Transfer Learning & Transformers

Artem Chernodub (guest speaker)
Grammarly & UCU

Kyiv, 2019

Short BIO

Kyiv Natural Sciences Lyceum # 145

Moscow Institute of Physics and Technology (B.Sc., M.Sc.), 2007

Institute of Mathematical Machines and Systems NASU, Kyiv (PhD), 2016

Grammarly

UCU

Agenda

- Transfer Learning
    - Recalling word embeddings
    - Multi-task Learning
    - Big Language Models (BERT, ELM and co.)

- Transformers
    - RNNs? ConvNets? Attention!
    - Recalling Attention Mechanism
    - Self-Attention
    - Transformers design

- Transformers + Transfer Learning: BERT

Transfer learning

Transfer learning

= storing knowledge gained while solving one problem and applying it to a different but related problem

Word embeddings

Distributional hypothesis

A word's meaning is given by the words that frequently appear close-by.

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent word "banking"!

Semantic similarity

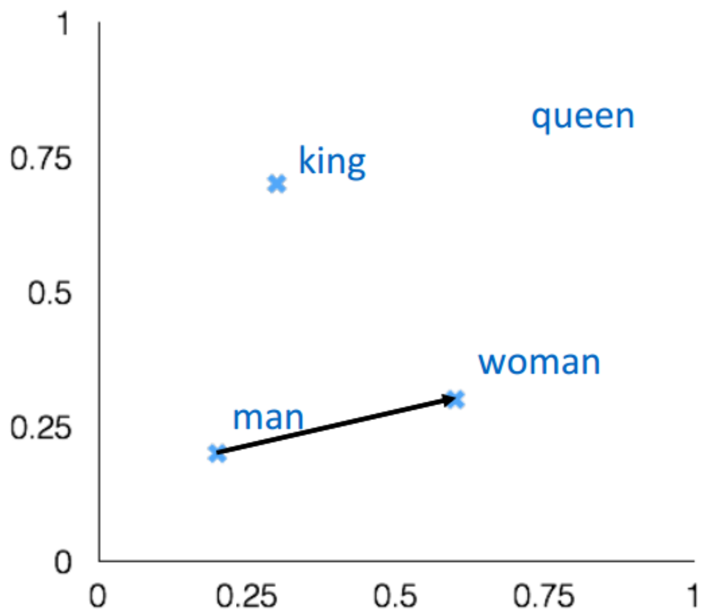
Word Analogies

Test for linear relationships, examined by Mikolov et al. (2014)

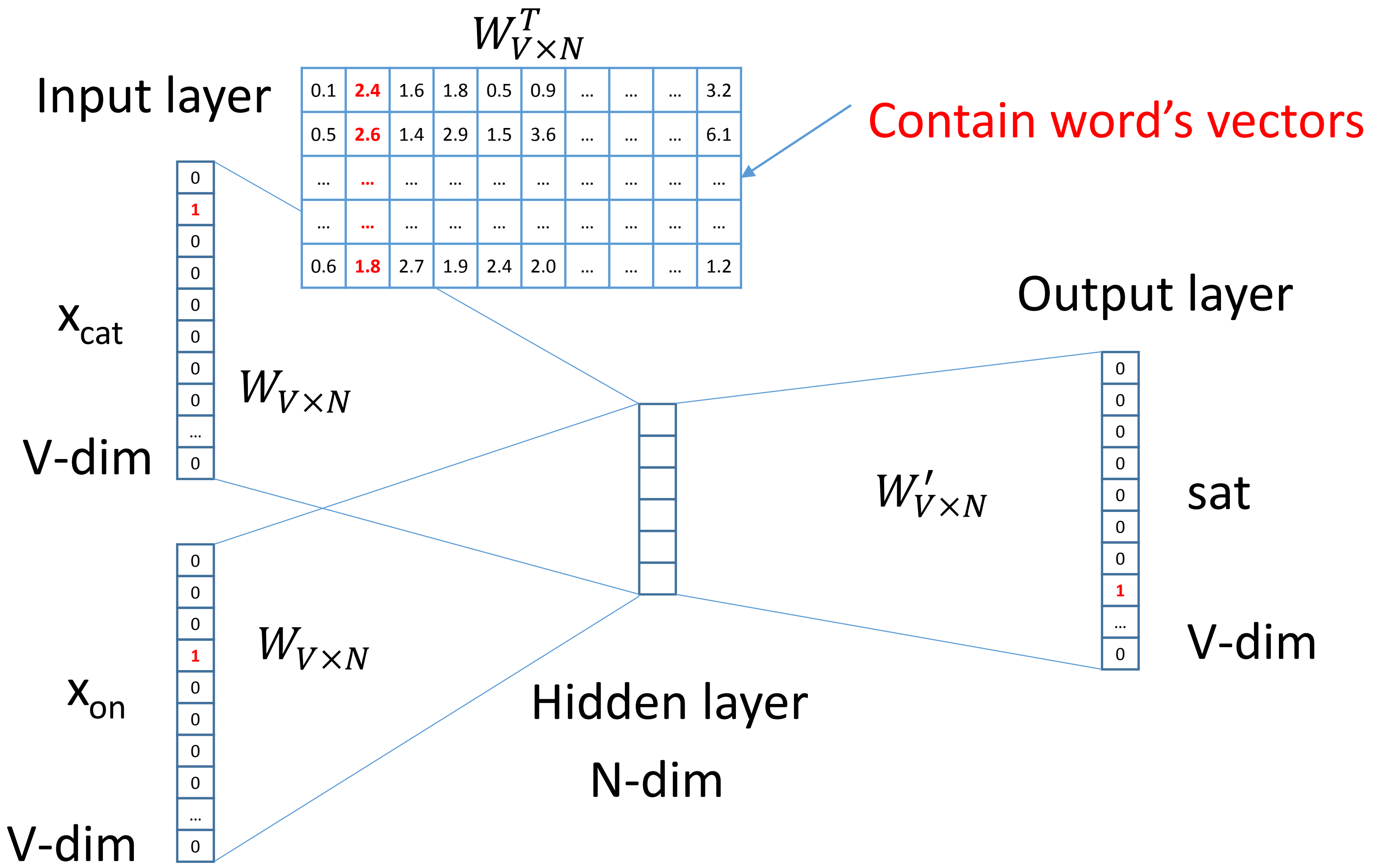
$\boxed{a:b :: c:?}$ $\longrightarrow$ $$\boxed{d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



word2vec – MLP



We can consider either W or W' as the word's representation. Or even take the average.

Example: Argument Mining

- detection and evaluation of the arguments in natural texts;
- develop automatic systems for making judgments, support decision making and finding contradictions in the natural text, document summarization, analysis of scientific papers, writing assistance, essay scoring.
- domains: law, decision making, philosophy.

Argument Component Detection

List of Argument Components:

"THE" -- this token is a part of the thesis of the argument (claim);
"PRO" -- this token is a part of a statement that supports the thesis (premise);
"CON" -- this token is a part of a statement that supports the opposite statement to the thesis.

Input: Let us discuss which technology to use for our new project. In my opinion, Python is a good choice for scientific programming, because it is open source and has a rich collection of libraries, such as NumPy.

