Convolutional Sequence to Sequence Learning

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

Katharina Korfhage

Heidelberg University

November 07, 2019

Outline

Introduction

What?

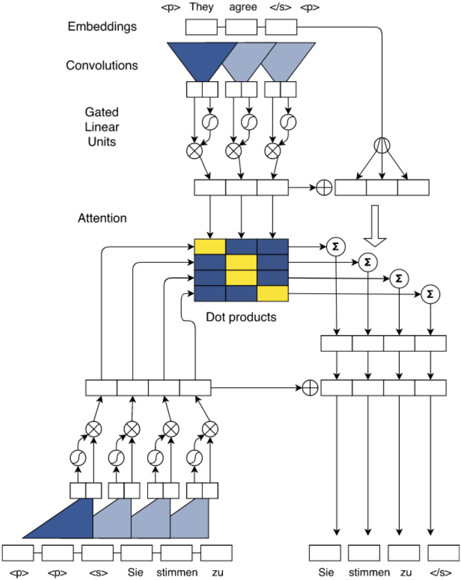
- The first fully convolutional model for sequence-to-sequence learning
  - gated linear units (GLUs)
  - residual connections
  - attention
- New state of the art results on several large benchmark datasets

Introduction

Why?

- Fixed size contexts, yet easily scalable through stacking of layers
  - → control of maximum length of dependencies to be modeled
- Hierarchical representations over input sequence
  - → shorter path to capturing long-range dependencies
- Fixed number of non-linearities
  - → easier optimization
- Batching, kernels independent of previous output
  - → parallelizable, faster learning

Overview


<p>
They
agree
</s>
<p>
Embeddings
Convolutions
Gated Linear Units
Attention
Dot products
<p>
<p>
<s>
Sie
stimmen
zu
Sie
stimmen
zu
</s>

Embeddings

input elements

$\mathbf{x} = (x_1, \ldots, x_m)$

↓

word embeddings

$\mathbf{w} = (w_1, \ldots, w_m)$

positions of input elements

↓

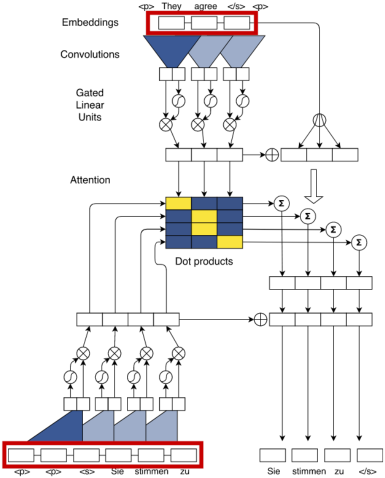
positional emb.

$\mathbf{p} = (p_1, \ldots, p_m)$

↘ ↙

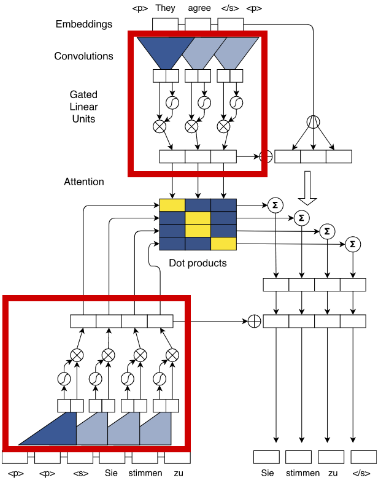
input element representations

$\mathbf{e} = (w_1 + p_1, \ldots w_m + p_m)$



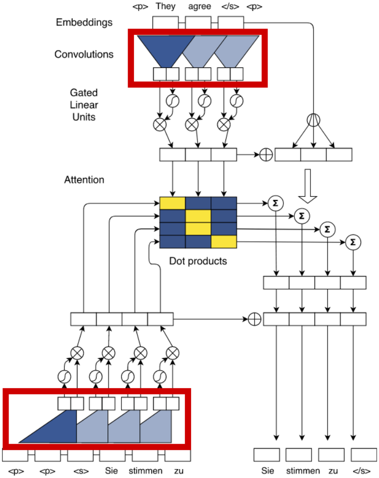
Convolutional Block Structure

- A.k.a. multi-layer network
- Block/layer:
  - one-dimensional convolution
  - non-linearity
- Kernel size $k \rightarrow$ fixed number of input elements
- Single block: output contains information about k input elements
- Stacking several blocks: increases the number of input elements represented in the state
- Output:
  - Encoder: $z^l = (z^l_1, ..., z^l_m)$
  - Decoder: $h^l = (h^l_1, ..., h^l_n)$



