



Natural Language Processing; Neural Machine Translation



[YouTube Playlist](#)

Maziar Raissi

Assistant Professor

Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu

Neural Machine Translation by Jointly Learning to Align and Translate


[YouTube Playlist](#)

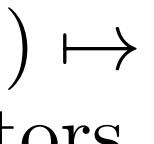
Translation

$$\arg \max_y p(y|x)$$

 source sentence
 target sentence

RNN Encoder-Decoder

$$x = (x_1, x_2, \dots, x_{T_x}) \mapsto c$$

 vector
 sequence of vectors
 $h_t = f(x_t, h_{t-1}) \rightarrow \text{LSTM}$
 $c = q(\{h_1, \dots, h_{T_x}\}) = h_{T_x}$
 $h_t \in \mathbb{R}^n \rightarrow \text{hidden state at time } t$
 $p(y) = \prod_{t=1}^{T_y} p(y_t | \underbrace{\{y_1, \dots, y_{t-1}\}}_{g(y_{t-1}, s_t, c)}, c)$
 $y = (y_1, y_2, \dots, y_{T_y})$
 $s_t \rightarrow \text{hidden state of the RNN}$

Learning to Align & Translate

$$p(y_t | \{y_1, \dots, y_{t-1}\}, x) = g(y_{t-1}, s_t, \textcolor{red}{c}_t)$$

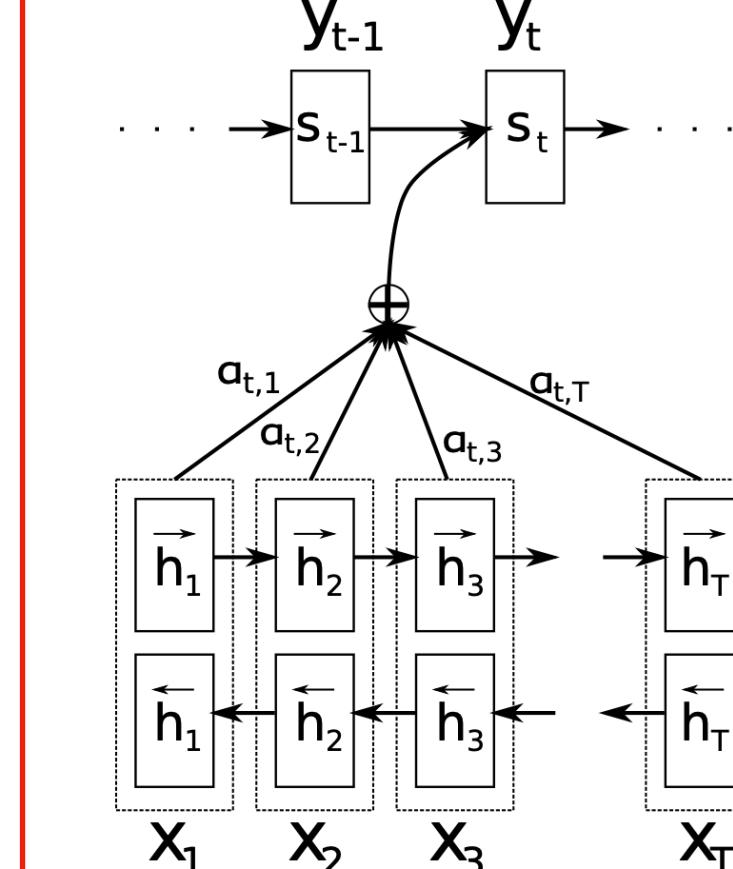
$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

$$c_t = \sum_{t'=1}^{T_x} \alpha_{tt'} h_{t'} \rightarrow \text{context vector}$$

$$\alpha_{tt'} = \frac{\exp(e_{tt'})}{\sum_{\tau=1}^{T_x} \exp(e_{t\tau})}$$

$$e_{tt'} = a(s_{t-1}, h_{t'}) \rightarrow \text{alignment score}$$

(attention mechanism)



$$s_i = f(s_{i-1}, y_{i-1}, c_i) = (1 - z_i) \circ s_{i-1} + z_i \circ \tilde{s}_i,$$

$$\tilde{s}_i = \tanh(W_e(y_{i-1}) + U[r_i \circ s_{i-1}] + Cc_i)$$

embedding

$$z_i = \sigma(W_z e(y_{i-1}) + U_z s_{i-1} + C_z c_i) \rightarrow \text{update gates}$$

$$r_i = \sigma(W_r e(y_{i-1}) + U_r s_{i-1} + C_r c_i) \rightarrow \text{reset gates}$$

encoder: ignore c_i , $s_i \leftrightarrow h_i$, $y_{i-1} \leftrightarrow x_i$

$$a(s_{i-1}, h_j) = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

$$p(y_i | s_i, y_{i-1}, c_i) \propto \exp(y_i^\top W_o t_i)$$

$$t_i = [\max \{\tilde{t}_{i,2j-1}, \tilde{t}_{i,2j}\}]_{j=1,\dots,l}^\top \rightarrow \text{maxout}$$

$$\tilde{t}_i = U_o s_{i-1} + V_o E y_{i-1} + C_o c_i$$

BLUE Score

<https://www.youtube.com/watch?v=DejHQYAGb7Q>

$$p_n = \frac{\sum_{C \in \{\text{Candidates}\}} \sum_{n\text{-gram} \in C} \text{Count}_{clip}(n\text{-gram})}{\sum_{C' \in \{\text{Candidates}\}} \sum_{n\text{-gram}' \in C'} \text{Count}(n\text{-gram}')}$$

modified n-gram precision score

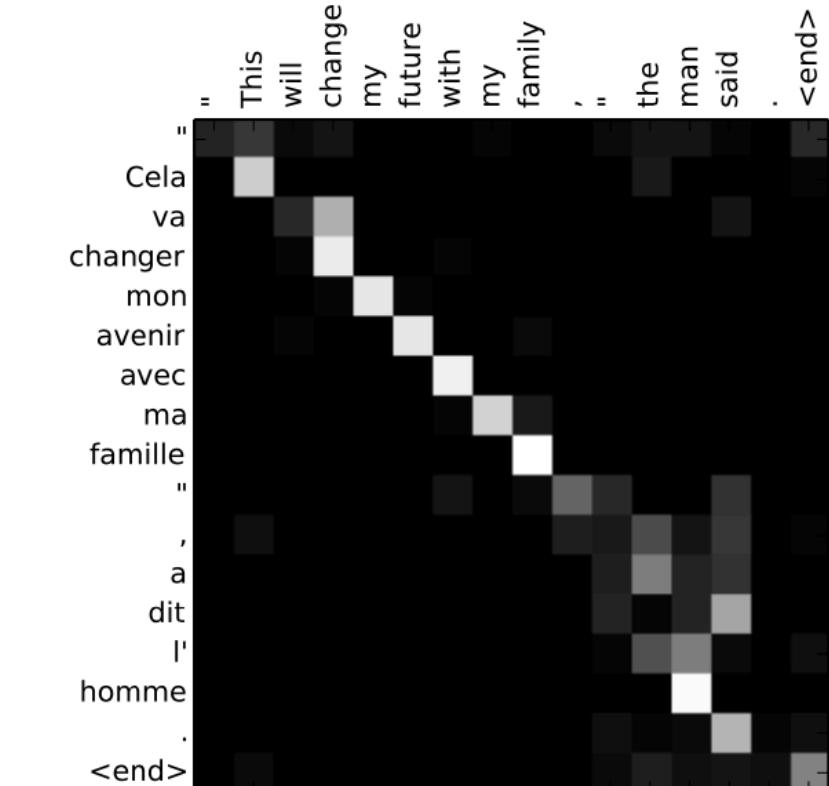
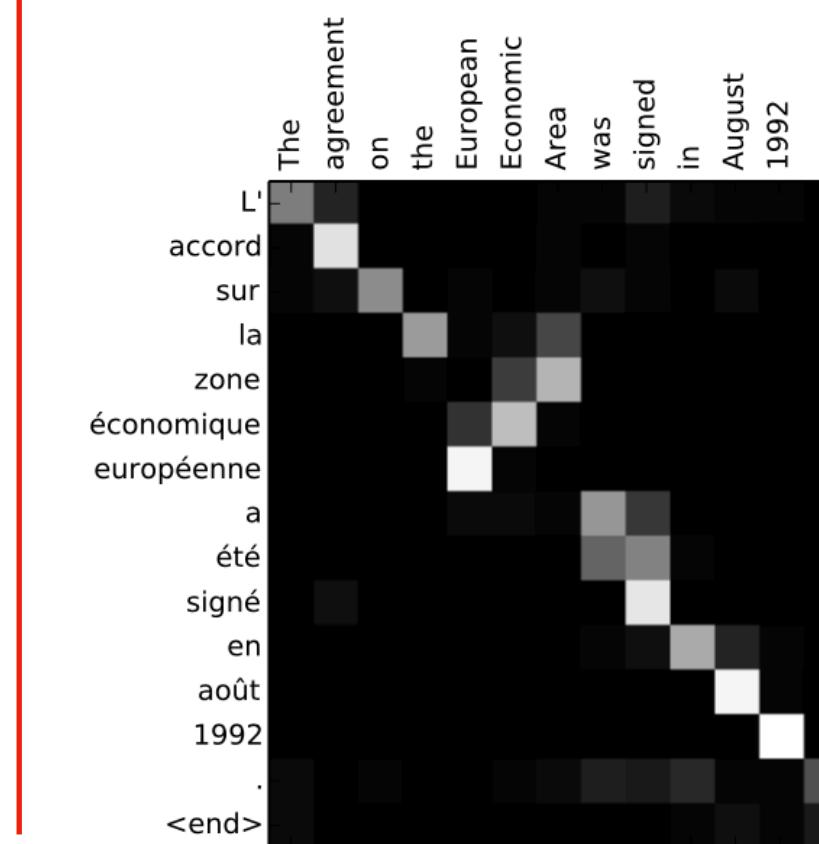
$$\text{BLEU} = \text{BP} \cdot \exp \left(\sum_{n=1}^N w_n \log p_n \right) \quad w_n = 1/N$$

brevity penalty $\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$

$$\log \text{BLEU} = \min(1 - \frac{r}{c}, 0) + \sum_{n=1}^N w_n \log p_n$$

$r \rightarrow$ reference translation length

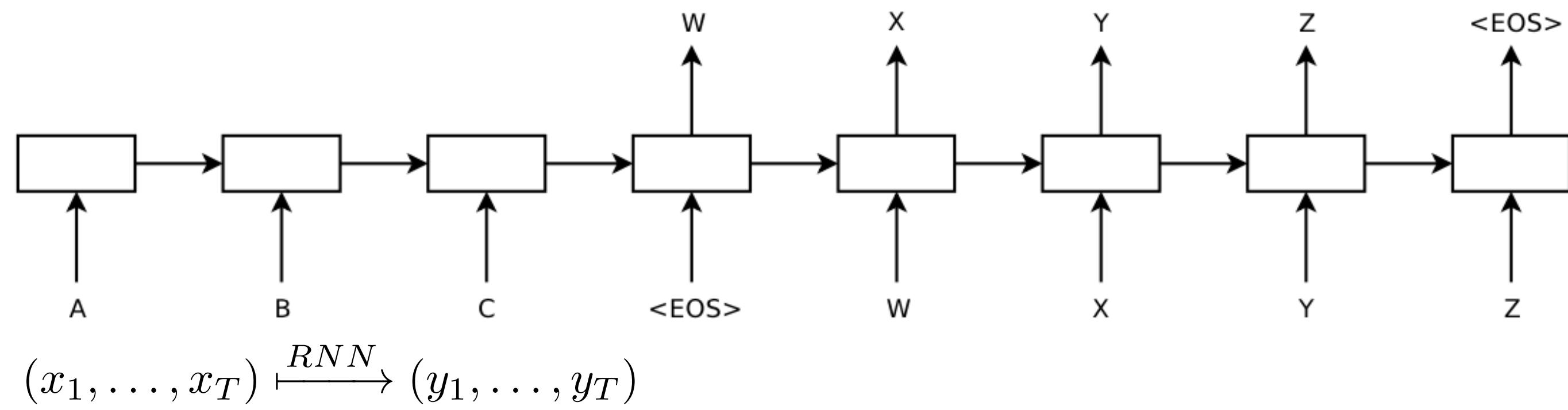
$c \rightarrow$ candidate translation length





Boulder

Sequence to Sequence Learning with Neural Networks


[YouTube Playlist](#)


$$h_t = \text{sigm}(W^{\text{hx}}x_t + W^{\text{hh}}h_{t-1})$$

$$y_t = W^{\text{yh}}h_t$$

$$T \neq T'$$

$$p(y_1, \dots, y_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} \underbrace{p(y_t | v, y_1, \dots, y_{t-1})}_{\text{softmax}}$$

1) Two different LSTMs: input & output sequences

2) Deep LSTMs

3) Reverse the order of the words of the input sentence
 $c, b, a \mapsto \alpha, \beta, \gamma$ rather than $a, b, c \mapsto \alpha, \beta, \gamma$

$$\frac{1}{|\mathcal{S}|} \sum_{(T, S) \in \mathcal{S}} \log p(T|S) \rightarrow \text{training objective}$$

<https://www.youtube.com/watch?v=RLWuzLLSIgw>

$$\hat{T} = \arg \max_T p(T|S) \rightarrow \text{beam search}$$

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	34.81

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
State of the art [9]	37.0
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	36.5
Oracle Rescoring of the Baseline 1000-best lists	~45

Type	Sentence
model	Ulrich UNK , membre du conseil d' administration du constructeur automobile Audi , affirme qu' il s' agit d' une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d' administration afin qu' ils ne soient pas utilisés comme appareils d' écoute à distance .
Truth	Ulrich Hackenberg , membre du conseil d' administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu' ils ne puissent pas être utilisés comme appareils d' écoute à distance , est une pratique courante depuis des années .

