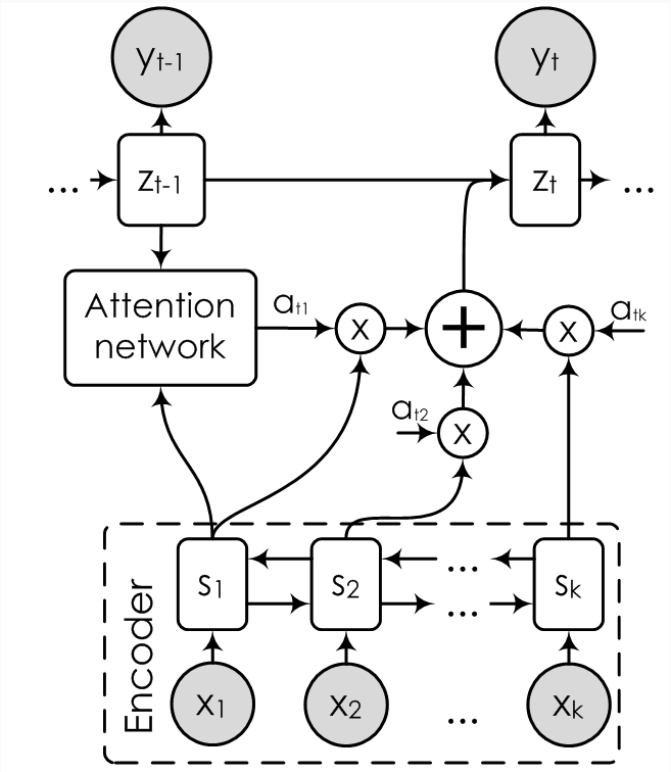
- *Мягкое внимание* (soft attention) (Luong et al. 2015a; 2015b; Jean et al. 2015):
  - encoder – двунаправленная RNN, есть оба контекста;
  - сеть внимания выдаёт оценку релевантности – надо ли переводить это слово прямо сейчас?



Внимание в нейронных сетях

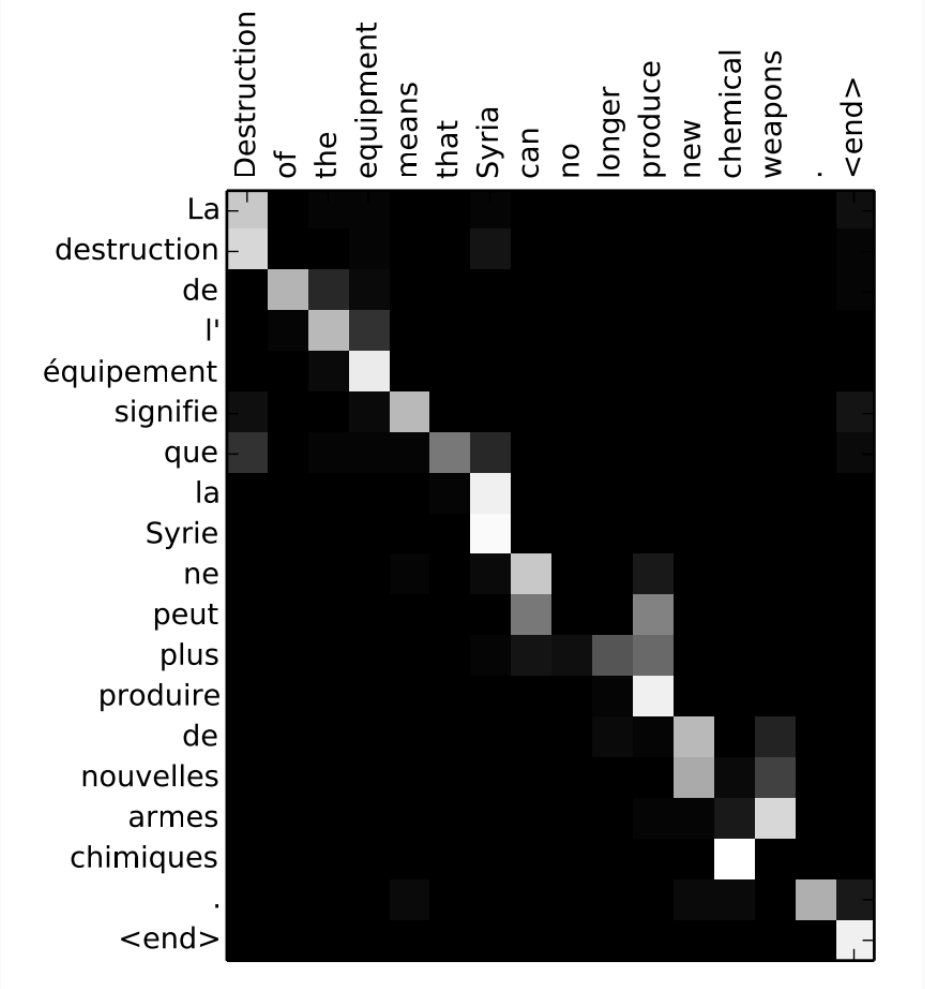
- Формально очень просто: считаем веса внимания α_{tj} и перевзвешиваем векторы контекстов:

$$e_{tj} = a(z_{t-1}, j), \quad \alpha_{tj} = \mathrm{softmax}(e_{tj}; e_{t*}),$$
$$c_t = \sum_j \alpha_{tj} h_j, \text{ и теперь } z_t = f(s_{t-1}, y_{t-1}, c_i).$$



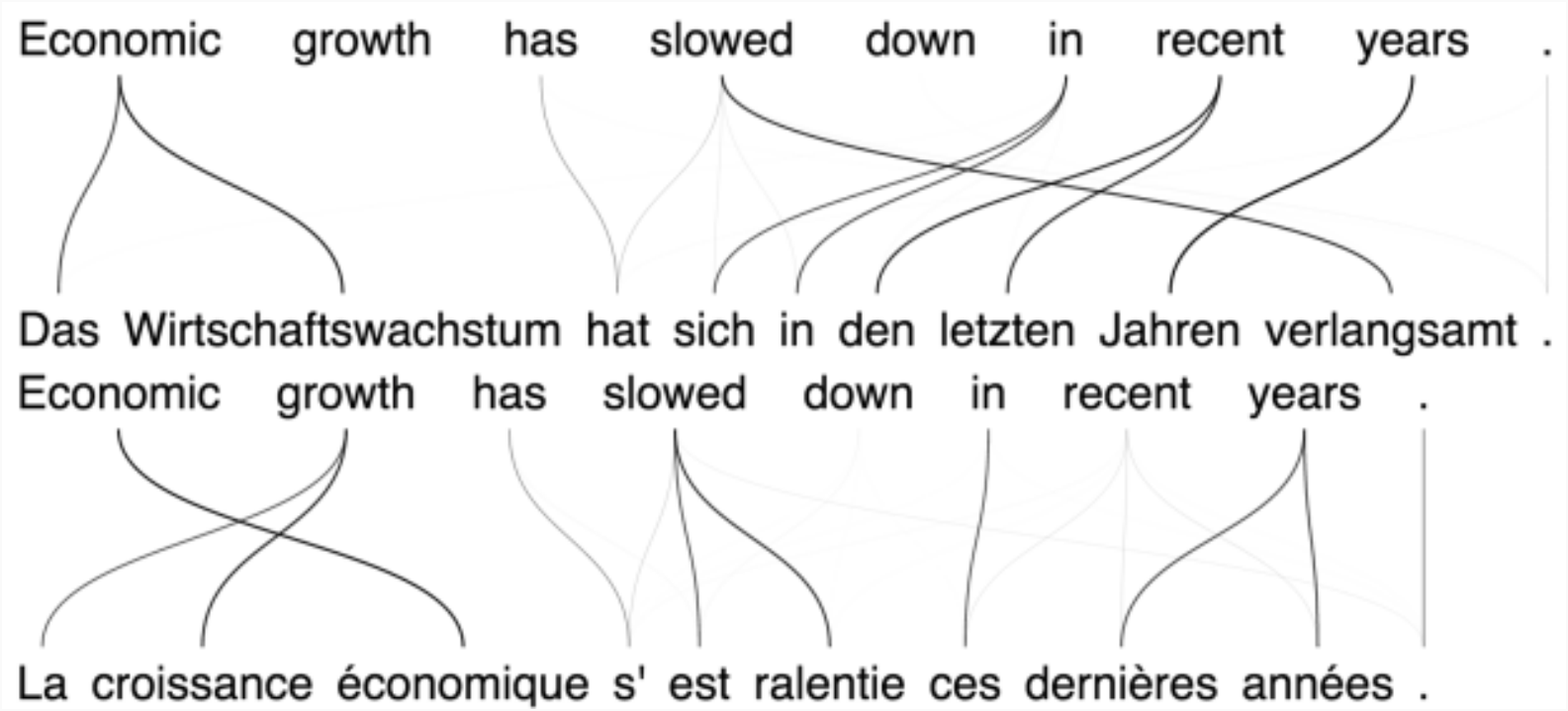
Внимание в нейронных сетях

- В результате можно визуализировать, на что смотрит сеть:



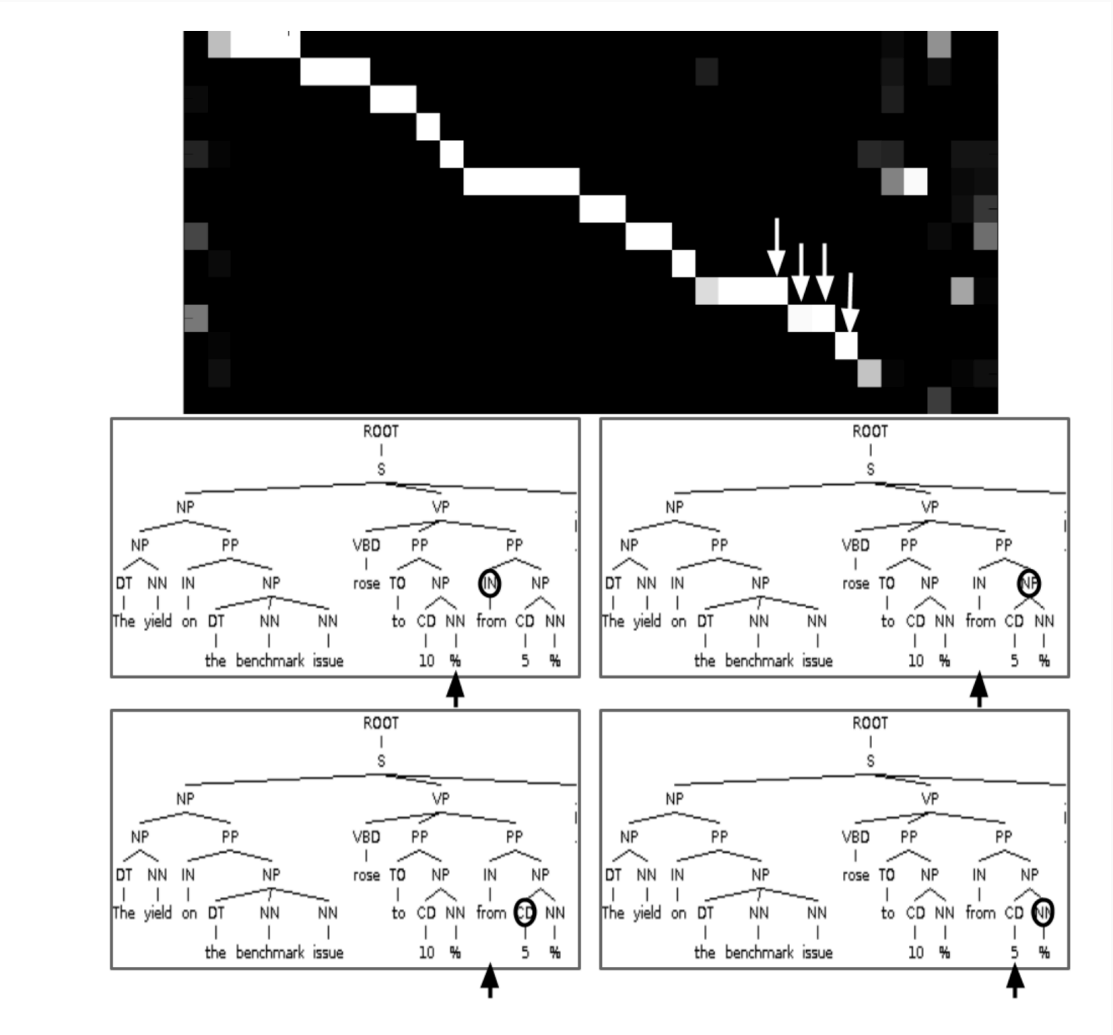
Внимание в нейронных сетях

- Получается гораздо лучше порядок слов:



