Фаза 2 • Неделя 3 • Вторник

ELBRUS / BOOTCAMP

Machine Translation • Attention

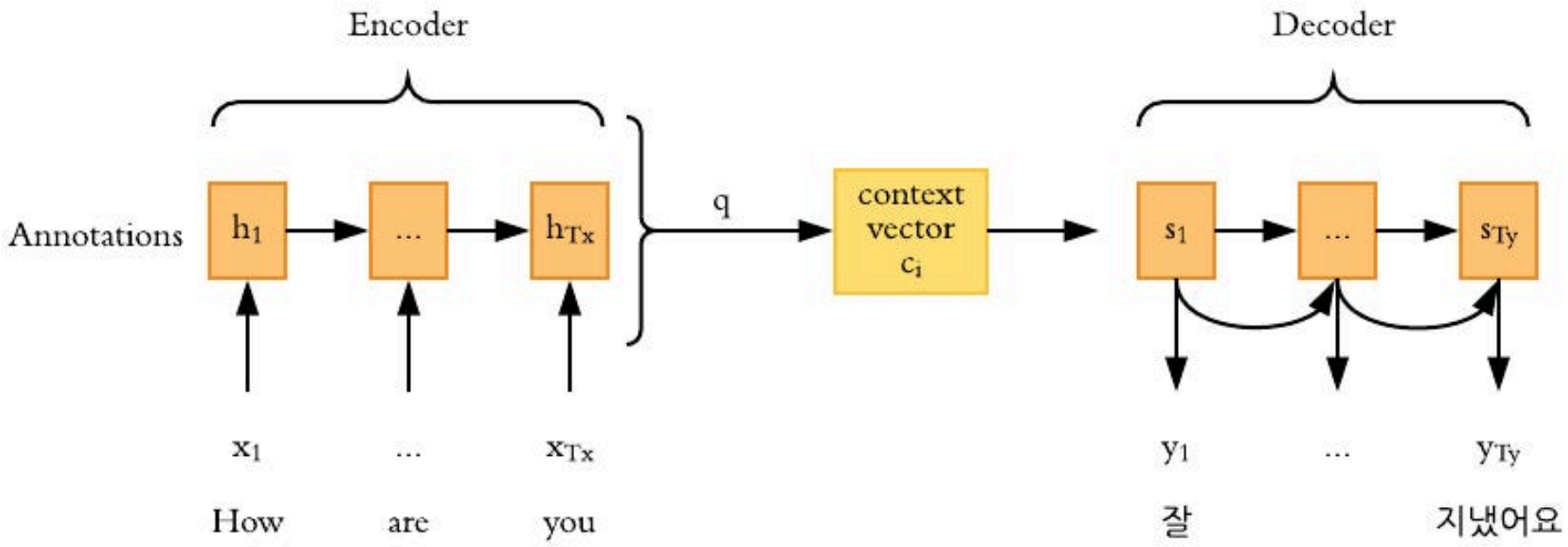
Сегодня

Эльбрус буткемп

- коснемся задачи машинного перевода
- разберем механизм внимания (attention)
- реализуем один из видов attention в задаче классификации

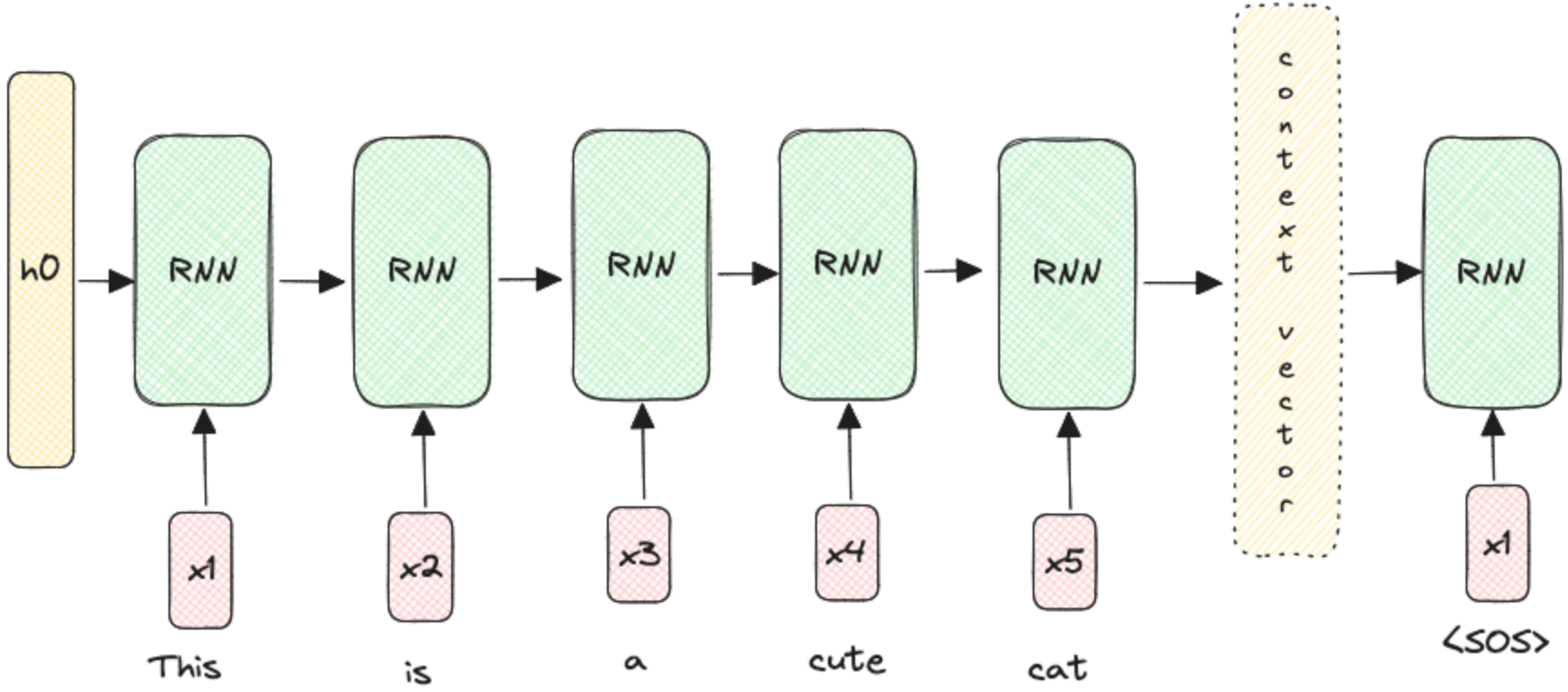
Общая архитектура рекуррентной модели перевода