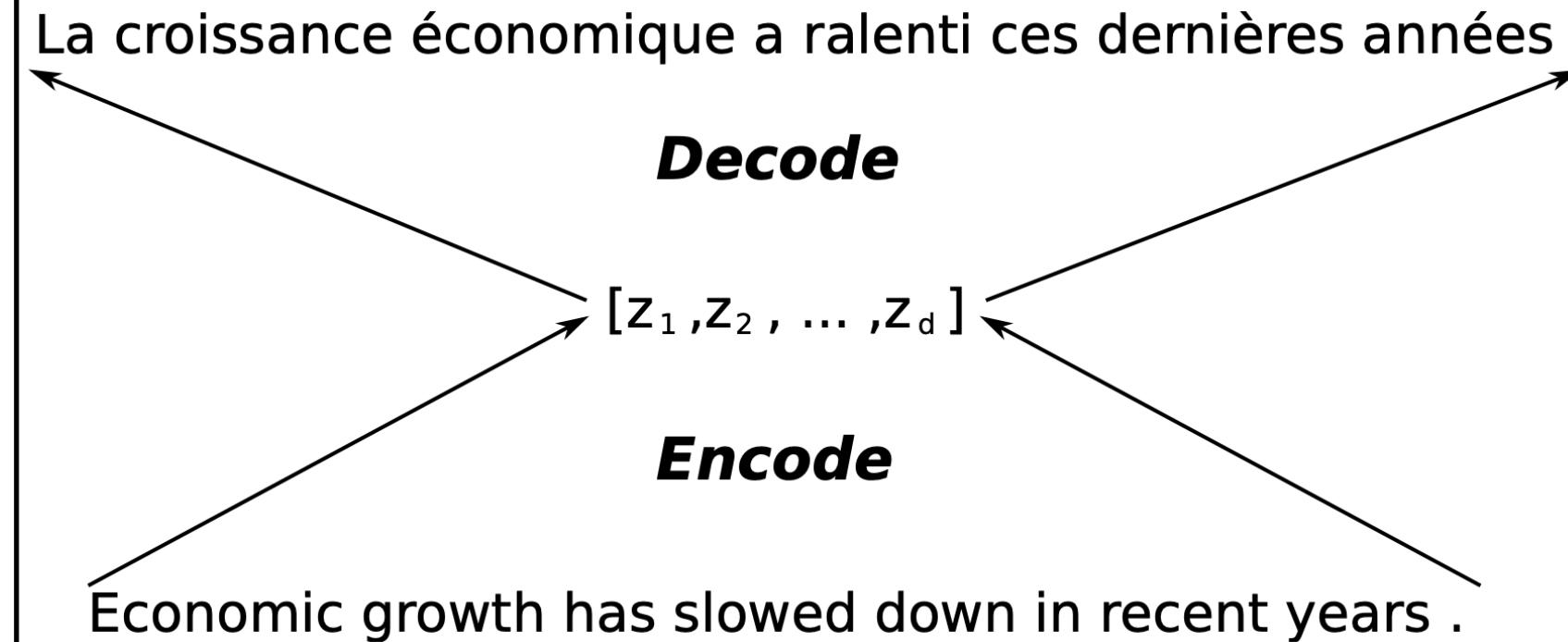


Boulder

On the Properties of Neural Machine Translation: Encoder–Decoder Approaches



[YouTube Playlist](#)



– sentence length – vocabulary size

RNN with Gated Hidden Neurons

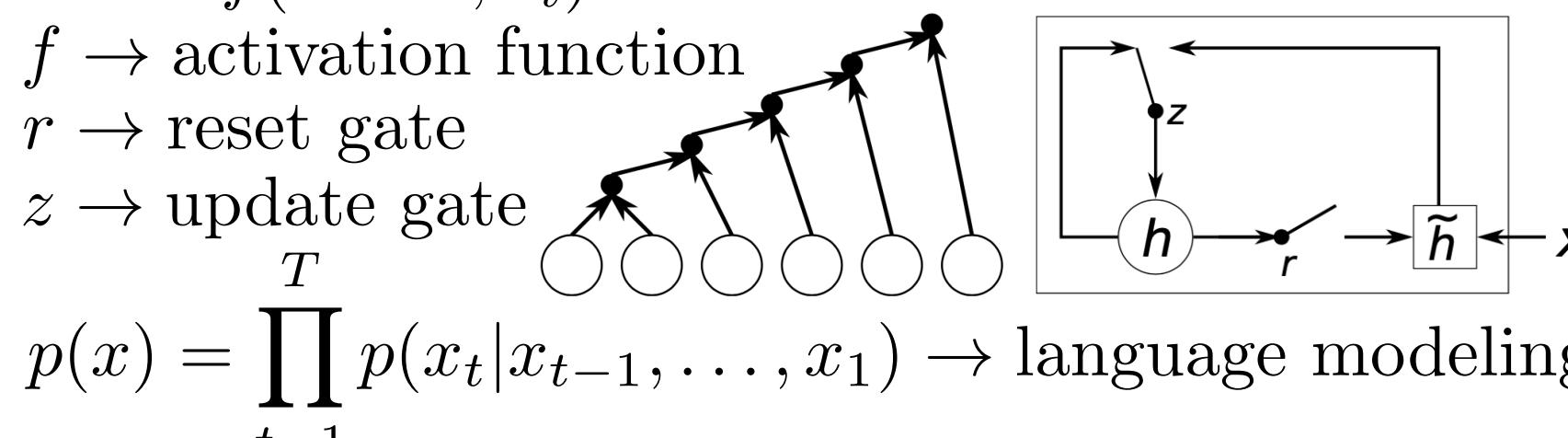
$x = (x_1, x_2, \dots, x_T) \rightarrow$ variable length sequence

$h^{(t)} = f(h^{(t-1)}, x_t) \rightarrow$ hidden state

$f \rightarrow$ activation function

$r \rightarrow$ reset gate

$z \rightarrow$ update gate



$p(x) = \prod_{t=1}^T p(x_t | x_{t-1}, \dots, x_1) \rightarrow$ language modeling

$$p(x_{t,j} = 1 | x_{t-1}, \dots, x_1) = \frac{\exp(w_j h_{<t>})}{\sum_{j'=1}^K \exp(w_{j'} h_{<t>})}$$

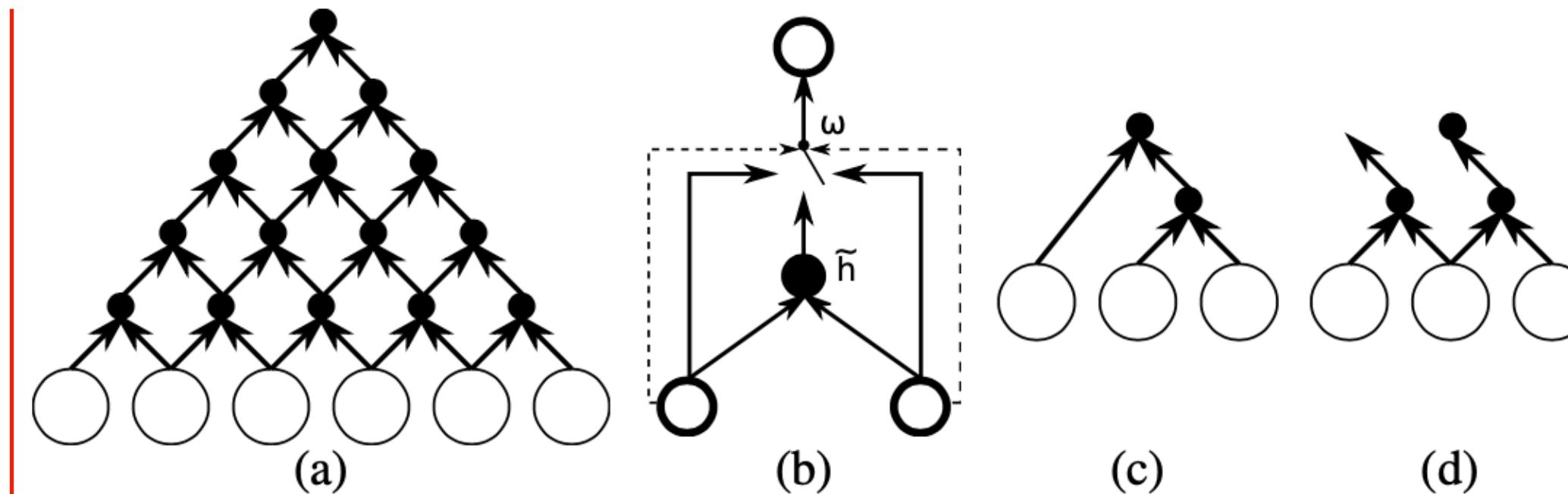
1-of- K encoding

Gated Recursive CNN

Parameters at each level are shared throughout the network

$x = (x_1, x_2, \dots, x_T) \rightarrow$ input sequence $x_j \in \mathbb{R}^d$

$h_j^{(t)} \rightarrow$ activation of the j -th hidden unit at recursion level $t \in [1, T - 1]$



$$h_j^{(t)} = w_c \tilde{h}_j^{(t)} + w_\ell h_{j-1}^{(t-1)} + w_r h_j^{(t-1)}$$

$$h_j^{(0)} = Ux_j \quad [w_c, w_\ell, w_r] = \text{softmax}(G^\ell h_{j-1}^{(t-1)} + G^r h_j^{(t-1)})$$

$$\tilde{h}_j^{(t)} = \phi(W^\ell h_{j-1}^{(t-1)} + W^r h_j^{(t-1)})$$

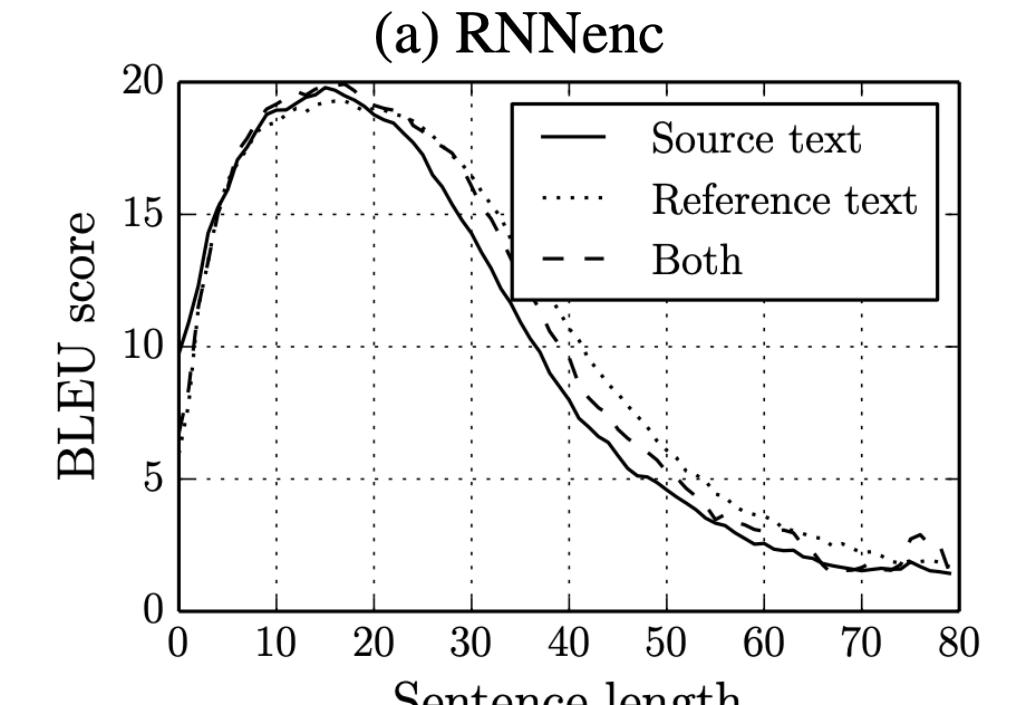
↳ element-wise nonlinearity

Source	And there are thorny mechanical questions that must be resolved during that time, like how to balance the state's mandate of "adequate access" to licensed marijuana with its prohibitions on cannabis businesses within 1,000 feet of a school, park, playground or child care center.
Reference	Pendant ce temps, des questions pratiques restent en suspens: comment équilibrer le mandat de l'état qui garantit un accès approprié à la marijuana agréée et interdit l'installation de commerces de vente de cannabis dans un rayon de 30 km autour d'une école, d'un parc, d'un terrain de jeu ou d'une crèche.
RNNEnc	Il y a des questions préventives qui se posent quant à l'équilibre des droits de l'enfant dans les limites d'une entreprise de collecte de sang.
grConv	De façon générale, il y a des raisons de sécurité pour que les entreprises aient accès à des milliers de centres de pêche, d'eau ou de recherche.
Moses	Et il y a des problèmes mécaniques complexes qui doivent être résolus au cours de cette période, comme la manière d'équilibrer le mandat de "l'accès adéquat" permis de marijuana avec l'interdiction du cannabis aux entreprises de 1000 pieds d'une école de jeu ou de parc, le service de garde.