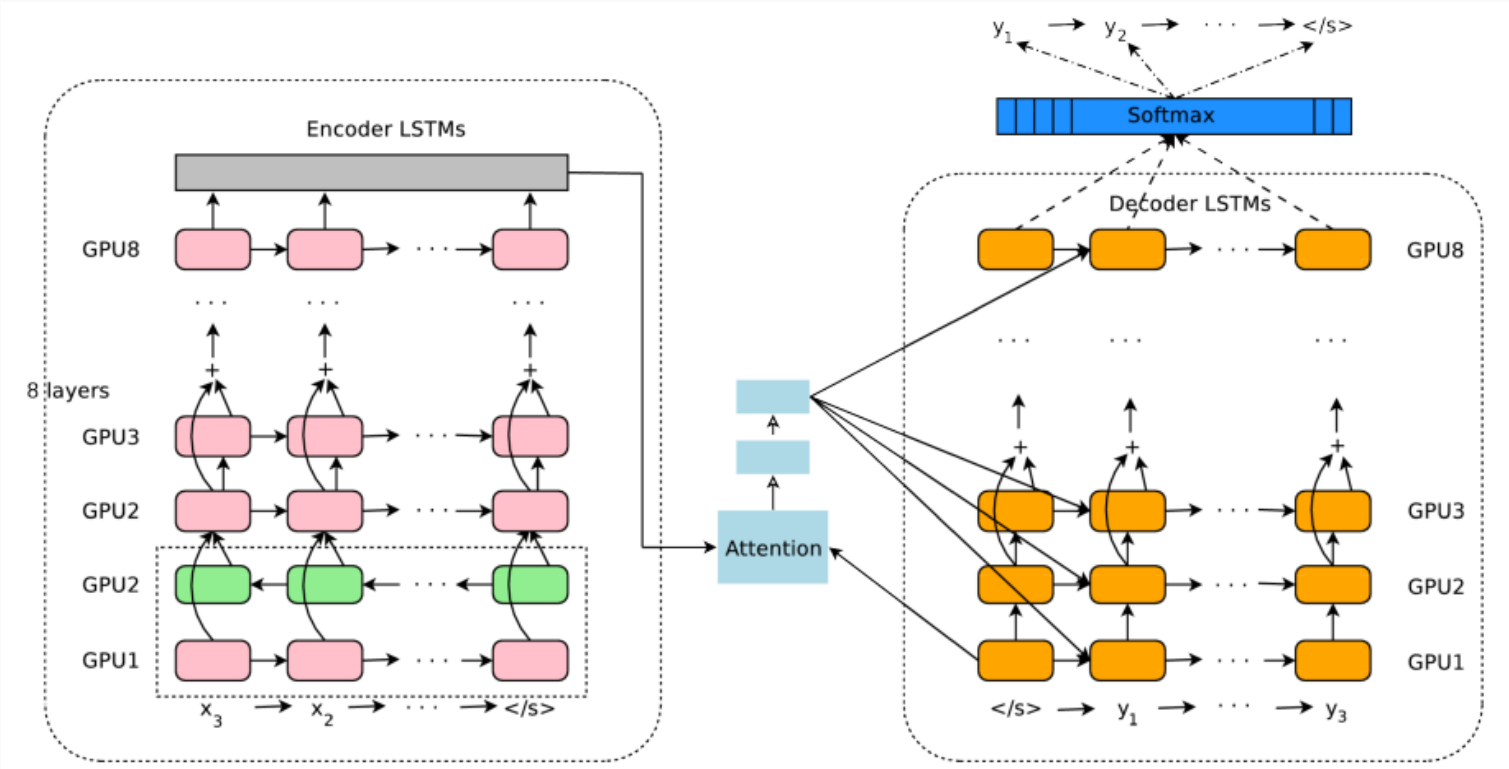
Внимание в нейронных сетях

- Другая необычная работа – «Grammar as a Foreign Language» (Vinyals et al., 2015)



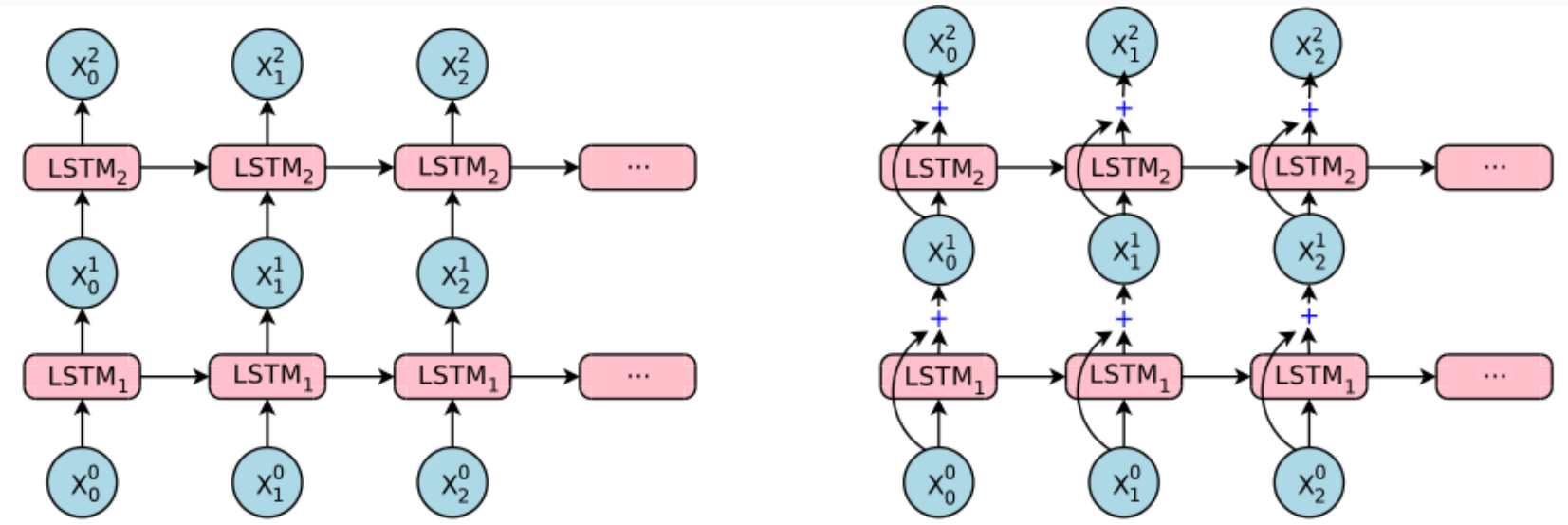
Google Translate

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
  - как на самом деле работает Google Translate;
  - базовая архитектура та же самая: encoder, decoder, attention;
  - RNN глубокие, по 8 уровней в encoder и decoder:



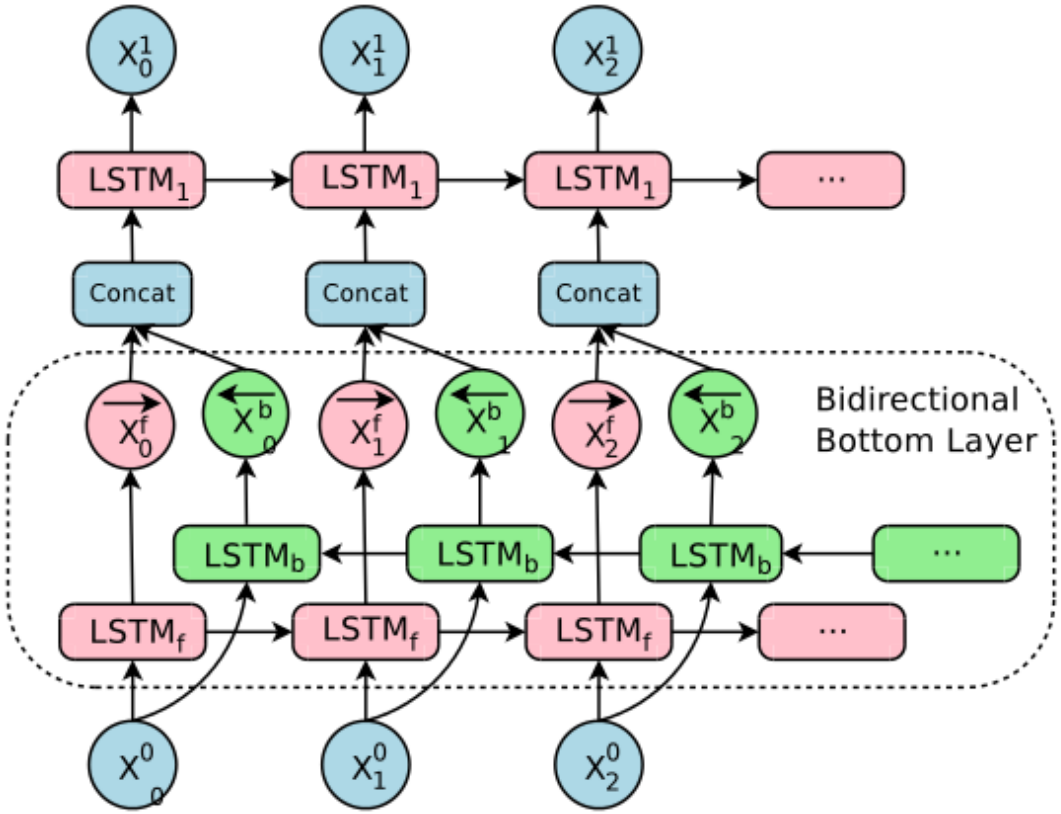
Google Translate

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
  - просто stacked LSTM перестают работать далее 4-5 уровней;
  - поэтому добавляют остаточные связи, как в ResNet:



Google Translate

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
  - нижний уровень, естественно, двунаправленный:



Google Translate

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
- в GNMT ещё две идеи о сегментации слов:
  - *wordpiece model*: разбить слова на кусочки (отдельной моделью); пример из статьи:

    `Jet makers feud over seat width with big orders at stake`

    превращается в

    `_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake`

  - *mixed word/character model*: конвертировать слова, не попадающие в словарь, в последовательность букв-токенов; пример из статьи:

    `Miki` превращается в `<B>M <M>i <M>k <E>i`

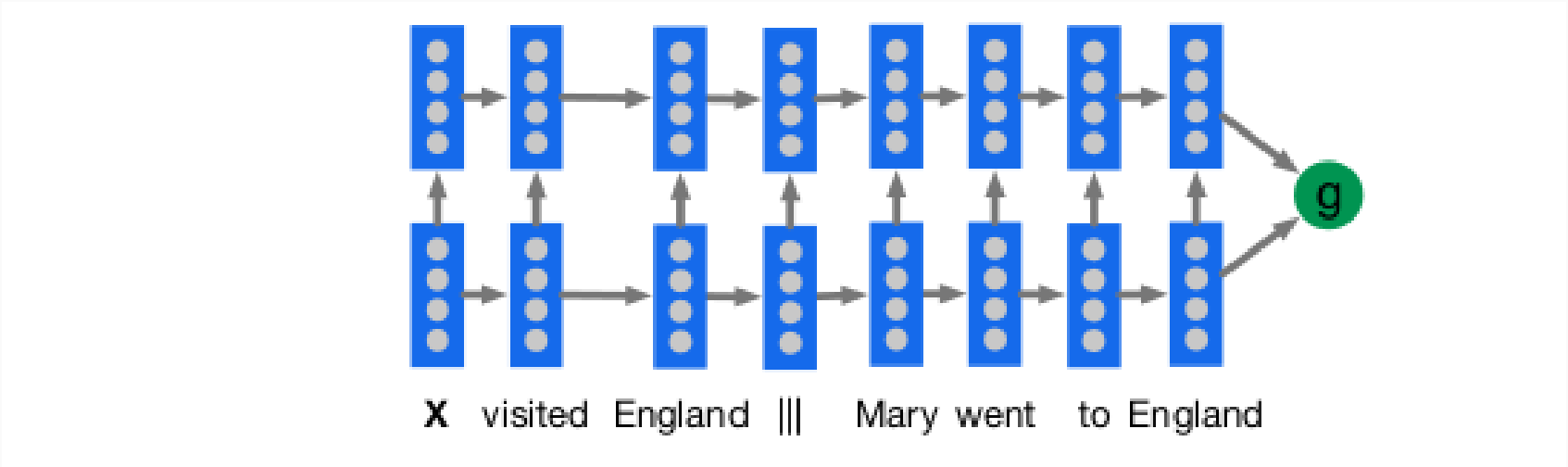
Teaching machines to read

- (Hermann et al., 2015): «Teaching machines to read and comprehend» (Google DeepMind)
- Предлагают новый способ построить датасет для понимания, автоматически создавая тройки (context, query, answer) из текстов новостей и т.п.