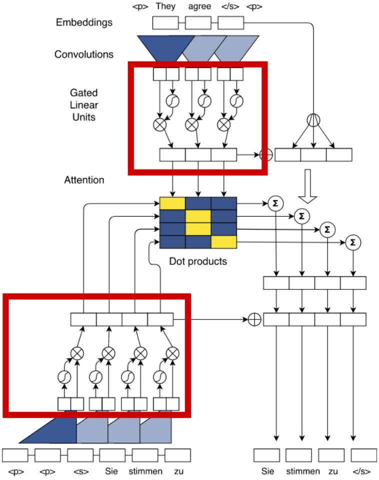
Convolutions

- Each convolution kernel of size k:
  - takes as input $X \in \mathbb{R}^{k \times d}$
  - multiplies the input with weights $W \in \mathbb{R}^{2d \times kd}$
  - adds a bias $b_w \in \mathbb{R}^{2d}$
  - maps the k input elements to a single output element $Y \in \mathbb{R}^{2d}$
- Subsequent blocks iterate over the k output elements of the previous block



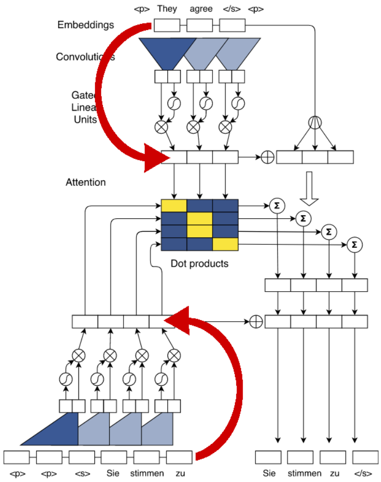
Non-Linearities/GLUs

- Take the output of the convolution as input: $Y = [A\ B] \in \mathbb{R}^{2d}$
- Gating mechanism: $v([A\ B]) = A \otimes \sigma(B)$
  - Input $[A\ B]$ is "split up" in two parts $A \in \mathbb{R}^d$ and $B \in \mathbb{R}^d$
  - Sigmoid activation function is applied to B
  - Pointwise multiplication of A and $\sigma(B)$
- Output $v([A\ B]) \in \mathbb{R}^d \rightarrow$ half the size of the input Y



Residual connections

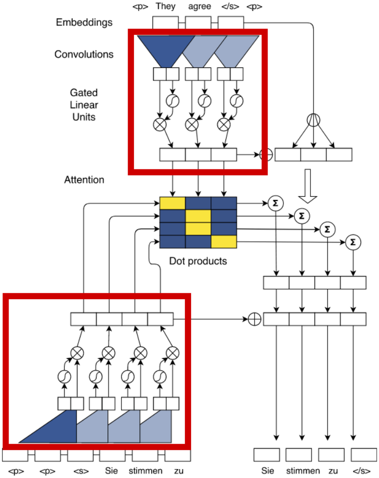
- input of the convolution added to the output of the block



Block Structure

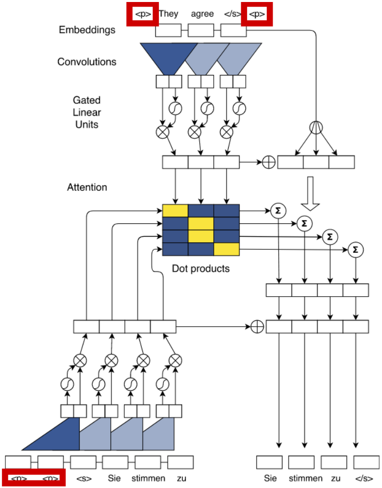
- Complete operation in one block:

$$\underbrace{\underbrace{h_i^l = v\Big(\underbrace{\underbrace{W^l}_{\text{weight}}\underbrace{[h^{l-1}_{i-k/2},\ldots,h^{l-1}_{i+k/2}]}_{\text{input}}+\underbrace{b^l_w}_{\text{bias}}}_{\text{convolution}}\Big)}_{\text{gating}} + h_i^l - 1}_{\text{residual connection}}$$