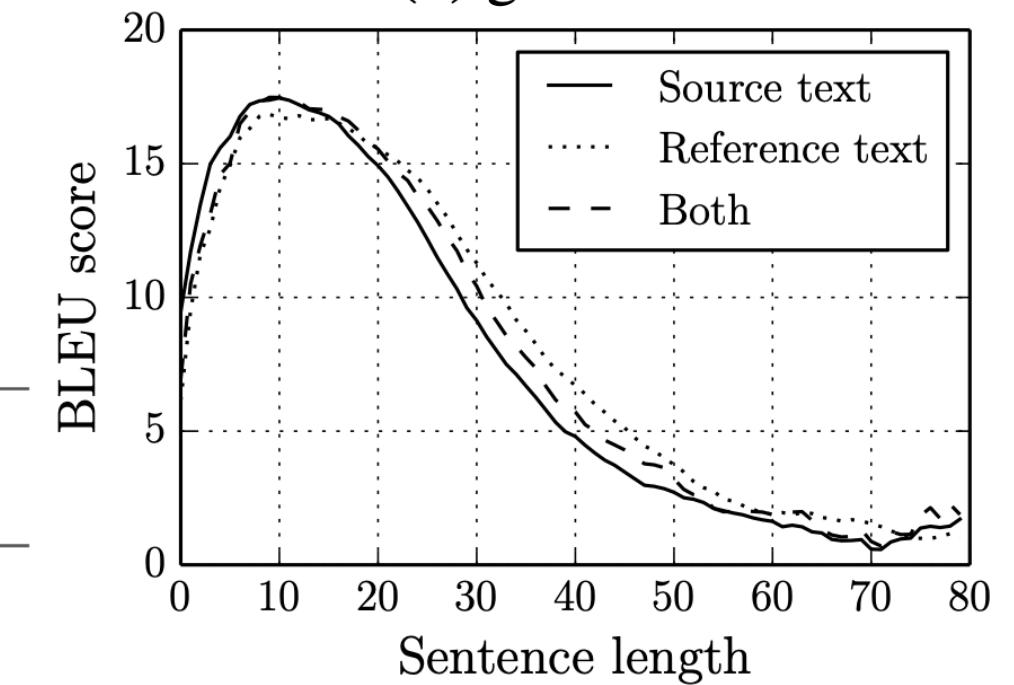
(a) Long Sentences

Source	According to them, one can find any weapon at a low price right now.
Reference	Selon eux, on peut trouver aujourd'hui à Moscou n'importe quelle arme pour un prix raisonnable.
RNNEnc	Selon eux, on peut se trouver de l'arme à un prix trop bas.
grConv	En tout cas, ils peuvent trouver une arme à un prix très bas à la fois.
Moses	Selon eux, on trouve une arme à bas prix pour l'instant.

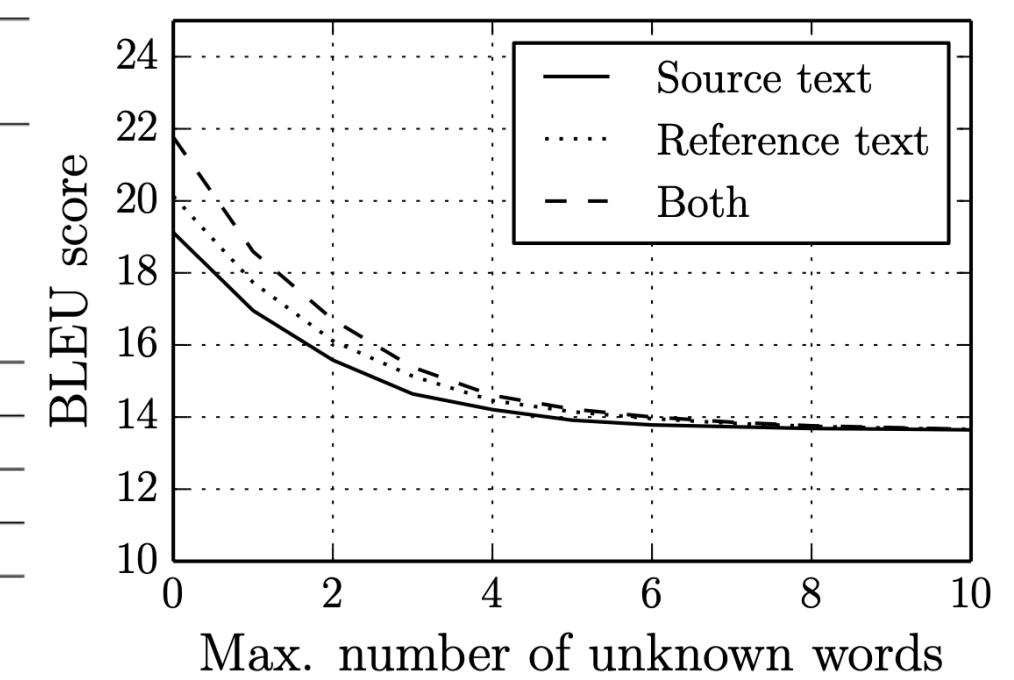
(b) Short Sentences



(b) grConv



(c) RNNenc



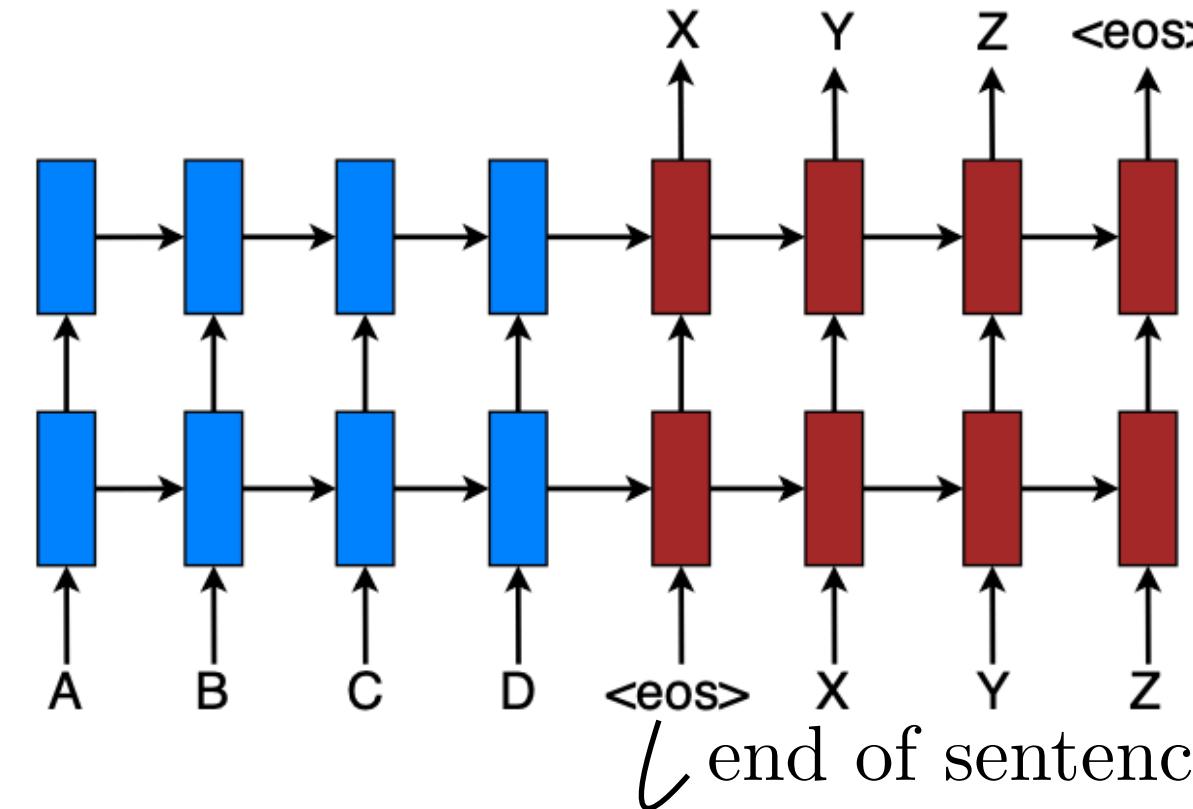


Boulder

Effective Approaches to Attention-based Neural Machine Translation



[YouTube Playlist](#)



$$p(y|x)$$

$x = (x_1, \dots, x_n) \rightarrow$ source sentence

$y = (y_1, \dots, y_m) \rightarrow$ target sentence

$$\log p(y|x) = \sum_{j=1}^m \log p(y_j|y_{<j}, s)$$

$s \rightarrow$ source sentence representation

$$p(y_j|y_{<j}, s) = \text{softmax}(g(\mathbf{h}_j))$$

vocabulary sized vector

$$\mathbf{h}_j = f(\mathbf{h}_{j-1}, s) \rightarrow \text{RNN hidden unit}$$

RNN, GRU, or LSTM

$$J_t = \sum_{(x,y) \in \mathbb{D}} -\log p(y|x)$$

training corpus

Attention-based Models

$$p(y_t|y_{<t}, x) = \text{softmax}(\mathbf{W}_s \tilde{\mathbf{h}}_t)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t])$$

$\mathbf{h}_t \rightarrow$ target hidden state

$\mathbf{c}_t \rightarrow$ source-side context vector

Global Attention

$$\mathbf{a}_t(s) = \text{align}(\mathbf{h}_t, \bar{\mathbf{h}}_s)$$

$$= \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'} \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_{s'}))}$$

$\bar{\mathbf{h}}_s \rightarrow$ source hidden state

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t^\top \bar{\mathbf{h}}_s & \text{dot} \\ \mathbf{h}_t^\top \mathbf{W}_a \bar{\mathbf{h}}_s & \text{general} \\ \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{h}_t; \bar{\mathbf{h}}_s]) & \text{concat} \end{cases}$$

content-based function

$$\mathbf{a}_t = \text{softmax}(\mathbf{W}_a \mathbf{h}_t) \rightarrow \text{location-based function}$$

Local Attention

$$p_t \rightarrow \text{aligned position } [p_t - D, p_t + D] \quad \mathbf{a}_t \in \mathbb{R}^{2D+1}$$

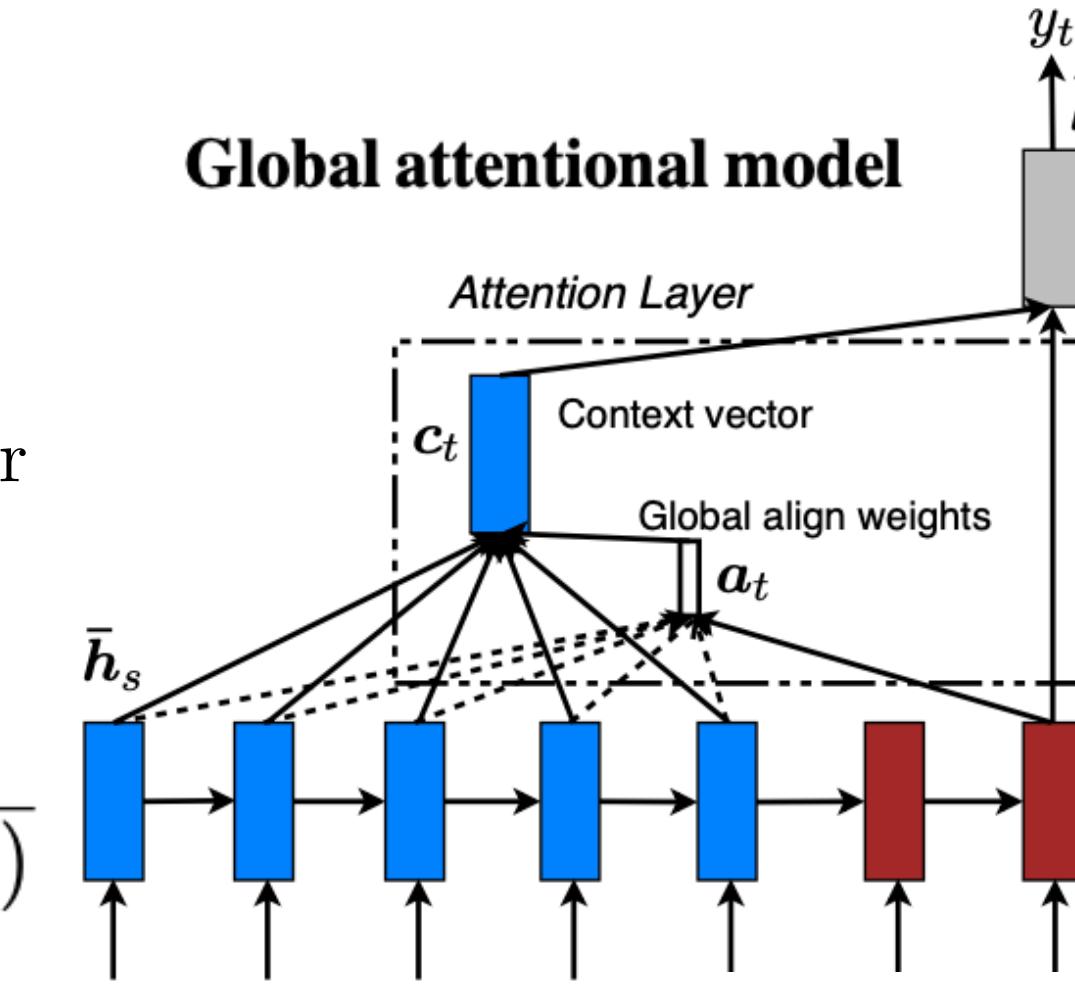
$p_t = t \rightarrow$ Monotonic alignment (local-m)