Original Version	Anonymised Version
Context	
The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." ...	the *ent381* producer allegedly struck by *ent212* will not press charges against the " *ent153* " host , his lawyer said friday . *ent212* , who hosted one of the most - watched television shows in the world , was dropped by the *ent381* wednesday after an internal investigation by the *ent180* broadcaster found he had subjected producer *ent193* " to an unprovoked physical and verbal attack . " ...
Query	
Producer **X** will not press charges against Jeremy Clarkson, his lawyer says.	producer **X** will not press charges against *ent212* , his lawyer says .
Answer	
Oisin Tymon	*ent193*

Teaching machines to read

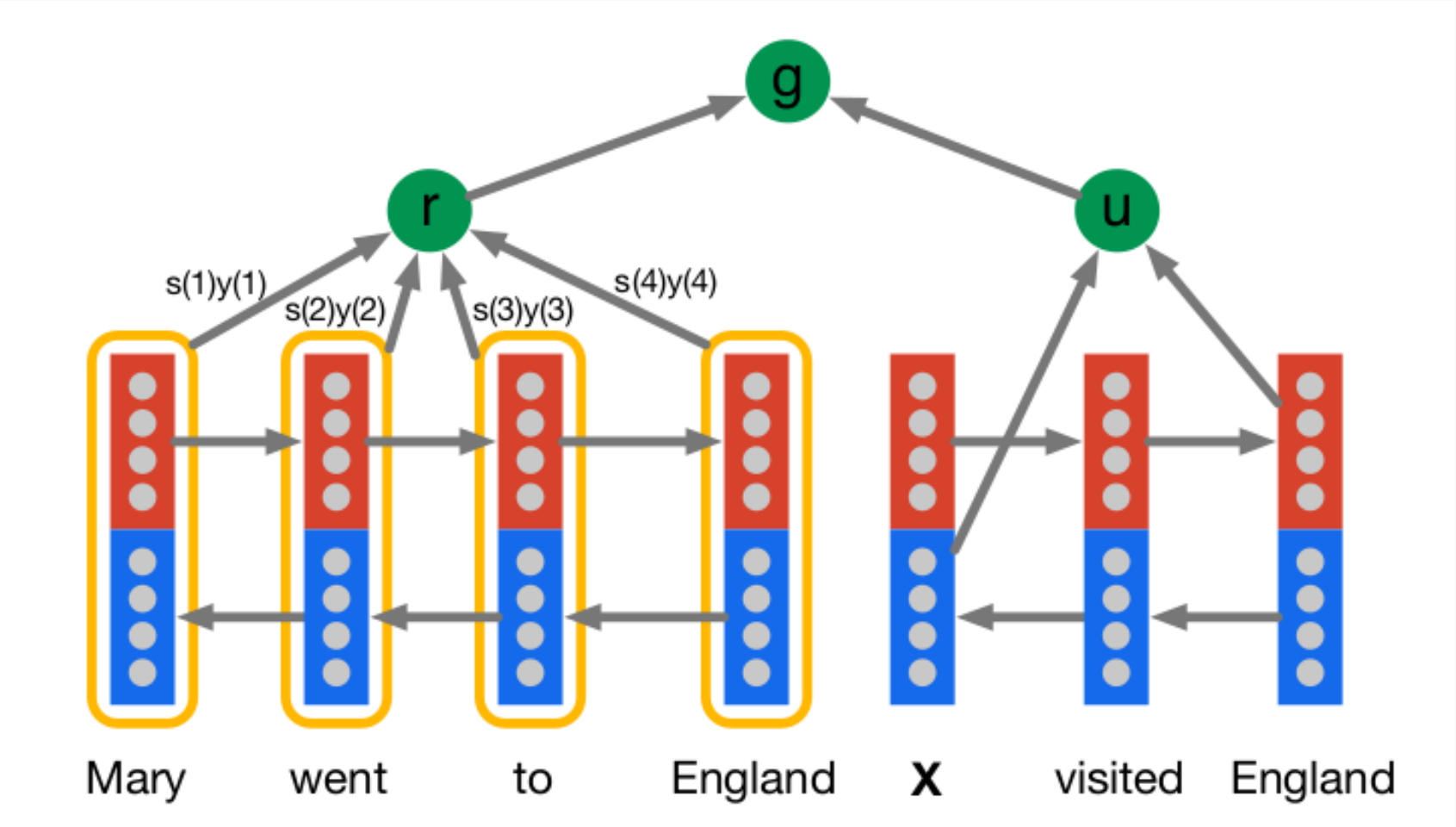
- Базовая модель – глубокий LSTM:



- Но так, конечно, совсем плохо работает.

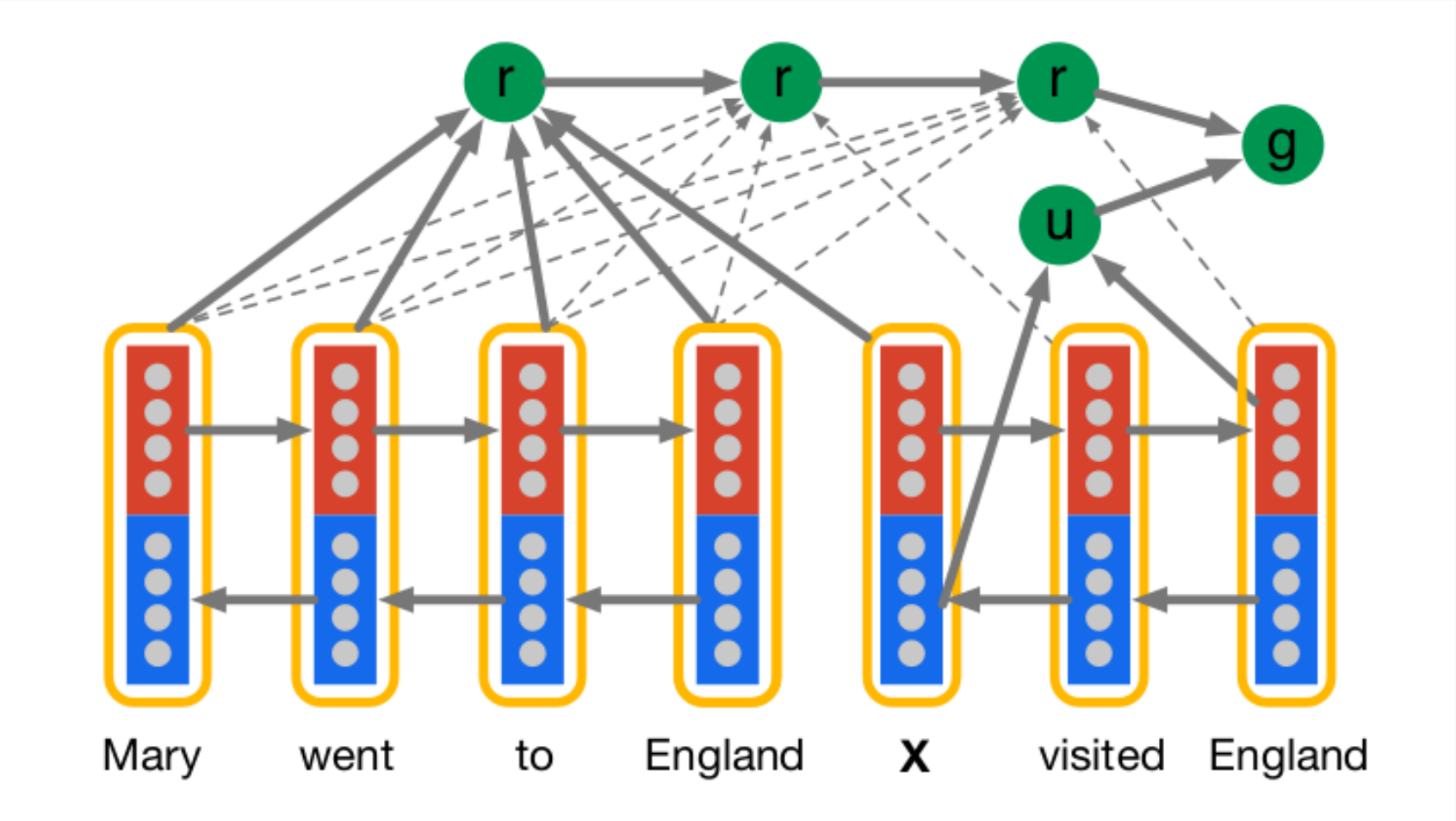
Teaching machines to read

- Attentive Reader: обучаемся, на какую часть документа смотреть



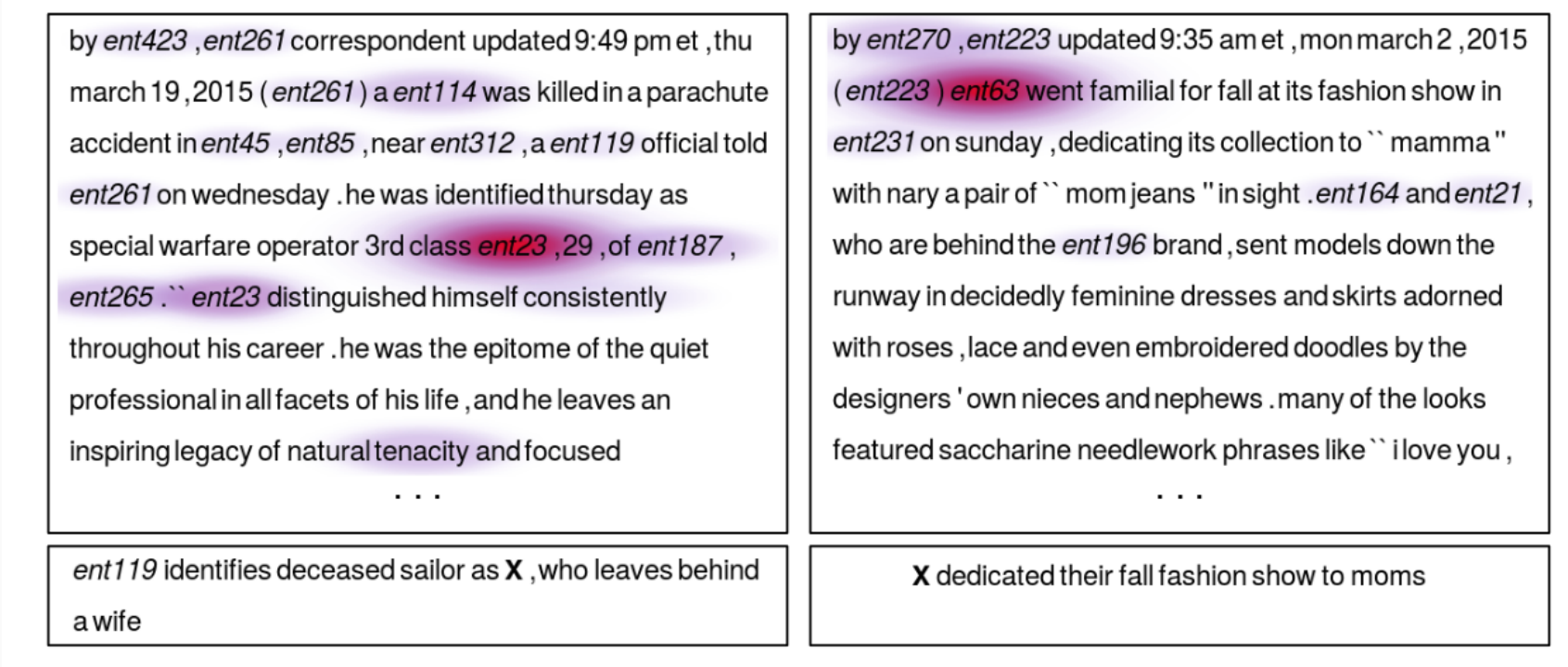
Teaching machines to read

- Impatient Reader: можем перечитывать нужные части документа по мере прочтения запроса



