Convolutional Sequence to Sequence Learning

Denis Yarats

with Jonas Gehring, Michael Auli, David Grangier, Yann Dauphin

Facebook AI Research

Sequence generation

- Need to model a conditional distribution

$$\Pr(\mathbf{x}) = \prod_t \Pr(x_t | x_{1:t-1})$$

- Repeatedly predict what will happen next, use your past predictions as if they were real

Sequence generation

- Language modeling
- Machine translation
- Speech generating
- Image generation
- etc.

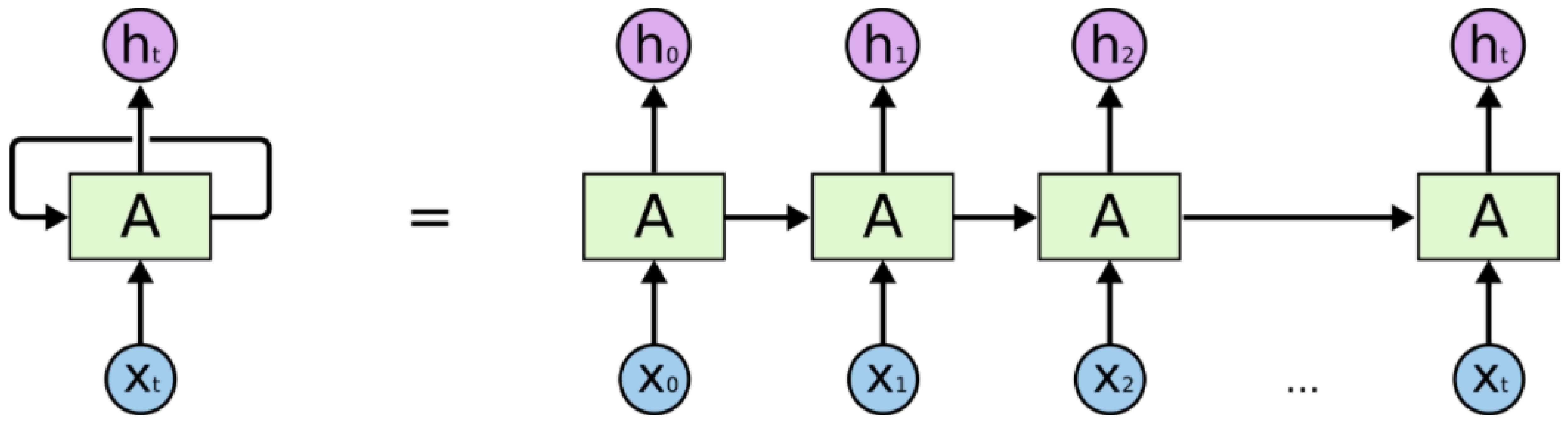
Sequence generation

- How to mode $\Pr(\mathbf{x}) = \prod_t \Pr(x_t | x_{1:t-1})$?

- Let's use Recurrent Neural Network

Recurrent Neural Network

- Like feed forward networks, except allows self connections
- Self connections are used to build an internal representation of past inputs
- They give the network memory



An unrolled recurrent neural network.

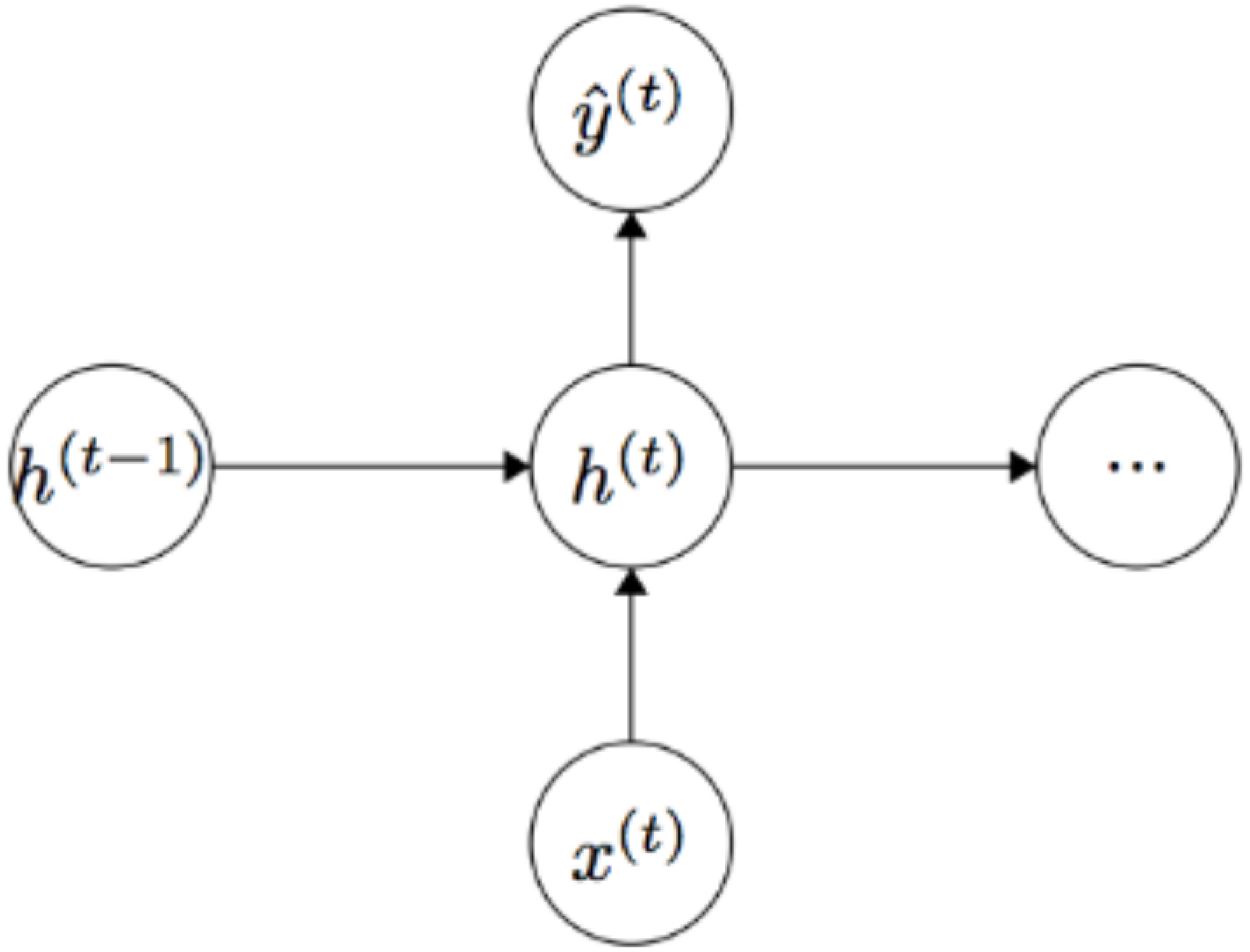
Figures from: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Recurrent Neural Network

- Given list of inputs $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$
- At each timestamp do:

$$
\begin{aligned}
h_t &= \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right) \\
\hat{y}_t &= \operatorname{softmax}\left(W^{(S)} h_t\right)
\end{aligned}
$$

- Then: $\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$



Figures from: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Recurrent Neural Network

- $\hat{y} \in \mathbb{R}^{|V|}$ is probability distribution over vocabulary
- To train the network, minimize cross-entropy:

$$J^{(t)}(\theta) = -\sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$

Figures from: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Recurrent Neural Network

- The notorious vanishing/exploding gradients problem

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W}$$

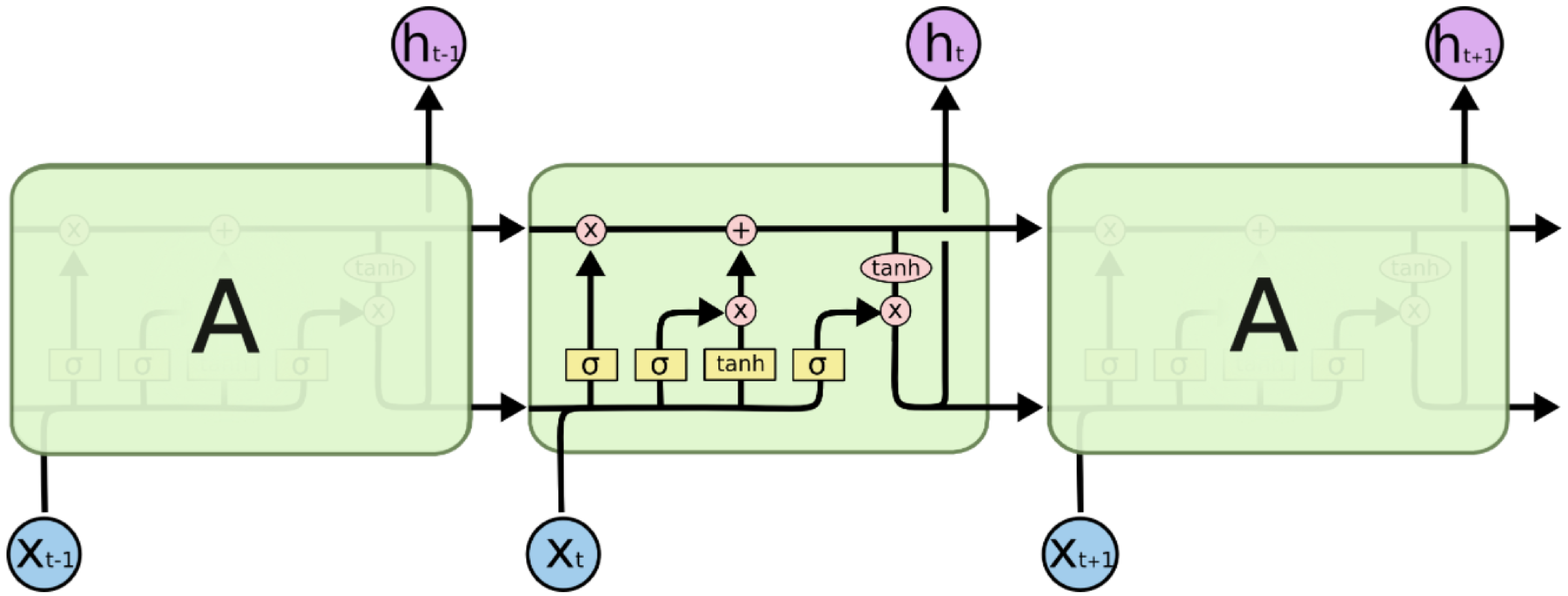
where $h_t = Wf(h_{t-1}) + W^{(hx)}x_{[t]}$

thus $$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

Figures from: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Long Short Term Memory (LSTM)

- Modification of RNN to have longer memory
- Additional memory cell to store information
- RNN overwrites the hidden state, LSTM adds to the hidden state
- Hochreiter ε

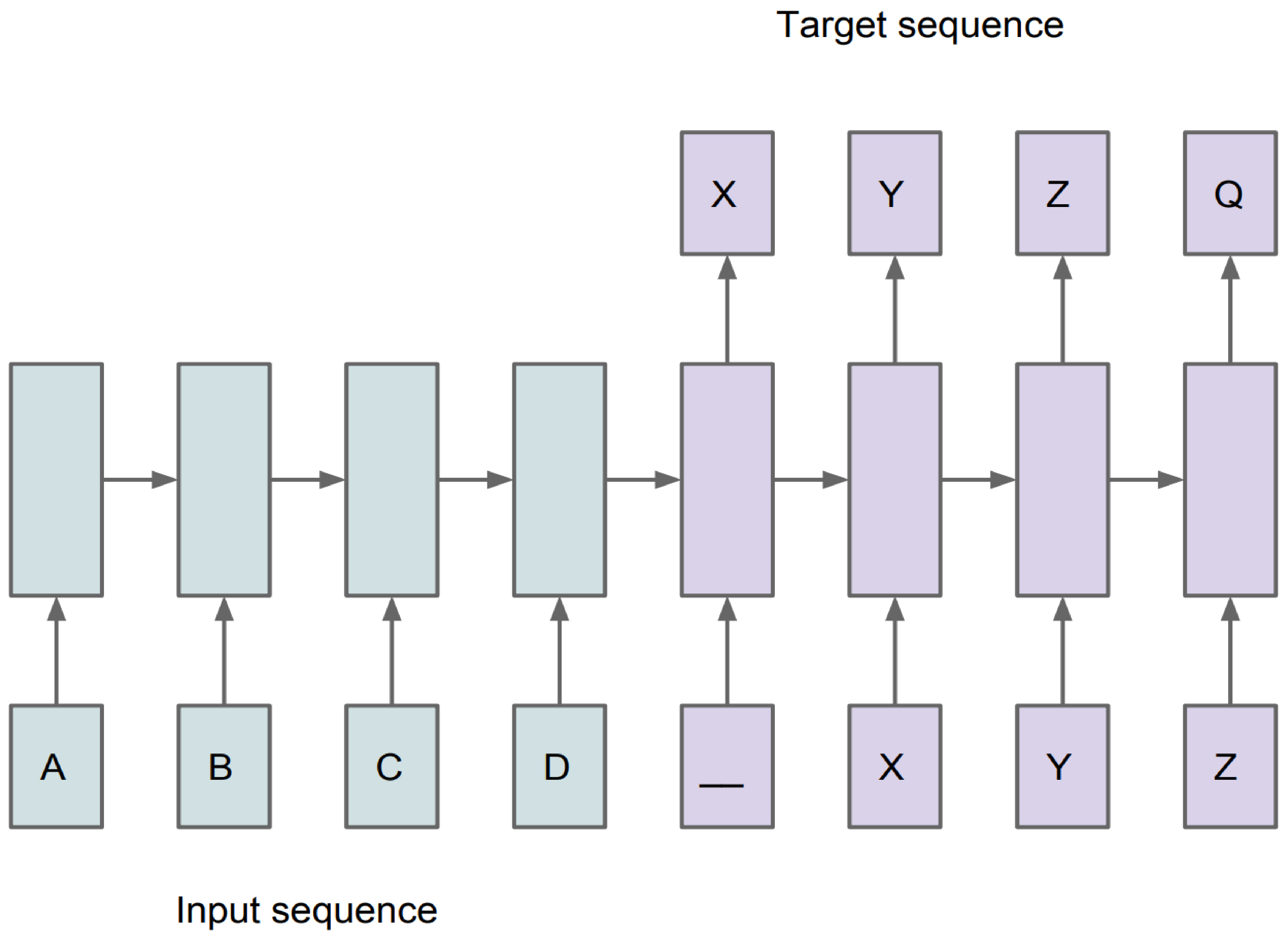


The repeating module in an LSTM contains four interacting layers.