Predictive alignment (local-p)

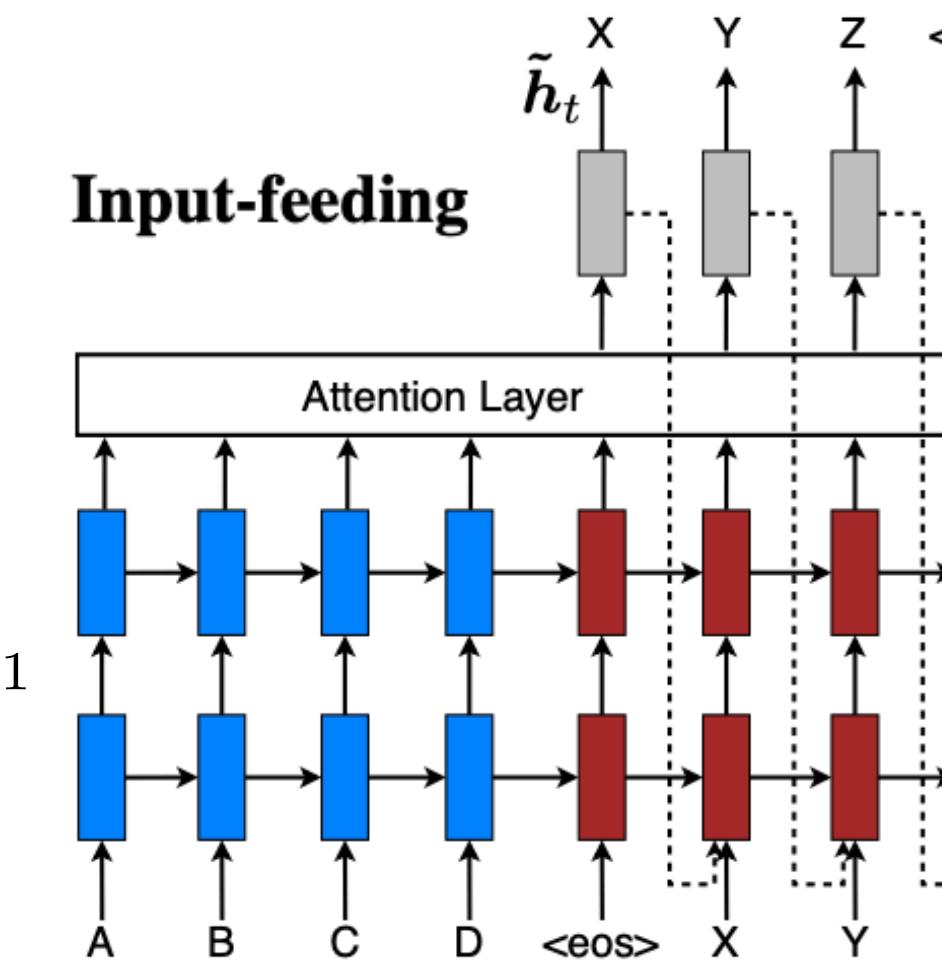
$$p_t = S \cdot \text{sigmoid}(\mathbf{v}_p^\top \tanh(\mathbf{W}_p \mathbf{h}_t))$$

$$\mathbf{a}_t(s) = \text{align}(\mathbf{h}_t, \bar{\mathbf{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right) \quad \sigma = \frac{D}{2}$$

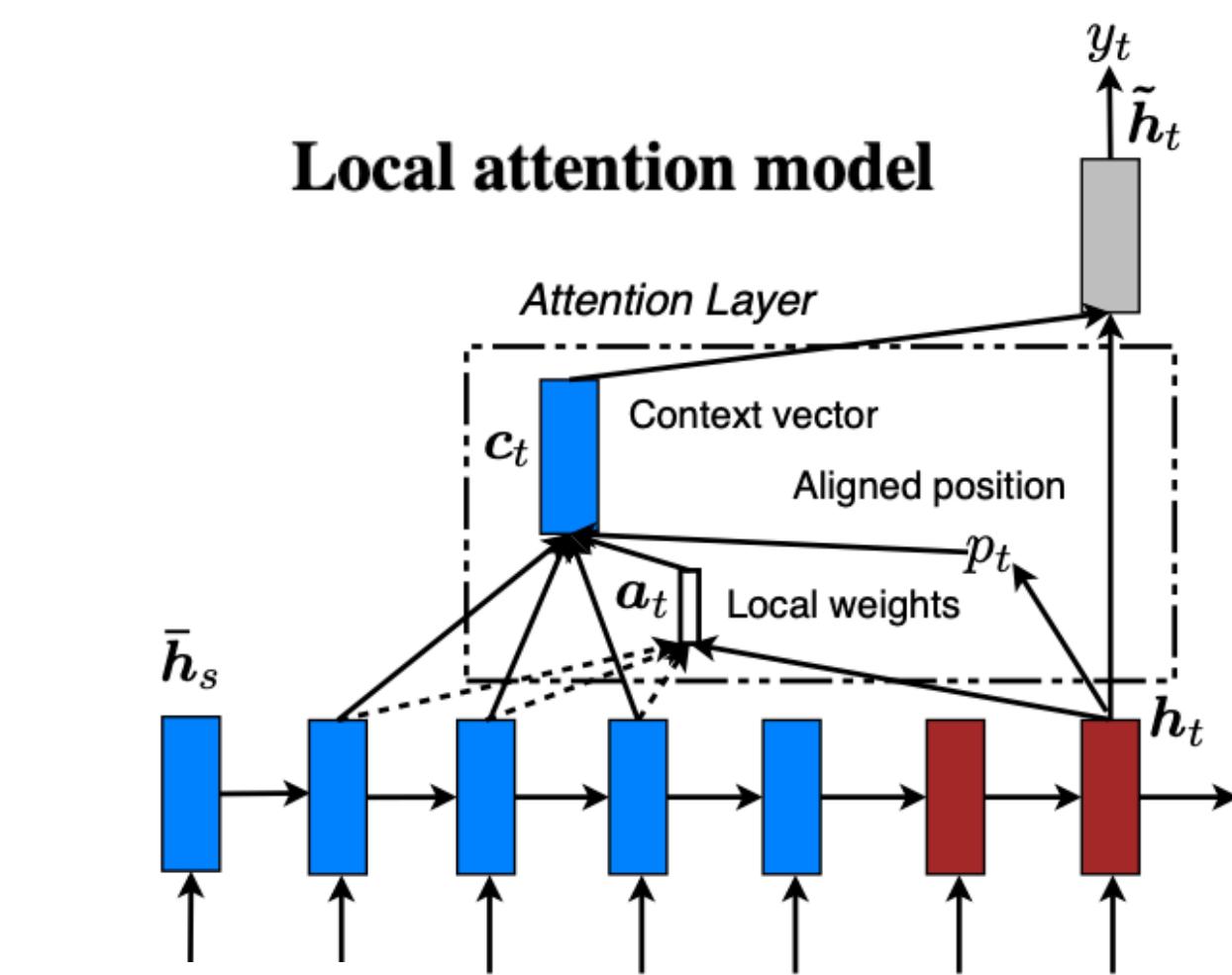
Global attentional model



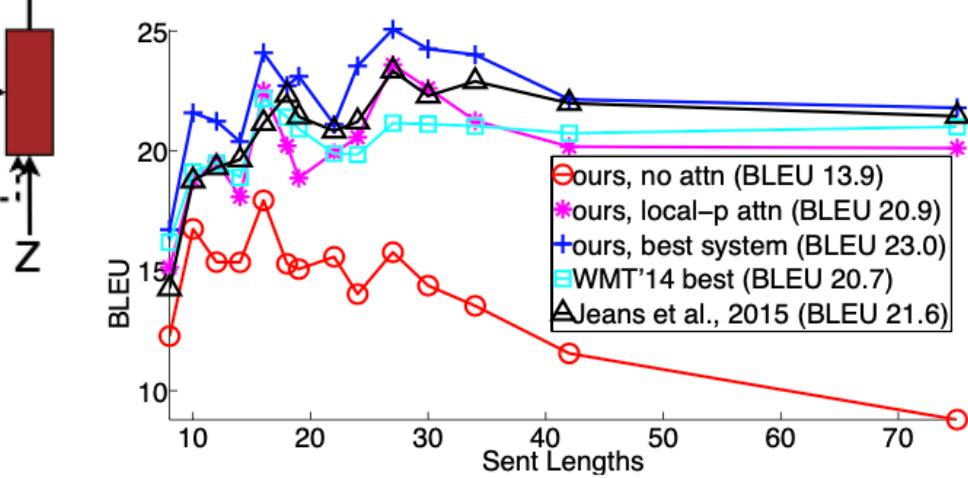
Input-feeding



Local attention model



System	Ppl	BLEU	
		Before	After unk
global (location)	6.4	18.1	19.3 (+1.2)
global (dot)	6.1	18.6	20.5 (+1.9)
global (general)	6.1	17.3	19.1 (+1.8)
local-m (dot)	>7.0	x	x
local-m (general)	6.2	18.6	20.4 (+1.8)
local-p (dot)	6.6	18.0	19.6 (+1.9)
local-p (general)	5.9	19	20.9 (+1.9)





Boulder

Neural Machine Translation Of Rare Words With Subword Units

[YouTube Playlist](#)

Byte Pair Encoding (BPE)

A simple data compression technique that iteratively replaces the most frequent pair of bytes in a sequence with
Byte: Character or character sequences a single, unused byte.

Byte Pair Encoding (BPE) merge operations learned from dictionary {'low', 'lower', 'newest', 'widest'}

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
    pairs = collections.defaultdict(int)
    for word, freq in vocab.items():
        symbols = word.split()
        for i in range(len(symbols)-1):
            pairs[symbols[i],symbols[i+1]] += freq
    return pairs

def merge_vocab(pair, v_in):
    v_out = {}
    bigram = re.escape(' '.join(pair))
    p = re.compile(r'(?!\S)' + bigram + r'(?!\S)')
    for word in v_in:
        w_out = p.sub(''.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out

vocab = {'l o w </w>': 5, 'l o w e r </w>': 2,
         'n e w e s t </w>': 6, 'w i d e s t </w>': 3}
num_merges = 10
for i in range(num_merges):
    pairs = get_stats(vocab)
    best = max(pairs, key=pairs.get)
    vocab = merge_vocab(best, vocab)
    print(best)
```

```
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}

('e', 's')
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}

('es', 't')
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}

('est', '</w>')
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}

('l', 'o')
{'lo w </w>': 5, 'lo w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}

('lo', 'w')
{'low </w>': 5, 'low e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}

('n', 'e')
{'low </w>': 5, 'low e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}

('ne', 'w')
{'low </w>': 5, 'low e r </w>': 2, 'new est </w>': 6, 'w i d e s t </w>': 3}

('new', 'est</w>')
{'low </w>': 5, 'low e r </w>': 2, 'newest </w>': 6, 'w i d e s t </w>': 3}

('low', '</w>')
{'low</w>': 5, 'low e r </w>': 2, 'newest </w>': 6, 'w i d e s t </w>': 3}