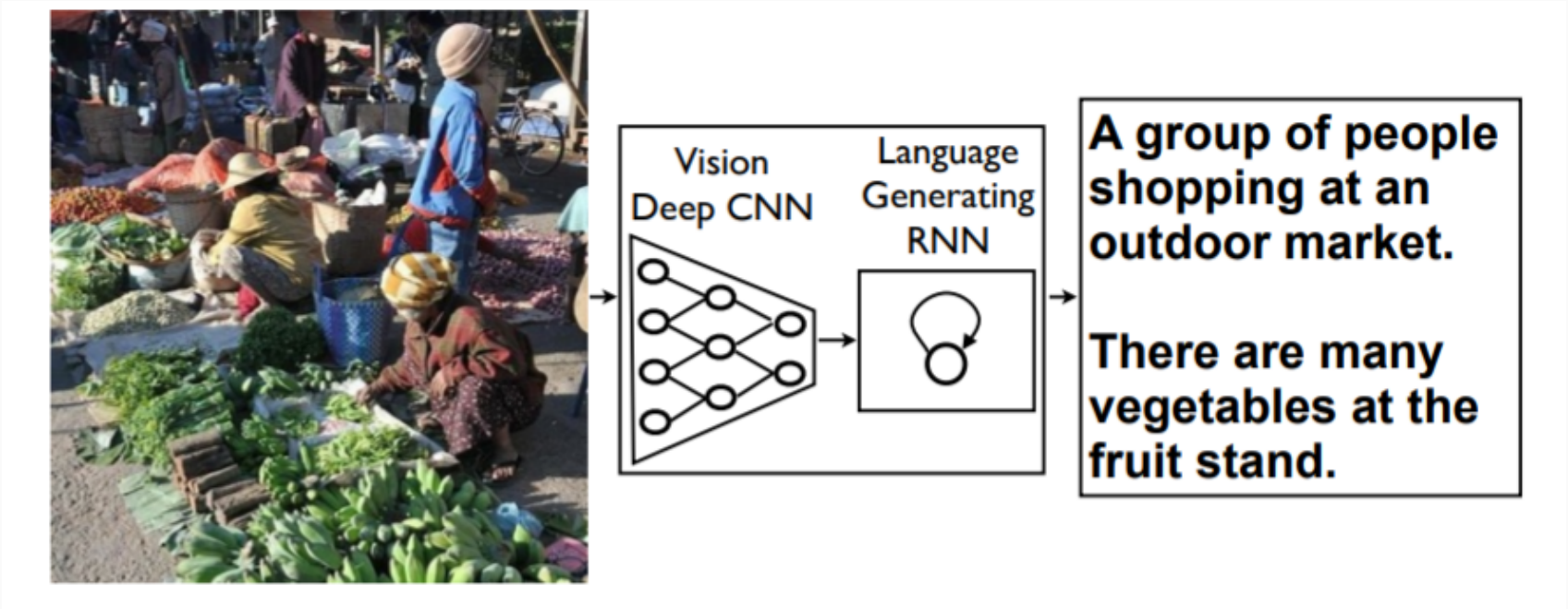
Teaching machines to read

- Получаются разумные карты внимания:



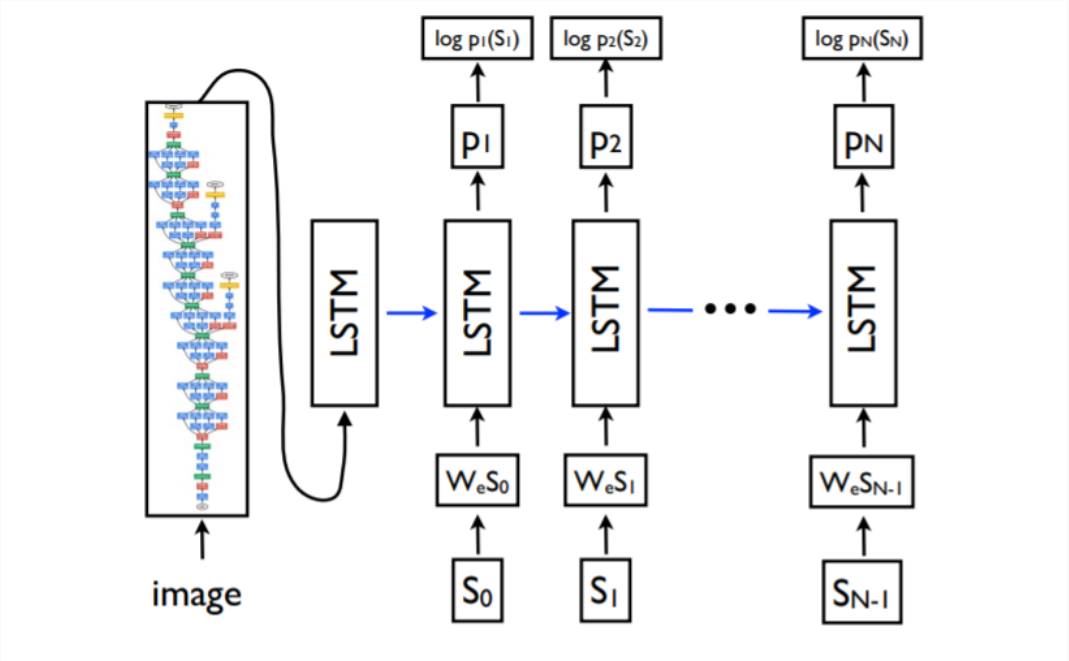
Show, Attend, and Tell

- Теперь давайте про подписи к картинкам.
- Сначала было «Show and Tell» (Vinyals et al., 2015):



Show, Attend, and Tell

- Довольно прямолинейная архитектура:
  - целевая функция – это просто $\sum_{(I,S)} \log p(S \mid I; \theta)$, где I – картинка, S – описание;
  - раскладываем и моделируем $p(S_t \mid I, S_0, \ldots, S_{t-1})$ рекуррентной сетью с LSTM;
  - а CNN используем, чтобы извлечь признаки.



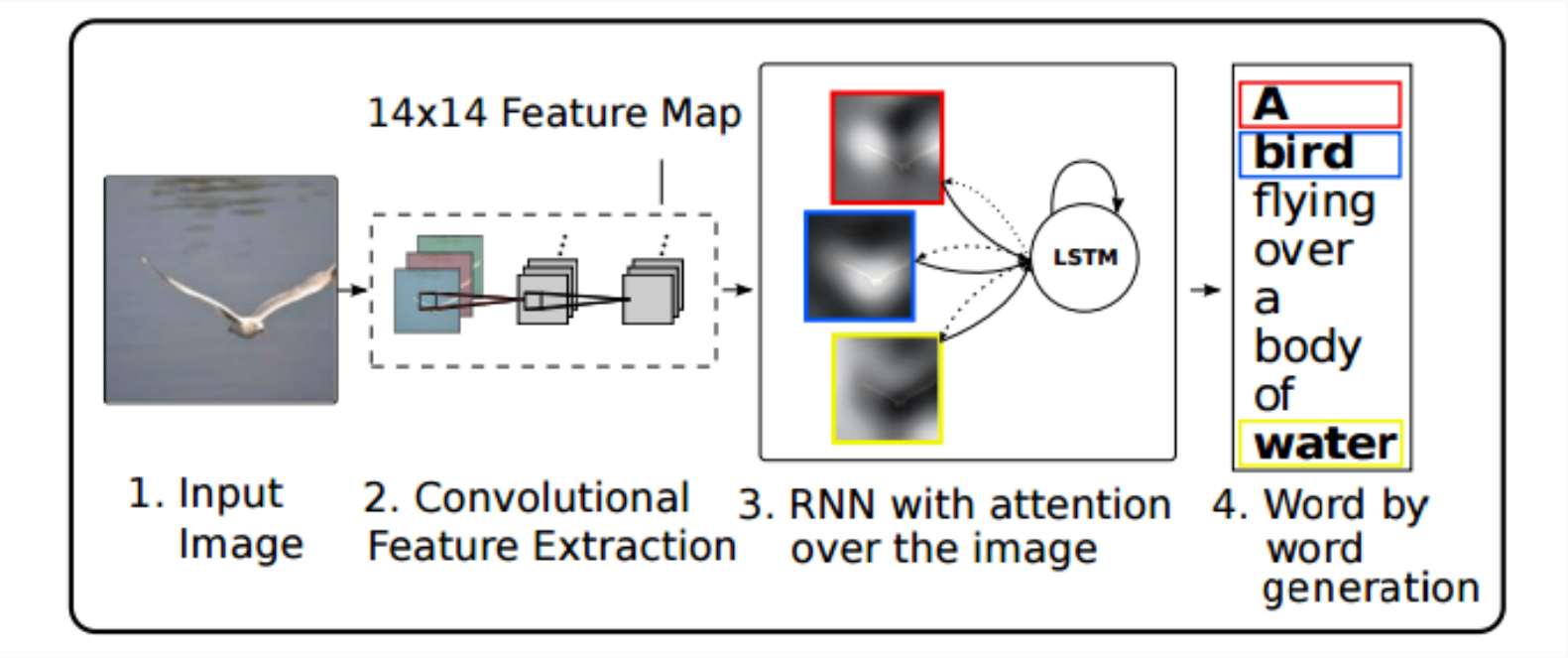
Show, Attend, and Tell

- Получалось хорошо, но можно лучше:



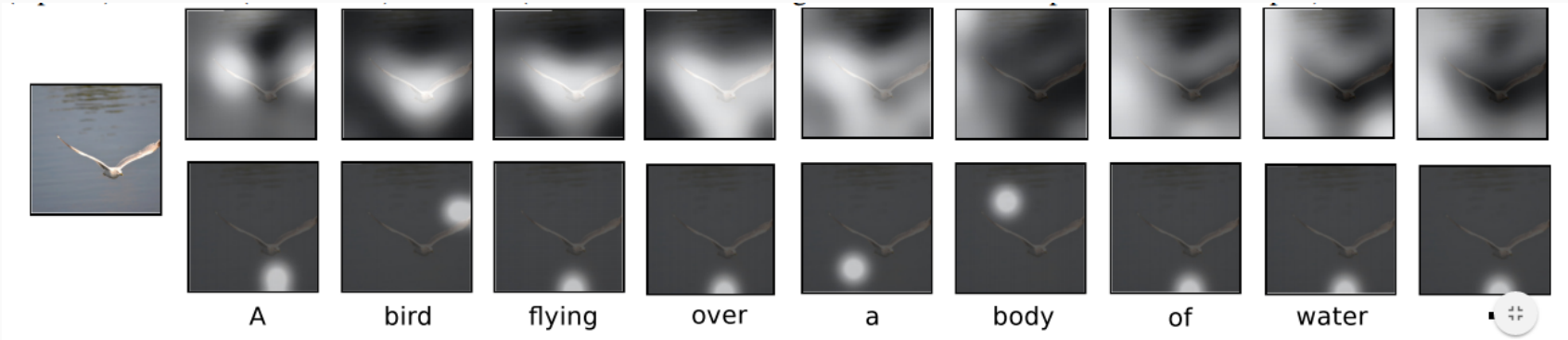
Show, Attend, and Tell

- Из этого появилось «Show, Attend, and Tell» (Xu et al., 2015)



Show, Attend, and Tell

- Soft attention vs. hard attention (стохастически выбираем однозначный кусок картинки).