Figures from: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Sequence to Sequence

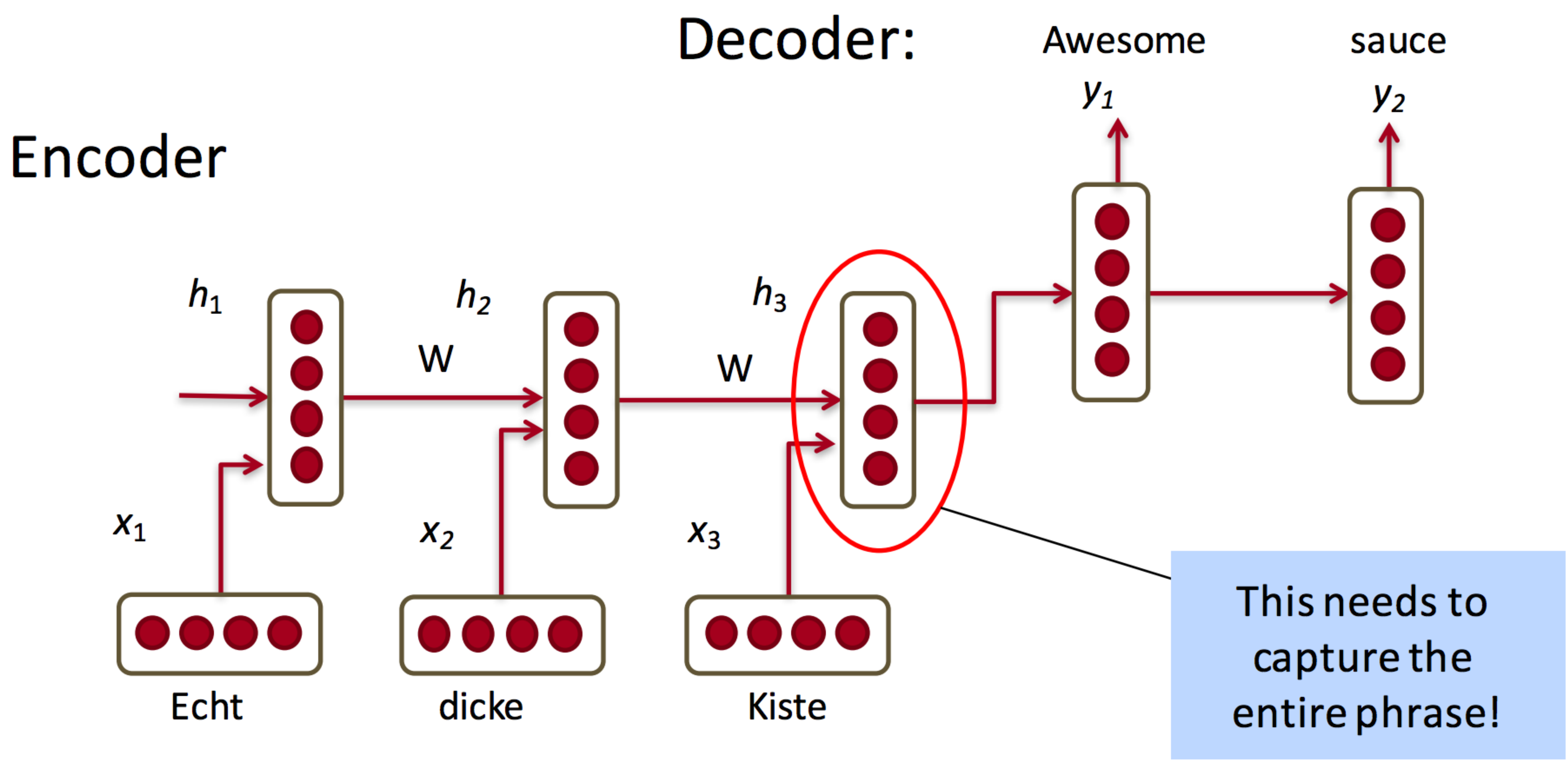
- Make NN to read one sequence and produce another
- Use 2 LSTMs
- Sutskever et al. 201



Figures from: http://www.iro.umontreal.ca/~bengioy/cifar/NCAP2014-summerschool/slides/Ilya_LSTMs_for_Translatior

Sequence to Sequence

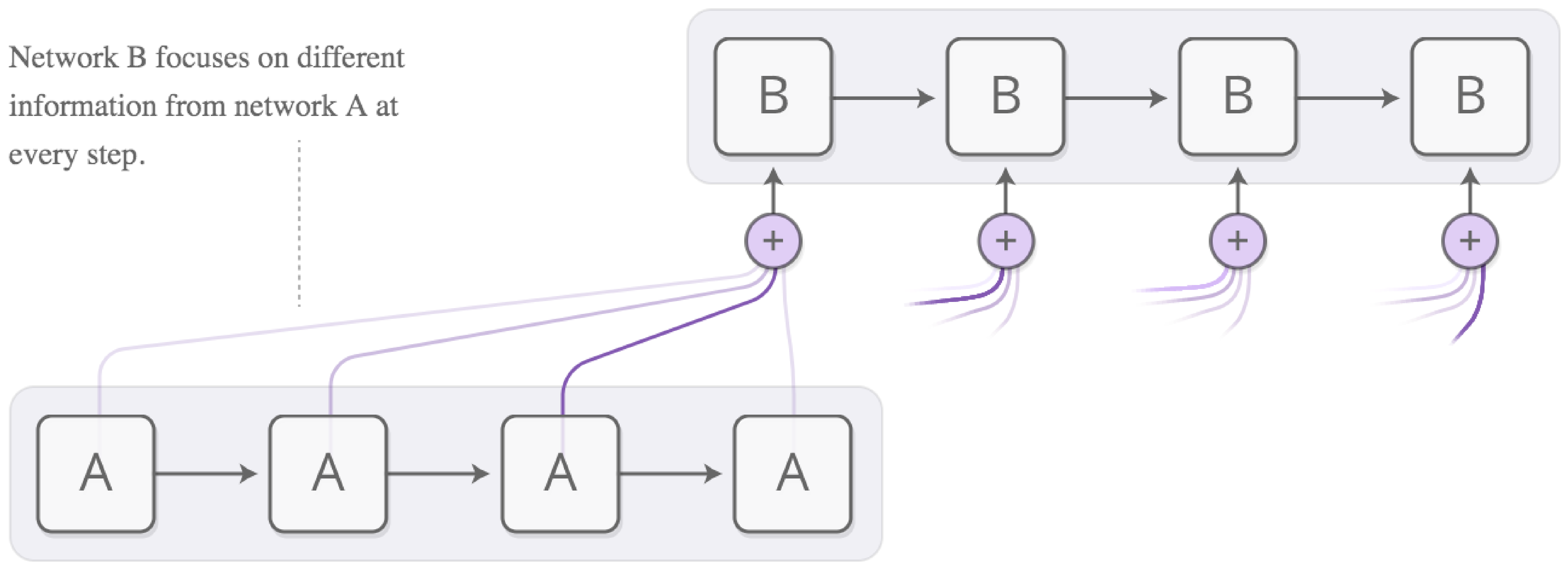
- Encoder - encodes input sequence
- Decoder - generates output sequence, conditioning on the input representation



Figures from: http://cs224d.stanford.edu/lectures/CS224d-Lecture9.pdf

Attention

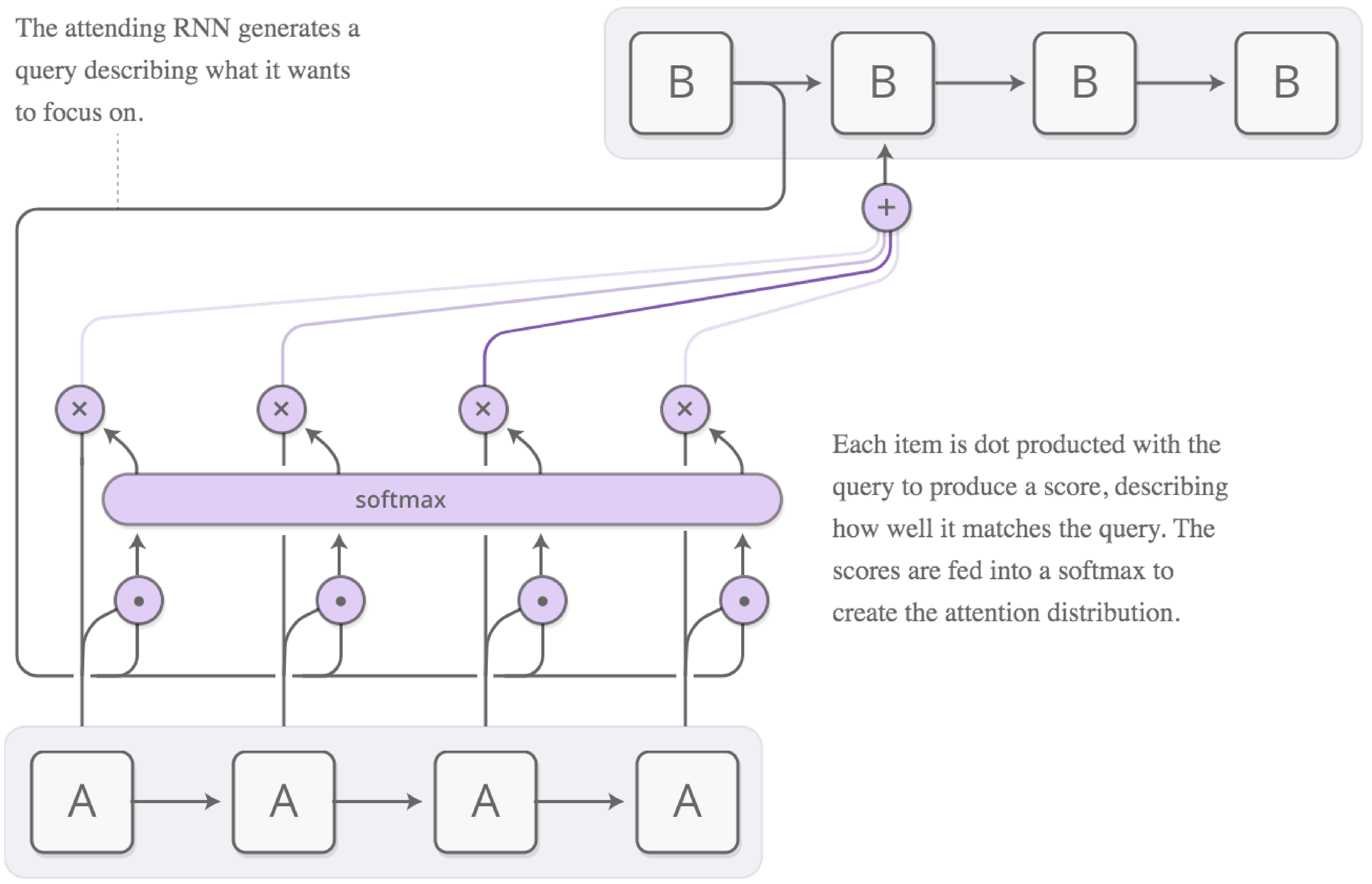
- Incorporate all hidden states of encoder, rather than the last one
- Bahdahau et al. 2014



Figures from: https://distill.pub/2016/augmented-rnns

Attention

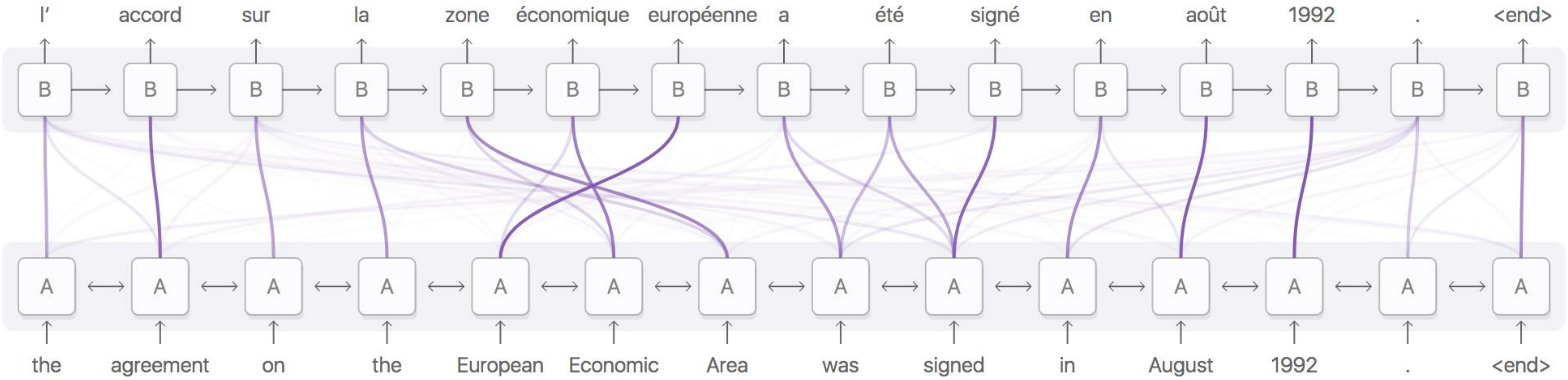
- Use weighted combination of each encoder hidden state
- Alignment mode $e_{ij} = a(s_{i-1}, h_j)$
- Weights: $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$
- Weighted sum $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$