('w', 'i')
{'low</w>': 5, 'low e r </w>': 2, 'newest </w>': 6, 'wi d est </w>': 3}
```

At test time, the out-of-vocabulary (OOV) word 'lowest' would be segmented into 'low est.'

Corpus statistics for German

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
character bigrams	306 m	20 000	34
character trigrams	214 m	120 000	59
compound splitting [△]	102 m	1 100 000	643
morfessor*	109 m	544 000	237
hyphenation [◊]	186 m	404 000	230
BPE	112 m	63 000	0
BPE (joint)	111 m	82 000	32
character bigrams (shortlist: 50 000)	129 m	69 000	34

sequence length (# tokens) & vocabulary size (# types)

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
WDict	Forschungsinstitute
C2-50k	Förlschungslitutioen
BPE-60k	Gesundheitsforschungsinstituten
BPE-J90k	Gesundheitsforschungsinstitute
source	asinine situation
reference	dumme Situation
WDict	asinine situation → UNK → asinine
C2-50k	aslinline situation → Aslinensituation
BPE-60k	aslinline situation → Alineline-Situation
BPE-J90K	aslinline situation → Aslinin-Situation

system	sentence
source	Mirzayeva
reference	Мирзаева (Mirzaeva)
WDict	Mirzayeva → UNK → Mirzayeva
C2-50k	Mirzayeva → Ми рз а е ва (Mirz ae va)
BPE-60k	Mirzayeva → Мир за ева (Mir zaleva)
BPE-J90k	Mirzayeva → Мир за ева (Mirzaleva)
source	rakfisk
reference	ракфиска (rakfiska)
WDict	rakfisk → UNK → rakfisk
C2-50k	rakfislk → па кф ис к (rakflisk)
BPE-60k	rakfisk → пра ф иск (prafisk)
BPE-J90k	rakfisk → рак ф иска (rakfiska)



Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation



[YouTube Playlist](#)

$(X, Y) \rightarrow$ source and target sentence pair

$$X = x_1, x_2, x_3, \dots, x_M$$

$$Y = y_1, y_2, y_3, \dots, y_N$$

$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M = EncoderRNN(x_1, x_2, x_3, \dots, x_M)$$

$$P(Y|X) = P(Y|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_M)$$

$$= \prod_{i=1}^N P(y_i | y_0, y_1, y_2, \dots, y_{i-1}; \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_M)$$

$$P(y_i | y_0, y_1, y_2, y_3, \dots, y_{i-1}; \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_M)$$

$$s_t = AttentionFunction(\underbrace{\mathbf{y}_{i-1}, \mathbf{x}_t}_{\text{decoder-RNN output}}) \quad \forall t, \quad 1 \leq t \leq M$$

$$p_t = \exp(s_t) / \sum_{t=1}^M \exp(s_t) \quad \forall t, \quad 1 \leq t \leq M$$

$$\mathbf{a}_i = \sum_{t=1}^M p_t \cdot \mathbf{x}_t \rightarrow \text{attention context}$$

Residual Connections

$$\mathbf{c}_t^i, \mathbf{m}_t^i = LSTM_i(\mathbf{c}_{t-1}^i, \mathbf{m}_{t-1}^i, \mathbf{x}_t^{i-1}; \mathbf{W}^i)$$

$$\mathbf{x}_t^i = \mathbf{m}_t^i + \mathbf{x}_t^{i-1}$$

$$\mathbf{c}_t^{i+1}, \mathbf{m}_t^{i+1} = LSTM_{i+1}(\mathbf{c}_{t-1}^{i+1}, \mathbf{m}_{t-1}^{i+1}, \mathbf{x}_t^i; \mathbf{W}^{i+1})$$

\mathbf{m}_t^i and \mathbf{c}_t^i are the hidden states and memory states
8 LSTM layers Bi-directional Encoder for First Layer

Wordpiece Model

Word: Jet makers feud over seat width with big orders at stake

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

$$\mathcal{D} \equiv \{(X^{(i)}, Y^{*(i)})\}_{i=1}^N$$

$$\mathcal{O}_{ML}(\boldsymbol{\theta}) = \sum_{i=1}^N \log P_{\theta}(Y^{*(i)} | X^{(i)})$$

$$\mathcal{O}_{Mixed}(\boldsymbol{\theta}) = \alpha * \mathcal{O}_{ML}(\boldsymbol{\theta}) + \mathcal{O}_{RL}(\boldsymbol{\theta})$$

Quantizable Model and Quantized Inference

$$\mathbf{c}'_t^i, \mathbf{m}_t^i = LSTM_i(\mathbf{c}_{t-1}^i, \mathbf{m}_{t-1}^i, \mathbf{x}_t^{i-1}; \mathbf{W}^i)$$

$$\mathbf{c}_t^i = \max(-\delta, \min(\delta, \mathbf{c}'_t^i)) \in [-\delta, \delta]$$

$$\mathbf{x}'_t^i = \mathbf{m}_t^i + \mathbf{x}_t^{i-1}$$

$$\mathbf{x}_t^i = \max(-\delta, \min(\delta, \mathbf{x}'_t^i)) \in [-\delta, \delta]$$

$$\mathbf{c}'_{t+1}^{i+1}, \mathbf{m}_t^{i+1} = LSTM_{i+1}(\mathbf{c}_{t-1}^{i+1}, \mathbf{m}_{t-1}^{i+1}, \mathbf{x}_t^i; \mathbf{W}^{i+1})$$

$$\mathbf{c}_t^{i+1} = \max(-\delta, \min(\delta, \mathbf{c}'_{t+1}^{i+1})) \in [-\delta, \delta]$$

$$\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5, \mathbf{W}_6, \mathbf{W}_7, \mathbf{W}_8]$$

$$\mathbf{i}_t = \text{sigmoid}(\mathbf{W}_1 \mathbf{x}_t + \mathbf{W}_2 \mathbf{m}_t)$$

$$\mathbf{i}'_t = \tanh(\mathbf{W}_3 \mathbf{x}_t + \mathbf{W}_4 \mathbf{m}_t)$$

$$\mathbf{f}_t = \text{sigmoid}(\mathbf{W}_5 \mathbf{x}_t + \mathbf{W}_6 \mathbf{m}_t)$$

$$\mathbf{o}_t = \text{sigmoid}(\mathbf{W}_7 \mathbf{x}_t + \mathbf{W}_8 \mathbf{m}_t)$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{i}'_t \odot \mathbf{i}_t$$

$$\mathbf{m}_t = \mathbf{c}_t \odot \mathbf{o}_t$$

$$\mathcal{O}_{RL}(\boldsymbol{\theta}) = \sum_{i=1}^N \sum_{Y \in \mathcal{Y}} P_{\theta}(Y | X^{(i)}) r(Y, Y^{*(i)})$$

GLEU score
(see paper)

replace all the floating point operations
with fixed-point integer operations
with either 8-bit or 16-bit resolution

weight matrix \mathbf{W} is represented
using an 8-bit integer matrix \mathbf{WQ}
and a float vector \mathbf{s}

$$\mathbf{s}_i = \max(\text{abs}(\mathbf{W}[i, :]))$$

$$\mathbf{WQ}[i, j] = \text{round}(\mathbf{W}[i, j] / \mathbf{s}_i \times 127.0)$$

\mathbf{c}_t^i and \mathbf{x}_t^i are represented using 16-bit integers
matrix multiplications

are done using 8-bit integer multiplication

All other operations (sigmoid, tanh) and $(\odot, +)$
are done using 16-bit integer operations

$$s(Y, X) = \log(P(Y|X)) / lp(Y) + cp(X; Y)$$

$$lp(Y) = \frac{(5 + |Y|)^{\alpha}}{(5 + 1)^{\alpha}}$$

$$cp(X; Y) = \beta * \sum_{i=1}^{|X|} \log(\min(\sum_{j=1}^{|Y|} p_{i,j}, 1.0)),$$

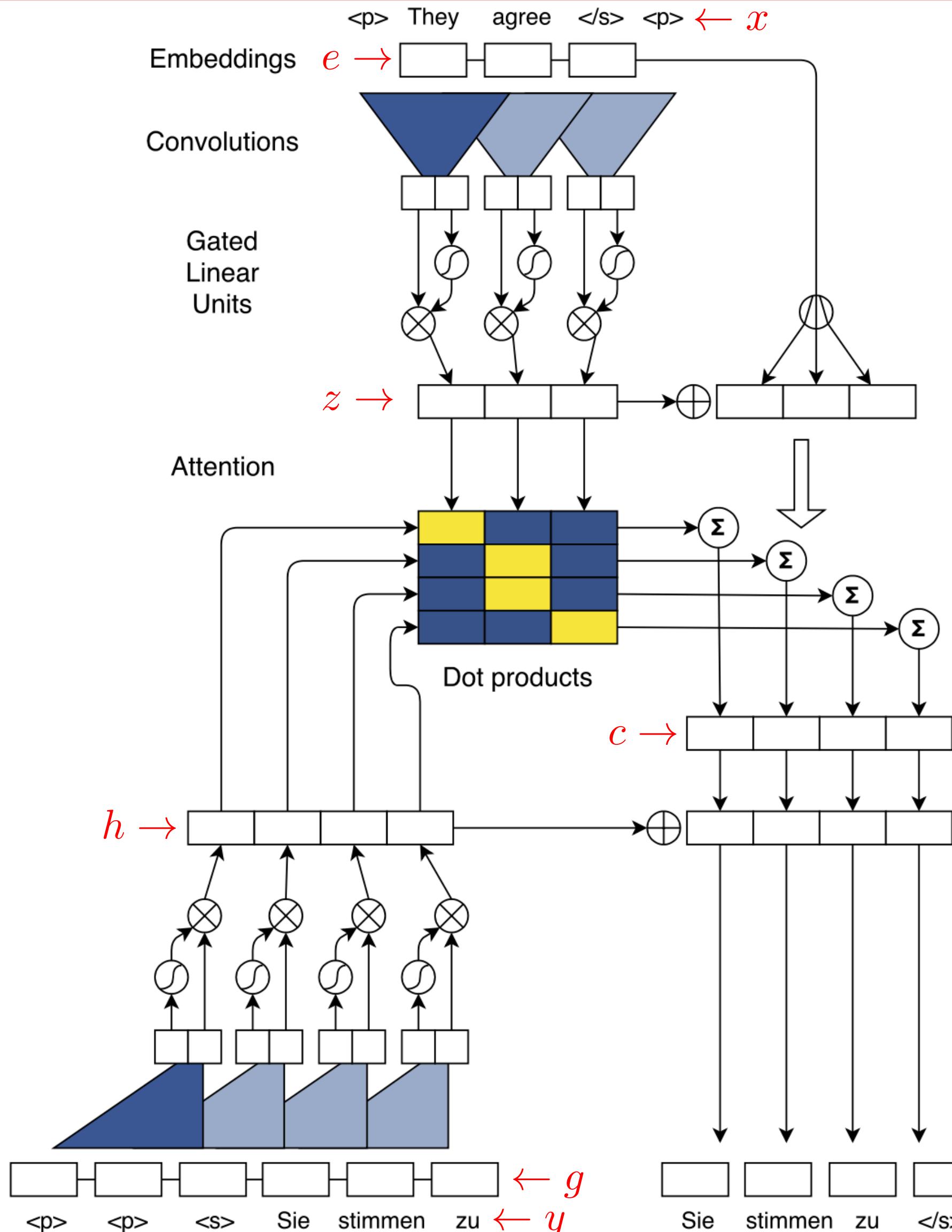
beam search



Boulder



Convolutional Sequence to Sequence Learning

[YouTube Playlist](#)


Position Embeddings

 $x = (x_1, \dots, x_m) \rightarrow$ input elements

 $w = (w_1, \dots, w_m) \rightarrow$ embedding of x in distributional space

 $w_j \in \mathbb{R}^f \rightarrow$ a column in an embedding matrix $\mathcal{D} \in \mathbb{R}^{f \times V}$