- Soft attention – строим аннотацию с весами

$$\phi(\{\mathbf{a}\}_i, \{\alpha_i\}) = \sum_{i=1}^{L} \alpha_{t,i} \mathbf{a}_i.$$

Show, Attend, and Tell

- Hard attention обучается максимизацией вариационной нижней оценки

$$L_s = \sum_s p(s \mid \mathbf{a}) \log p(\mathbf{y} \mid s, \mathbf{a}) \leq \log \sum_s p(s \mid \mathbf{a}) p(\mathbf{y} \mid s, \mathbf{a}) = \log p(\mathbf{y} \mid \mathbf{a}).$$

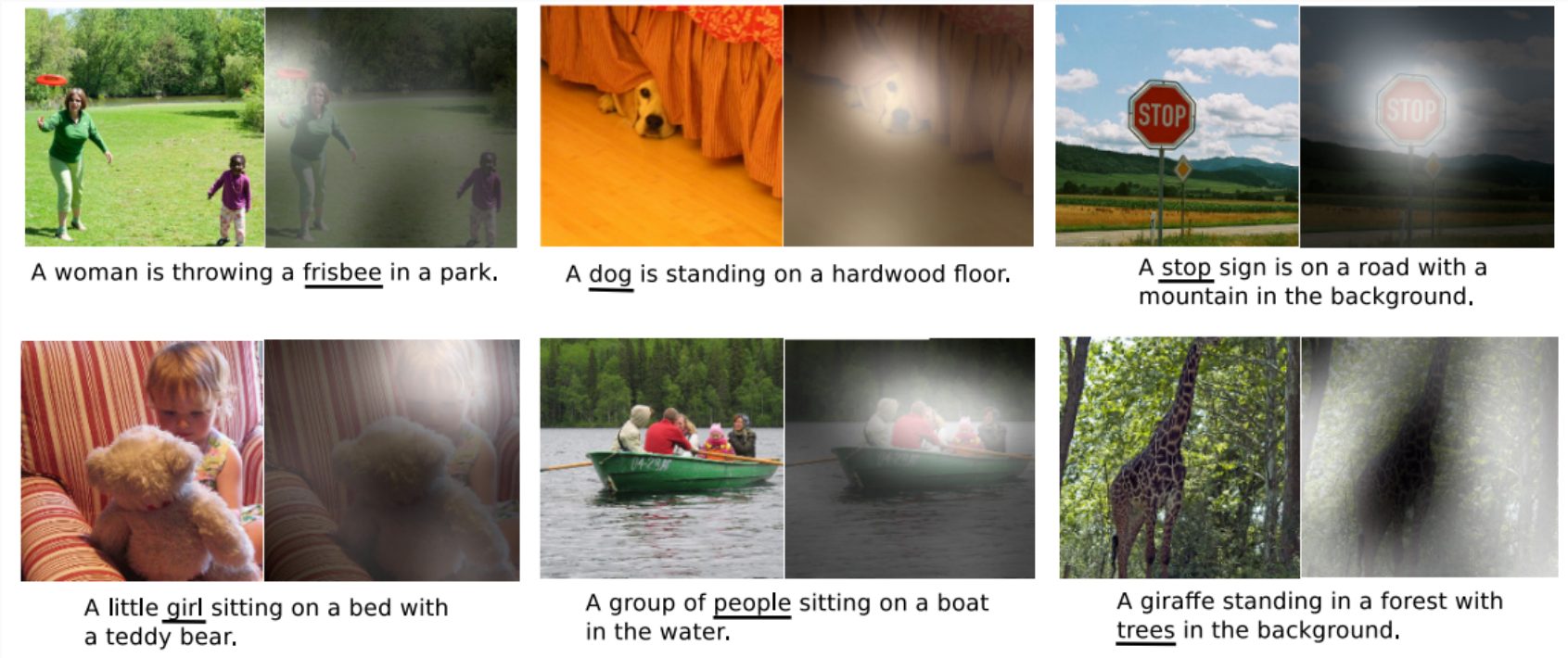
- От L_s можно брать производные:

$$\frac{\partial L_s}{\partial W} = \sum_s p(s \mid \mathbf{a}) \left[\frac{\partial \log p(\mathbf{y} \mid s, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid s, \mathbf{a}) \frac{\partial \log p(s \mid \mathbf{a})}{\partial W} \right].$$

- И дальше сэмплируем s_t с вероятностями α_i и приближаем ожидание выборкой.
- Опять те же трюки, вычитаем baseline, всё такое.

Show, Attend, and Tell

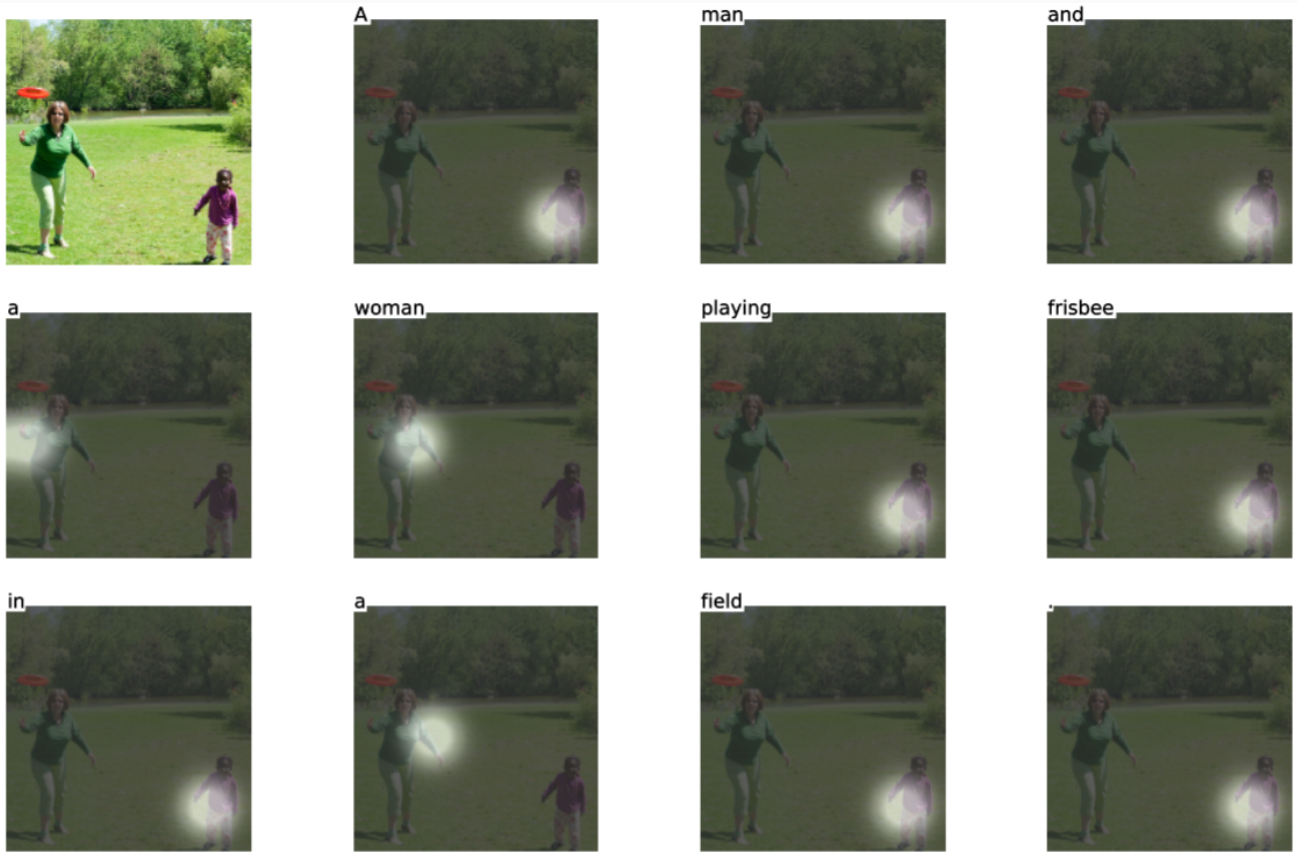
- Часто получаются очень хорошие результаты:



- А когда плохие, можно посмотреть почему.

Show, Attend, and Tell

- Примеры – hard attention:



(a) A man and a woman playing frisbee in a field.

Show, Attend, and Tell

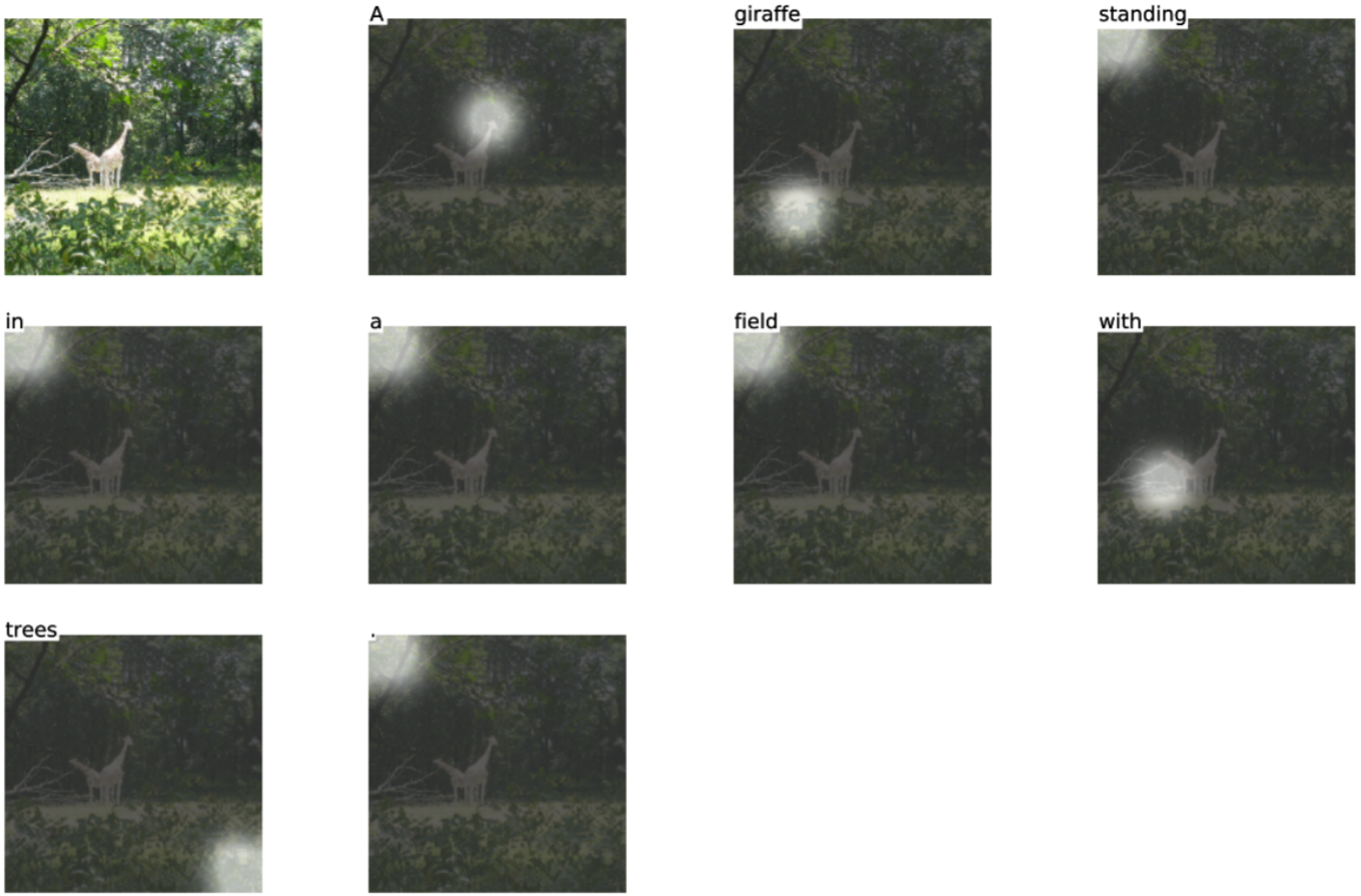
- Примеры – soft attention:



(b) A woman is throwing a frisbee in a park.