Эльбрус буткемп



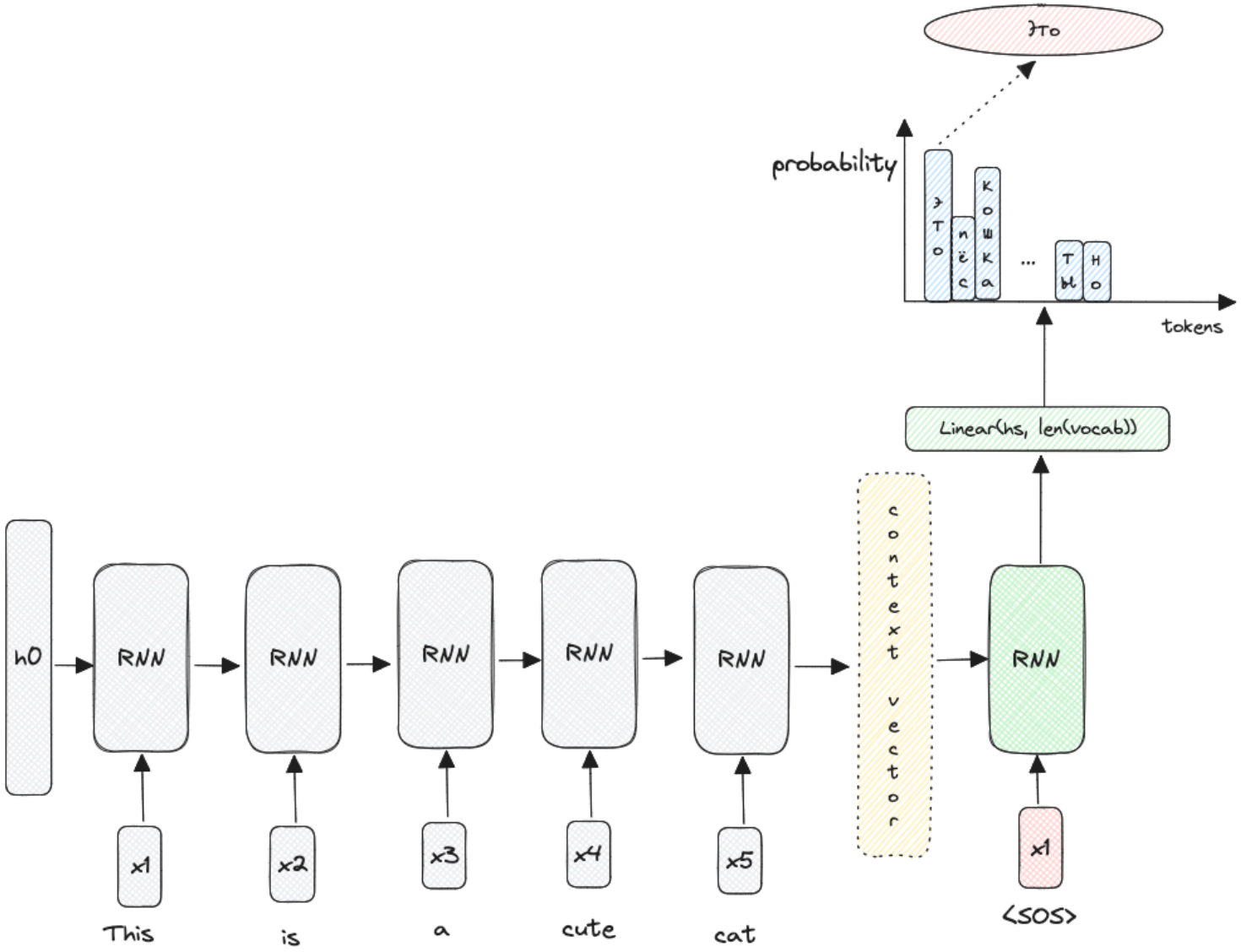
Процесс перевода

Эльбрус буткемп

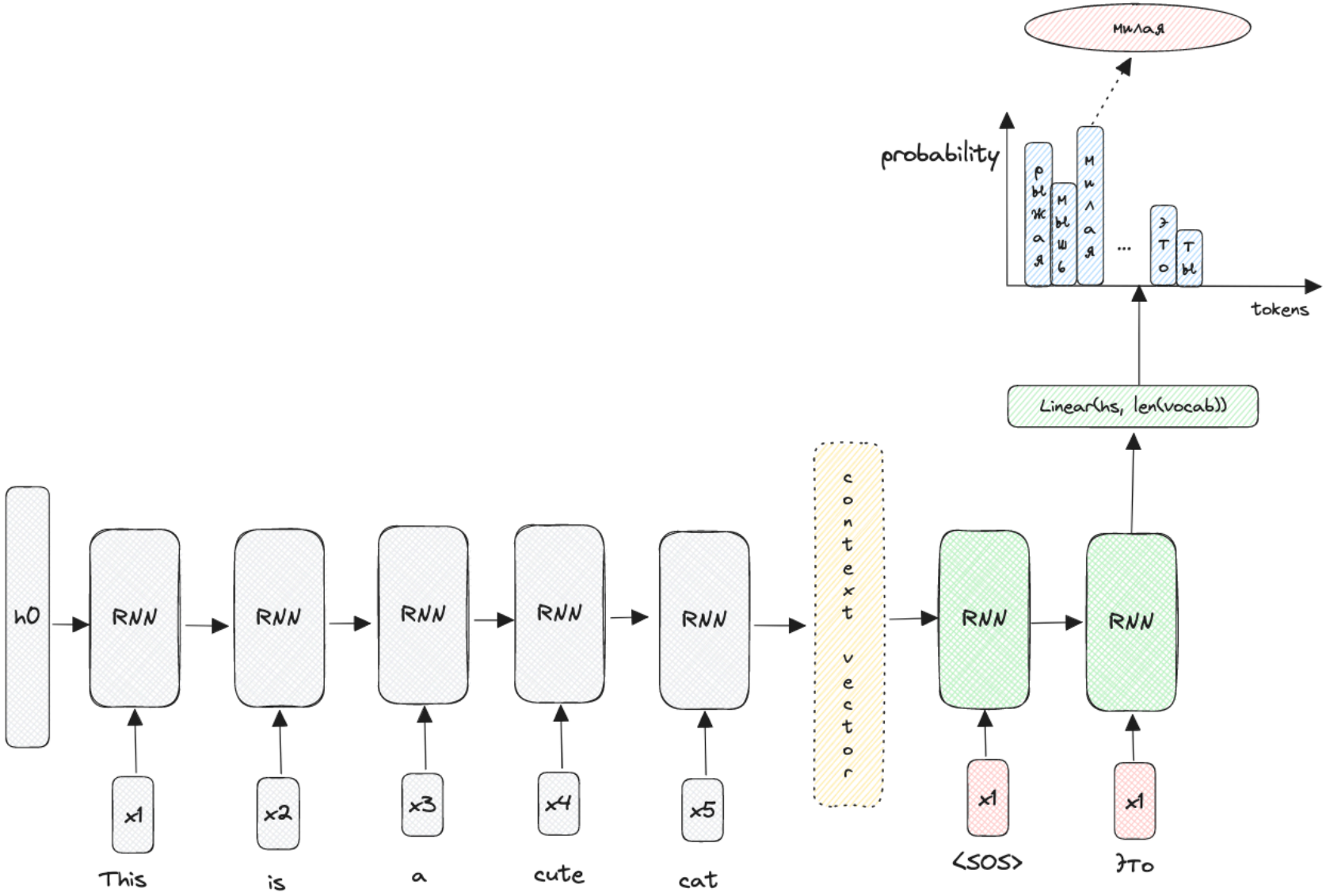


! `<SOS>` – start of sequence – служебный токен старта процесса

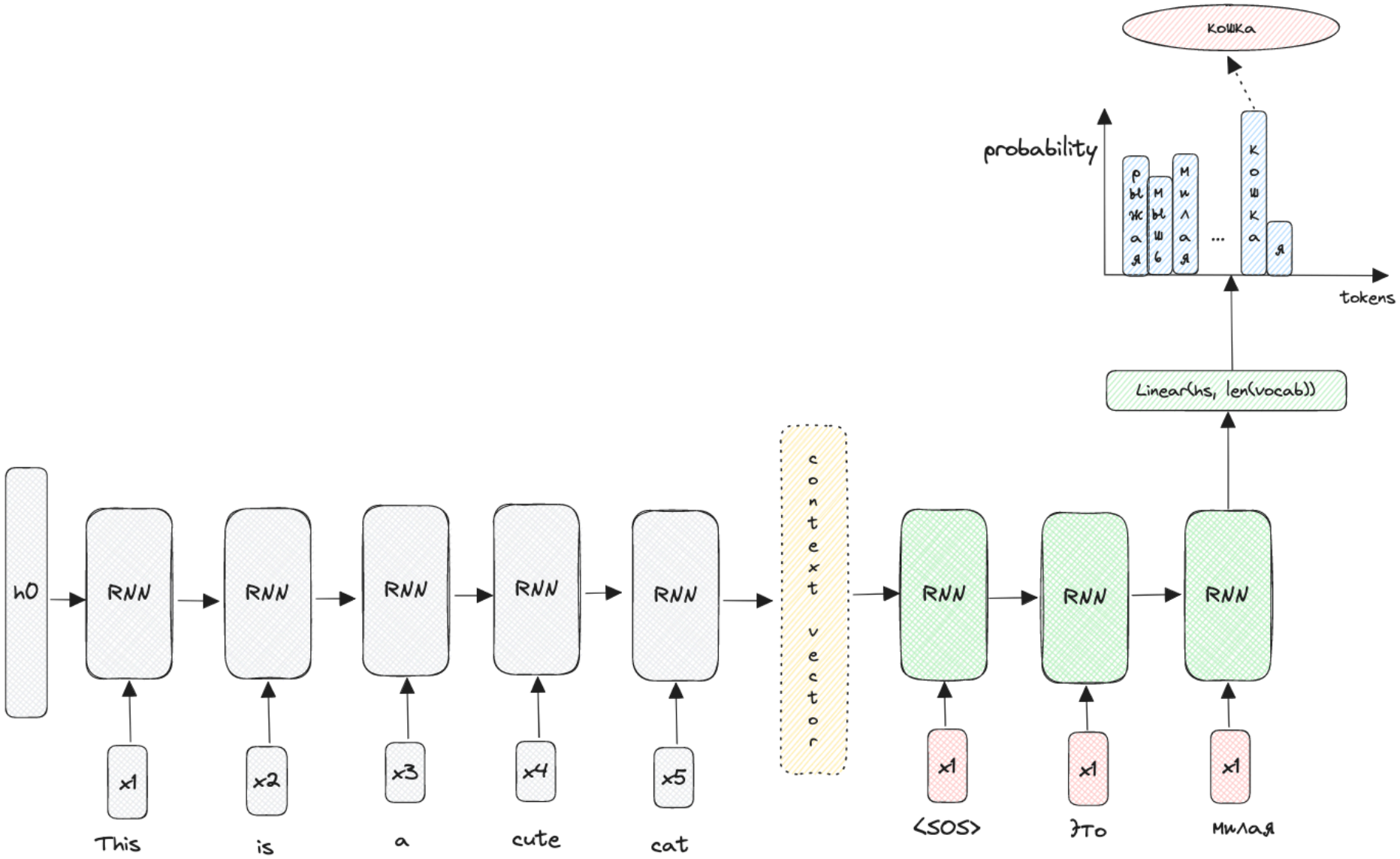