Figures from: https://distill.pub/2016/augmented-rnns

Attention

- Example of attention for machine translation



Figures from: https://distill.pub/2016/augmented-rnns

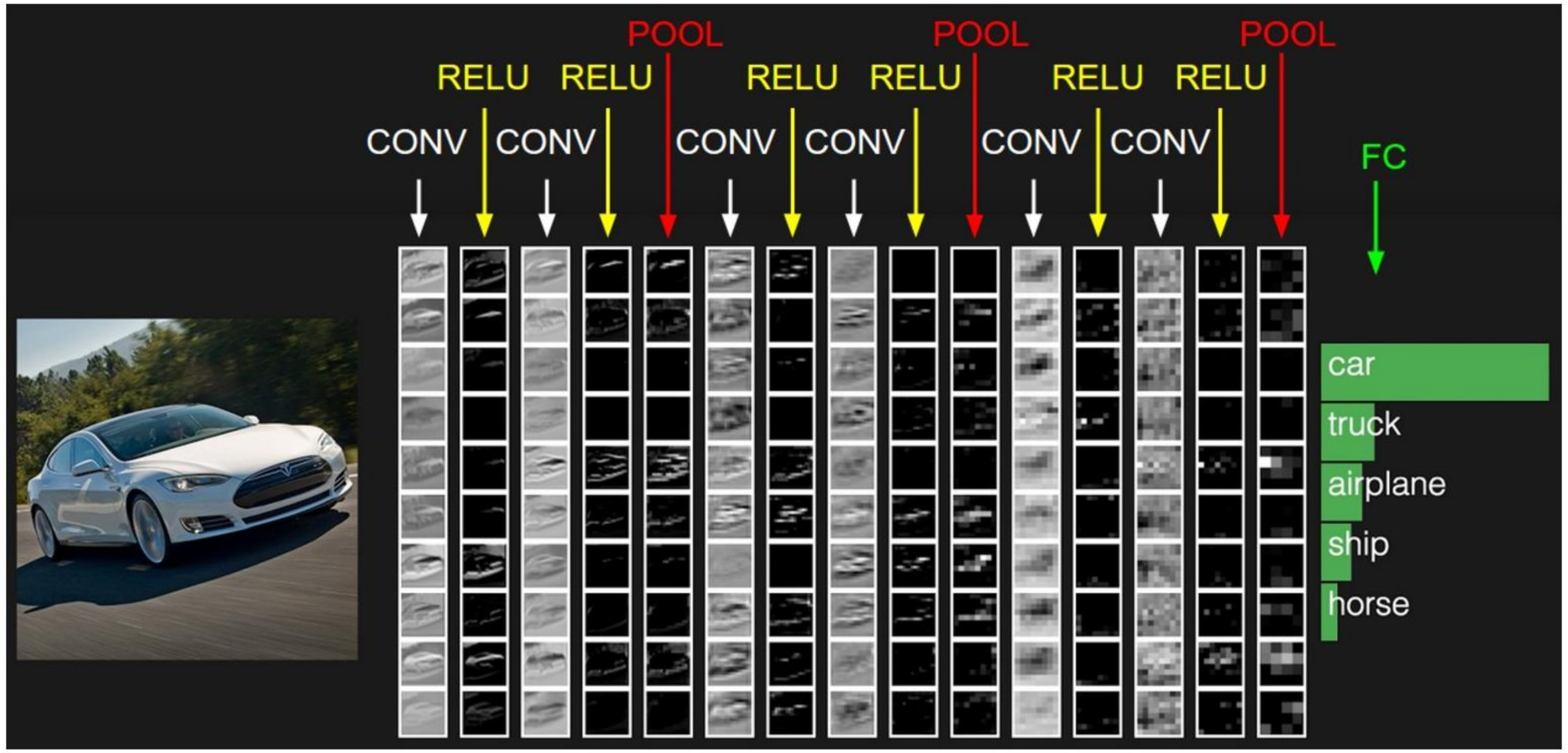
LSTM based Seq2Seq: Problems

- Still challenging to model long term dependencies
- RNNs are hard to parallelize because of non-homogeneous nature

- Convolutional neural networks to the rescue?

Convolutional Neural Network

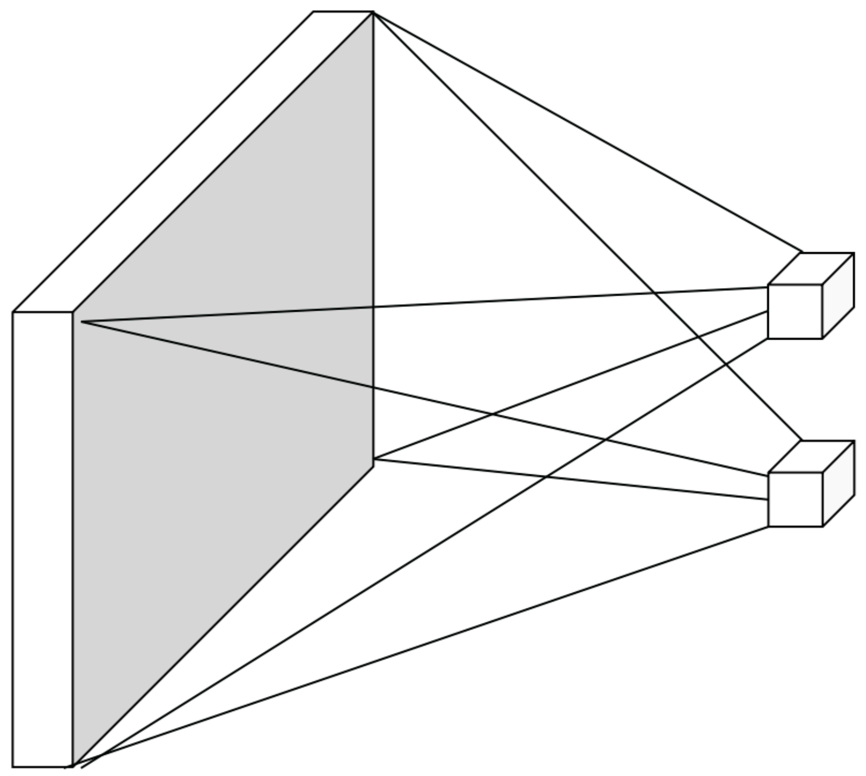
- Great success in computer vision: AlexNext, VGG, ResNet, ...
- Efficient implementation on GPU



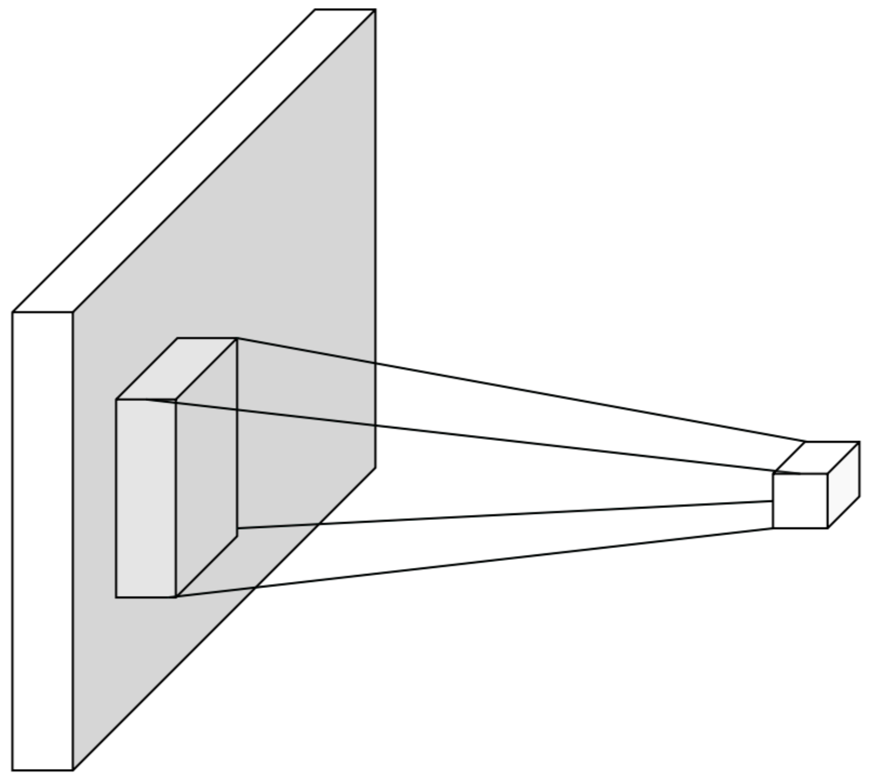
Figures from: http://cs231n.github.io/convolutional-networks

Convolutional Neural Network

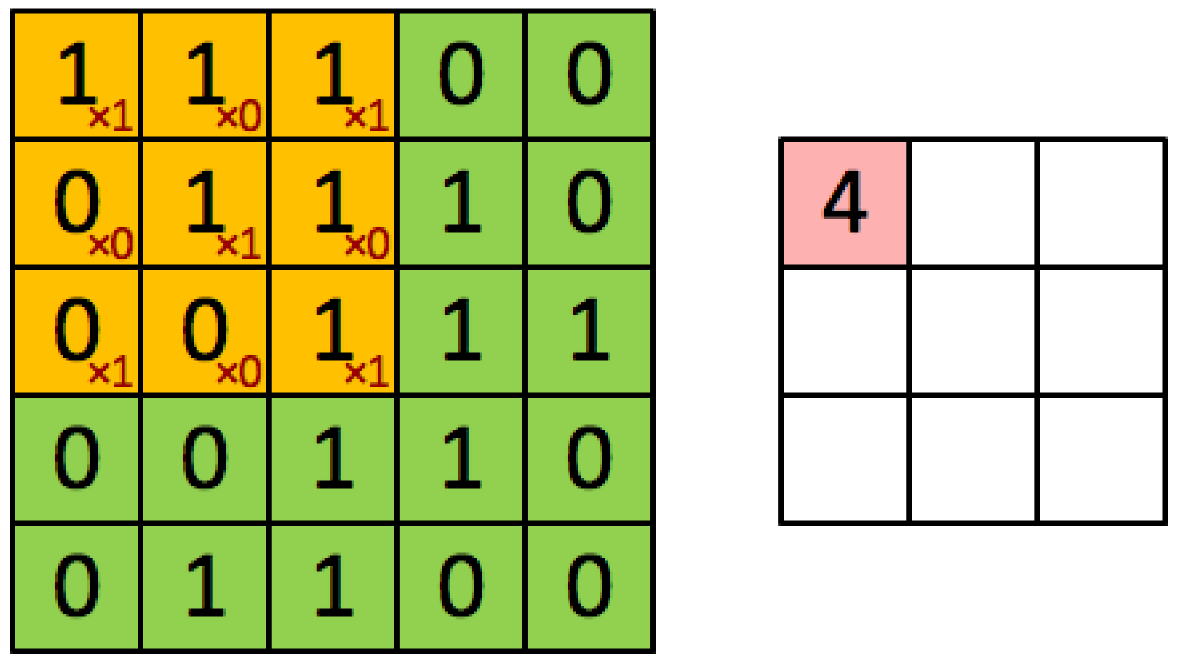
- Each output neuron is a linear combination of input neurons in some spatial neighborhood
- Unlike a feed forward network, where it is connected to every input neurons
- Parameter sharing and better scalability