Show, Attend, and Tell

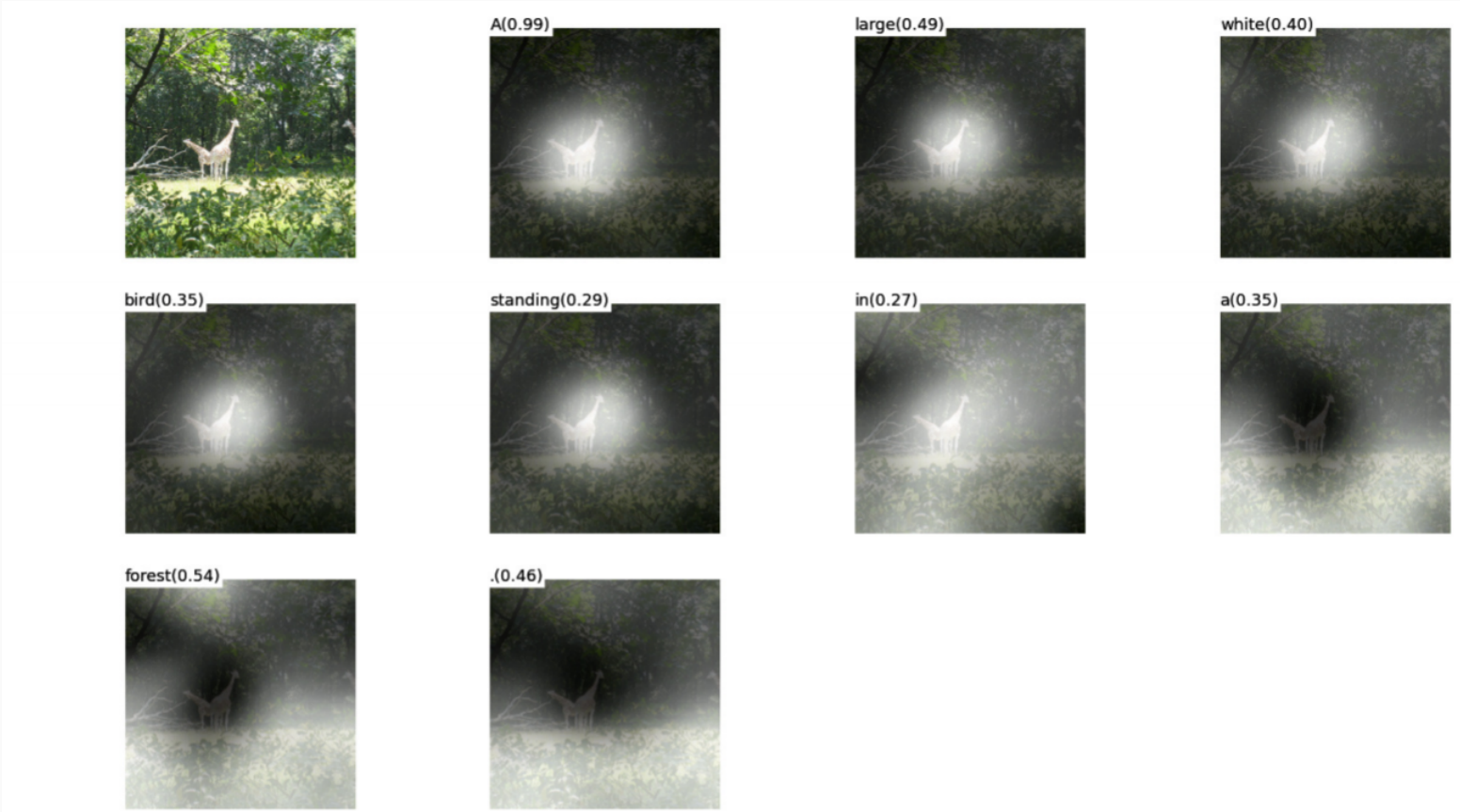
- Примеры – hard attention:



(a) A giraffe standing in the field with trees.

Show, Attend, and Tell

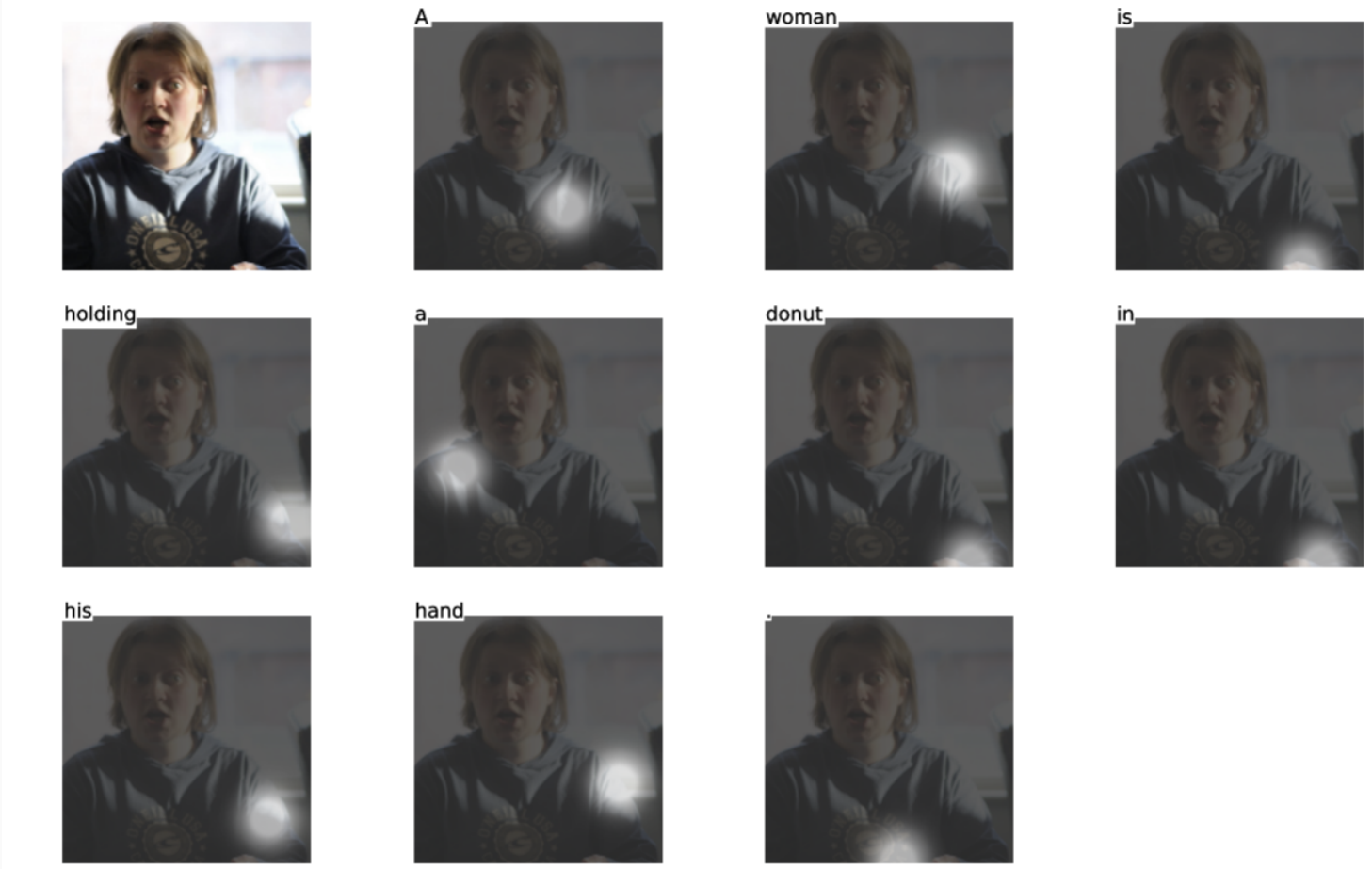
- Примеры – soft attention:



(b) A large white bird standing in a forest.

Show, Attend, and Tell

- Примеры – hard attention:



(a) A woman is holding a donut in his hand.

Show, Attend, and Tell

- Примеры – soft attention:



(b) A woman holding a clock in her hand.

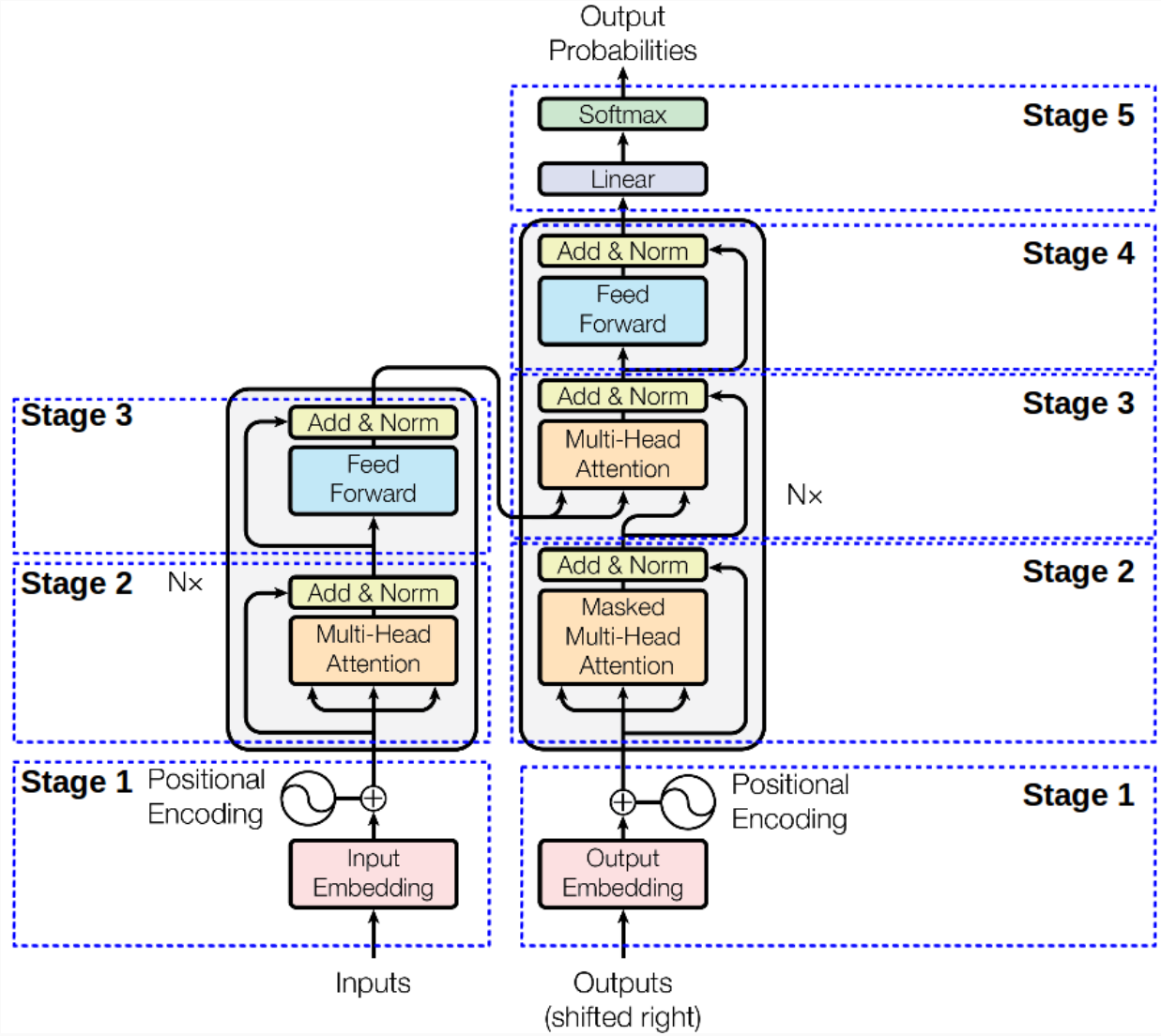
Самовнимание

Transformer

- Мы изучали перевод на рекуррентных сетях и дошли до архитектуры Google NMT
- Но в 2017 году оказалось, что всё может быть ещё проще и интереснее
- Google: «Attention is all you need» (Vaswani et al., 2017)
- Основная идея – self-attention; оказывается очень плодотворной для всевозможных seq2seq задач
- Главная мотивация – попробовать всё-таки уйти от кодирования вектором постоянной длины