 $p = (p_1, \dots, p_m) \rightarrow$ position embedding (sense of order)

 $p_j \in \mathbb{R}^f$
 $e = (e_1, \dots, e_m) \rightarrow$ input element representation

 $e_j = w_j + p_j$
 $g = (g_1, \dots, g_n) \rightarrow$ output element representation (similarly)
(fed back into the decoder network)

 $h^l = (h_1^l, \dots, h_n^l) \rightarrow$ output of the l -th block for the decoder network

 $z^l = (z_1^l, \dots, z_m^l) \rightarrow$ output of the l -th block for the encoder network

Block: One dimensional convolution followed by a non-linearity

 $X \in \mathbb{R}^{k \times d} \rightarrow$ input to the convolution

 \hookrightarrow concatenation of k input elements embedded in d dimensions

 $X \in \mathbb{R}^{kd} \rightarrow$ flatten

 $W \in \mathbb{R}^{2d \times kd}, b_w \in \mathbb{R}^{2d} \rightarrow$ parameters of the convolutional kernel

 $Y = WX + b_w \in \mathbb{R}^{2d} \rightarrow$ convolution

 $Y = [A, B] \in \mathbb{R}^{2d}, A, B \in \mathbb{R}^d$
 $v([A, B]) = A \odot \sigma(B) \in \mathbb{R}^d \rightarrow$ GLU (Gated Linear Units)

 $h_i^l = v(W^l[h_{i-k/2}^{l-1}, \dots, h_{i+k/2}^{l-1}] + b_w^l) + h_i^{l-1} \rightarrow$ residual connections

 $p(y_{i+1}|y_1, \dots, y_i, x) = \text{softmax}(W_o h_i^L + b_o) \in \mathbb{R}^T$

Multi-step Attention

 $h_i^l \rightarrow$ current decoder state

 $g_i \rightarrow$ embedding of the previous target element

$$d_i^l = W_d^l h_i^l + b_d^l + g_i$$

 \hookrightarrow decoder state summary

 $a_{ij}^l \rightarrow$ attention of state i and source element j for decoder layer l
 $z_j^u \rightarrow$ output of the the last encoder block u

$$a_{ij}^l = \frac{\exp(d_i^l \cdot z_j^u)}{\sum_{t=1}^m \exp(d_i^l \cdot z_t^u)}$$

$$c_i^l = \sum_{j=1}^m a_{ij}^l (z_j^u + e_j)$$

 \hookrightarrow conditional input to the current decoder layer

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

	BLEU
WMT'14 English-German	
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

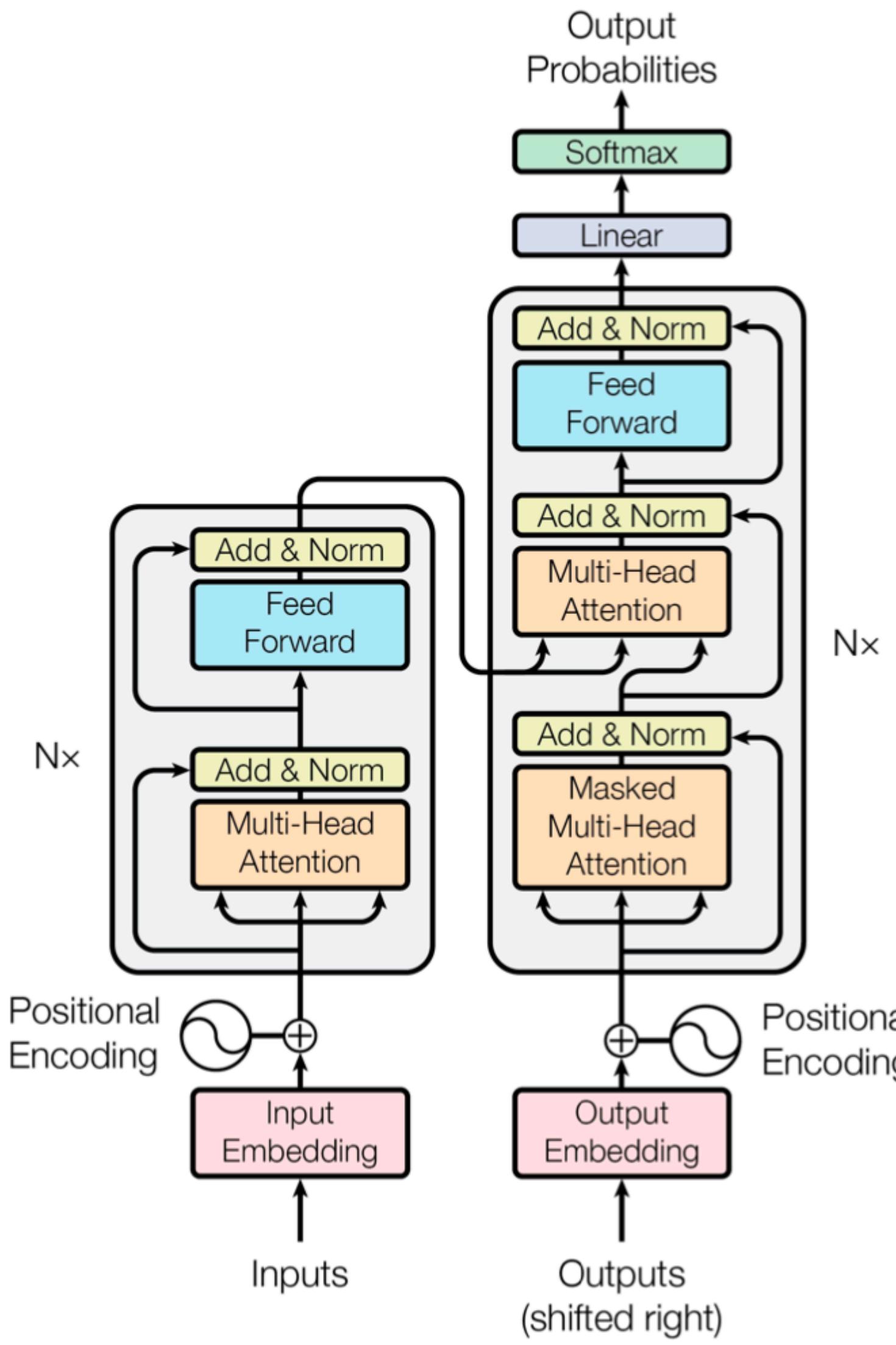
	BLEU
WMT'14 English-French	
Wu et al. (2016) GNMT (Word 80K)	37.90
Wu et al. (2016) GNMT (Word pieces)	38.95
Wu et al. (2016) GNMT (Word pieces) + RL	39.92
ConvS2S (BPE 40K)	40.51



Boulder



Attention Is All You Need

[YouTube Playlist](#)


$(x_1, x_2, \dots, x_n) \rightarrow$ input sequence of symbol representations
 $(z_1, z_2, \dots, z_n) \rightarrow$ sequence of continuous representations
 $(y_1, y_2, \dots, y_m) \rightarrow$ output sequence of symbols
 encoder : $(x_1, \dots, x_n) \mapsto (z_1, \dots, z_n)$
 decoder : $(y_0, \dots, y_{t-1}; z_1, \dots, z_n) \mapsto p(y_t | y_0, \dots, y_{t-1}; z_1, \dots, z_n)$
 $x_t \in \{1, \dots, |V|\}$ $V \rightarrow$ input vocabulary
 $E \in \mathbb{R}^{|V| \times d_{\text{model}}}$ \rightarrow input embedding matrix
 $e_t = E(x_t) \in \mathbb{R}^{d_{\text{model}}}$
 $p_t \in \mathbb{R}^{d_{\text{model}}} \rightarrow$ positional encoding

$$p_t^{2i-1} = \sin\left(\frac{t}{10000^{\frac{2i-1}{d_{\text{model}}}}}\right)$$

$$p_t^{2i} = \cos\left(\frac{t}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

 $\epsilon_t = e_t + p_t \in \mathbb{R}^{d_{\text{model}}}$
 $q_t^{\ell} = W_q^{\ell} \epsilon_t \in \mathbb{R}^d \rightarrow$ query ($W_q^{\ell} \in \mathbb{R}^{d \times d_{\text{model}}}$)
 $k_t^{\ell} = W_k^{\ell} \epsilon_t \in \mathbb{R}^d \rightarrow$ key ($W_k^{\ell} \in \mathbb{R}^{d \times d_{\text{model}}}$)
 $v_t^{\ell} = W_v^{\ell} \epsilon_t \in \mathbb{R}^d \rightarrow$ value ($W_v^{\ell} \in \mathbb{R}^{d \times d_{\text{model}}}$)

$$a_{tt'}^{\ell} = \frac{q_t^{\ell T} k_{t'}^{\ell}}{\sqrt{d}}$$
 $d = d_{\text{model}}/m$

$$\alpha_{tt'}^{\ell} = \frac{\exp(a_{tt'}^{\ell})}{\sum_{\tau=1}^n \exp(a_{t\tau}^{\ell})}$$

$$h_t = \sum_{t'=1}^n \alpha_{tt'}^{\ell} v_{t'}^{\ell} \in \mathbb{R}^d$$

$$h_t = [h_t^1; \dots; h_t^m] \in \mathbb{R}^{d_{\text{model}}}$$

$$o_t = W^o h_t \in \mathbb{R}^{d_{\text{model}}}$$

$$W^o \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$$

$$f_t = \epsilon_t + o_t \in \mathbb{R}^{d_{\text{model}}}$$

$$\mu_t = \text{mean}_{i=1, \dots, d_{\text{model}}} f_t^i \in \mathbb{R}$$

$$\sigma_t = \text{std}_{i=1, \dots, d_{\text{model}}} f_t^i \in \mathbb{R}$$

$$g_t = \frac{f_t - \mu_t}{\sigma_t}$$

$$\text{FFN}(x) = \underbrace{\max(0, xW_1 + b_1)W_2 + b_2}_{\in \mathbb{R}^{d_{\text{ff}}}} \in \mathbb{R}^{d_{\text{model}}}$$

$$h_t = \sum_{t'=1}^{t-1} \alpha_{tt'} v_{t'} \rightarrow$$
 masking & shift right

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	3.3 \cdot 10^{18}	
Transformer (big)	28.4	41.0		$2.3 \cdot 10^{19}$



SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing



[YouTube Video](#)

SentencePiece comprises four main components:

Normalizer, Trainer, Encoder, and Decoder.

Normalizer is a module to normalize semantically-equivalent Unicode characters into canonical forms.

By default, SentencePiece normalizes the input text with the Unicode NFKC normalization.

SentencePiece implements two subword segmentation algorithms, byte-pair-encoding (BPE) and unigram language model, with the extension of **direct training from raw sentences** (i.e., no need for pre-tokenized input).

Given an input sentence (or word) of length N , BPE segmentation requires $O(N^2)$ computational cost when we naively scan the pair of symbols in every iteration. Sentence-Piece adopts an $O(N \log(N))$ algorithm in which the merged symbols are managed by a binary heap (priority queue).

The basic idea of **lossless tokenization** is to treat the input text just as a sequence of Unicode characters.

Whitespace is handled as a normal symbol. For the sake of clarity, SentencePiece first escapes the whitespace with a meta symbol `_` (U+2581), and tokenizes the input into an arbitrary subword sequence, for example:

- **Raw text:** Hello_world.

- **Tokenized:** [Hello] [_wor] [ld] [.]