fully connected



convolution



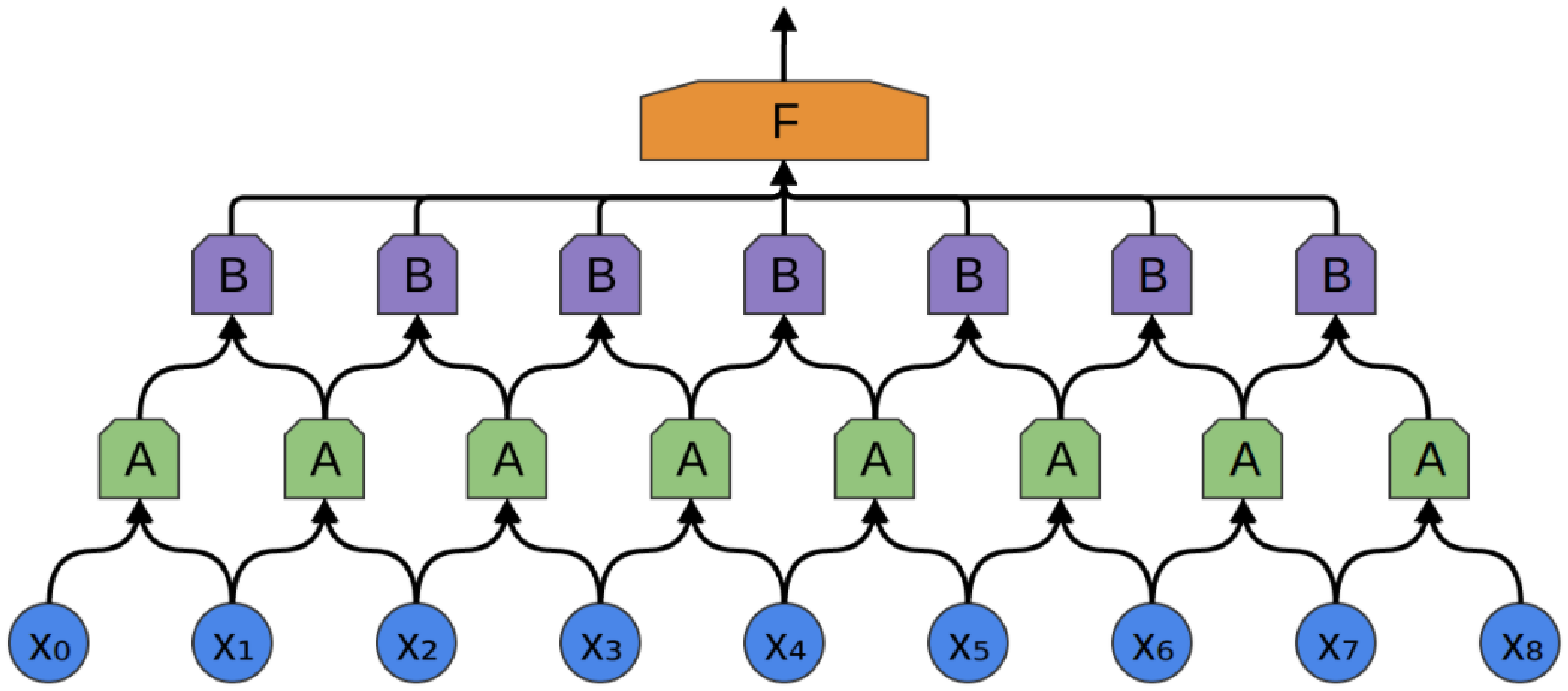
Image

Convolved Feature

Figures from: http://cs231n.github.io/convolutional-networks

Convolutional Neural Network

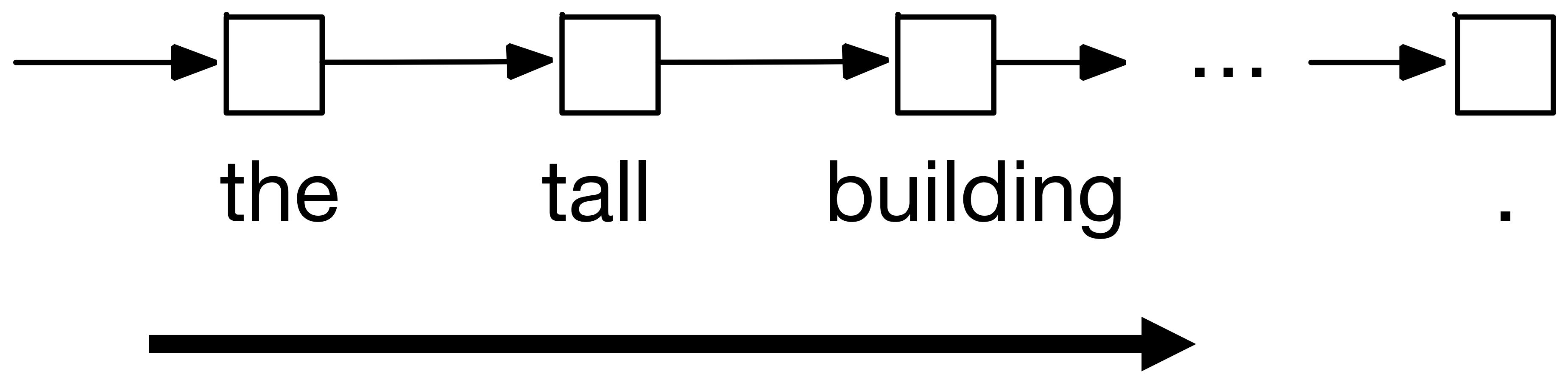
- Convolution can also be used for sequences
- Temporal convolution



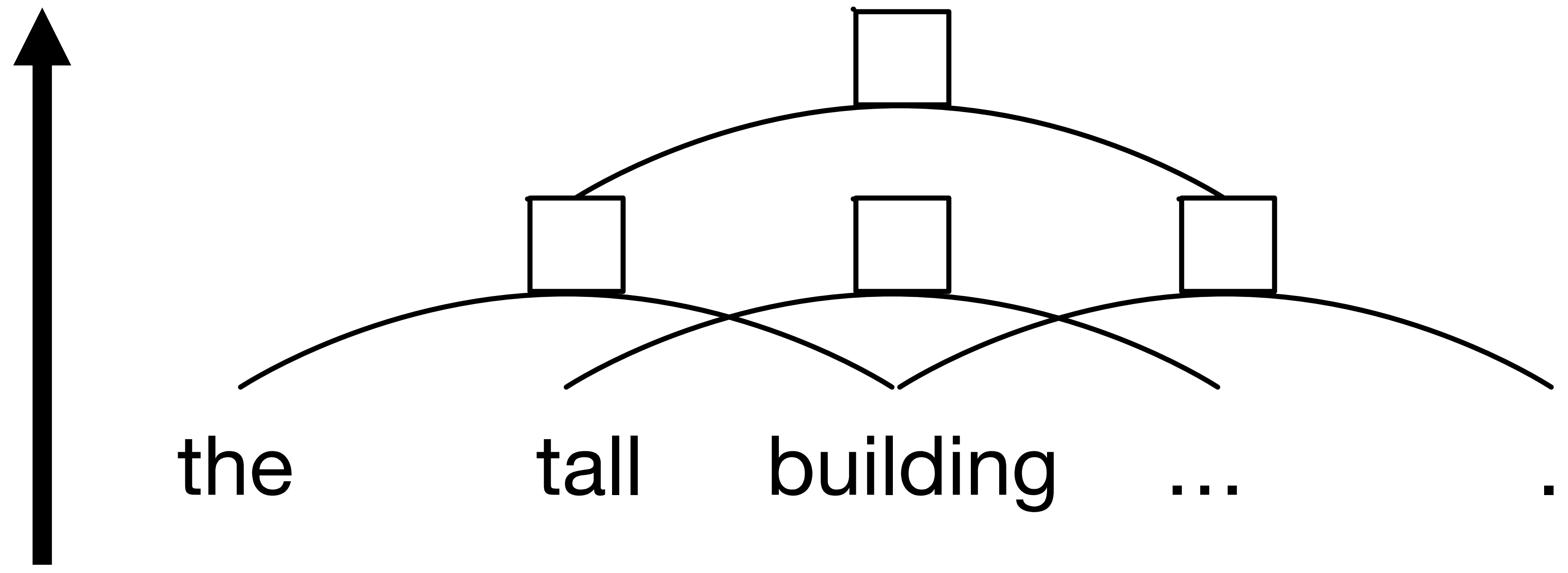
Figures from: http://colah.github.io/posts/2014-07-Conv-Nets-Modular

Convolutional Neural Network

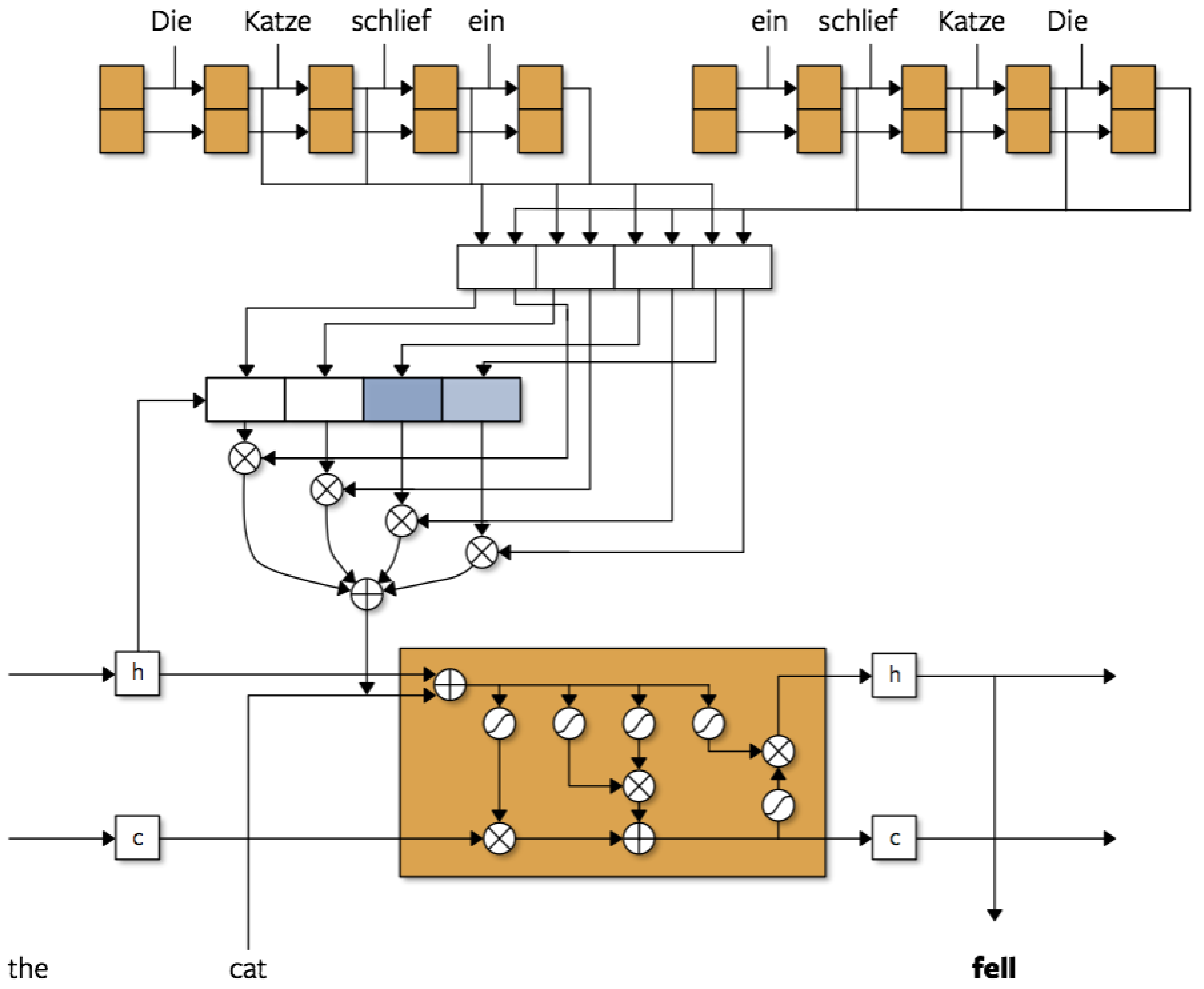
- Hierarchical processing: bottom-up vs. left-right
- Homogeneous: all elements processed in the same way
- Scalable computation: parallelizable, suited for GPUs



vs.