Transformer

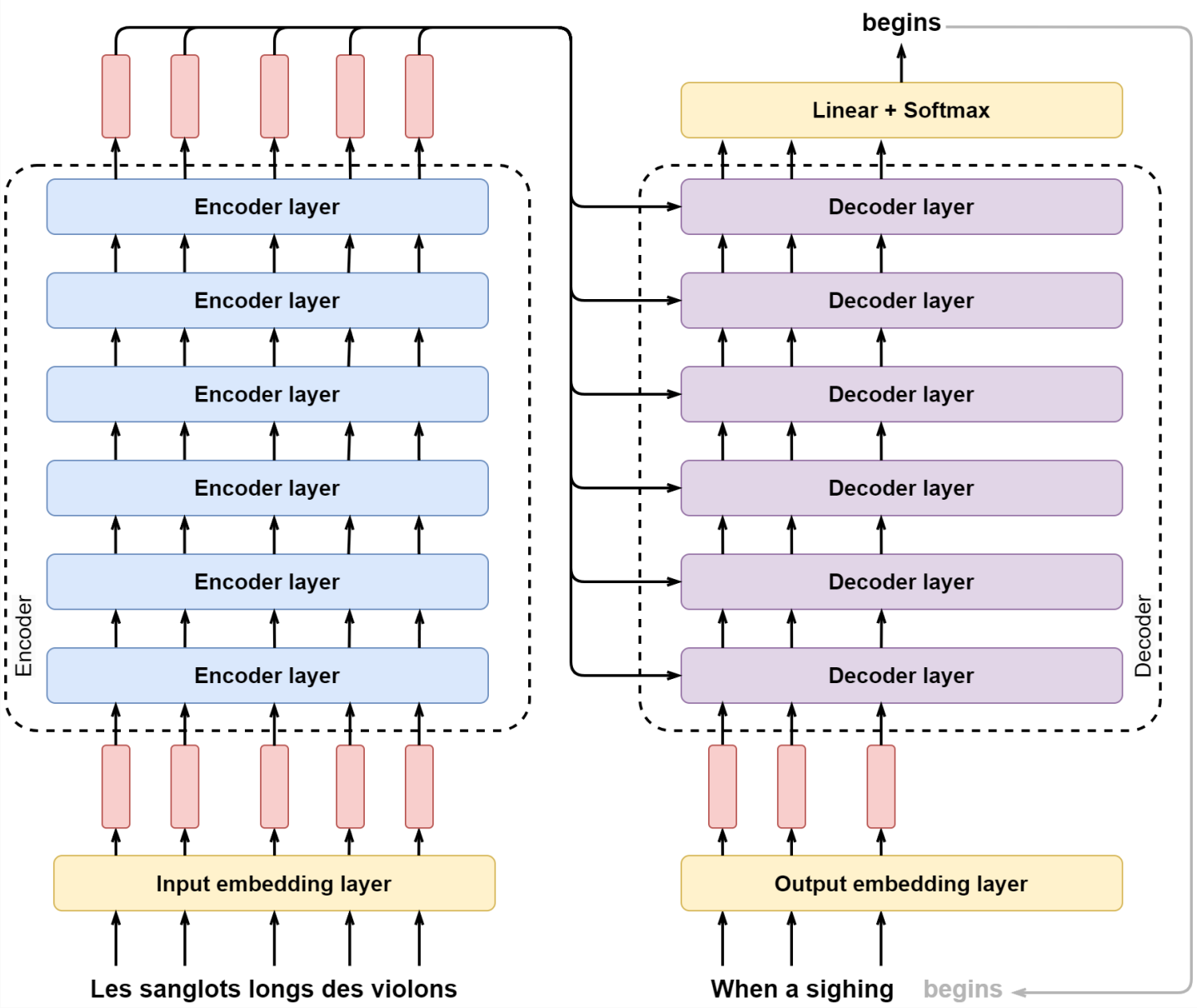
- Общая схема:



- Теперь подробнее...

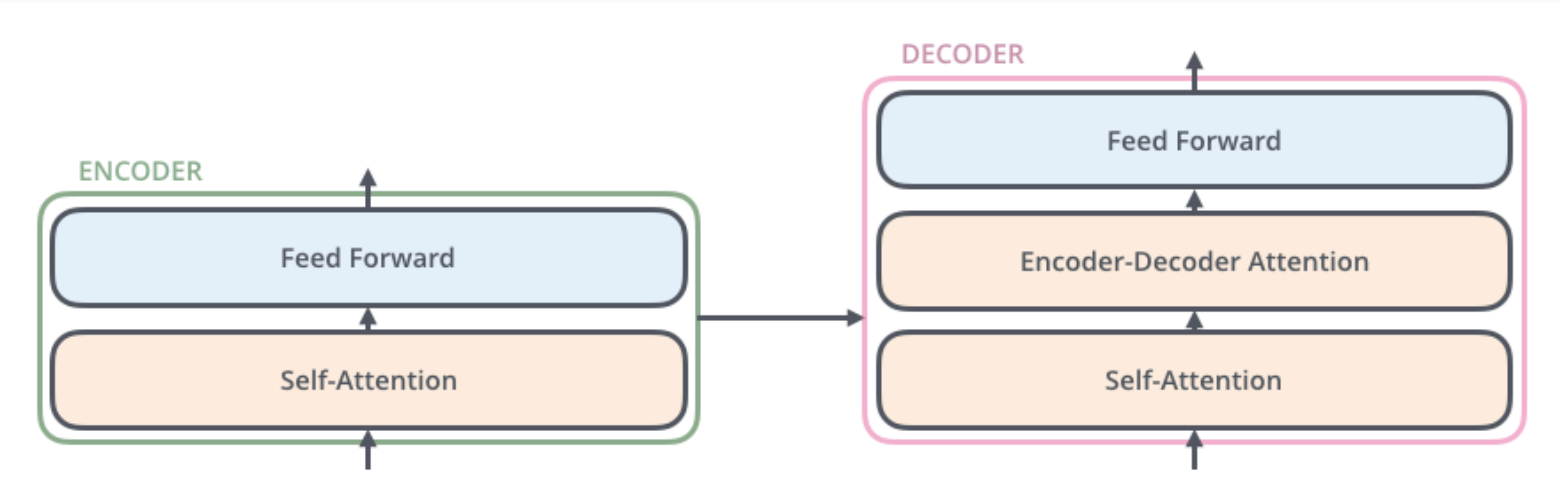
Transformer

- Суть, как и раньше, – encoder-decoder:



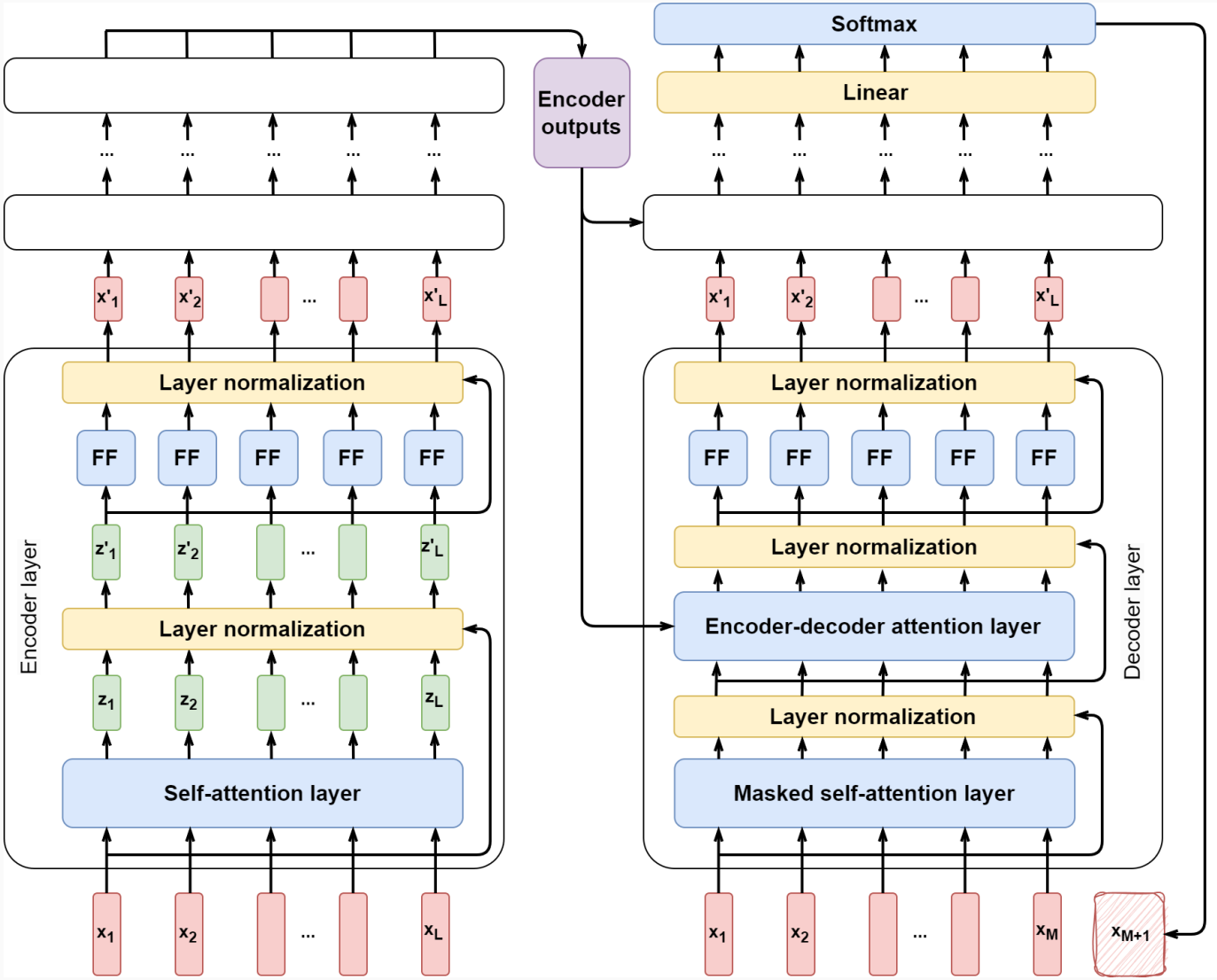
Transformer

- В каждом слое – слой self-attention, а потом feedforward layer, который независимо применяется к каждой позиции входа. У декодера ещё есть attention между ними:



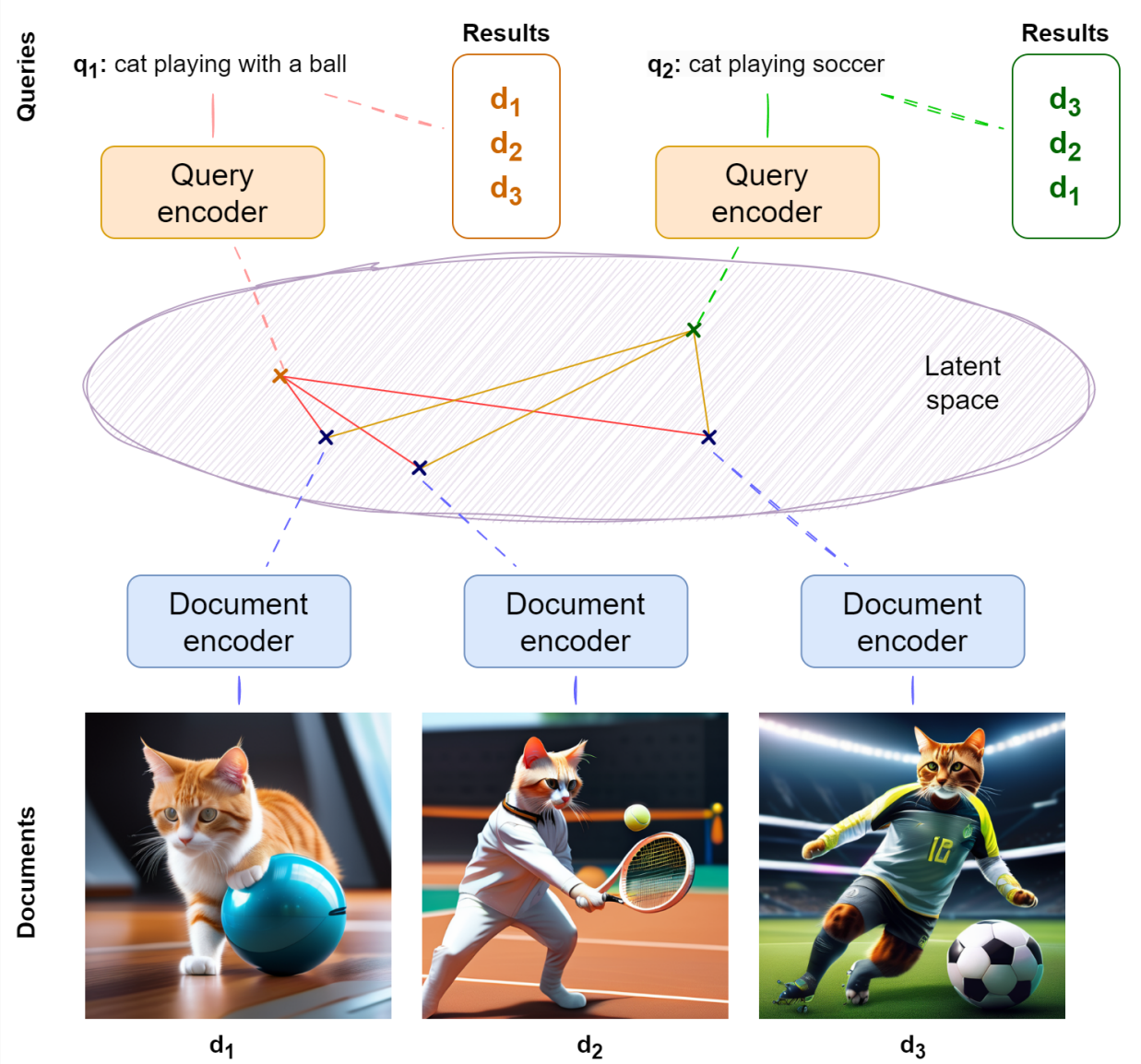
Transformer

- Слова, естественно, представляются векторами, в feedforward слое всё параллельно:



Self-attention

- Чтобы понять самовнимание (self-attention), начнём с информационного поиска (information retrieval)



Self-attention

- Запросы и документы отображаются в одно и то же латентное пространство (но разными кодировщиками — это могут быть вообще объекты разной природы)
- Запрос идёт через query encoder
- Документы (на иллюстрации это картинки) — через другой encoder, но в то же пространство
- Чтобы найти самые релевантные документы, мы ищем ближайших соседей в латентном пространстве среди документов; часто предполагают, что латентное пространство линейно, и расстояние там — это просто скалярное произведение $\mathrm{dist}(q, d) = \mathrm{Enc}_q(q)^\top \mathrm{Enc}_d(d)$.