Процесс перевода



Процесс перевода

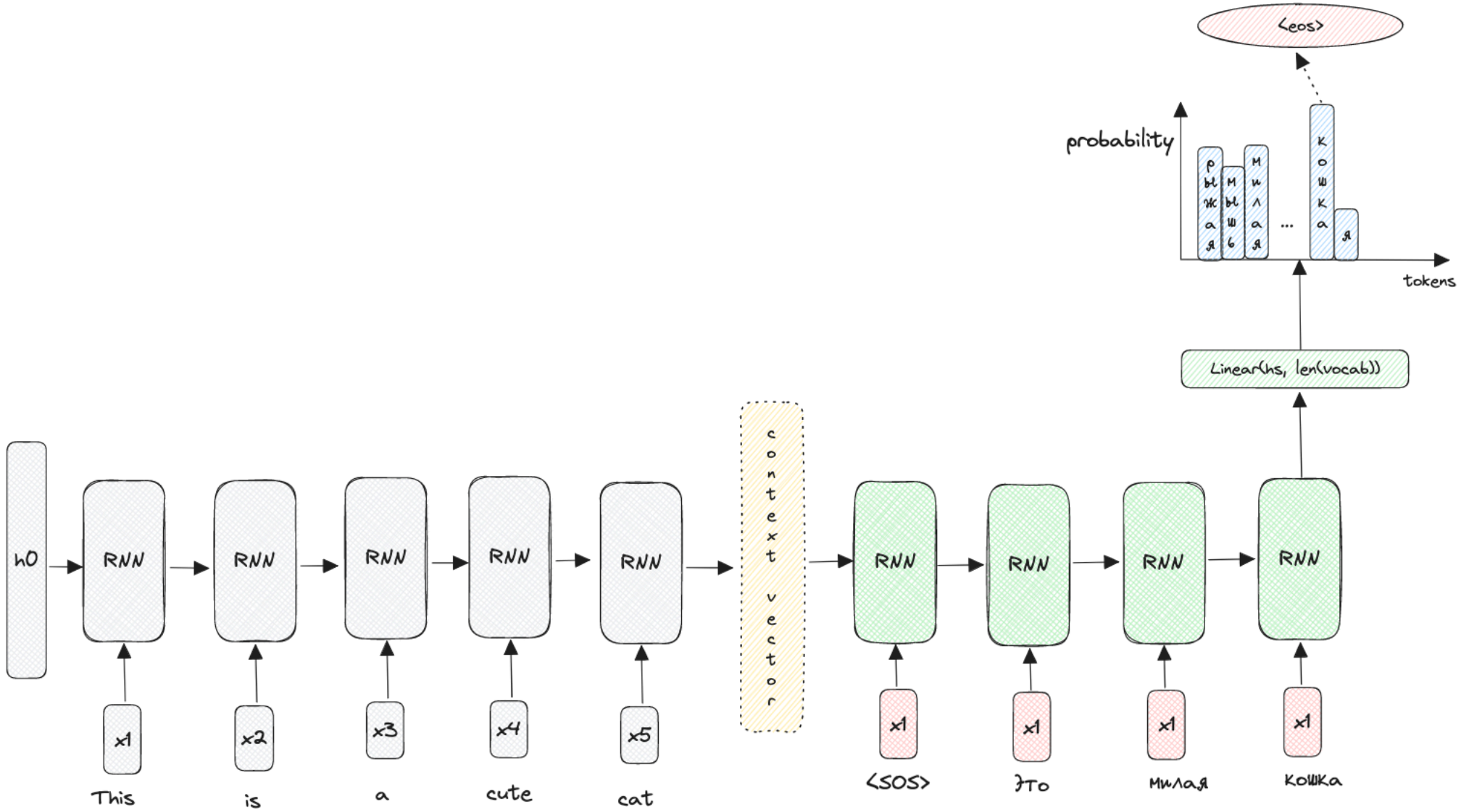


Процесс перевода

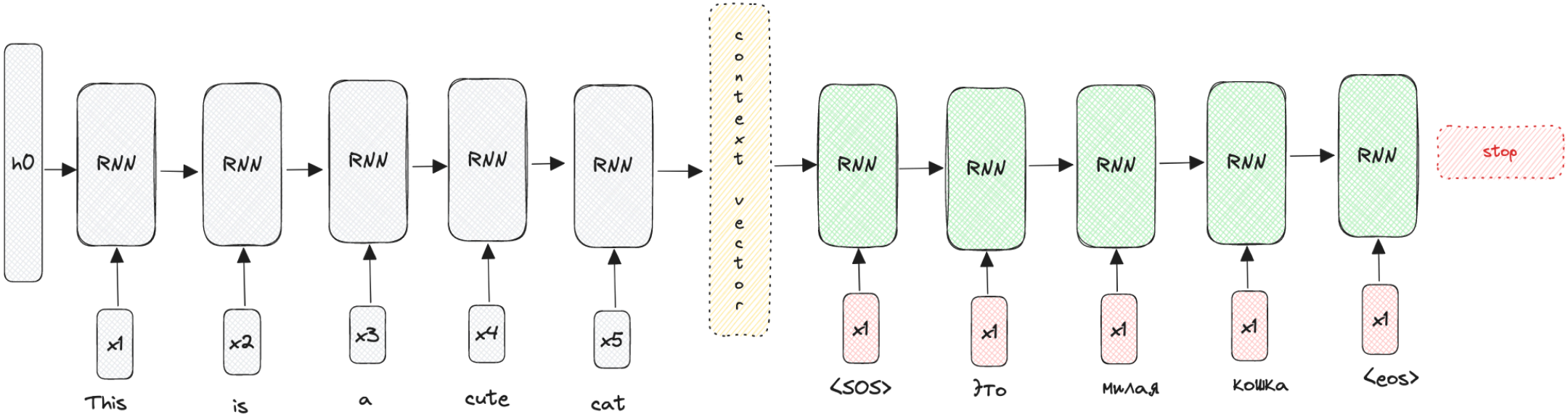


Процесс перевода

Эльбрус буткемп



Процесс перевода



! `<EOS>` – end of sequence – служебный токен окончания процесса