```
detok = '' . join(tokens) . replace('_', ' ')
```

We can find larger improvements in BLEU when 1) SentencePiece is applied to Japanese, and 2) the target sentence is Japanese.

Kudo, Taku, and John Richardson. "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." *arXiv preprint arXiv:1808.06226* (2018).

Normalize(*text*) → Decode(Encode(Normalize(*text*))) = Normalize(*text*).

```
% spm_train --input=data/input.txt
--model_prefix=spm --vocab_size=1000
```

```
% echo "Hello_world." | spm_encode --
model=spm.model
Hello_world.
```

```
% echo "Hello_world." | spm_encode
--model=spm.model --output_format=id
151 88 21 887 6
```

```
% echo "_He_ll_o__world_." |
spm_decode --model=spm.model
Hello world.
```

```
% echo "151_88_21_887_6" | spm_decode
--model=spm.model
--input_format=id
Hello world.
```

Lang pair	setting (source/target)	# vocab.	BLEU
ja→en	Word model (baseline)	80k/80k	28.24
	SentencePiece	8k (shared)	29.55
	SentencePiece w/ pre-tok.	8k (shared)	29.85
	Word/SentencePiece	80k/8k	27.24
	SentencePiece/Word	8k/80k	29.14
en→ja	Word model (baseline)	80k/80k	20.06
	SentencePiece	8k (shared)	21.62
	SentencePiece w/ pre-tok.	8k (shared)	20.86
	Word/SentencePiece	80k/8k	21.41
	SentencePiece/Word	8k/80k	19.94



Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates


[YouTube Playlist](#)

Unigram language model

A new subword segmentation algorithm based on a unigram language model, which is capable of outputting multiple subword segmentations with probabilities.

Subwords (- means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/l/o/_world	7 18085 356 356 137 255
_H/el/l/o/_world	320 585 356 137 7 12295

$\mathcal{V} \rightarrow$ vocabulary

$$\mathcal{L} = \sum_{s=1}^{|D|} \log(P(X^{(s)})) = \sum_{s=1}^{|D|} \log \left(\sum_{\mathbf{x} \in \mathcal{S}(X^{(s)})} P(\mathbf{x}) \right) \rightarrow \text{likelihood}$$

$X^{(s)} \in D \rightarrow$ sentence

$|D| \rightarrow$ size of the dataset

$\mathcal{S}(X^{(s)}) \rightarrow$ set of segmentation candidates built from the input sentence $X^{(s)}$

$\mathbf{x} = (x_1, \dots, x_M) \rightarrow$ subword sequence

$$p(\mathbf{x}) = \prod_{i=1}^M p(x_i) \rightarrow \text{unigram language model}$$

$p(x_i) \rightarrow$ subword occurrence probabilities

$$x_i \in \mathcal{V}, \forall i \quad \text{and} \quad \sum_{x \in \mathcal{V}} p(x) = 1$$

Given \mathcal{V} , the hidden variables $p(x_i)$ are estimated via the EM algorithm that maximizes the marginal likelihood \mathcal{L} .

Kudo, Taku. "Subword regularization: Improving neural network translation models with multiple subword candidates." *arXiv preprint arXiv:1804.10959* (2018).

Comparison of different segmentation algorithms (WMT14 en→de)

Model	BLEU
Word	23.12
Character (512 nodes)	22.62
Mixed Word/Character	24.17
BPE	24.53
Unigram w/o SR ($l = 1$)	24.50
Unigram w/ SR ($l = 64, \alpha = 0.1$)	25.04

$$P(\mathbf{x}_i | X) \cong P(\mathbf{x}_i)^\alpha / \sum_{i=1}^l P(\mathbf{x}_i)^\alpha$$

l -best segmentations according to the probability $P(\mathbf{x}|X)$

1. Heuristically make a reasonably big seed vocabulary from the training corpus.
2. Repeat the following steps until $|\mathcal{V}|$ reaches a desired vocabulary size.
 - (a) Fixing the set of vocabulary, optimize $p(x)$ with the EM algorithm.
 - (b) Compute the $loss_i$ for each subword x_i , where $loss_i$ represents how likely the likelihood \mathcal{L} is reduced when the subword x_i is removed from the current vocabulary.
 - (c) Sort the symbols by $loss_i$ and keep top η % of subwords (η is 80, for example). Note that we always keep the subwords consisting of a single character to avoid out-of-vocabulary.

Neural Machine Translation (NMT) with Subword Regularization (SR)

$$\mathcal{L}_{marginal}(\theta) = \sum_{s=1}^{|D|} \mathbb{E}_{\mathbf{x} \sim P(\mathbf{x}|X^{(s)})} [\log P(\mathbf{y}|\mathbf{x}; \theta)]$$

$$P(\mathbf{y}|\mathbf{x}; \theta) = \prod_{n=1}^N P(y_n|\mathbf{x}, y_{<n}; \theta)$$



Boulder



[YouTube Playlist](#)

Reformer: The Efficient Transformer

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V$$

$Q, K, V \in \mathbb{R}^{b \times l \times d}$ → query, key, value

b → batch size

l → sequence length

d → dimension of queries, keys and values

$QK^T \in \mathbb{R}^{b \times l \times l}$

- memory footprint

- computational cost

Memory-efficient attention

If $l = 64K$ and $b = 1$, the memory footprint of QK^T is $64K \times 64K \times \text{Float32} \approx 16\text{GB}$.

The attention can be computed for each query q_t separately.

$$o_t = \sum_{s \in \mathcal{P}_t} \exp(q_t \cdot k_s - z(t, \mathcal{P}_t)) v_s$$

$z(t, \mathcal{P}_t)$ → normalization term in softmax

$\mathcal{P}_t = \{s : s \leq t\}$ → set that t attends to

Locality Sensitive Hashing (LHS)

Computational complexity is quadratic in l .

Softmax is dominated by the largest elements!

Only need to focus on keys closest to q_t .

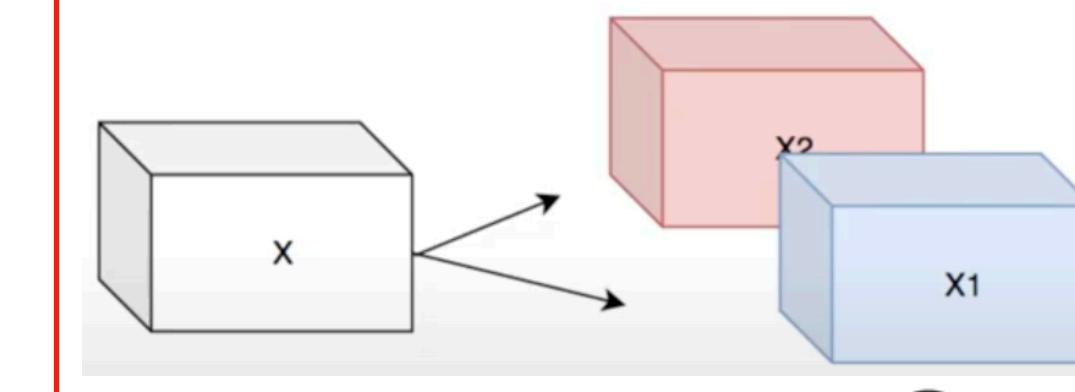
Nearby vectors get the same hash with high probability and distant ones do not.

h → hash function

$$\mathcal{P}_t = \{s : s \leq t, h(k_s) = h(q_t)\}$$

Computational complexity can be reduced to $O(l \log l)$ in l .

Reversible Transformer

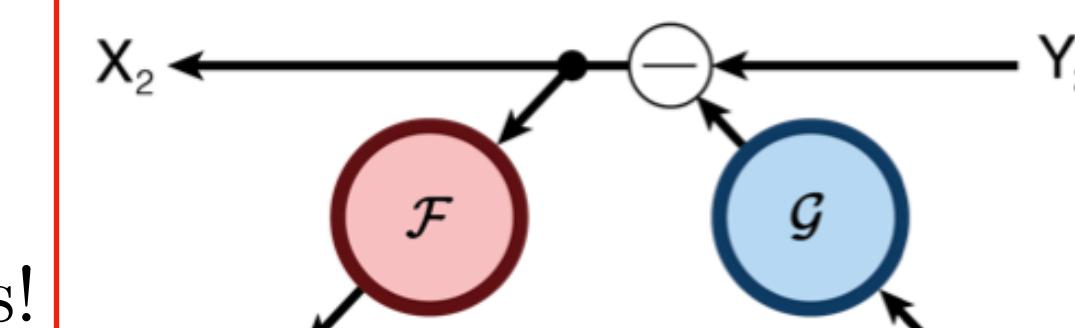