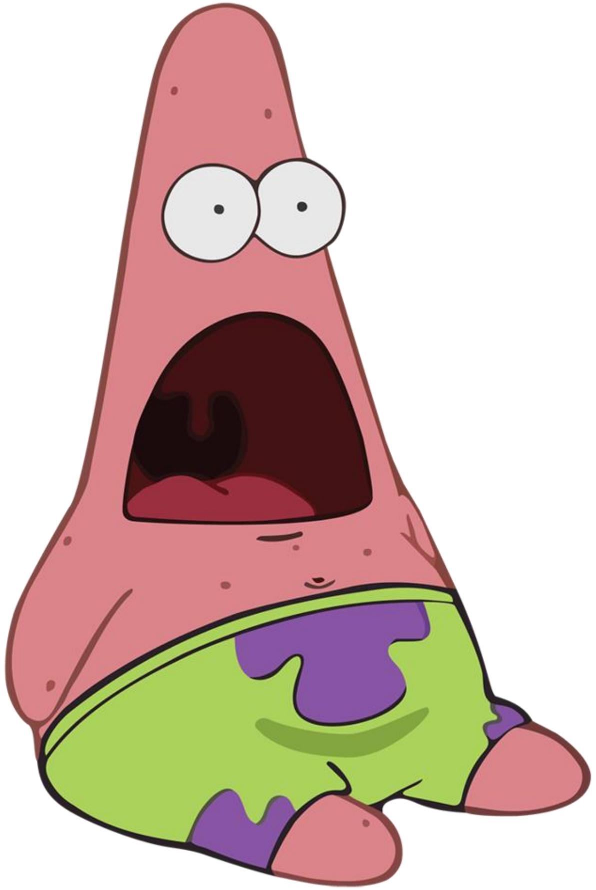
Output: Let$_{O}$ us$_{O}$ discuss$_{O}$ which$_{O}$ technology$_{O}$ to$_{O}$ use$_{O}$ for$_{O}$ our$_{O}$ new$_{O}$ project$_{O}$. In$_{O}$ my$_{O}$ opinion$_{O}$, Python$_{B\text{-}THE}$ is$_{I\text{-}THE}$ a$_{I\text{-}THE}$ good$_{I\text{-}THE}$ choice$_{I\text{-}THE}$ for$_{I\text{-}THE}$ scientific$_{I\text{-}THE}$ programming$_{I\text{-}THE}$, because$_{I}$ it$_{B\text{-}PRO}$ is$_{I\text{-}PRO}$ open$_{I\text{-}PRO}$ source$_{I\text{-}PRO}$ and$_{I\text{-}PRO}$ has$_{I\text{-}PRO}$ a$_{I\text{-}PRO}$ rich$_{I\text{-}PRO}$ collection$_{I\text{-}PRO}$ of$_{I\text{-}PRO}$ libraries$_{I\text{-}PRO}$, such$_{I\text{-}PRO}$ as$_{I\text{-}PRO}$ NumPy$_{I\text{-}PRO}$.

AM Feature set

Group	*Feature*	*Description*
Structural	Token position	Token present in introduction or conclusion*; token is first or last token in sentence; relative and absolute token position in document, paragraph and sentence
	Punctuation	Token precedes or follows any punctuation, full stop, comma and semicolon; token is any punctuation or full stop
	Position of covering sentence	Absolute and relative position of the token's covering sentence in the document and paragraph
Syntactic	Part-of-speech	The token's part-of-speech
	Lowest common ancestor (LCA)	Normalized length of the path to the LCA with the following and preceding token in the parse tree
	LCA types	The two constituent types of the LCA of the current token and its preceding and following token
LexSyn	Lexico-syntactic	Combination of lexical and syntactic features as described by Soricut and Marcu (2003)
Prob	Probability	Conditional probability of the current token being the beginning of a component given its preceding tokens

Stab C., Gurevych I. Parsing Argumentation Structures in Persuasive Essays // Computational Linguistics. – 2017. – T. 43. – №. 3. – C. 619-659.

Embeddings vs hand-crafted features in AM



In-domain scenario

Domain	Feature set combinations 0	01	012	0123	01234	1234	234	34	4
HS	0.134	0.162	0.167	0.165	0.187	0.176	**0.205**	0.203	0.193
MS	0.072	0.123	0.138	0.151	0.198	0.216	0.165	0.190	**0.226**
PIS	0.152	0.174	0.178	0.168	**0.212**	0.192	0.175	0.177	0.181
PPS	0.235	0.233	0.230	0.240	**0.265**	0.250	0.239	0.250	0.243
RS	0.090	0.156	0.156	0.144	0.195	0.201	0.204	0.190	**0.225**
SSE	0.141	0.176	0.200	0.185	0.206	**0.216**	0.189	0.202	0.201
Aggregated	0.182	0.200	0.205	0.206	**0.236**	0.230	0.218	0.228	0.229

Cross-domain scenario

Domain	Feature set combinations 0	01	012	0123	01234	1234	234	34	4
HS	0.087	0.063	0.044	0.106	0.072	0.075	0.065	0.063	**0.197**
MS	0.072	0.060	0.070	0.058	0.038	0.062	0.045	0.060	**0.188**
PIS	0.078	0.073	0.083	0.074	0.086	0.073	0.096	0.081	**0.166**
PPS	0.070	0.059	0.070	0.132	0.059	0.062	0.071	0.067	**0.203**
RS	0.067	0.067	0.082	0.110	0.097	0.092	0.075	0.075	**0.257**
SSE	0.092	0.089	0.066	0.036	0.120	0.091	0.071	0.066	**0.194**
Aggregated	0.079	0.086	0.072	0.122	0.094	0.088	0.089	0.076	**0.209**

Stab C., Gurevych I. Parsing Argumentation Structures in Persuasive Essays // Computational Linguistics. – 2017. – T. 43. – №. 3. – C. 619-659.

Multi-task learning

Multi-task Learning in NLP

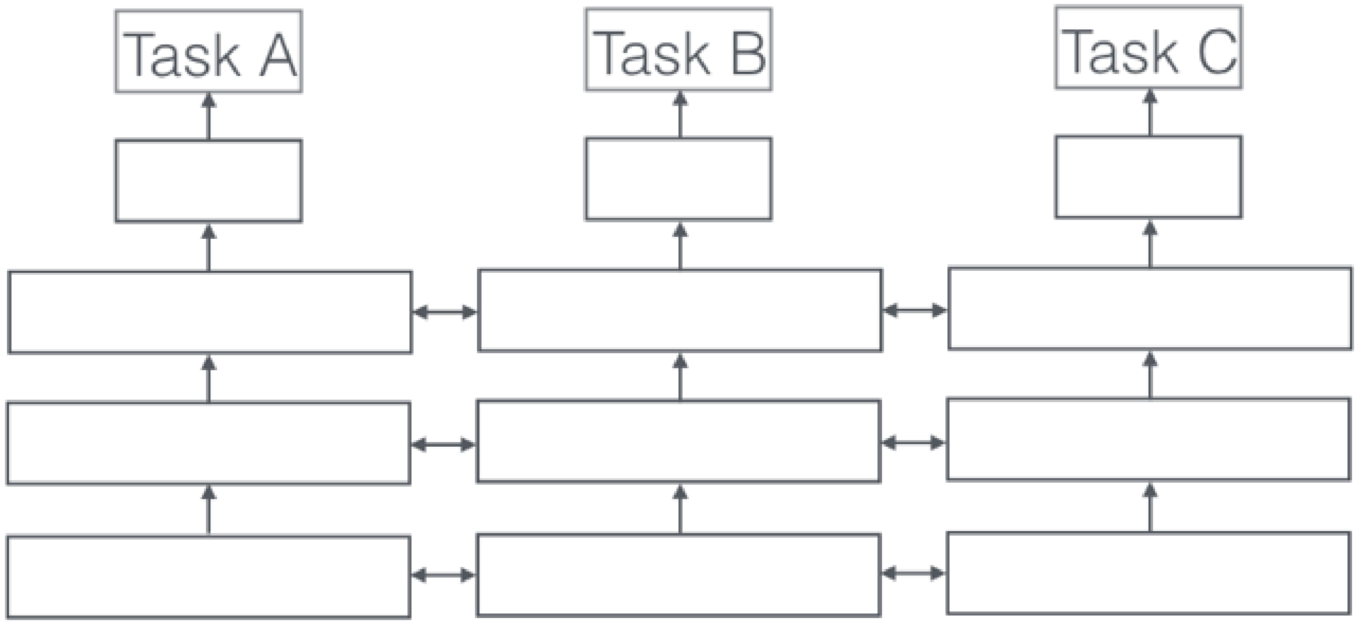
- Proposed by Caruana [*], popularized for NLP by Collobert et.al [**], and recently by Anders Sogaard [***].
- Idea: training few datasets simultaneously, information/parameters sharing between similar problems
- Generalization is improved by using training data for similar or related tasks (datasets).