Проблемы

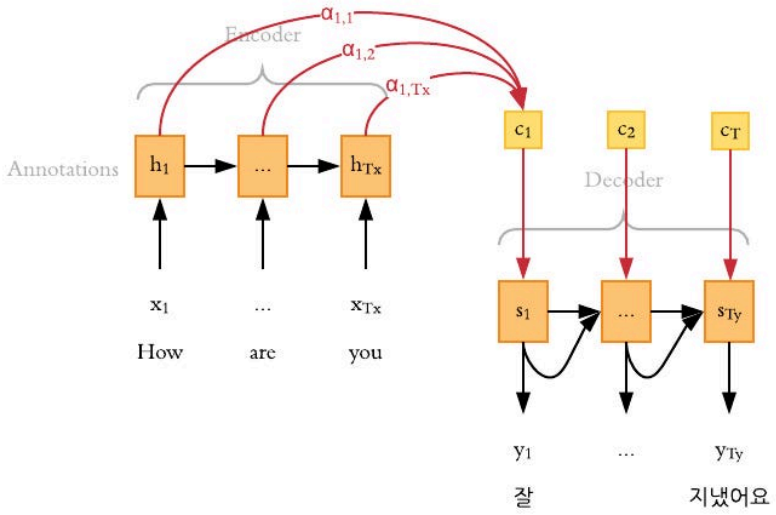
Эльбрус буткемп

- context vector слишком маленький для длинной последовательности, он не может уместить в себе всю информацию
- сеть теряет контекст (следствие компактности вектора)