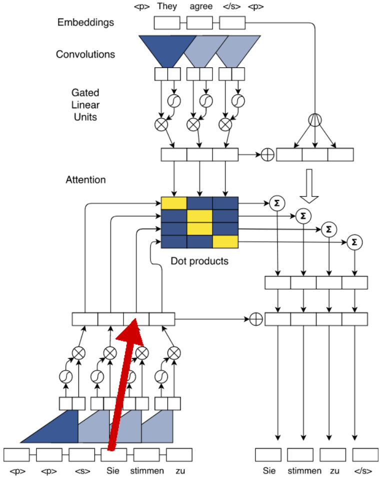
Padding

- Deep network: stacking several layers on top of each other
- Higher layer takes k output elements of lower layer as input
- Input size to higher layers should match the output size of lower layers $\rightarrow$ padding!
- Decoder: no future information must be available!
  - $\rightarrow$ Solution: Pad by *k-1* elements on both sides, then remove k elements from the ends



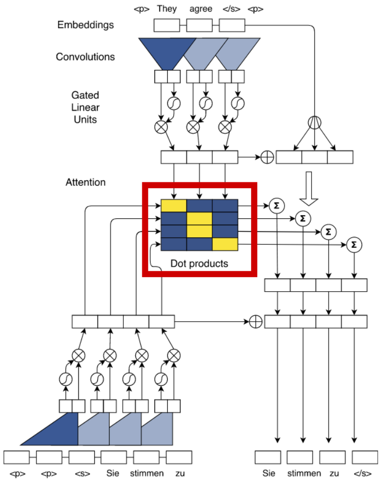
Multi-step Attention

- Separate attention mechanism for each decoder layer
- "Decoder state summary":
  - Combine current decoder state h_i^l with an embedding of the previous target element g_i
  - $d_i^l = W_d^l h_i^l + b_d^l + g_i$



Multi-step Attention

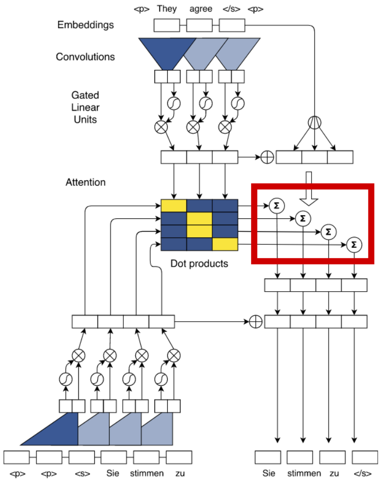
- Dot-product attention between:
  - Decoder state summary d_i^l
  - Output of last encoder block z_j^u
- $a_{ij}^l = \frac{exp(d_i^l \cdot z_j^u)}{\sum_{t=1}^m exp(d_i^l \cdot z_t^u)}$



Conditional decoder input

- Weighted sum of encoder outputs and input element embeddings e_j
- $c_i^l = \sum_j {}_1^m a_{ij}^l (z_j^u + e_j)$

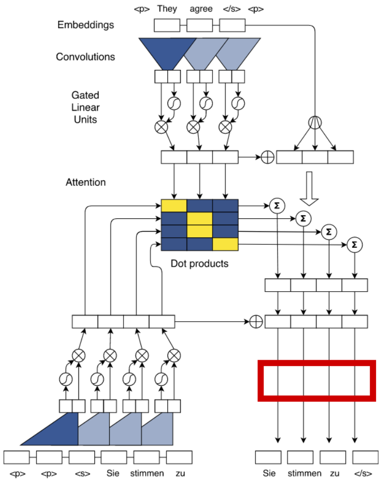
! Difference to recurrent approaches: calculate the weighted sum over z_j^u only