[*] Rich Caruana, Multi-task Leaning: A Knowledge-Based Source of Inductive Bias. Proceedings of ICML, 1993.

[**] Collobert, Ronan, et al. Natural language processing (almost) from scratch, Journal of Machine Learning Research 12, 2011.

[***] Augenstein, Isabelle, and Anders Søgaard. "Multi-Task Learning of Keyphrase Boundary Classification." arXiv preprint arXiv:1704.00514, 2017.

Multi-task Learning in NLP

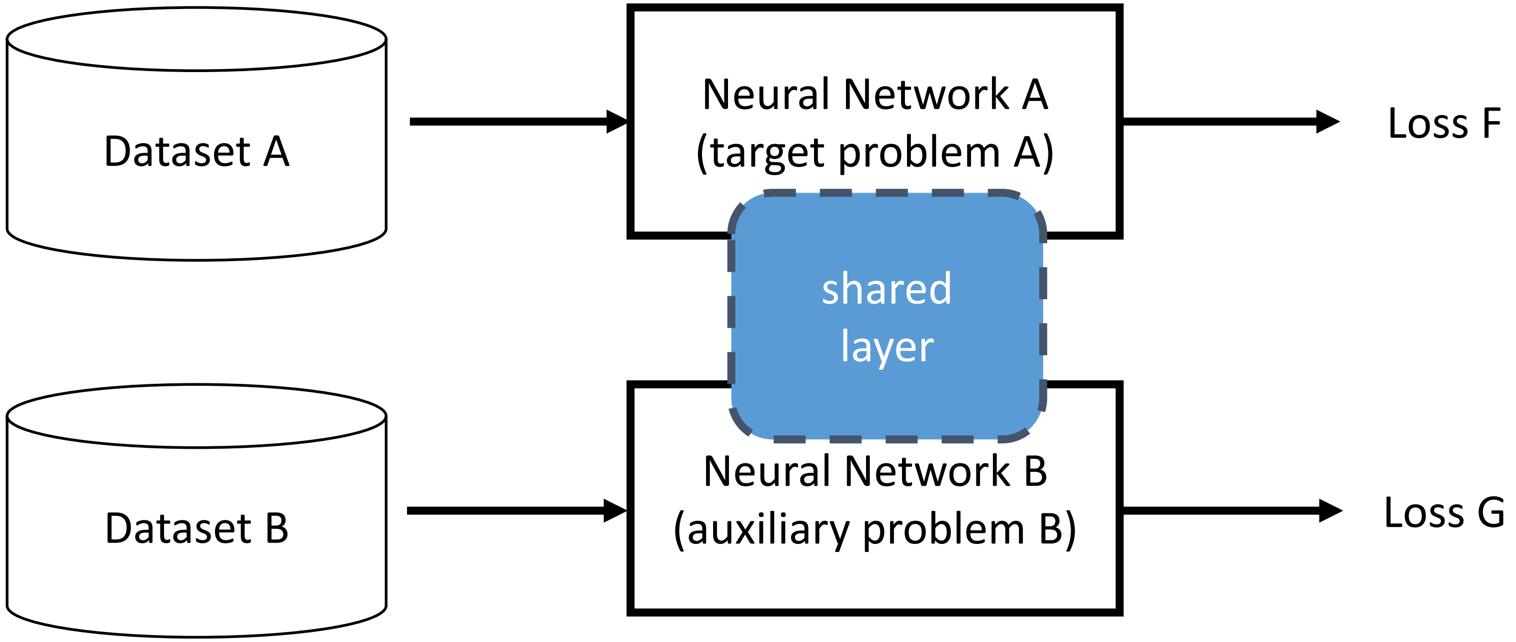


- Auxiliary objectives from the same datasets (language modelling, predict data statistics, learning the inverse)
- Joint training on similar NLP tasks (machine translation, semantic parsing, chunking, speech recogntion)

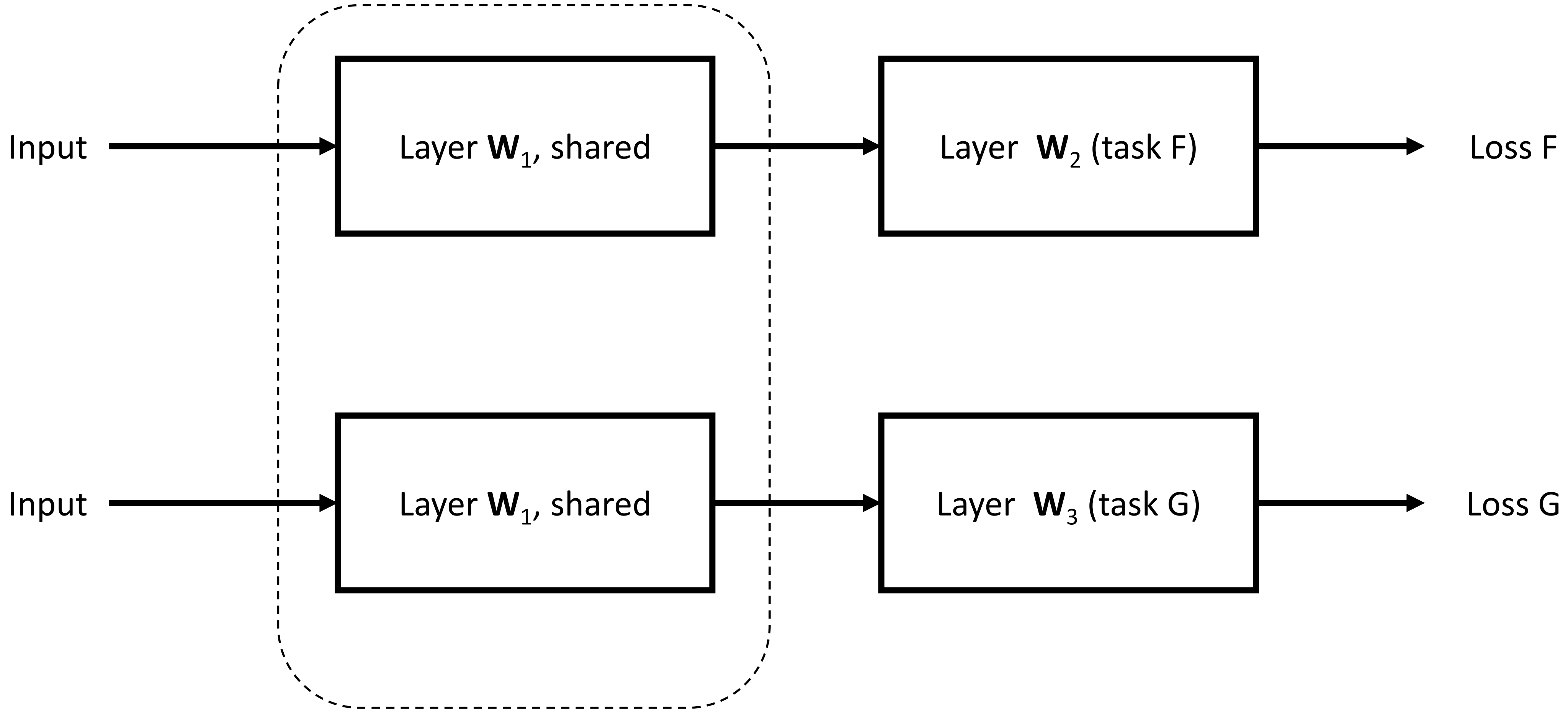
Source: http://ruder.io/multi-task-learning-nlp/