Attention

Основная идея: каждый раз, когда модель предсказывает слово, она использует лишь часть входных релевантных данных, а не всю последовательность.

Проще: модель "обращает *внимание*" только на некоторые слова, которые важны для перевода текущего слова.



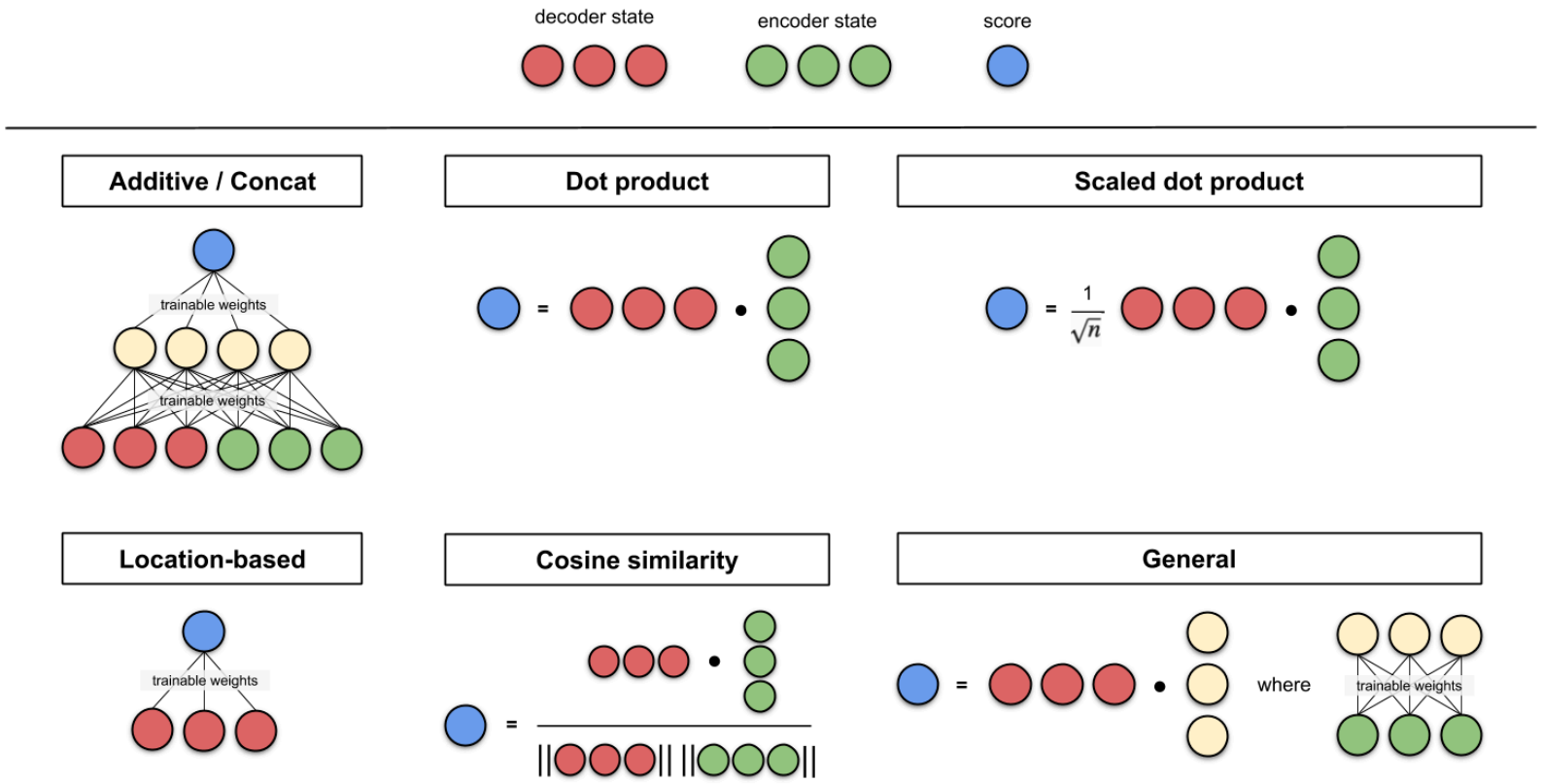
Типы механизмов внимания