ConvS2S: LSTM based Seq2Seq

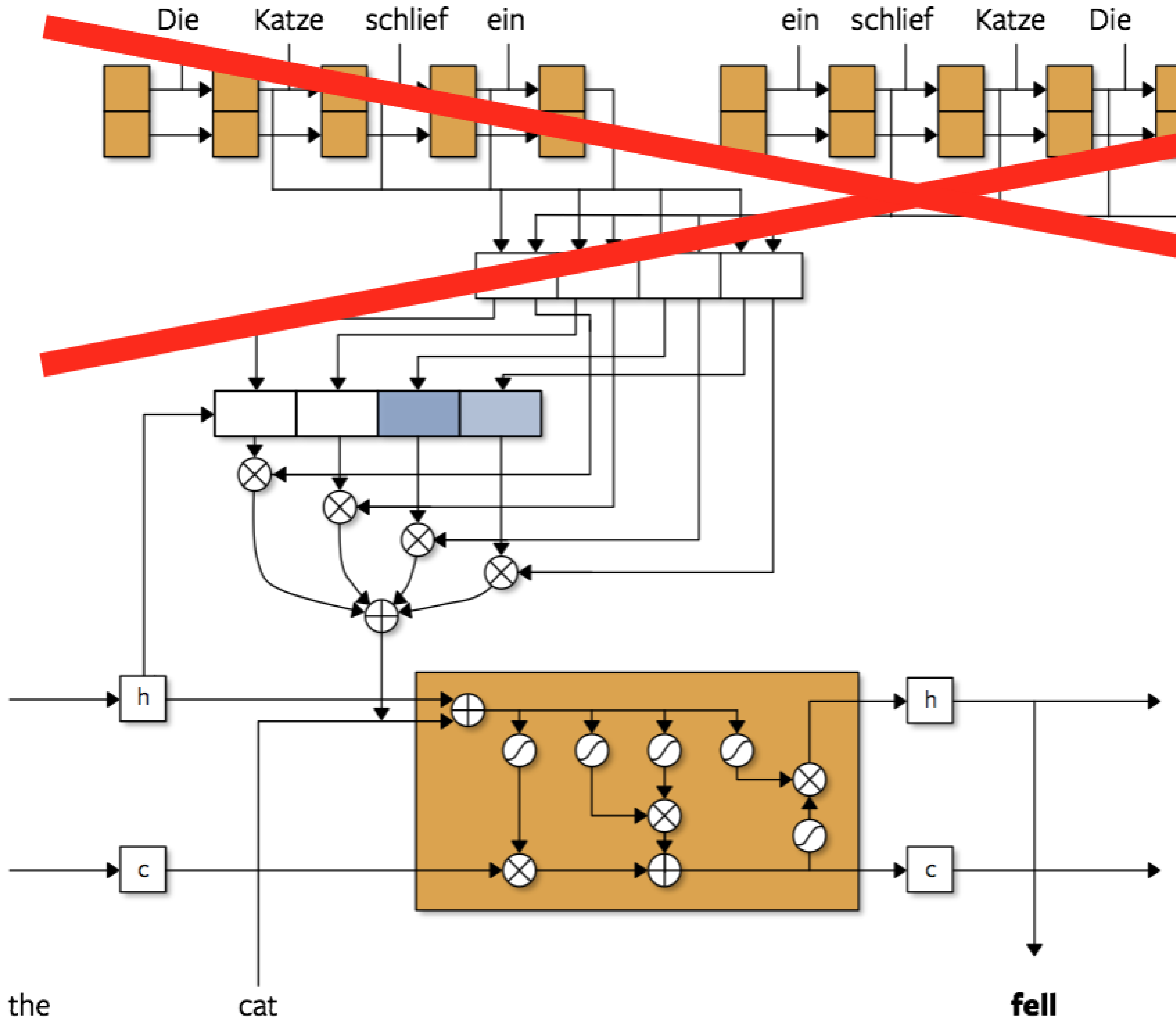


Bi-directional LSTM Encoder

Soft Attention

LSTM Decoder

ConvS2S: Convolutional Encoder

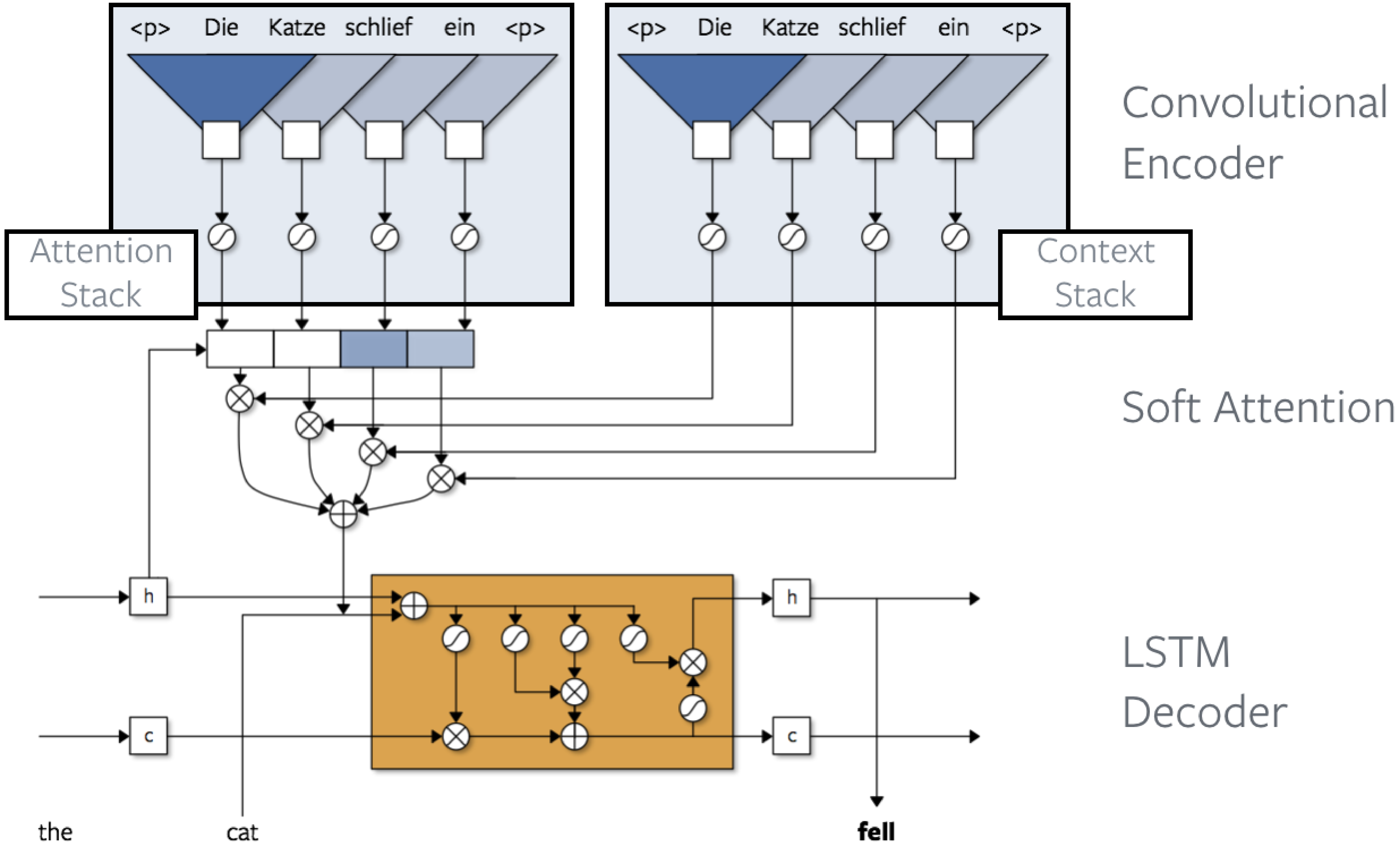


~~Bi-directional LSTM Encoder~~

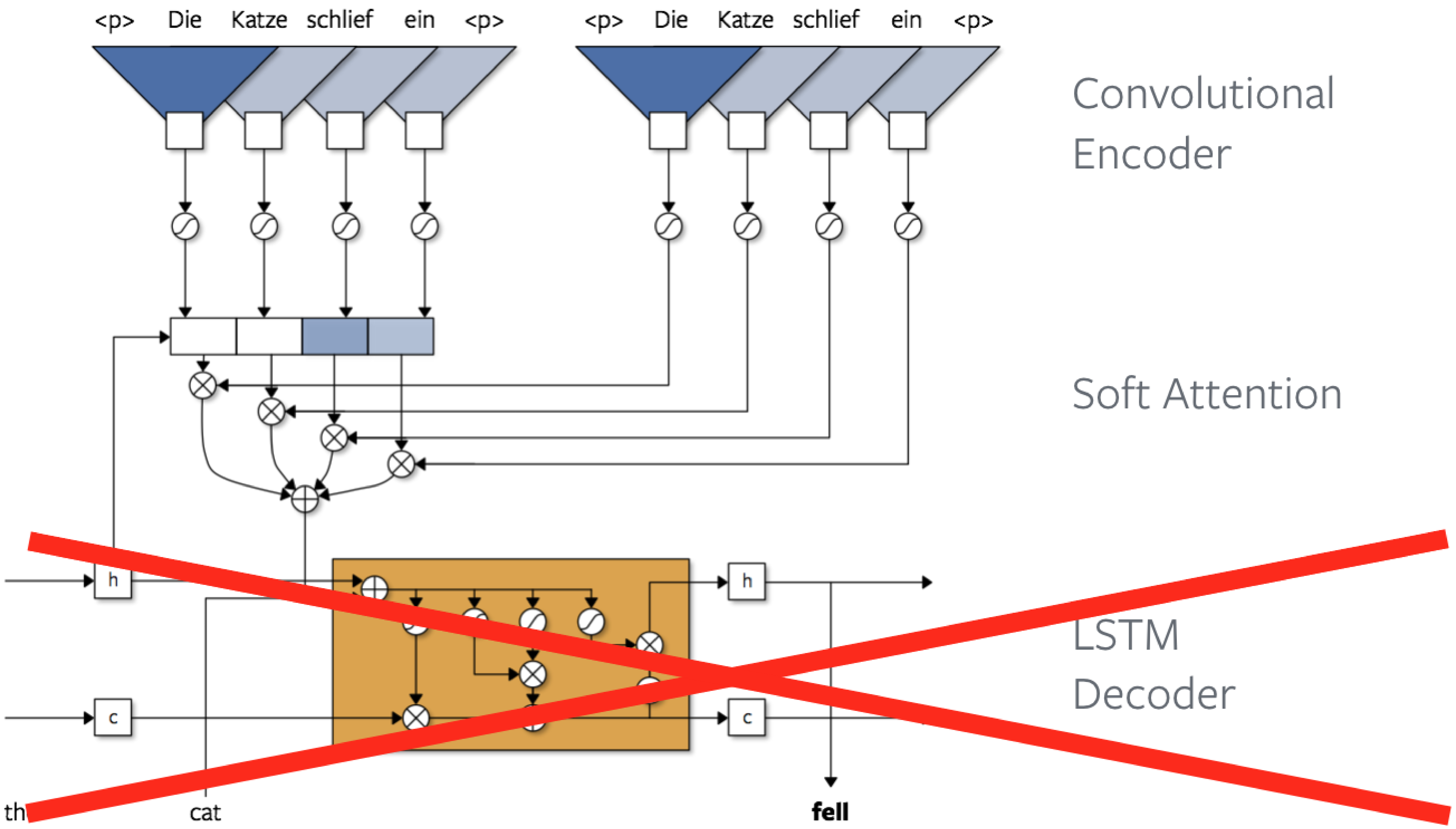
Soft Attention

LSTM Decoder

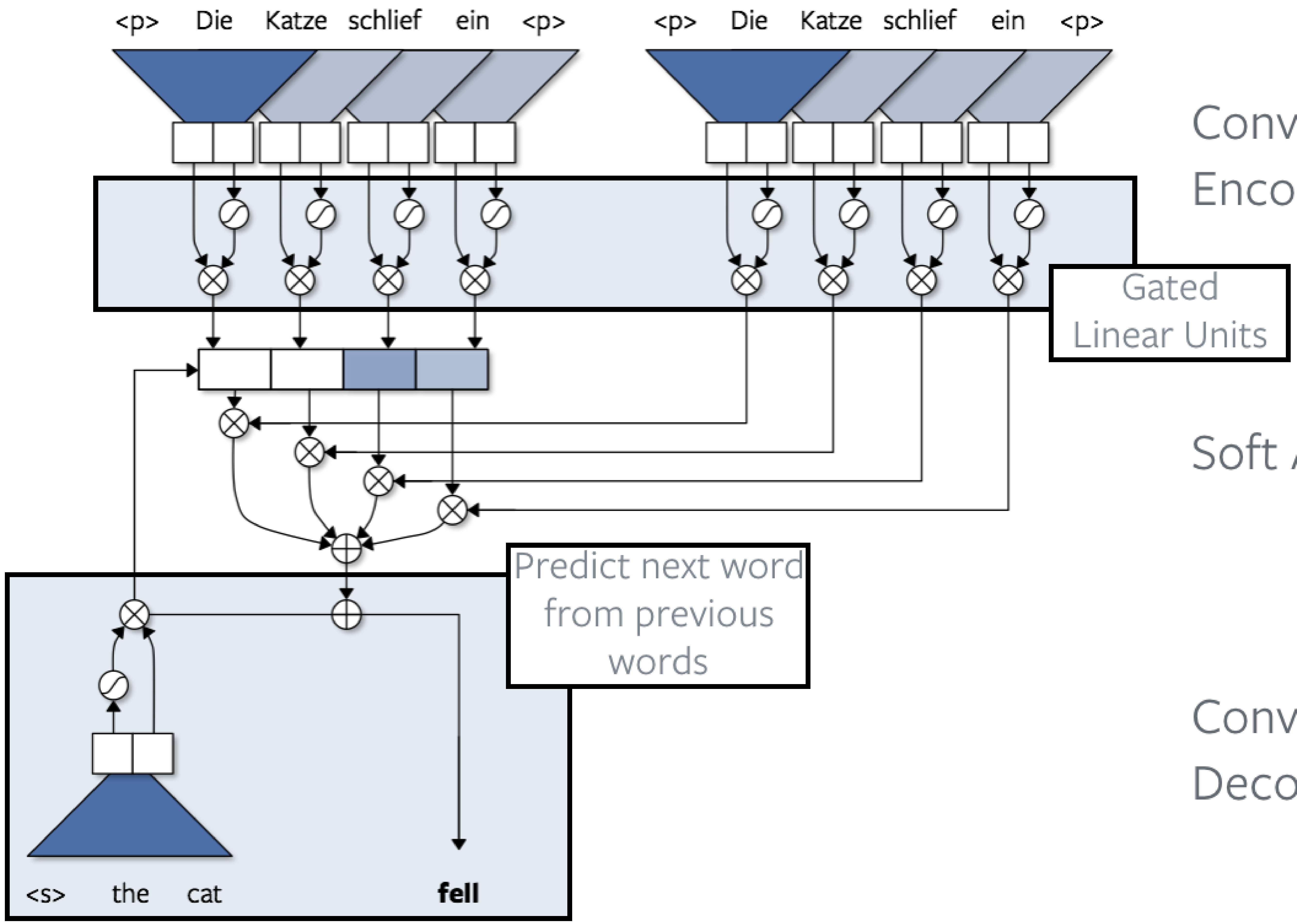
ConvS2S: Convolutional Encoder



ConvS2S: Convolutional Decoder



ConvS2S: Convolutional Decoder



Convolutional Encoder

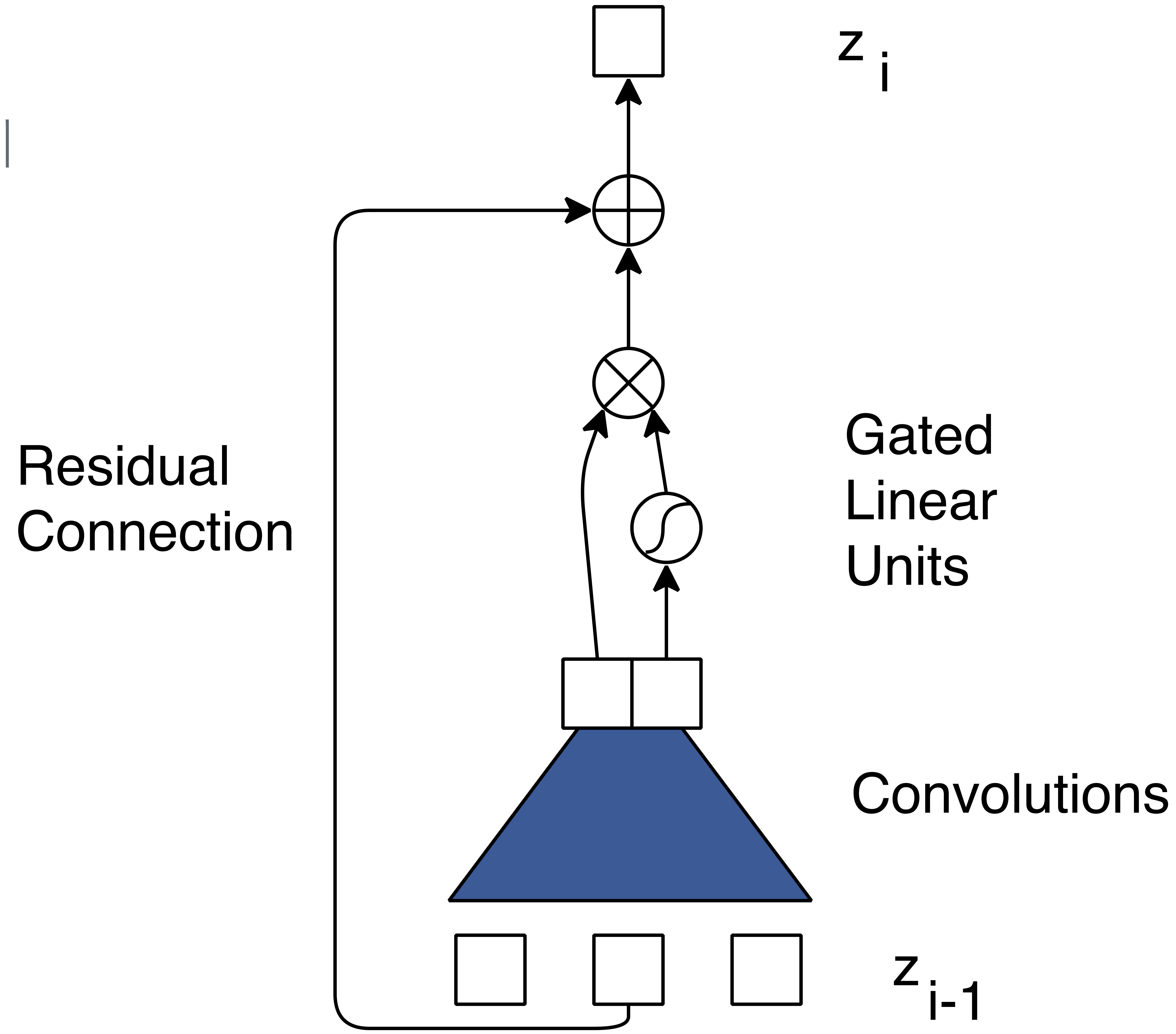
Soft Attention

Convolutional Decoder

ConvS2S: Encoder

Convolutional block structure for encoder:

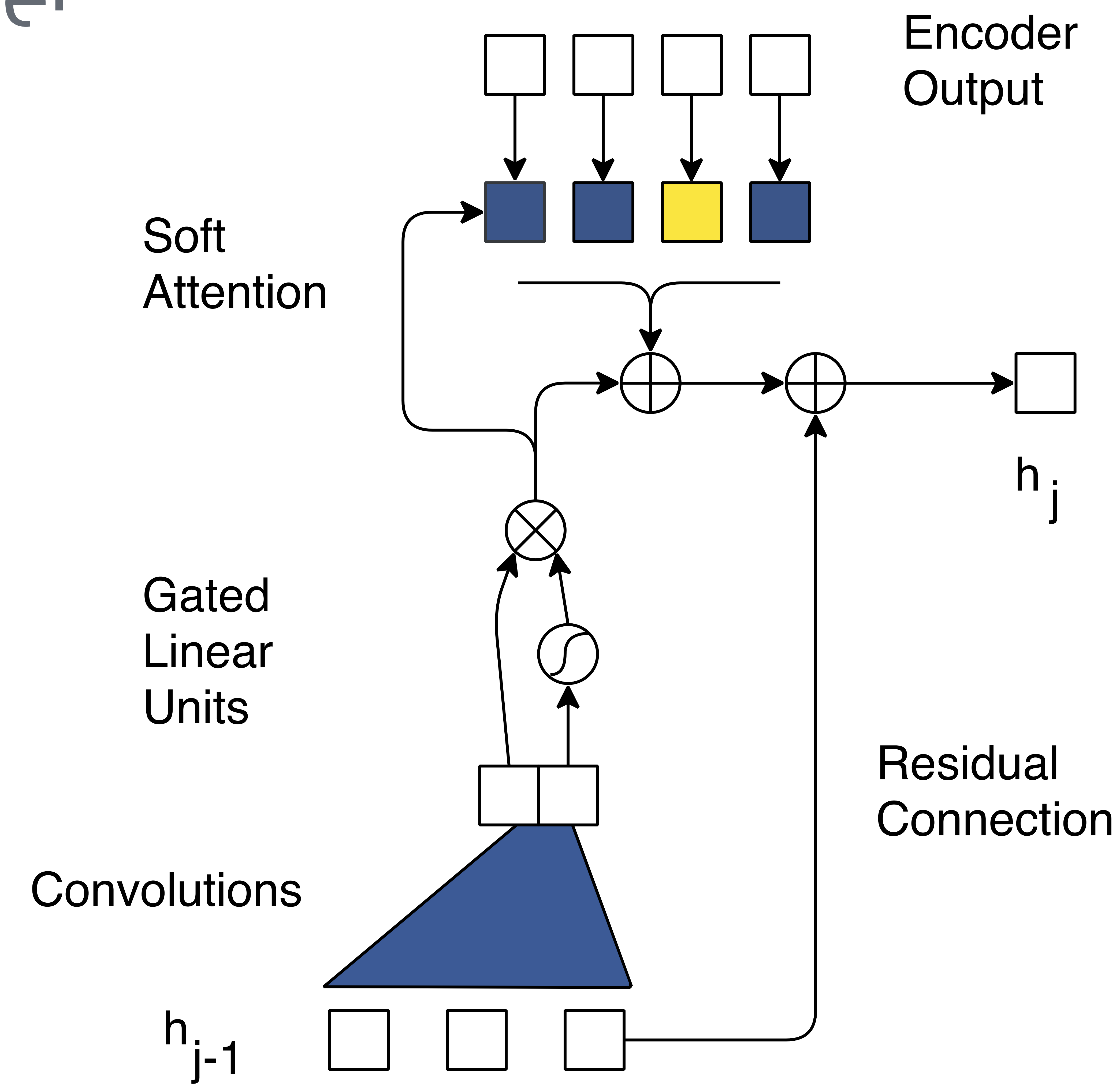
- Gated linear units, residual connectio
- z_0 = embeddings
  - word or sub-word embedding
  - plus position embedding: 0,1,2,...



ConvS2S: Decoder

Convolutional block structure for decoder:

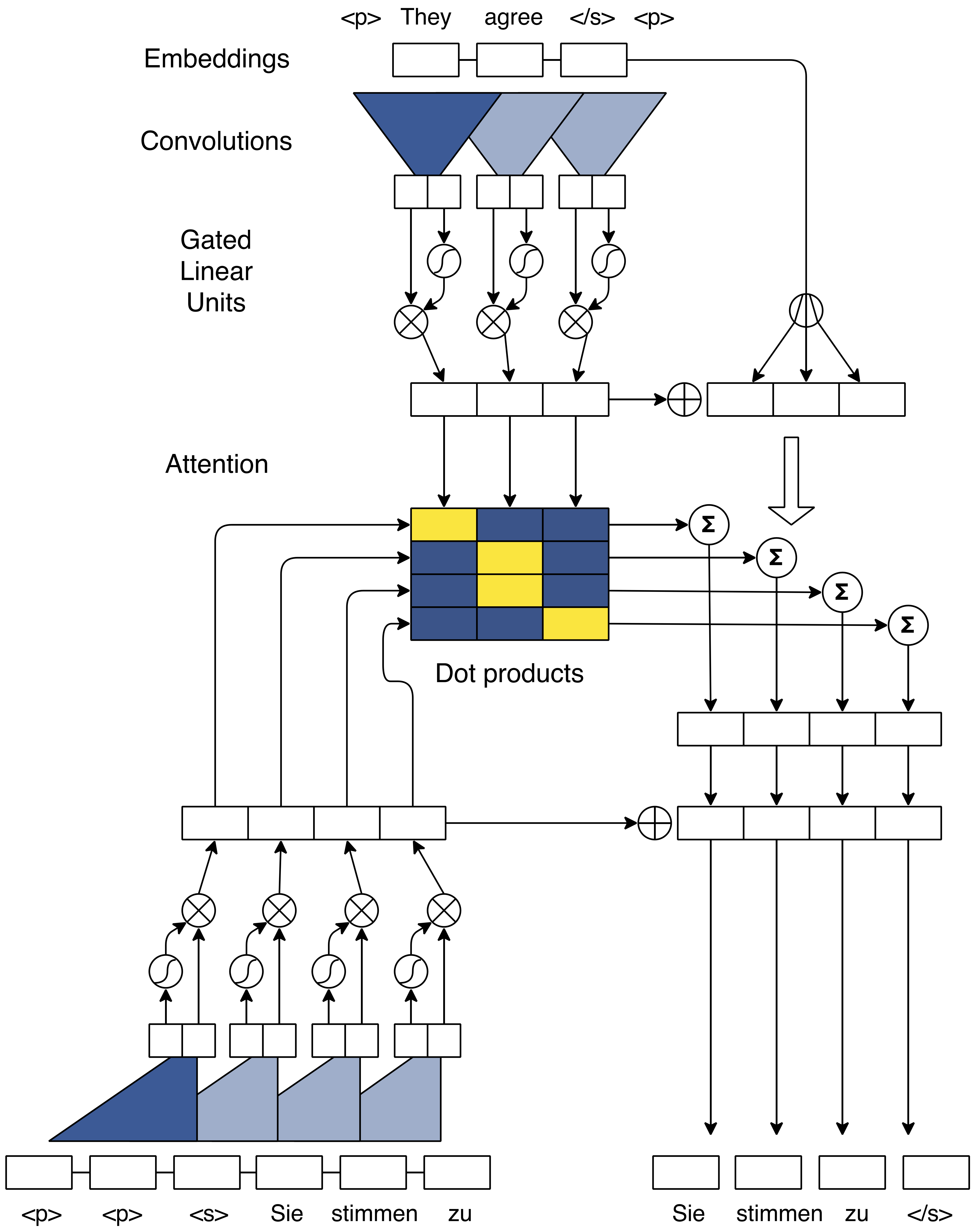
- Gated linear units and residual connections
- h_0 = embeddings (word + pos)
- Soft attention pass at every layer



ConvS2S: All together

Putting it all together:

- Here: single-layer encoder and decoder
- High training efficiency due to parallel computation in decoder



ConvS2S: Architecture

- 15 layers in both encoder and decoder