Using Multi-Objective Loss Function for Multi-Task Learning

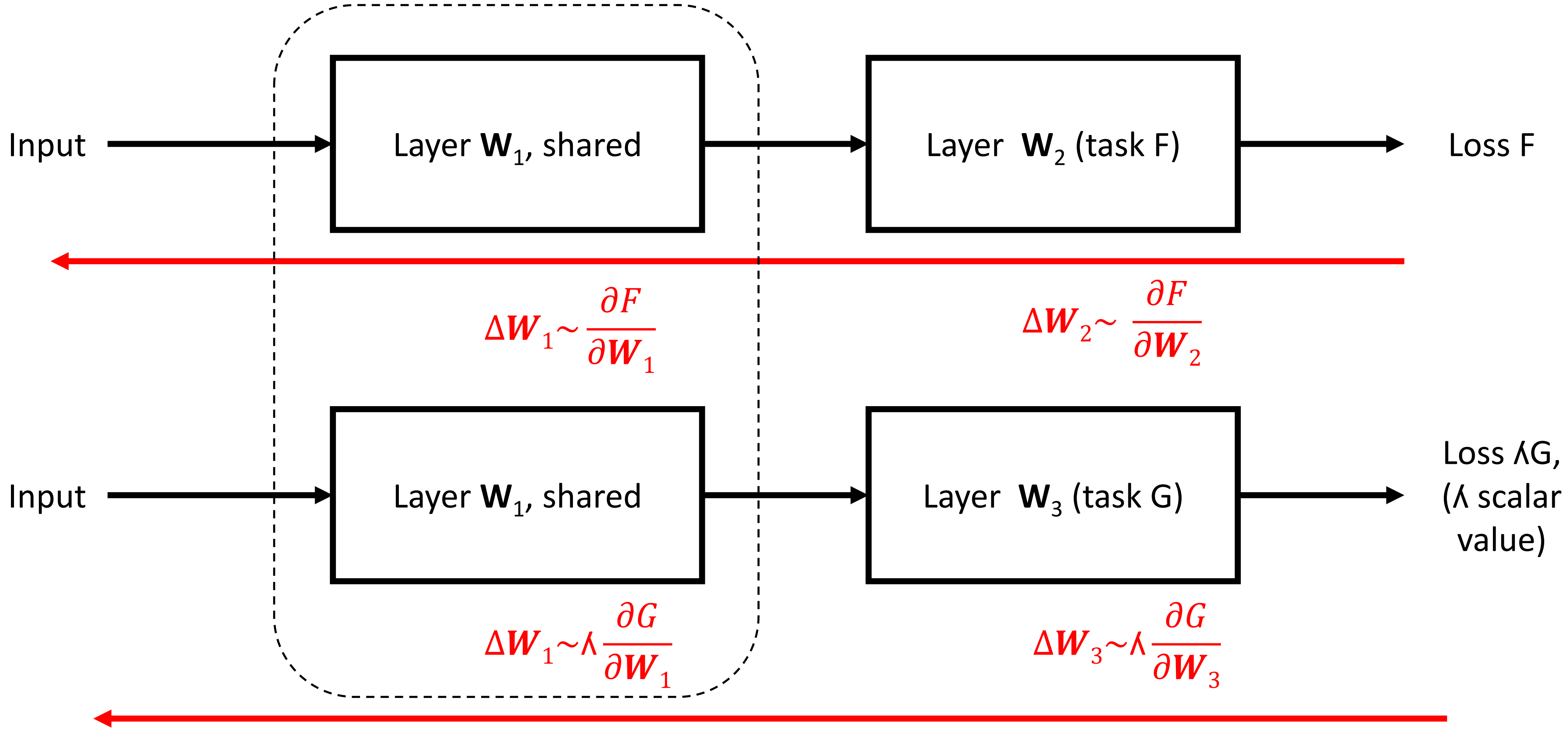
$$H(w) = F(w) + \lambda G(w)$$



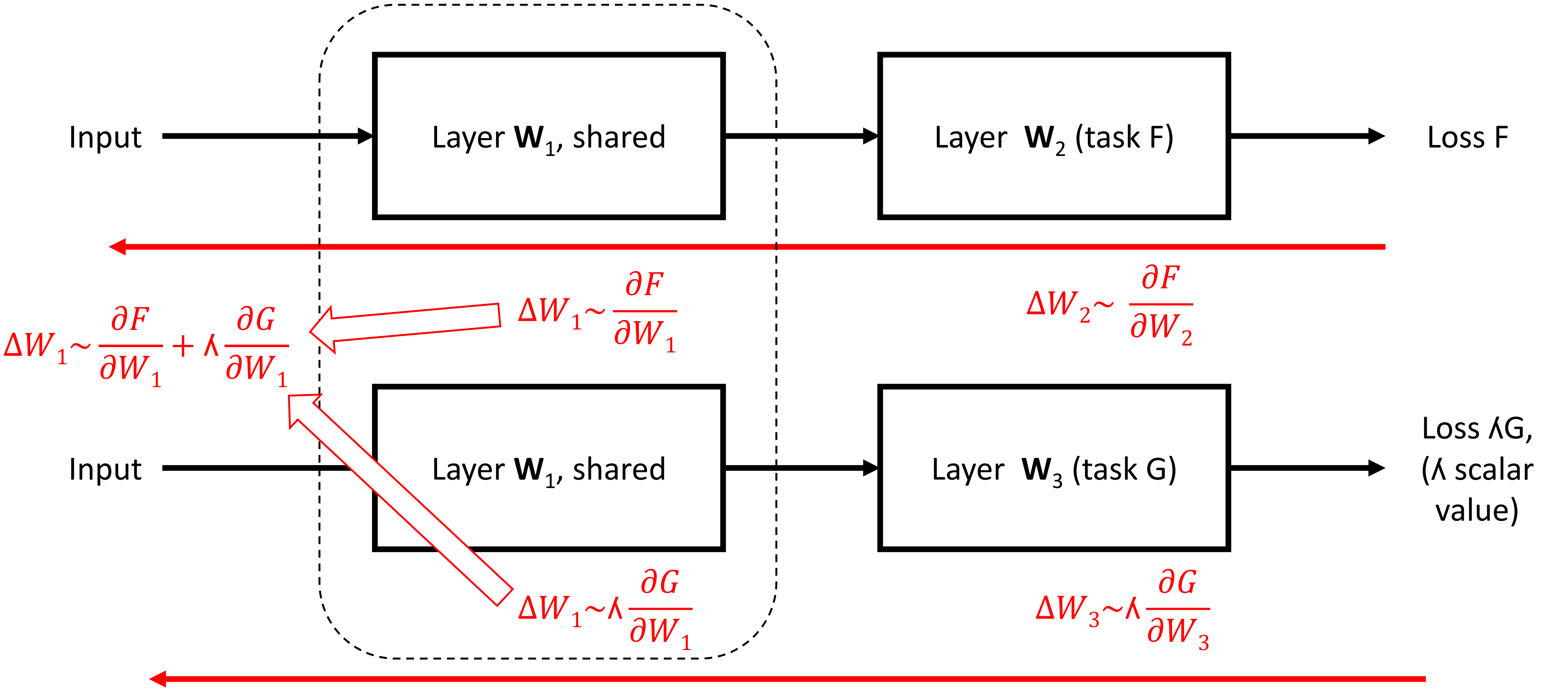
Using Parameters Sharing for Multi-Task Learning