Эльбрус буткемп

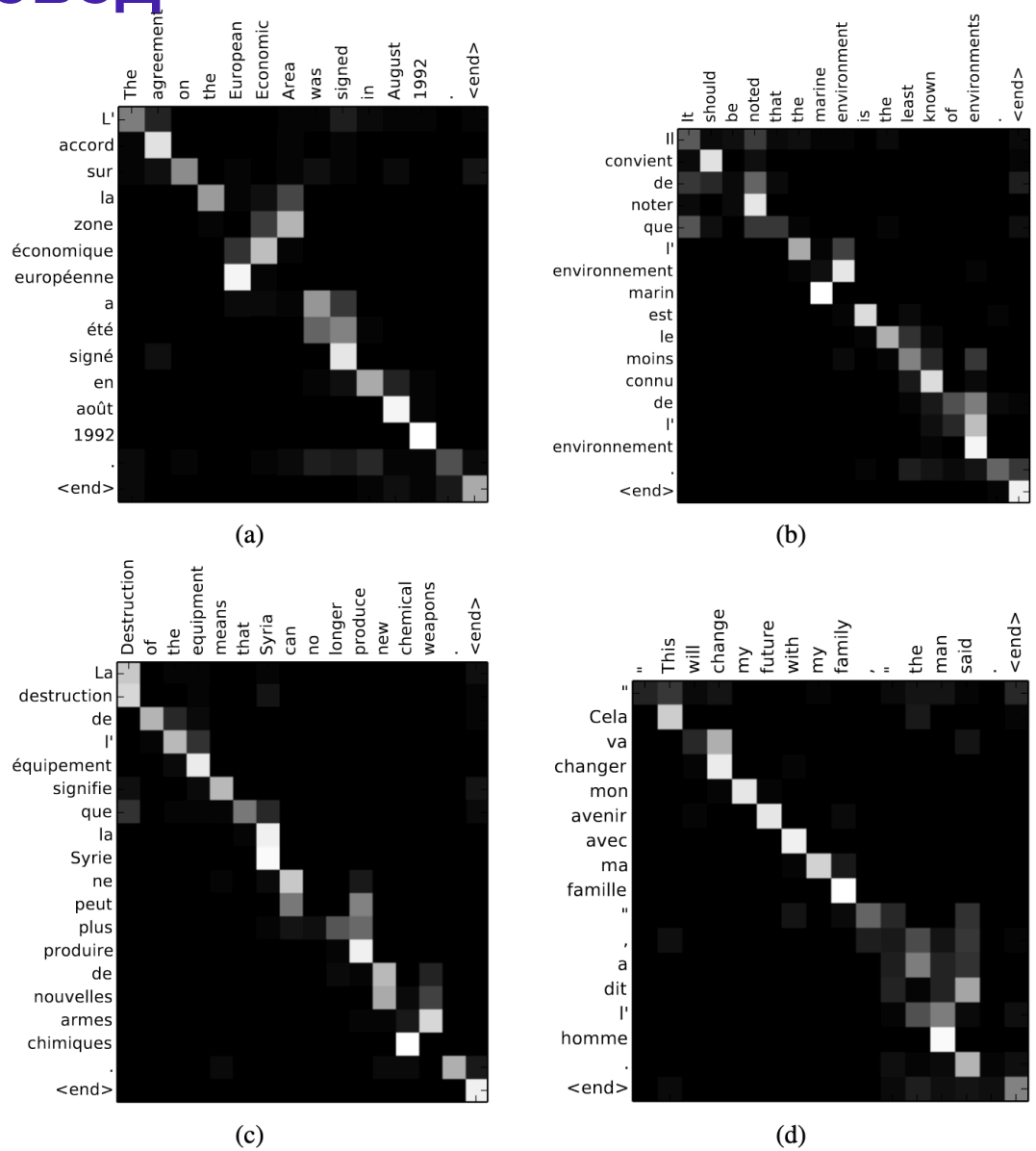
Их много

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Типы механизмов внимания



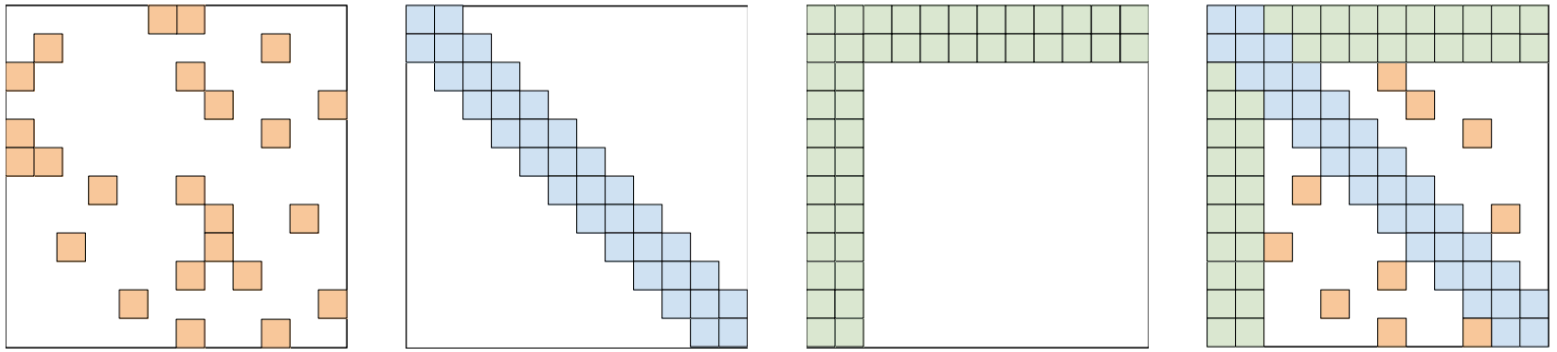
Машинный перевод



Global vs. local attention

Эльбрус буткемп

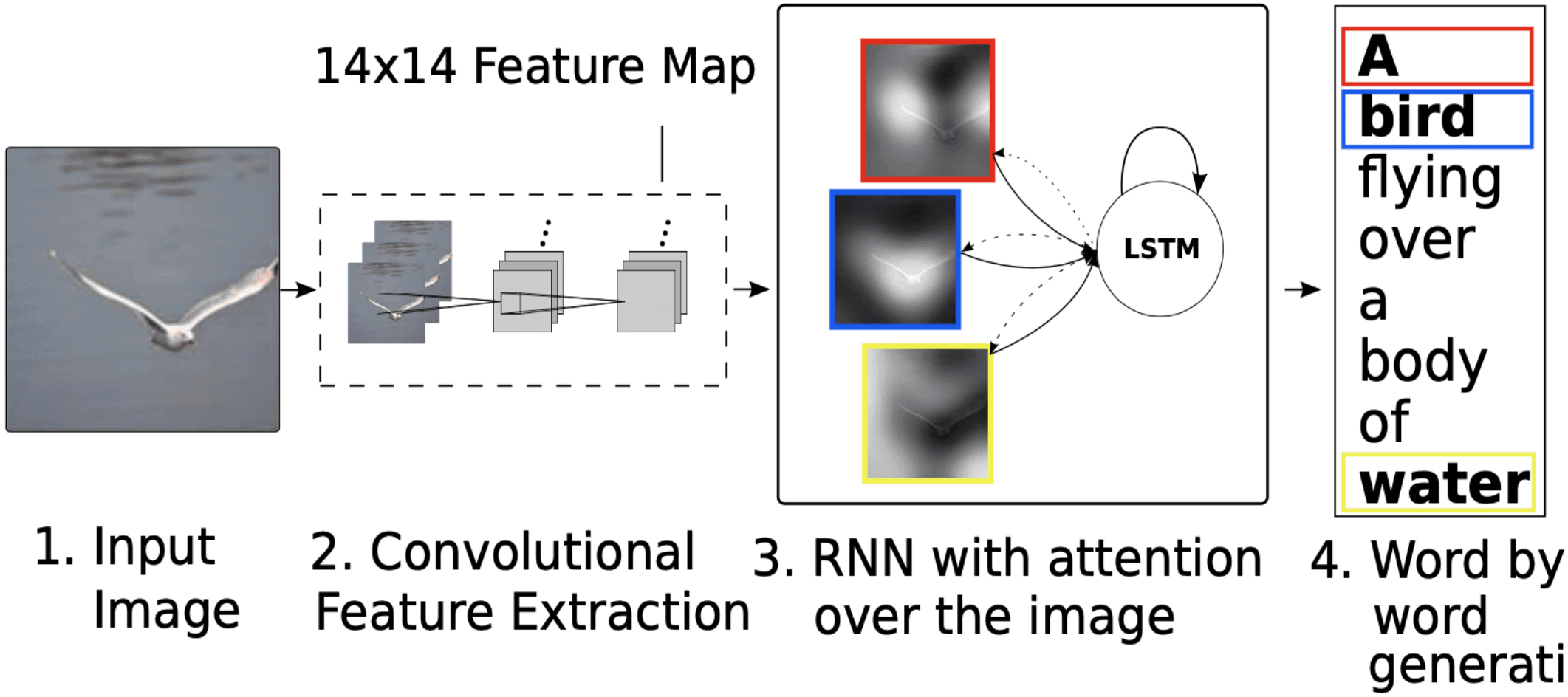