Predictions

- Probability distribution over the T possible next target elements y_{i+1}
- Transforming the top decoder output h_i^L with weights W_o and bias b_o:

$$p(y_{i+1}|y_1,\ldots,y_i,x) = softmax(W_o h_i^L + b_o)$$



Datasets

WMT'16 Eng-Rum	WMT'14 Eng-Ger	WMT'14 Eng-Fre	Abstractive summarization
2.8M sentence pairs	4.5M sentence pairs	35.5M sentence pairs	Gigaword: 3.8M training examples
Evaluation on newstest2016	Evaluation on newstest2014	Evaluation on newstest2014	Evaluation on DUC-2004 test data
200/80K word types	40K BPE types	40K BPE types	30K source/ target word vocabulary
40K BPE types		Max. 175 words/sentence	outputs at least 14 words
Max. 175 words/sentence		Max. source/ target length ratio 1.5	

Results

WMT'16 English-Romanian	BLEU
Sennrich et al. (2016b) GRU (BPE 90K)	28.1
ConvS2S (Word 80K)	29.45
ConvS2S (BPE 40K)	30.02

WMT'14 English-German	BLEU
Luong et al. (2015) LSTM (Word 50K)	20.9
Kalchbrenner et al. (2016) ByteNet (Char)	23.75
Wu et al. (2016) GNMT (Word 80K)	23.12
Wu et al. (2016) GNMT (Word pieces)	24.61
ConvS2S (BPE 40K)	25.16

WMT'14 English-French	BLEU
Wu et al. (2016) GNMT (Word 80K)	37.90
Wu et al. (2016) GNMT (Word pieces)	38.95
Wu et al. (2016) GNMT (Word pieces) + RL	39.92
ConvS2S (BPE 40K)	40.51

Results: Ensembling

WMT'14 English-German	BLEU
Wu et al. (2016) GNMT	26.20
Wu et al. (2016) GNMT + RL	26.30
ConvS2S	26.43

WMT'14 English-French	BLEU
Zhou et al. (2016)	40.4
Wu et al. (2016) GNMT	40.35
Wu et al. (2016) GNMT + RL	41.16
ConvS2S	41.44
ConvS2S (10 models)	41.62

Results: Generation speed

	BLEU	Time (s)
GNMT GPU (K80)	31.20	3,028
GNMT CPU 88 cores	31.20	1,322
GNMT TPU	31.21	384
ConvS2S GPU (K40) $b = 1$	33.45	327
ConvS2S GPU (M40) $b = 1$	33.45	221
ConvS2S GPU (GTX-1080ti) $b = 1$	33.45	142
ConvS2S CPU 48 cores $b = 1$	33.45	142
ConvS2S GPU (K40) $b = 5$	34.10	587
ConvS2S CPU 48 cores $b = 5$	34.10	482
ConvS2S GPU (M40) $b = 5$	34.10	406
ConvS2S GPU (GTX-1080ti) $b = 5$	34.10	256

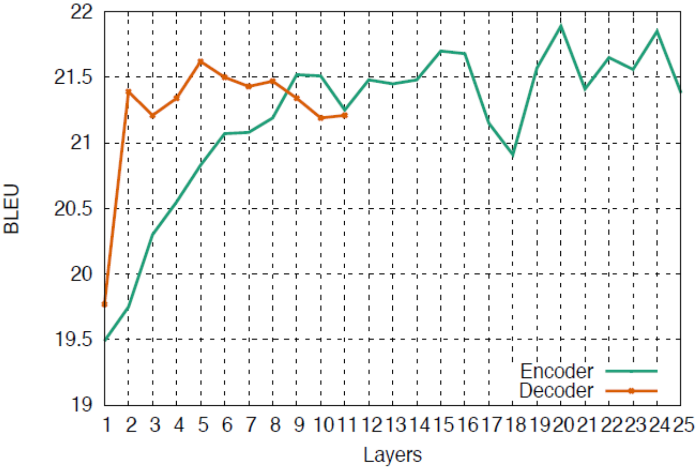
Results: Positional Embeddings

	PPL	BLEU
ConvS2S	6.64	21.7
-source position	6.69	21.3
-target position	6.63	21.5
-source & target position	6.68	21.2