Using Parameters Sharing for Multi-Task Learning (backward pass)



Using Parameters Sharing for Multi-Task Learning (backward pass)



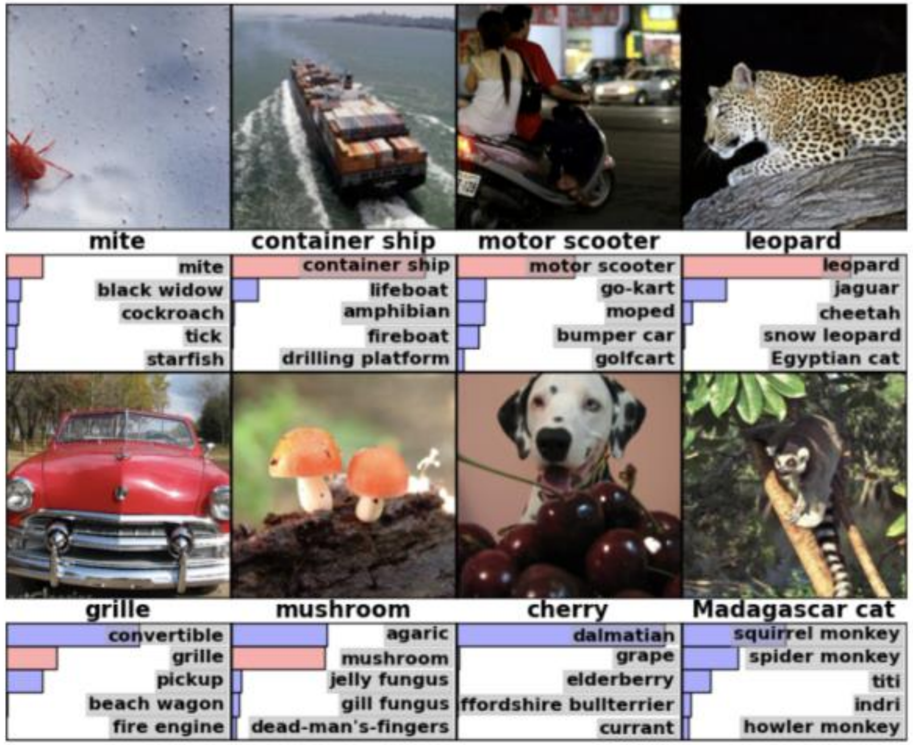
Example: keyphrase boundary classification

	Unlabelled			Labelled		
Method	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Finkel et al. (2005)	77.89	50.27	61.10	49.90	27.97	35.85
Lample et al. (2016)	71.92	49.37	58.55	41.36	28.47	33.72
BiLSTM	81.58	57.86	67.71	45.80	32.48	38.01
BiLSTM + Chunking	82.88	52.08	63.96	55.54	34.90	42.86
BiLSTM + Framenet	77.86	56.05	65.18	54.04	38.91	45.24
BiLSTM + Hyperlinks	76.59	60.53	67.62	46.99	44.09	41.13
BiLSTM + Multi-word	74.80	70.18	**72.42**	46.99	44.09	**45.49**
BiLSTM + Super-sense	83.70	51.76	63.93	56.94	35.25	43.54

Augenstein, Isabelle, and Anders Søgaard. "Multi-Task Learning of Keyphrase Boundary Classification." arXiv preprint arXiv:1704.00514, 2017.

Inspiration: Transfer learning in Computer Vision

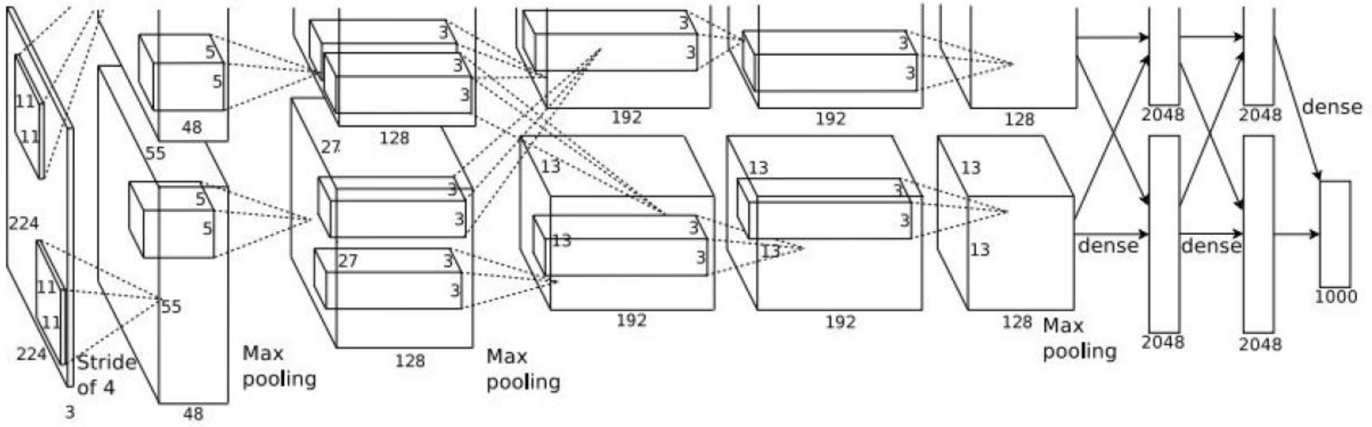
ILSVRC 2012 results on ImageNet



#	Team name	Method	Top-5 error
1	SuperVision	AlexNet + extra data (CNN)	0.15315
2	SuperVision	AlexNet (CNN)	0.16422
3	ISI	SIFT+FV, LBP+FV, GIST+FV	0.26172
5	ISI	Naive sum of scores from classifiers using each FV	0.26646
7	OXFORD_VGG	Mixed selection from High-Level SVM scores and Baseline Scores	0.26979

Transfer Learning in CV: AlexNet vs ImageNet, 2012

[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[55x55x96] RELU
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[27x27x256] RELU
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] RELU
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x384] RELU
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[13x13x256] RELU
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)



Images copyright © A. Krizhevsky et. al, 2012

Details:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate reduced by 10 manually when training become slow
- Top 5 on ImageNet error 16,4%.