Идея локального механизма внимания: зачем смотреть на всю последовательность, если можно на наиболее вероятное подмножество?



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

Visual Attention

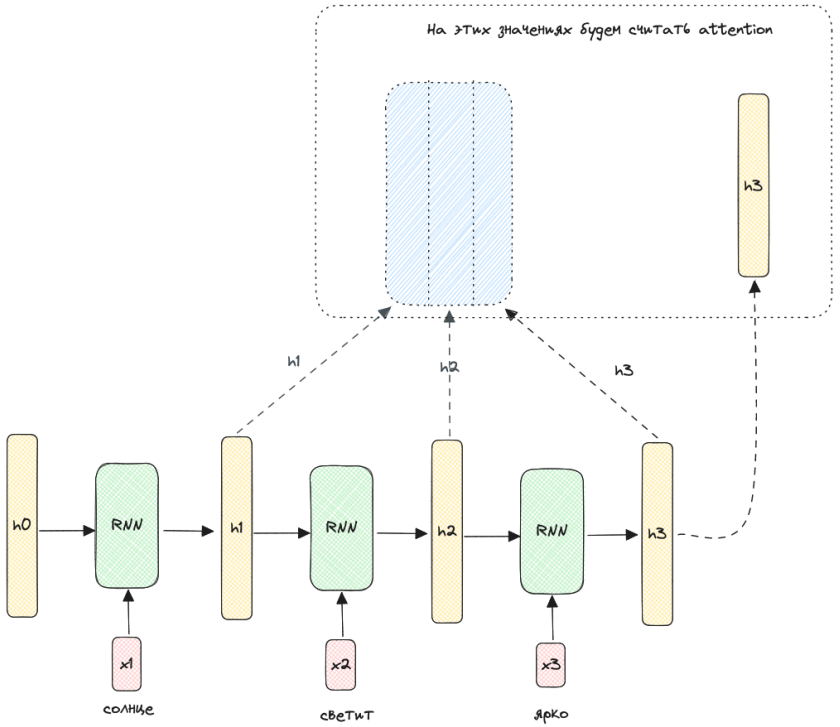


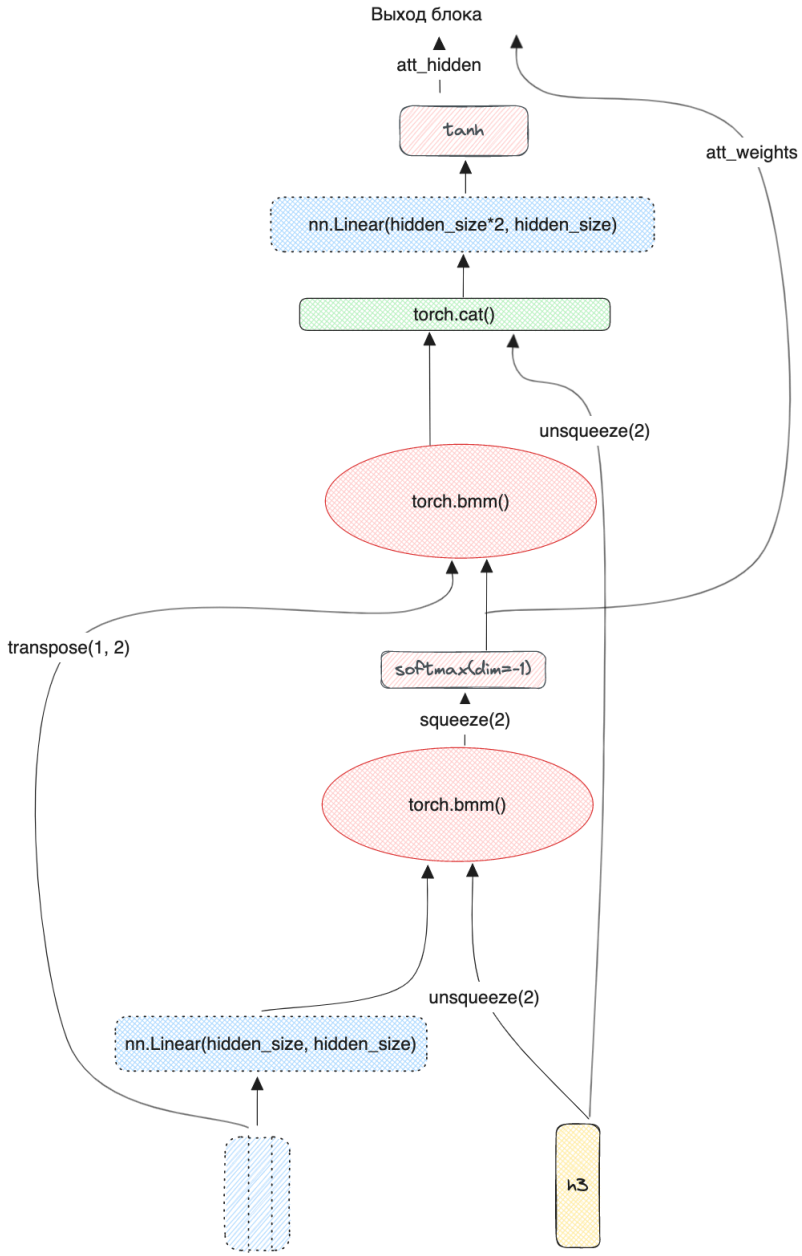
📝 Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