Self-attention

- В самовнимании эта интуиция используется очень абстрактно. Слой получает на вход последовательность векторов $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L$, т.е. матрицу $X \in \mathbb{R}^{d \times L}$.
- Запросы, ключи и документы берутся из самих $\mathbf{x}_i$:
  - умножая $\mathbf{x}_1$ на матрицу весов W^Q, мы получаем вектор *запроса* $\mathbf{q}_1 = W^Q \mathbf{x}_1$; обратите внимание, что размерность q векторов запросов, $\mathbf{q}_i \in \mathbb{R}^q$, может отличаться (обычно меньше) от входной размерности d, $\mathbf{x}_i \in \mathbb{R}^d$;
  - умножая $\mathbf{x}_i$ на матрицу весов W^K, мы получаем векторы *ключей* $\mathbf{k}_i = W^K \mathbf{x}_i$ для $i = 1, \dots, L$; поскольку мы хотим, чтобы запросы и ключи находились в одном латентном пространстве, ключи имеют ту же размерность, что и запросы, $\mathbf{k}_i \in \mathbb{R}^q$, таким образом $W^V \in \mathbb{R}^{q \times d}$;
  - третья матрица весов W^V даёт векторы *значений* $\mathbf{v}_i = W^V \mathbf{x}_i$ для $i = 1, \dots, L$; это те *документы*, которые мы будем «извлекать» с помощью ключей $\mathbf{k}_i$; формально мы имеем другую размерность v для значений, $\mathbf{v}_i \in \mathbb{R}^v$ и $W^V \in \mathbb{R}^{v \times d}$; на практике обычно $v = q$.

Self-attention

- Затем мы выполняем поиск, вычисляя оценки внимания как скалярные произведения между запросами и документами
- Нормализуем $\mathbf{q}_i^\top \mathbf{v}_j$, деля на $\sqrt{q}$, и получить оценки через softmax, чтобы преобразовать их в вероятности:

$$\alpha_{ij} = \mathrm{softmax}\left(\frac{1}{\sqrt{q}}\mathbf{q}_i K^\top\right)_j,$$

где $K \in \mathbb{R}^{q \times L}$ — это все ключи, объединенные в матрицу, $K = W^K X$.

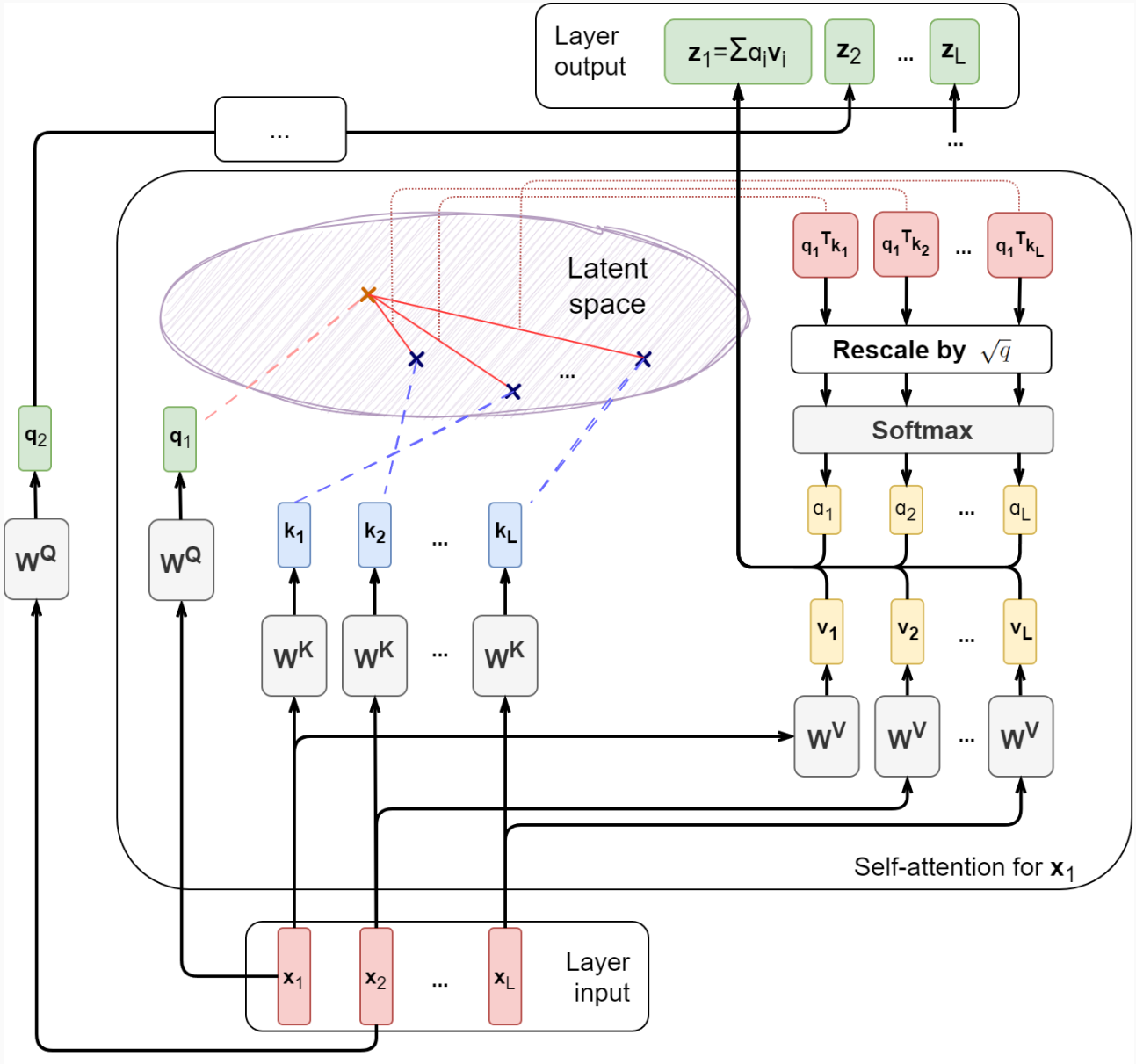
- Затем используем результат в качестве коэффициентов для выпуклой комбинации *значений* $\mathbf{v}_j$:

$$\mathbf{z}_i = \mathrm{softmax}\left(\frac{1}{\sqrt{q}}\mathbf{q}_i K^\top\right) V,$$

где $V \in \mathbb{R}^{q \times L}$ — значения (values), $V = W^V X$.

Self-attention

- Общая структура слоя самовнимания (self-attention):



Self-attention

- И это вся интуиция! Мы можем объединить вычисления каждого $\mathbf{z}_i$ в одну формулу в матричном виде:

$$Z = \mathrm{softmax}\left(\frac{1}{\sqrt{q}} QK^\top\right) V.$$

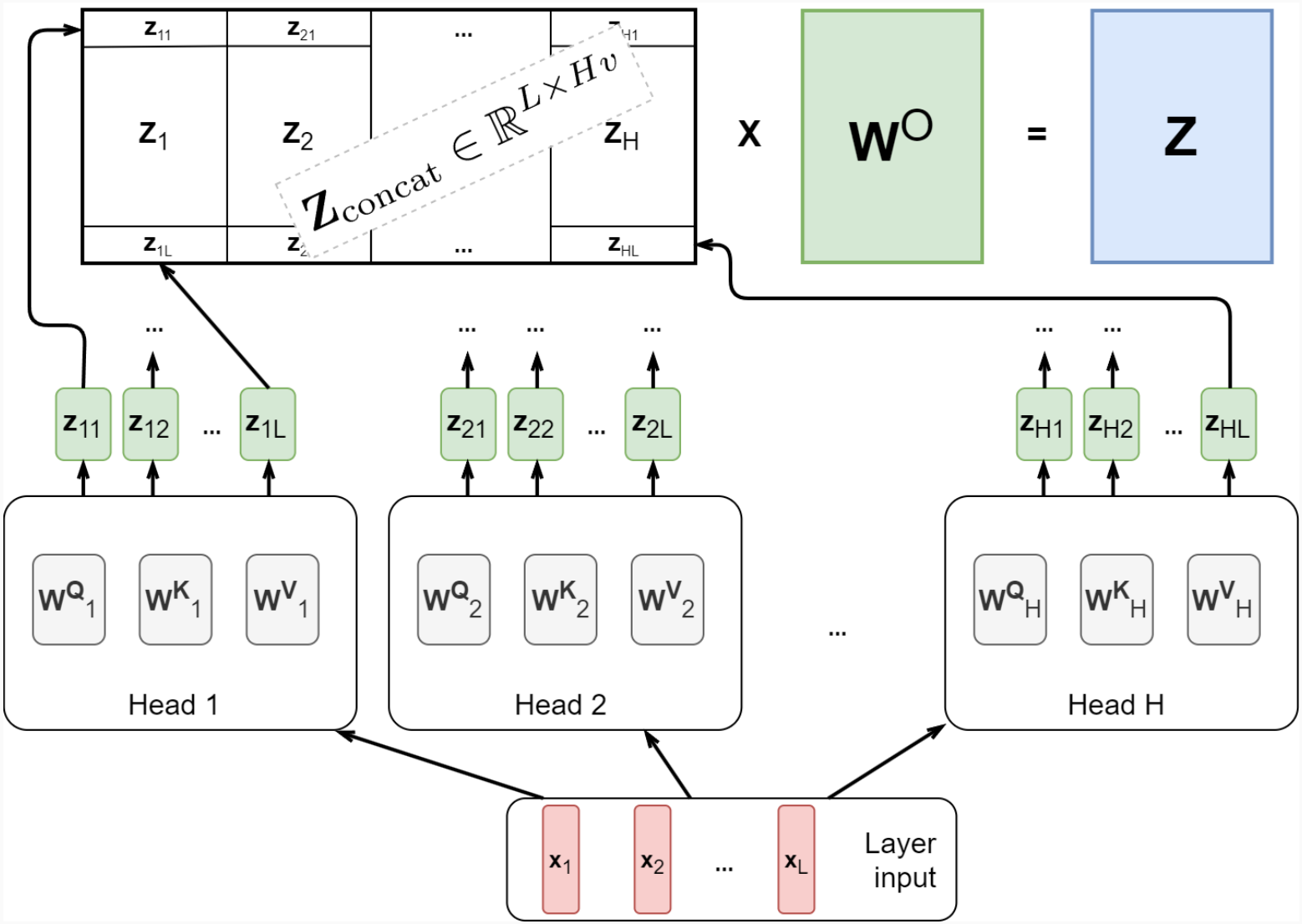
- Но это только один способ «смотреть» на входные векторы; то, что мы сейчас определили, — это не полный слой самовнимания, а только одна *голова* (self-attention head)
- Можно распараллелить эти вычисления вдоль H различных голов, используя разные матрицы весов $W_1^Q, \dots W_H^Q$, $W_1^K, \dots W_H^K$ и $W_1^V, \dots W_H^V$. Это *многоголовое внимание* (multi-head attention)

Self-attention

- После H параллельных голов мы получаем матрицы выходов $Z_1, Z_2, \ldots, Z_H$ размерности $v \times L$, объединяем их все в $Z_{\text{concat}} \in \mathbb{R}^{L \times Hv}$ и добавляем ещё одну матрицу весов $W^O \in \mathbb{R}^{Hv \times d}$, чтобы сжать результат обратно до необходимой размерности:
$$Z = \left(Z_{\text{concat}} W^O\right)^\top.$$
- С помощью W^O слой самовнимания может комбинировать представления, полученные из различных голов внимания; это также важный способ добавить гибкость и выразительность в архитектуру.

Self-attention

- Многоголовое внимание:



- Это самая важная часть, но это ещё не всё.

Спасибо за внимание!