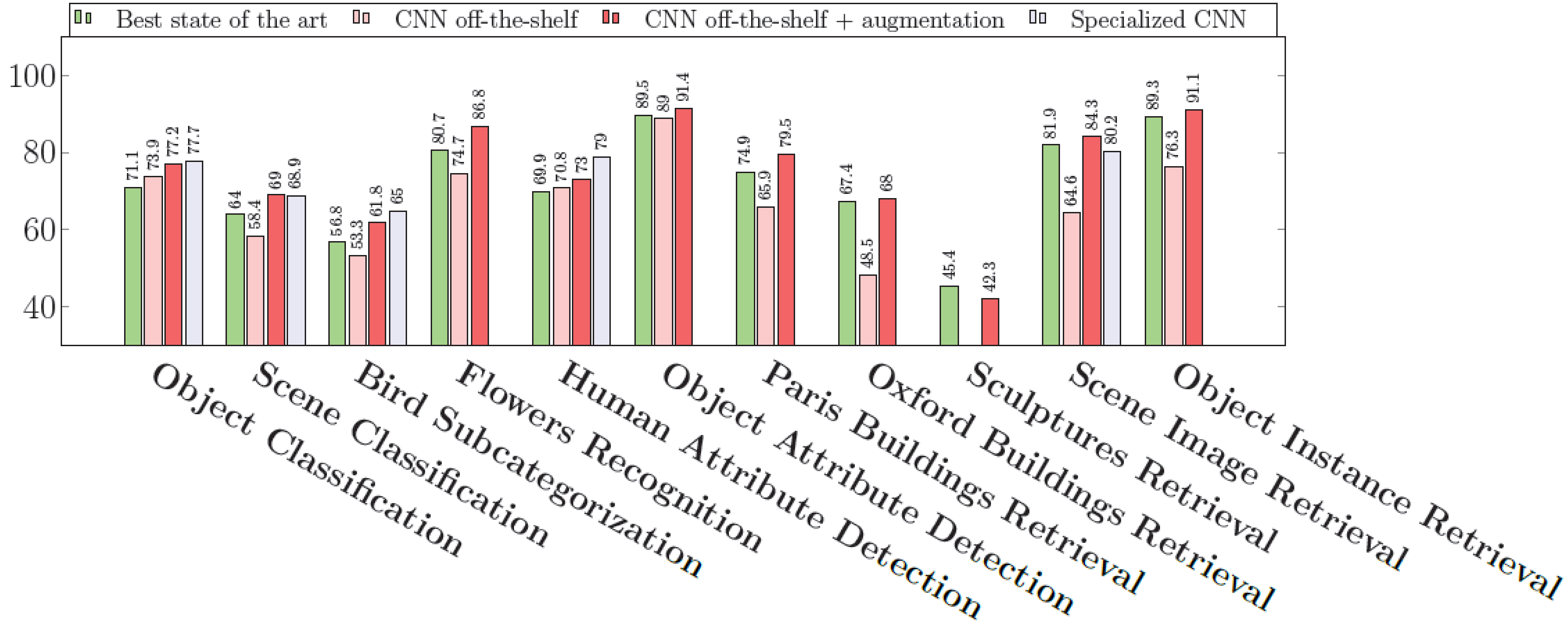
Transfer Learning in CV, three main scenarios

Training deep neural network from scratch is very difficult task (e.g. ImageNet, which contains 1.2 million images with 1000 categories). The three major Transfer Learning scenarios:

- Trained deep neural networks as feature extractors: remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest as a fixed feature extractor.
- Fine-tuning deep neural networks. Train the weights of the pre-trained network by continuing the training via backpropagation
- Use pre-trained models, e.g. Model Zoo https://modelzoo.co/

Source: http://cs231n.github.io/transfer-learning/

Transfer Learning in CV: CNNs as feature extractors



A. S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition //2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 23-28 June 2014, Columbus, USA, p. 512 – 519.

Transfer learning task for NLP?

Transfer learning task for NLP – Language Modelling?

Language model: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

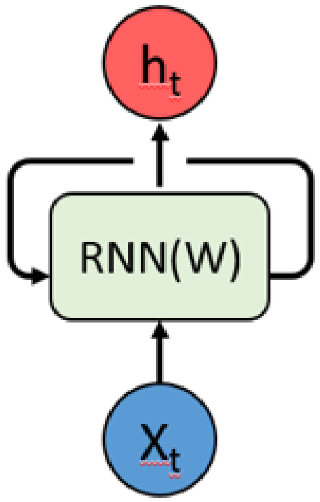
$$P(\boldsymbol{x}^{(t+1)} | \; \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|V|}\}$

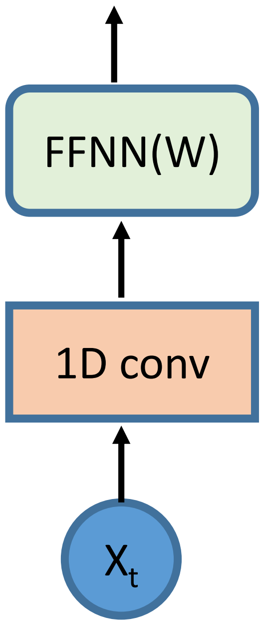
- ELMo ("Embeddings from Language Models") https://arxiv.org/pdf/1802.05365.pdf Deep LSTM language model: biLSTM layers with 4096 units and 512 dimension projections;
- ULM-Fit ("Universal Language Model Fine-tuning for Text Classification") https://arxiv.org/pdf/1801.06146.pdf - AWD-LSTM language model: embedding size of 400, 3 layers, 1150 hidden size;
- coming soon: BERT, OpenAI Transformer…

Dynamic Neural Network Models

Recurrent Neural Networks:
adding recurrent Connection



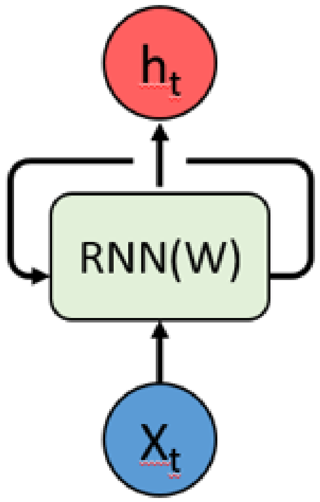
Convolutional Neural Networks 1D:
adding memory banks to input



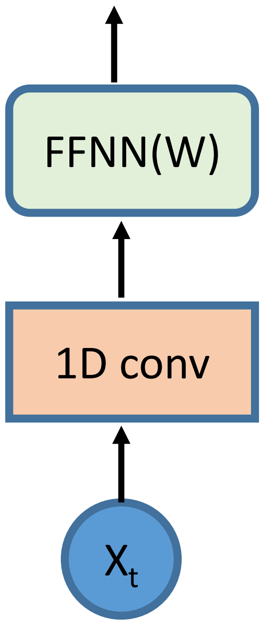
X_t - network's input for time t;
h - network's state for time t.