Attention в задаче классификации последовательности

Эльбрус буткемп

В задаче перевода есть скрытые состояния декодера, в задаче классификации такой последовательности состояний нет, но attention все равно можно применить





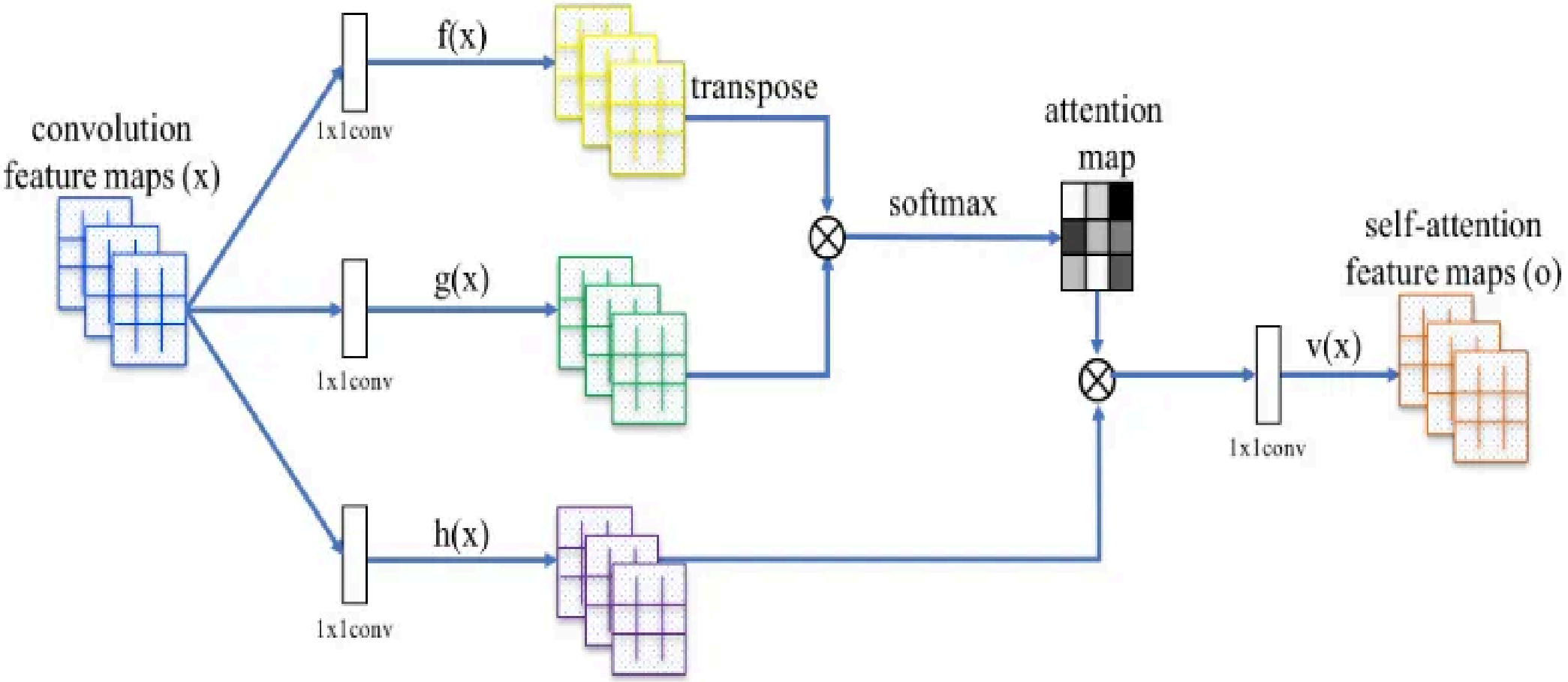

```python
class ConcatAttention(nn.Module):
  def __init__(
        self,
        hidden_size: int = HIDDEN_SIZE
        ) -> None:

    super().__init__()
    self.hidden_size = hidden_size
    self.linear = nn.Linear(hidden_size, hidden_size)
    self.align  = nn.Linear(hidden_size * 2, hidden_size)
    self.tanh   = nn.Tanh()

  def forward(
        self,
        lstm_outputs: torch.Tensor, # BS x SEQ_LEN x HIDDENS
        final_hidden: torch.Tensor  # BS x HS
        ) -> Tuple[torch.Tensor, torch.Tensor]:

    att_weights = self.linear(lstm_outputs)
    att_weights = torch.bmm(
      att_weights,
      final_hidden.unsqueeze(2)
      )
    att_weights = F.softmax(att_weights.squeeze(2), dim=1)
    cntxt = torch.bmm(
      lstm_outputs.transpose(1, 2),
      att_weights.unsqueeze(2)
      )
    concatted = torch.cat(
      (cntxt, final_hidden.unsqueeze(2)),
      dim=1
      )
    att_hidden = self.tanh(self.align(concatted.squeeze(-1)))
    return att_hidden, att_weights
```

И даже в сверточной архитектуре



📝Self Attention in Convolutional Neural Networks

Итоги

Эльбрус буткемп

- задача перевода – задача генерации последовательности токенов
- такой подход называется **авторегрессионным**, т.к. каждый следующий токен зависит от того, что было сгенерировано раньше
- attention позволяет сети «сфокусироваться» на нужных элементах исходного текста
- одного attention не существует – это набор архитектур, которые нужно применять и анализировать результат
- attention – это обычная обучаемая часть нейронной сети