- Convolutional kernel size is 3
- Hidden size gradually increases from 512 to 4096
- Embedding size is 512

ConvS2S: Training

- Optimizer: Nesterov with momentum
- Learning rate 0.25, momentum 0.99
- Gradient clipping if norm exceeds 0.1
- Batch size 64
- Data parallel training: all-reduct gradients after each iteration
- Model is implemented in Torch (PyTorch implementation is coming)

ConvS2S: Tricks

- WeighNorm (Salimans et al. 2016)
- Dropout (Srivastava et. al. 2014)
- Scale outputs of each layer to normalize variance
- Careful weight initialization to ensure normally distributed variance

ConvS2S: Results

Results on English-German (WMT'14, newstest2014)

System	Vocabulary	BLEU
ByteNet v2 (Kalchbrenner et al., 2016)	Characters	23.75
GNMT (Wu et al., 2016)	Word 80k	23.12
GNMT (Wu et al., 2016)	Word pieces	24.61
ConvS2S	**BPE 40k**	**25.16**
Transformer (Vaswani et al., 2017)	Word pieces	**28.4***

ConvS2S: Results

Results on English-French (WMT'14, newstest2014)

System	Vocabulary	BLEU
GNMT (Wu et al., 2016)	Word 80k	37.90
GNMT (Wu et al., 2016)	Word pieces	38.95
GNMT + RL (Wu et al., 2016)	Word pieces	39.92
ConvS2S	**BPE 40k**	**40.46**
Transformer (Vaswani et al., 2017)	Word pieces	**41.0***