Dynamic Neural Network Models
= dynamized feedforward models

Recurrent Neural Networks:
adding recurrent Connection

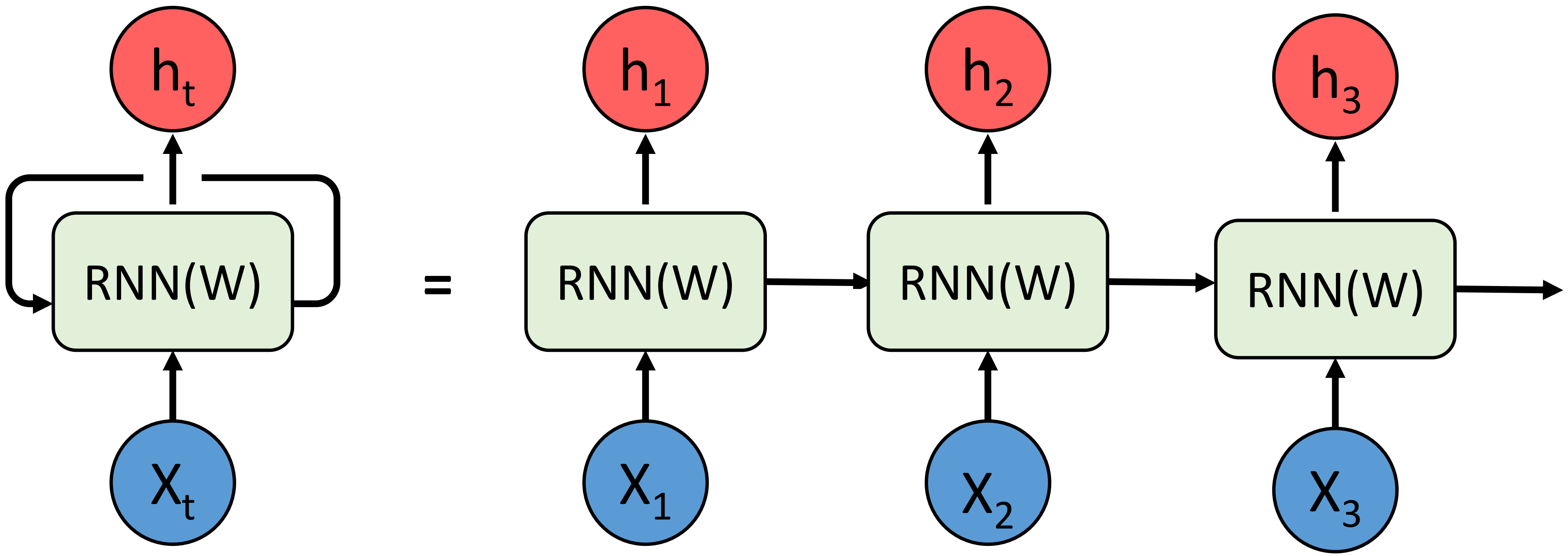


Convolutional Neural Networks 1D:
adding memory banks to input



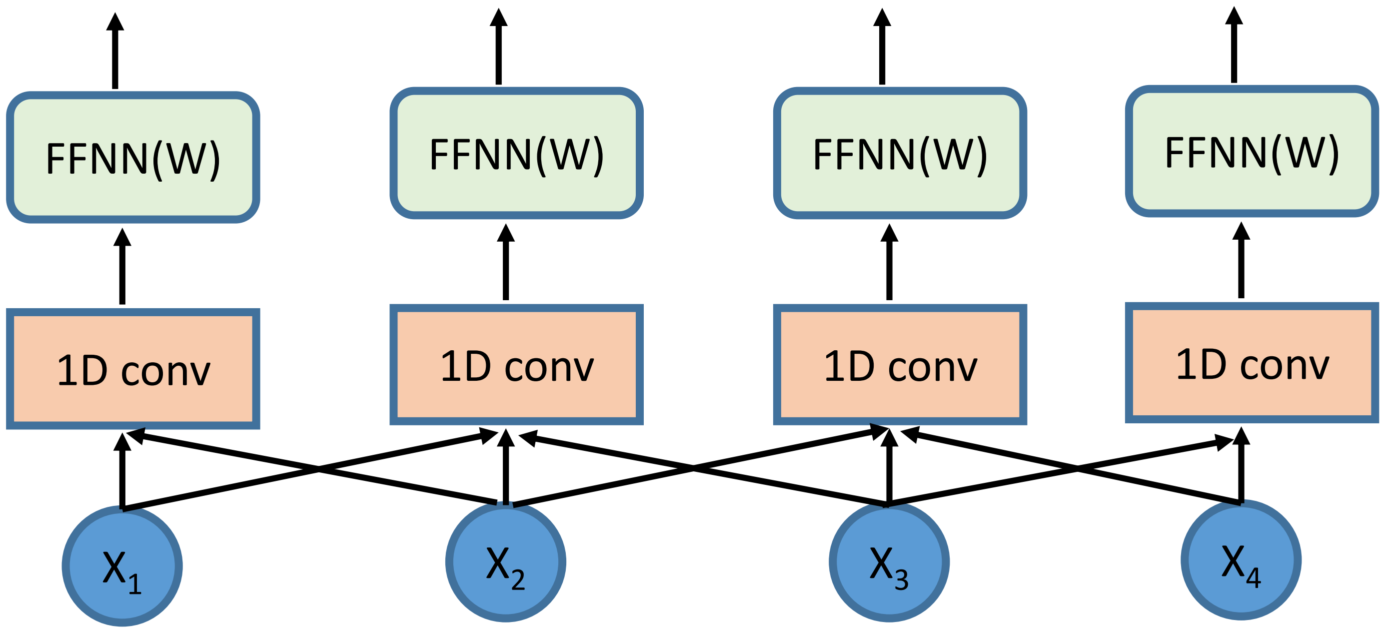
X_t - network's input for time t;
h - network's state for time t.

Recurrent Neural Networks



- sequential computation inhibits parallelization
- no explicit modeling of long and short range dependencies

1D Convolutional Neural Networks



- easy to parallelize!
- exploits local dependencies
- long-distance dependencies require many layers

Numerical Complexity

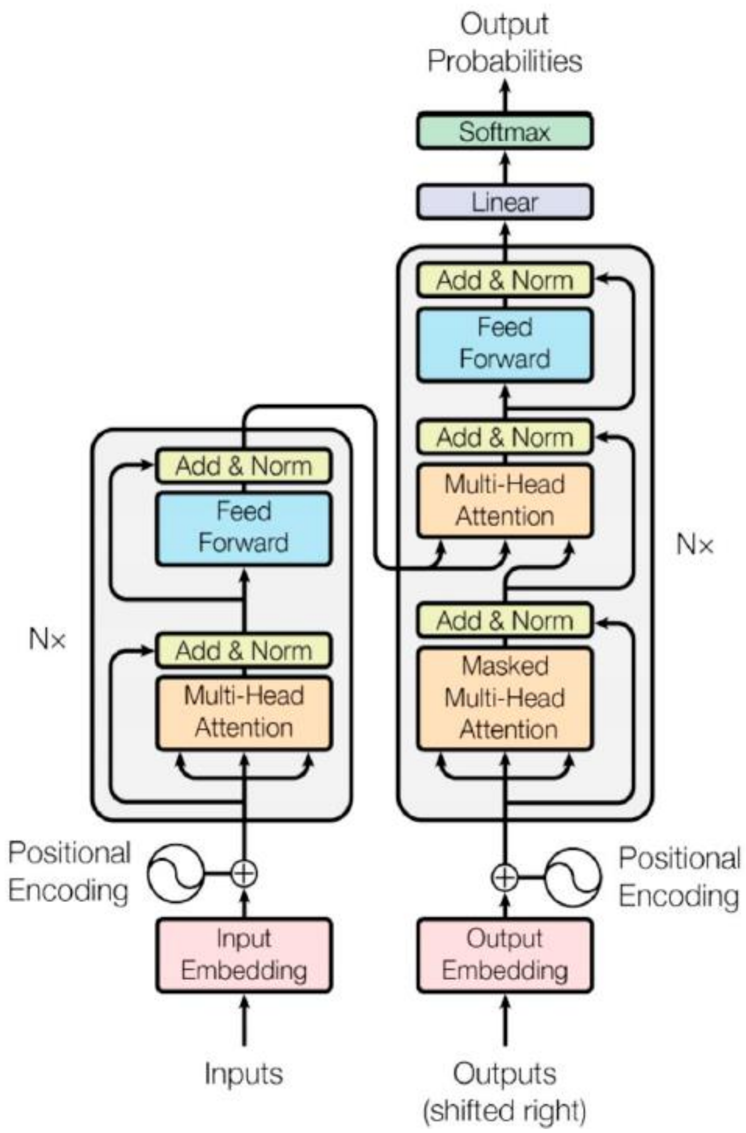
Model	**FLOPs**
RNN	$O(length \cdot dim^2)$
1D ConvNet	$O(length \cdot dim^2 \cdot K)$