Results: Attention

Attn Layers	PPL	BLEU
1,2,3,4,5	6.65	21.63
1,2,3,4	6.70	21.54
1,2,3	6.95	21.36
1,2	6.92	21.47
1,3,5	6.97	21.10
1	7.15	21.26
2	7.09	21.30
3	7.11	21.19
4	7.19	21.31
5	7.66	20.24

Results: Number of layers



Results: Kernel sizes

Kernel width	Encoder layers		
	5	9	13
3	20.61	21.17	21.63
5	20.80	21.02	21.42
7	20.81	21.30	21.09

Kernel width	Decoder layers		
	3	5	7
3	21.10	21.71	21.62
5	21.09	21.63	21.24
7	21.40	21.31	21.33

Results: Summarization

	DUC-2004			Gigaword		
	RG-1 (R)	**RG-2 (R)**	**RG-L (R)**	**RG-1 (F)**	**RG-2 (F)**	**RG-L (F)**
RNN MLE (Shen et al., 2016)	24.92	8.60	22.25	32.67	15.23	30.56
RNN MRT (Shen et al., 2016)	30.41	10.87	26.79	36.54	16.59	33.44
WFE (Suzuki & Nagata, 2017)	32.28	10.54	27.80	36.30	17.31	33.88
ConvS2S	30.44	10.84	26.90	35.88	17.48	33.29

Discussion

Thank you for your attention!

Any questions?

Discussion

- [Original question in German]
  The input to the kernel has a dimension of $k \times d$. Then how can the kernel itself have dimensions of $2d \times kd$? (Section 3.2)

- [Original question in German]
  "We compute a distribution over the T possible next target elements" (Section 3.2).
  - → Does this mean that for translations, for example, T elements are created "at once", and then the next T elements?
  - → How is the output created?

Discussion

- "We proceed similarly for output elements that were already generated by the decoder network to yield output element representations that are being fed back into the decoder network g= (g1,...,gn)." (Section 3.1)
  → Does it here mean that the outputs from lower blocks/layers and fed into the higher blocks/layers, or there are two different computation process with the same decoder network?
- According to the authors, removing the position features in both encoder and decoder doesn't affect the result a lot, and the model with convolutional block structure is able to capture the sequence information. (Section 5.4)
  → Why is this? Residual connections? Because every state output results from k continuous input elements?

Discussion

- I noticed that the whole structure of the model resembles the transformer, but with convolution doing the self-attention job. Are there even more similarities?

- Could we discuss where in the model the Gated Linear Units are exactly used? I am just not sure if I understood correctly what A and B is.
- Is there a special reason why the output kernel size is 2d, meaning that a size o d times k inputs is mapped to a single vector of 2d?

Discussion

- Judging from paragraph 3.1 to me it reads like these are simple one-hot vectors. However, later on the authors write "the models can learn relative position information" (Section 5.4). This suggest that they far more complex. [...]
  - → How exactly are those positional embeddings created or even learned?

- The authors find that encoders and to a smaller degree decoders work better with narrow kernels and many layers than with wider kernels (Section 5.6).
  - → Why do the encoders (and decoders) work better with these parameters?

Discussion

- What are the advantages of using GLUs in this work?
- Why are deeper architectures more beneficial for the encoder than for the decoder?

Discussion

- What's the justification for using the gated linear unit? Isn't it linear in terms of it's input $Av([AB]) = A \otimes \sigma(B)$ or am I missing something?
- This paper doesn't really seem to describe the structure of the decoder network, how does it work? Are the convolutions transposed?

Discussion

- Why are residual connections able to allow deep convolutional networks? (Section 3.2)
- "Multi-layer convolutional neural networks create hierarchical representations over the input sequence in which nearby input elements interact at lower layers while distant elements interact at higher layers." (Section 1)
  - → Are there approaches which exploit these hierarchical representations to improve the translation?
  - → Have these representations been used on other tasks?

Paper

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin.
Convolutional sequence to sequence learning, 2017.