ConvS2S: Speed

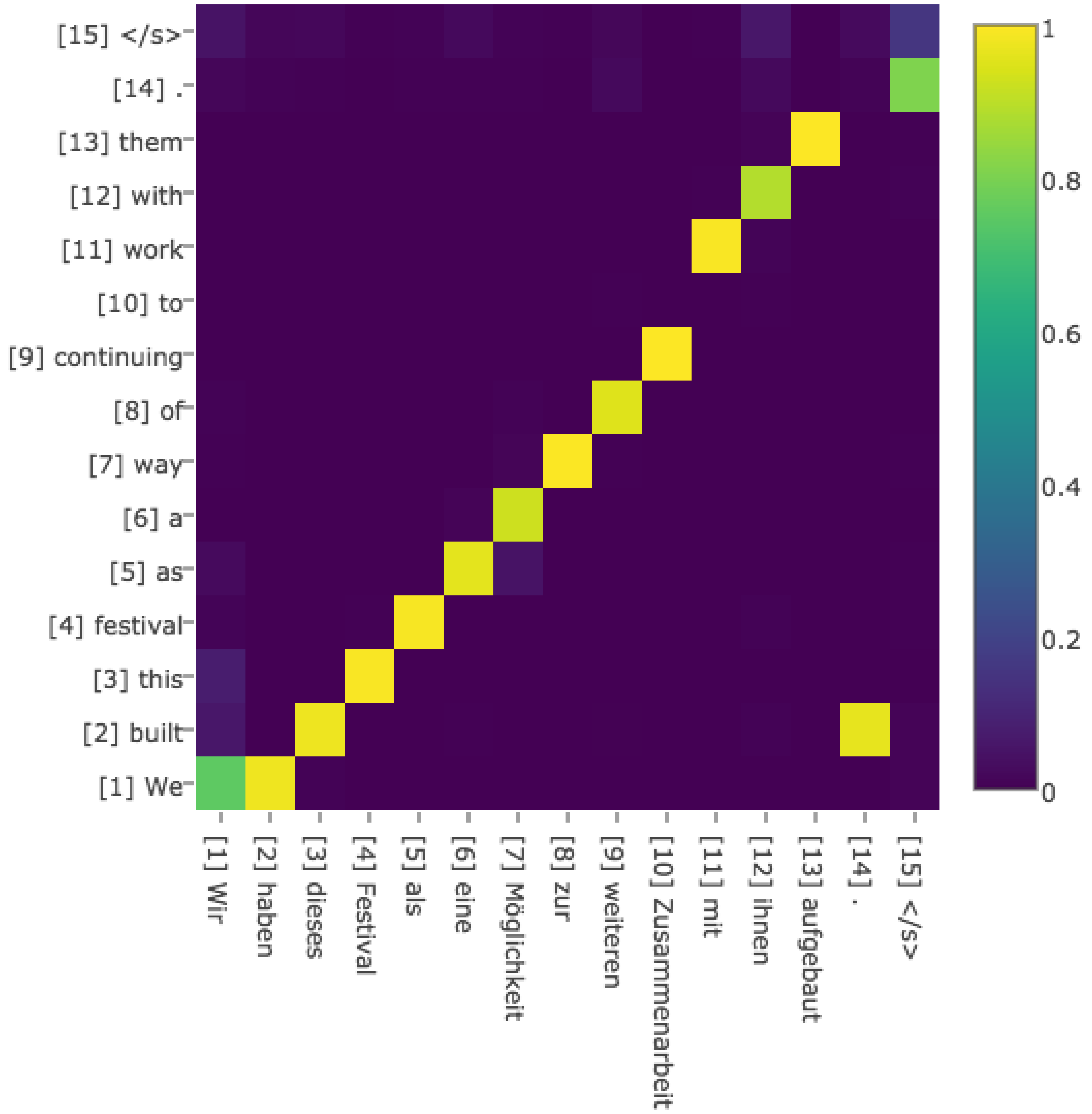
Translation speed on English-French (WMT'14, dev set)

System	Hardware	BLEU	Time (s)
GNMT (Wu et al., 2016)	CPU (88 cores)	31.20	1322
GNMT (Wu et al., 2016)	GPU (K80)	31.20	3028
GNMT + RL (Wu et al., 2016)	TPU	31.21	384
ConvS2S, beam=5	**CPU (48 cores)**	**34.10**	**482**
ConvS2S, beam=5	**GPU (K40)**	**34.10**	**587**
ConvS2S, beam=5	**GPU (GTX-1080ti)**	**34.10**	**406**
ConvS2S, beam=1	**CPU (48 cores)**	**33.45**	**142**

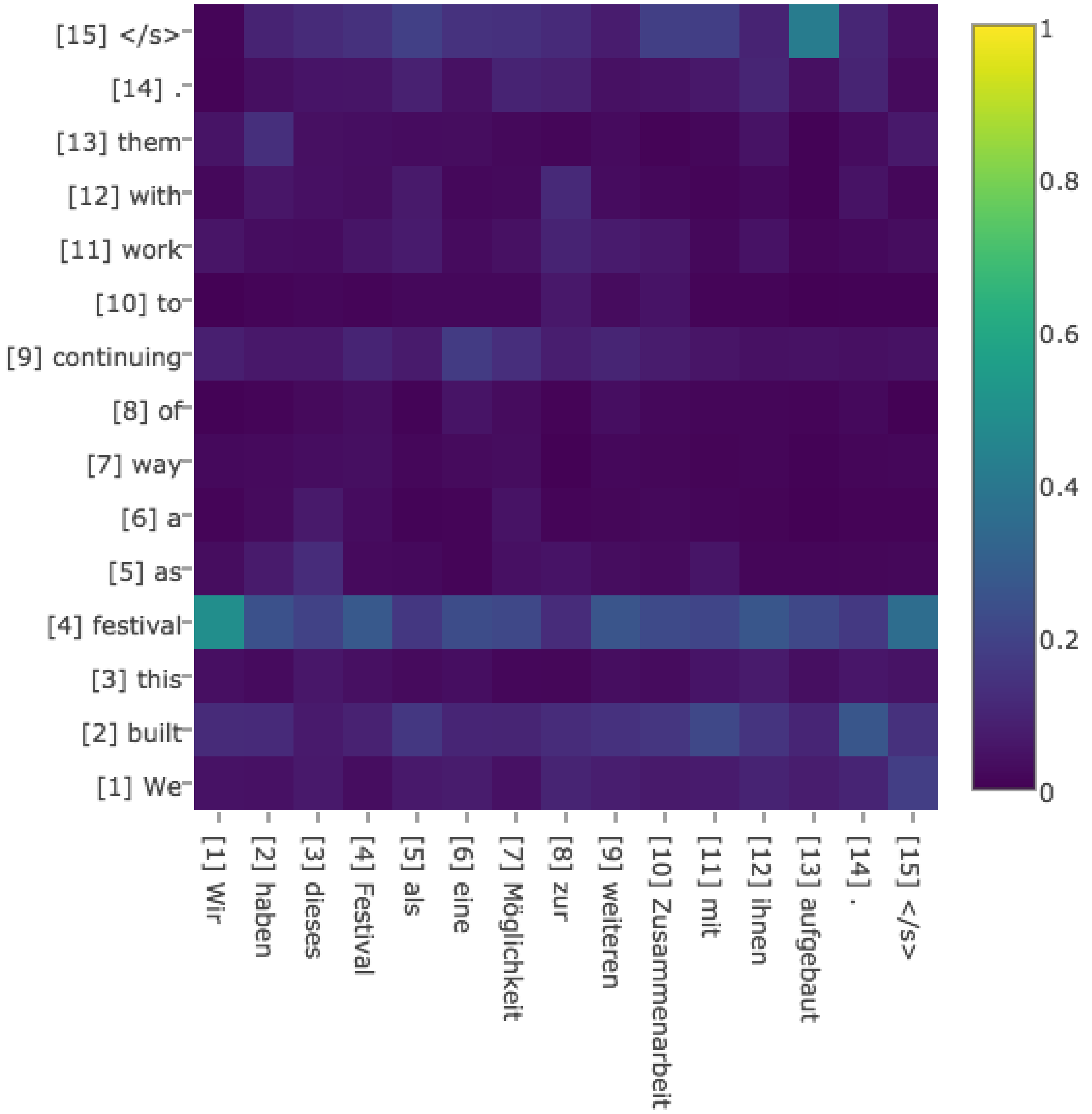
ConvS2S: Training

Dataset	# Pairs (in million)	Hardware	Days
WMT14 EN-DE	4.5	1 x GPU (K40)	18
WMT14 EN-FR	36	8 x GPU (K40)	37