\mathcal{F} → Attention

\mathcal{G} → FeedForward

$$y_1 = x_1 + \mathcal{F}(x_2)$$

$$y_2 = x_2 + \mathcal{G}(y_1)$$



$$x_2 = y_2 - \mathcal{G}(y_1)$$

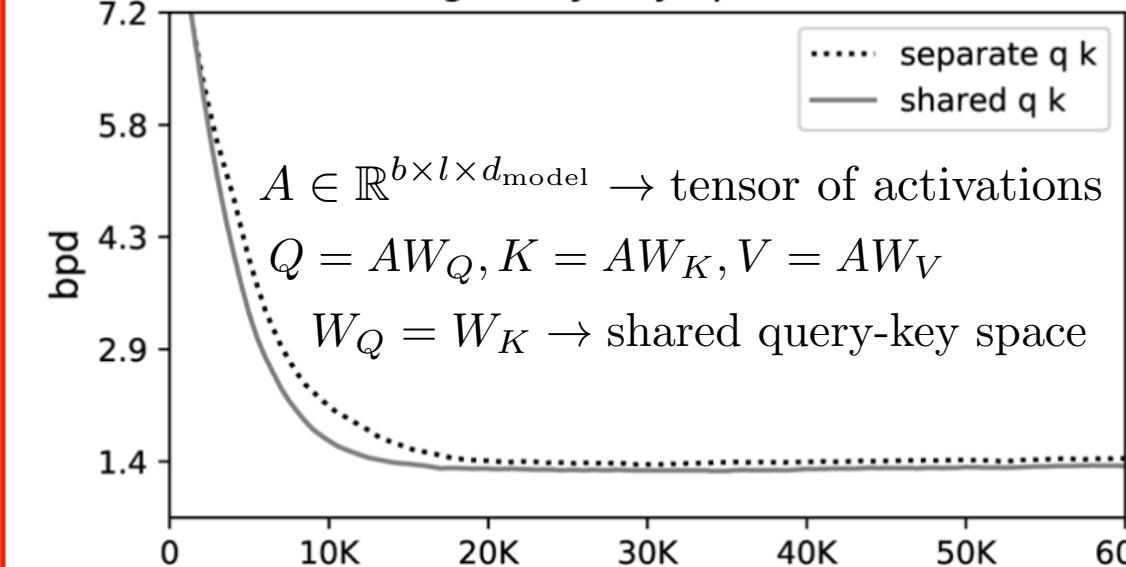
$$x_1 = y_1 - \mathcal{F}(x_2)$$

Backpropagation without storing activations!

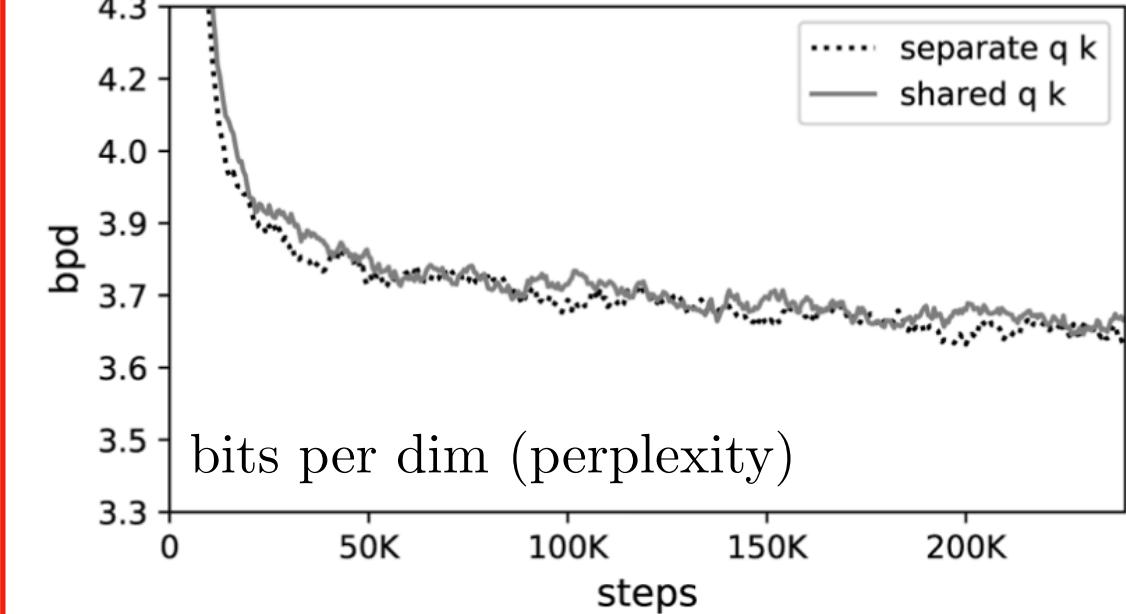
Chunking across positions in a sequence

$$Y_2 = [Y_2^{(1)}; \dots; Y_2^{(c)}] \quad Y_2^{(i)} = X_2^{(i)} + \mathcal{G}(Y_1^{(i)})$$

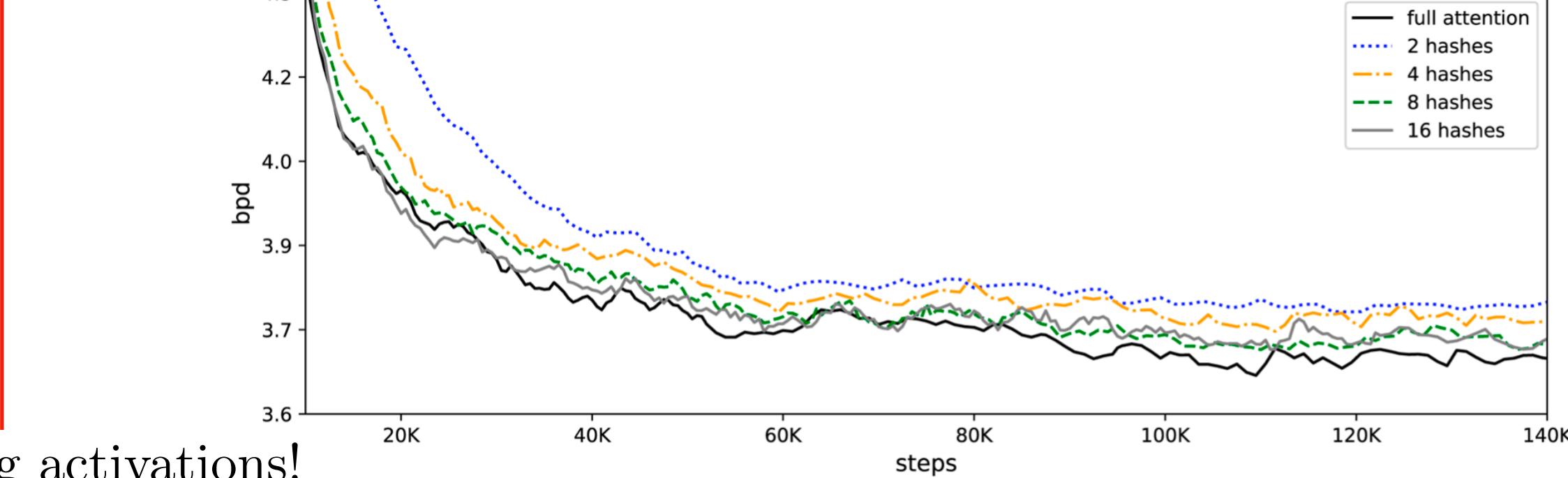
Sharing Query-Key Space - enwik8



Sharing Query-Key Space - imagenet64



LSH Attention on Imagenet64





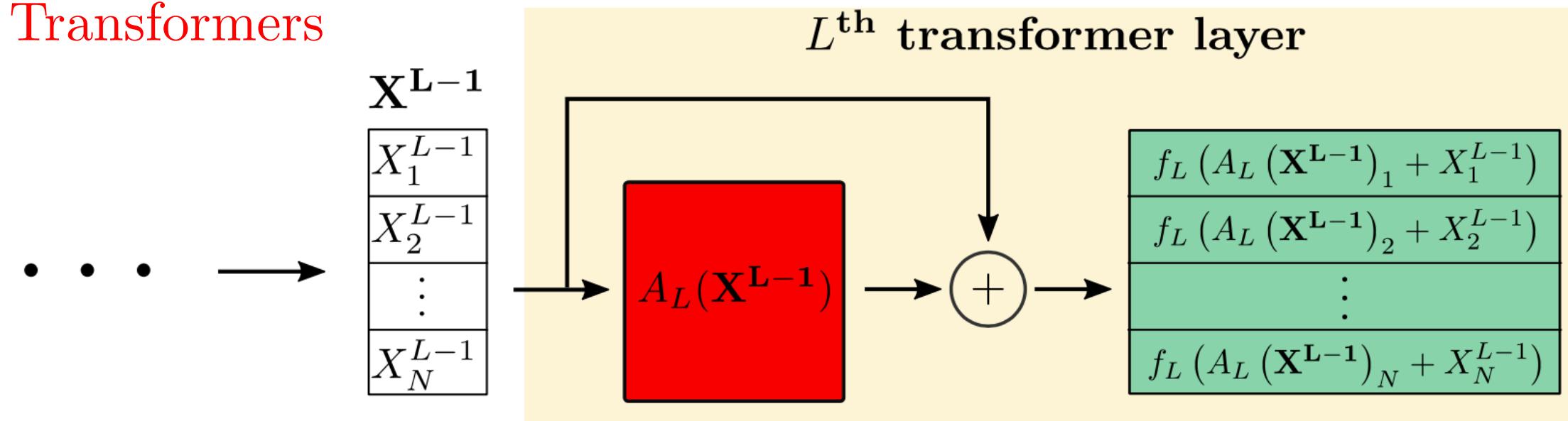
Boulder

Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention



YouTube Video

Transformers



Linearized Attention

 $\text{sim}(q, k) = \phi(q)^T \phi(k), \phi \rightarrow$ feature representation of the kernel

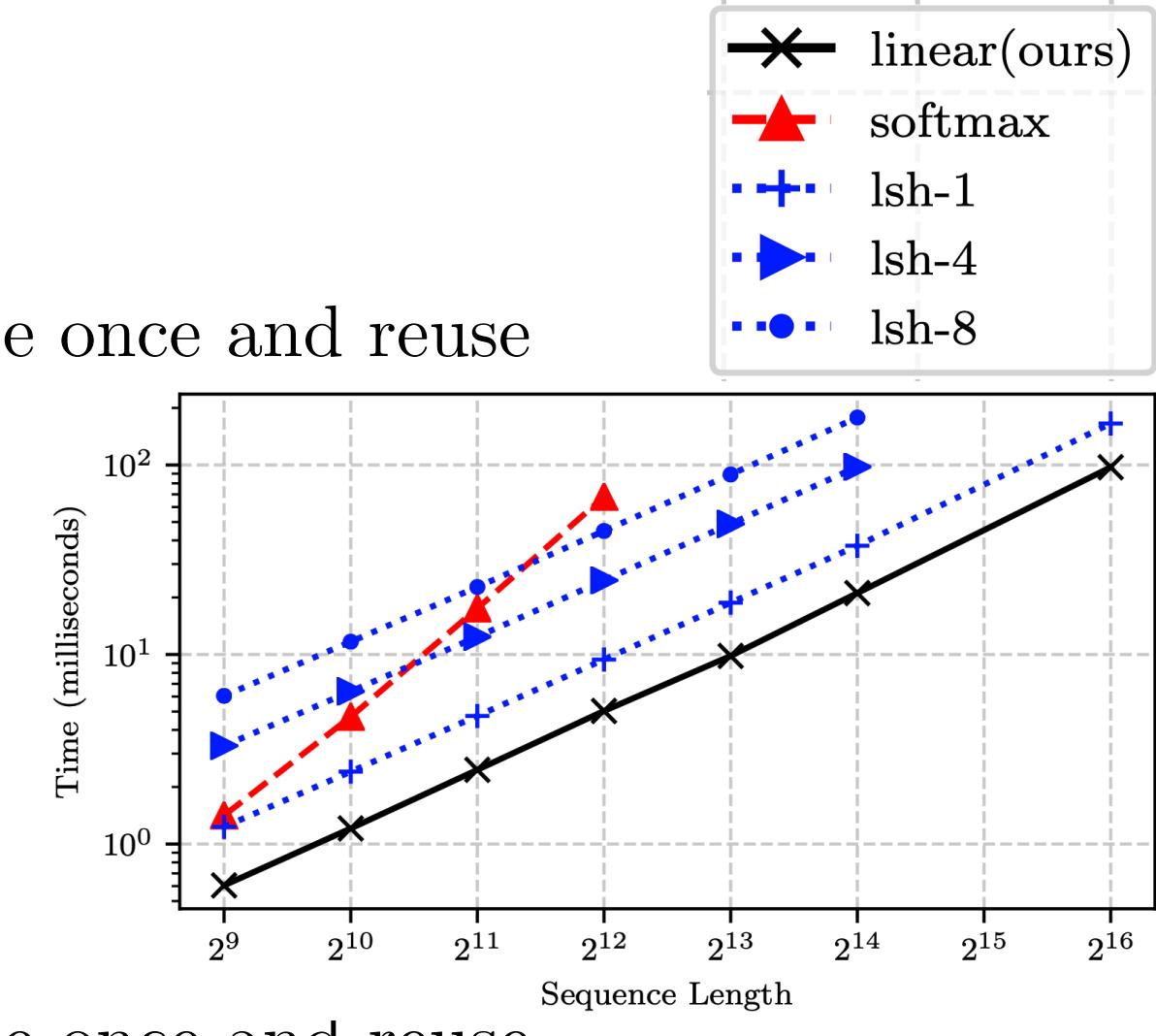
$$\begin{aligned} \phi(x) &= \text{elu}(x) + 1 \\ V'_i &= \frac{\sum_{j=1}^N \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^N \phi(Q_i)^T \phi(K_j)} \\ &=: S \rightarrow \text{compute once and reuse} \\ &= \frac{\phi(Q_i)^T \sum_{j=1}^N \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^N \phi(K_j)} \\ &=: Z \rightarrow \text{compute once and reuse} \end{aligned}$$

Causal Masking $=: S_i \rightarrow$ can be computed from S_{i-1}

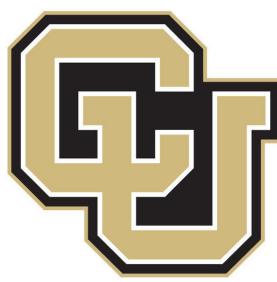
$$V'_i = \frac{\phi(Q_i)^T \sum_{j=1}^i \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^i \phi(K_j)}$$

Gradient Computation $=: Z_i \rightarrow$ can be computed from Z_{i-1}

A naive implementation would require storing all intermediate values S_i in order to compute the gradients! See the paper for a derivation of the gradients of the numerator as cumulative sums!



$$\begin{aligned} \text{Transformers are RNNs} \\ s, z \rightarrow \text{attention, normalizer memory} \\ s_0 = 0, z_0 = 0 \\ s_i = s_{i-1} + \phi(x_i W_K) (x_i W_V)^T \\ z_i = z_{i-1} + \phi(x_i W_K) \\ y_i = f_l \left(\frac{\phi(x_i W_Q)^T s_i}{\phi(x_i W_Q)^T z_i} + x_i \right) \end{aligned}$$



Boulder



[YouTube Playlist](#)

Rethinking Attention with Performers

Fast Attention Via positive Orthogonal Random features (FAVOR+)

Bidirectional (non-directional) dot-product attention

$L \rightarrow$ size of an input sequence of tokens

$d \rightarrow$ dimension of the latent representation

$Q, K, V \in \mathbb{R}^{L \times d} \rightarrow$ input matrices (queries, keys, and values)

$A = \exp(QK^T / \sqrt{d}) \in \mathbb{R}^{L \times L} \rightarrow$ attention matrix

$D = \text{diag}(A\mathbf{1}_L)$

$\mathbf{1}_L \rightarrow$ all ones vector of length L

$\text{Att}_{\leftrightarrow}(Q, K, V) = D^{-1}AV$

$O(L^2d) \rightarrow$ time complexity

$O(L^2 + Ld) \rightarrow$ space complexity (A has to be stored explicitly)

Positive Random Features (PRFs) for Softmax

$\text{SM}(q, k) := \exp(q^T v) \rightarrow$ softmax kernel (omit \sqrt{d} -renormalization without loss of generality)

$\text{SM}(q, k) = \mathbb{E}_{w \sim \mathcal{N}(0, I_d)} [\exp(w^T q - \|q\|^2/2) \exp(w^T k - \|k\|^2/2)]$
 $= \exp(-(\|q\|^2 + \|k\|^2)/2) \mathbb{E}_{w \sim \mathcal{N}(0, I_d)} [\cosh(w^T (q + k))]$

$\widehat{\text{SM}}_m^+, \widehat{\text{SM}}_m^{\text{hyp}+} \rightarrow$ estimators

$m \rightarrow$ number of Monte-Carlo samples

$\widehat{\text{SM}}_m^+(q, v) = \phi^+(q)^T \phi^+(k)$

$\phi^+(x) = \frac{h^+(x)}{\sqrt{m}} (f^+(w_1^T x), \dots, f^+(w_m^T x))$

$h^+(x) = \exp(-\|x\|^2/2), f^+(x) = \exp(x)$

$\widehat{\text{SM}}_m^{\text{hyp}+}(q, v) = \phi^{\text{hyp}+}(q)^T \phi^{\text{hyp}+}(k) \rightarrow$ variance reduction

$\phi^{\text{hyp}+}(x) = \frac{h^{\text{hyp}+}(x)}{\sqrt{m}} (f_1^{\text{hyp}+}(w_1^T x), \dots, f_1^{\text{hyp}+}(w_m^T x), f_2^{\text{hyp}+}(w_1^T x), \dots, f_2^{\text{hyp}+}(w_m^T x))$

$h^{\text{hyp}+}(x) = \frac{1}{\sqrt{2}} \exp(-\|x\|^2/2), f_1^{\text{hyp}+}(x) = \exp(x), f_2^{\text{hyp}+}(x) = \exp(-x)$

Orthogonal Random Features (ORFs)

$w_1, w_2, \dots, w_m \rightarrow$ orthogonal (Gram-Schmidt)

Generalized Kernelizable Attention

$Q', V' \in \mathbb{R}^{L \times r}$ with rows given by $\phi(q_i^T)^T, \phi(k_i^T)^T$

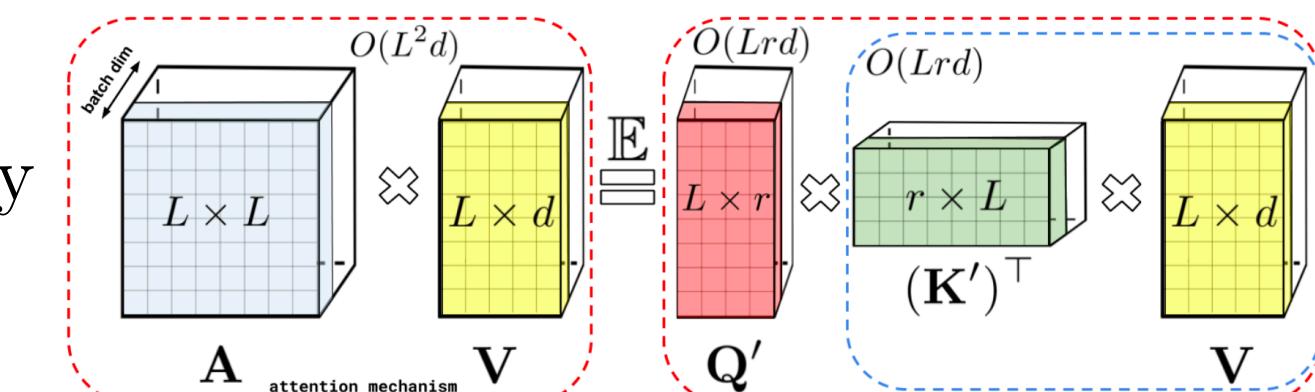
$r = m$ for $\widehat{\text{SM}}^+$ and $r = 2m$ for $\widehat{\text{SM}}^{\text{hyp}+}$

$\widehat{\text{Att}}_{\leftrightarrow}(Q, K, V) = \widehat{D}^{-1}(Q'((K')^T V)) \rightarrow$ approximate attention

$\widehat{D} = \text{diag}(Q'((K')^T \mathbf{1}_L))$

$O(Lr + Ld + rd) \rightarrow$ space complexity

$O(Lrd) \rightarrow$ time complexity



Unidirectional Case and Prefix Sums

$\text{tril} \rightarrow$ lower triangular part including the diagonal

$\text{tril}(Q'((K')^T)C)$ where $C = [V \ \mathbf{1}]_T \in \mathbb{R}^{L \times (d+1)}$

$Q'((K')^T) \rightarrow$ we don't want to create and store this matrix

$[\text{tril}(\mathbf{Q}'(\mathbf{K}')^\top) \mathbf{C}]_i = \mathbf{G}_{i,:,:}^{\text{PS}} \times \mathbf{Q}'_i, \quad \mathbf{G}_{i,:,:}^{\text{PS}} = \sum_{j=1}^i \mathbf{G}_{j,:,:}, \quad \mathbf{G}_{j,:,:} = \mathbf{K}'_j \mathbf{C}_j^\top \in \mathbb{R}^{r \times (d+1)}$

$\mathbf{G}, \mathbf{G}^{\text{PS}} \in \mathbb{R}^{L \times r \times (d+1)}$ are 3d-tensors



Boulder



Questions?

[YouTube Playlist](#)