ConvS2S: Multi-Hop Attention

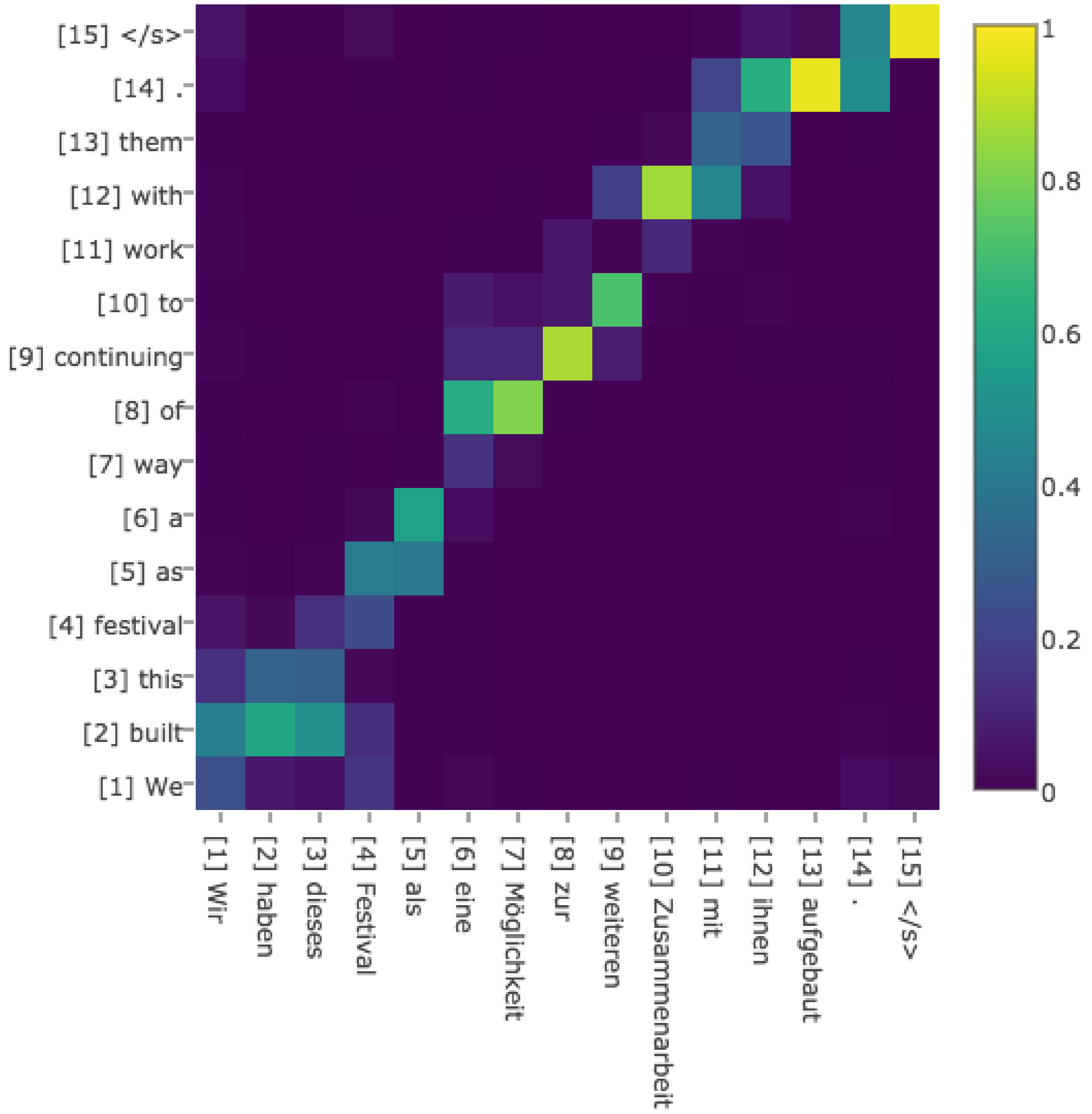


1st Layer

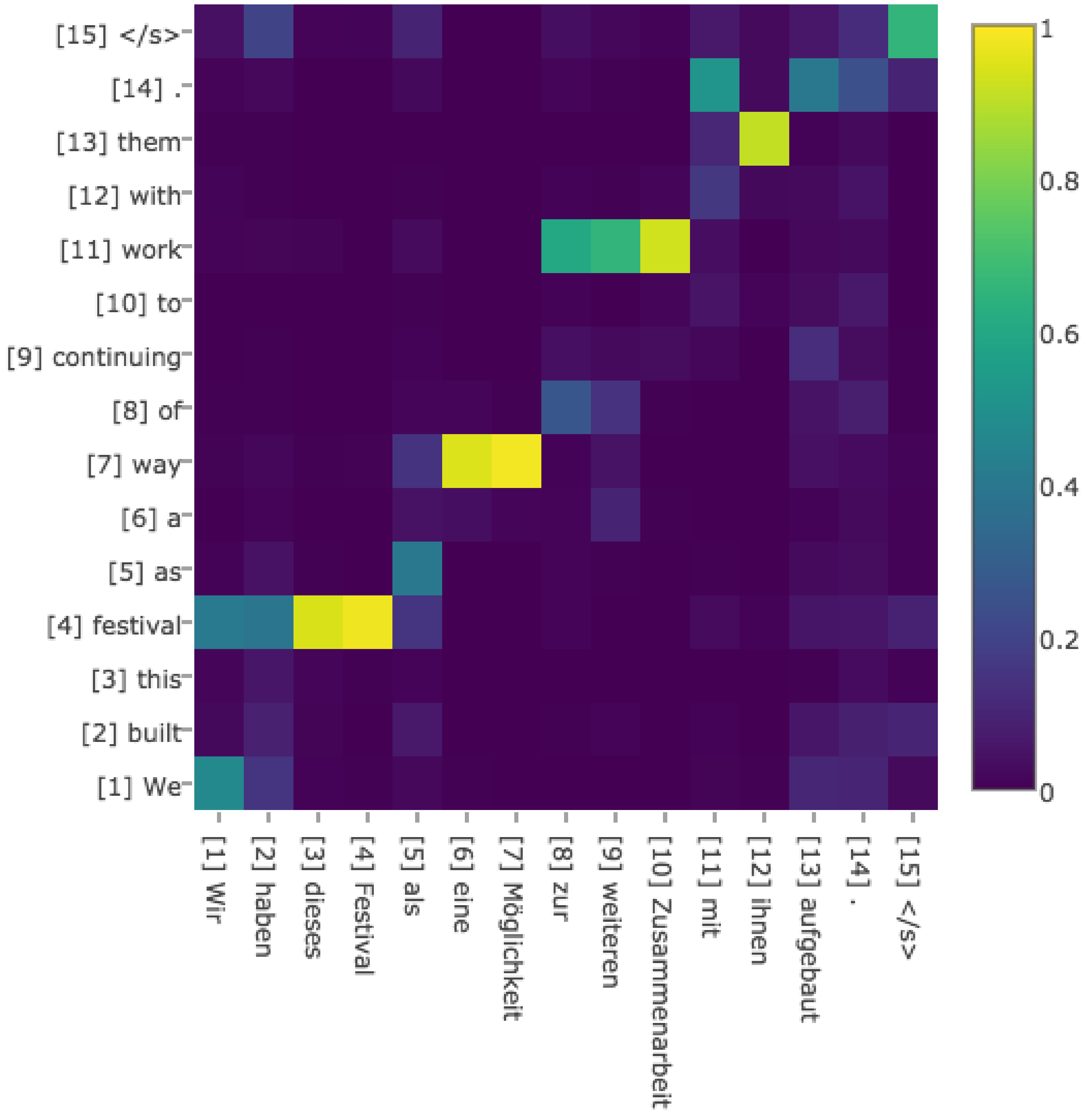


2nd Layer

ConvS2S: Multi-Hop Attention

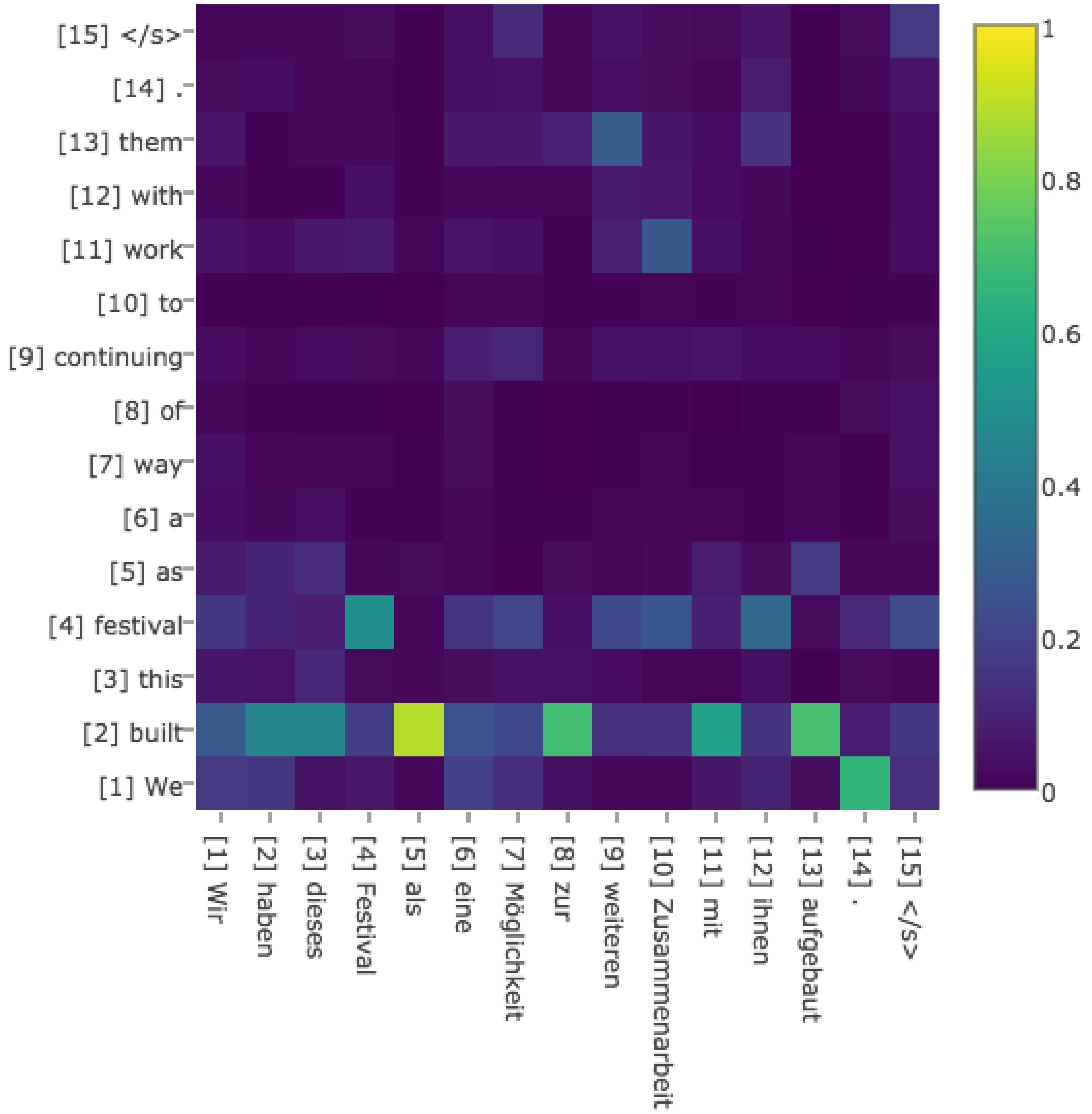


3rd Layer

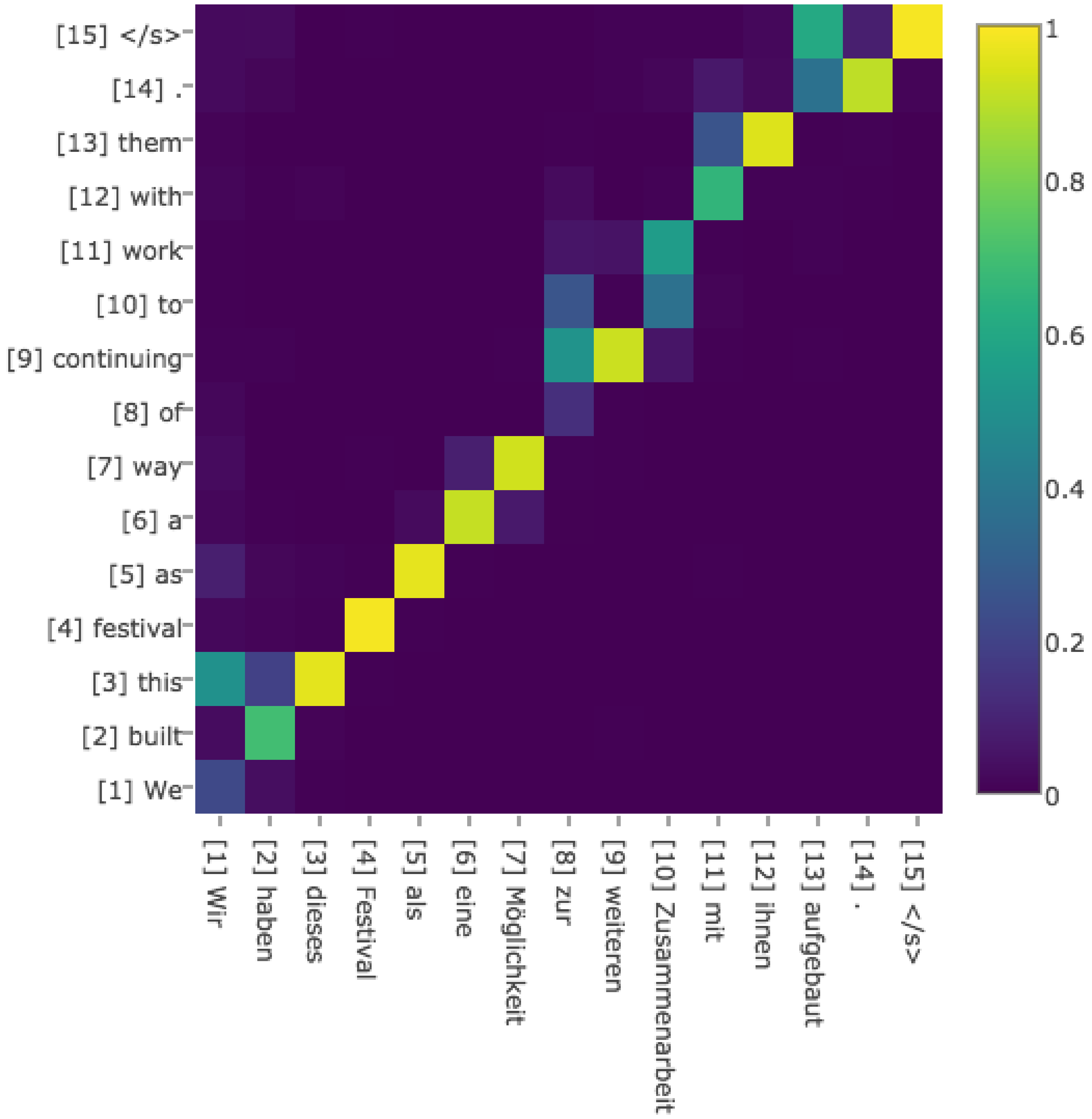


4th Layer

ConvS2S: Multi-Hop Attention



5th Layer



6th Layer

Blogpost

May 9 ARTIFICIAL INTELLIGENCE · RESEARCH

A novel approach to neural machine translation

Jonas Gehring Michael Auli David Grangier Denis Yarats Yann N. Dauphin

Research Paper

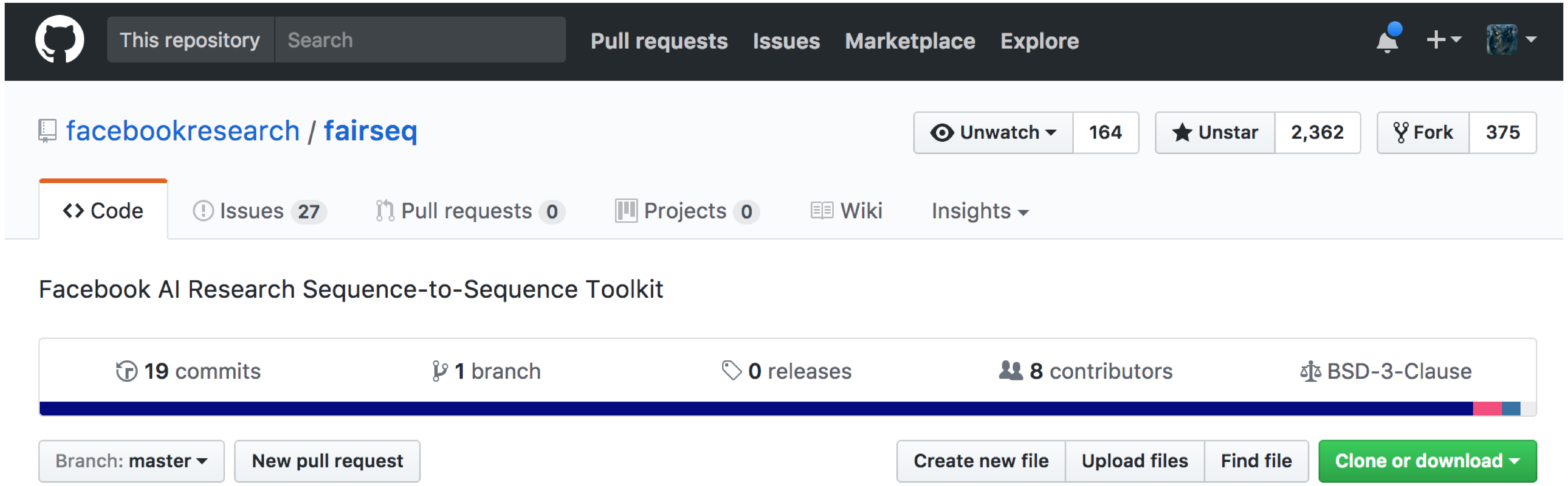
Convolutional Sequence to Sequence Learning

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin

(Submitted on 8 May 2017 (v1), last revised 25 Jul 2017 (this version, v3))

The prevalent approach to sequence to sequence learning maps an input sequence to a variable length output sequence via recurrent neural networks. We introduce an architecture based entirely on convolutional neural networks. Compared to recurrent models, computations over all elements can be fully parallelized during training and optimization is easier since the number of non-linearities is fixed and independent of the input length. Our use of gated linear units eases gradient propagation and we equip each decoder layer with a separate attention module. We outperform the accuracy of the deep LSTM setup of Wu et al. (2016) on both WMT'14 English-German and WMT'14 English-French translation at an order of magnitude faster speed, both on GPU and CPU.

Source Code


This repository
Search
Pull requests
Issues
Marketplace
Explore
facebookresearch / fairseq
Unwatch 164
Unstar 2,362
Fork 375
Code
Issues 27
Pull requests 0
Projects 0
Wiki
Insights
Facebook AI Research Sequence-to-Sequence Toolkit
19 commits
1 branch
0 releases
8 contributors
BSD-3-Clause
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download