Transformers

Transformers
= "Attention Is All You Need"

Transformers – model architecture

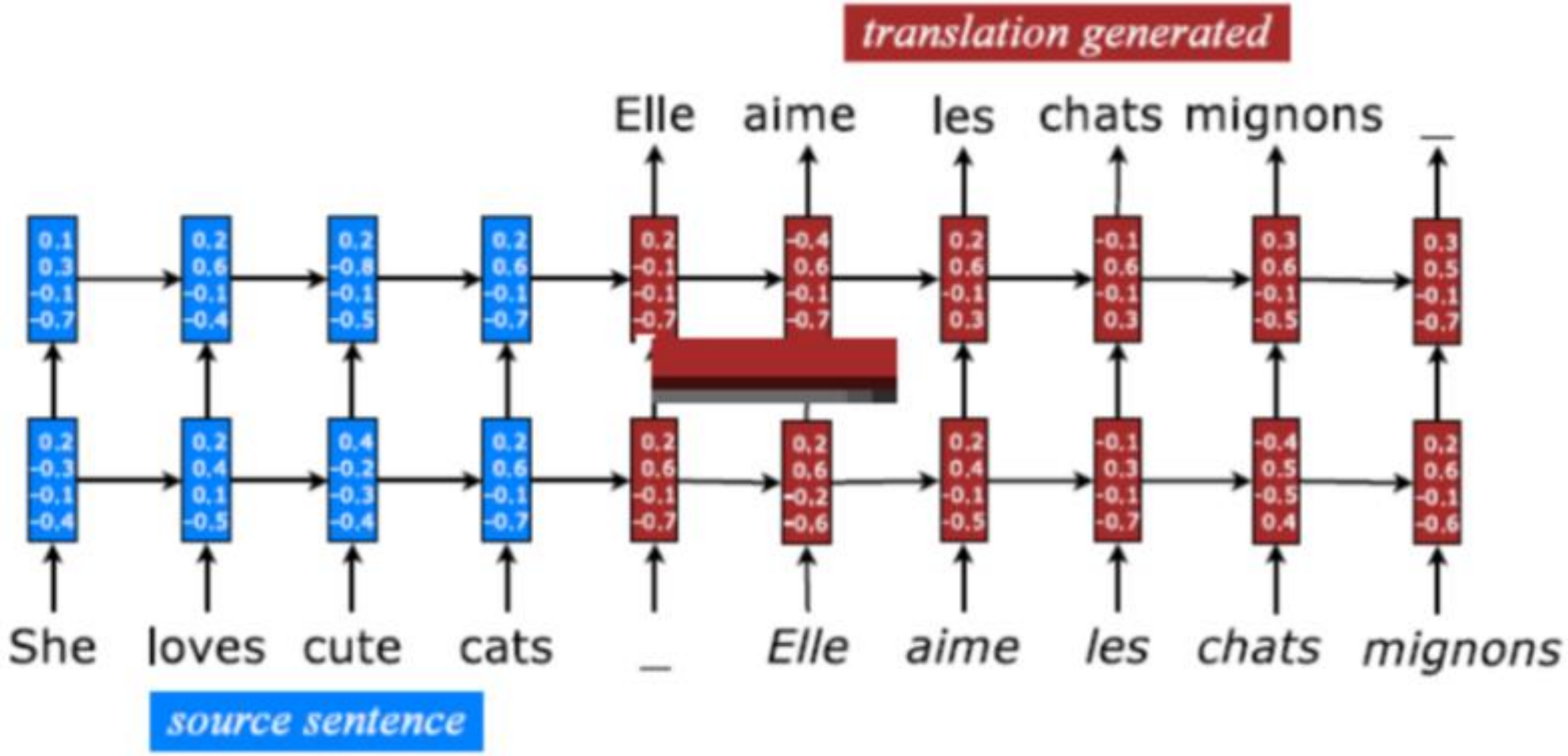
- encoder-decoder architecture
- 6 layers stacked one-by-one in encoder an decoder
- positional encoding + residuals
- multihead attention in encoder + *masked* multi-head attentions in decoder



Vaswani et. al., "Attention Is All You Need", NIPS 2017.

Recall
Neural Machine Translation

Seq2Seq Architecture for NMT



- 100 % ~~cotton~~ end-to-end learning
- better accuracy than for SMT (standard Google Translate since mid 2016)
- encoder-decoder architecture

https://github.com/tensorflow/nmt

Implementation details

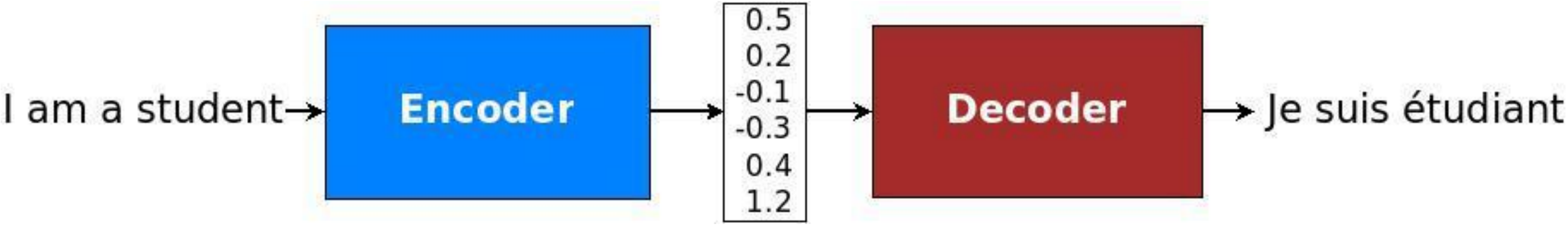
- deep LSTMs with 4 layers, 1000 cells at each layer and 1000 dimensional word embeddings (in original paper are random, but could be initialized by word2vec, GloVe, fasttext, etc.)
- input vocabulary 160,000 words, output vocabulary 80,000 words
- 384M parameters of which 64M are pure recurrent connections
- 32M for the “encoder” LSTM and 32M for the “decoder” LSTM)
- BLEU score on WMT’14 English-to-French:
  ensemble of 5 reversed LSTMs = 34.81 vs basiline SMT = 33.30

Sutskever, I., Vinyals, O., & Le, Q. V. Sequence to Sequence Learning with Neural Networks // Advances in Neural Information Processing Systems, 2014, (pp. 3104-3112).

RNN encoder-decoder

RNN

RNN



<u>Input:</u>

sequence of words

the last encoder’s hidden state

<u>Output:</u>

sequence of words

Sutskever, I., Vinyals, O., & Le, Q. V. Sequence to Sequence Learning with Neural Networks // Advances in Neural Information Processing Systems, 2014, (pp. 3104-3112).

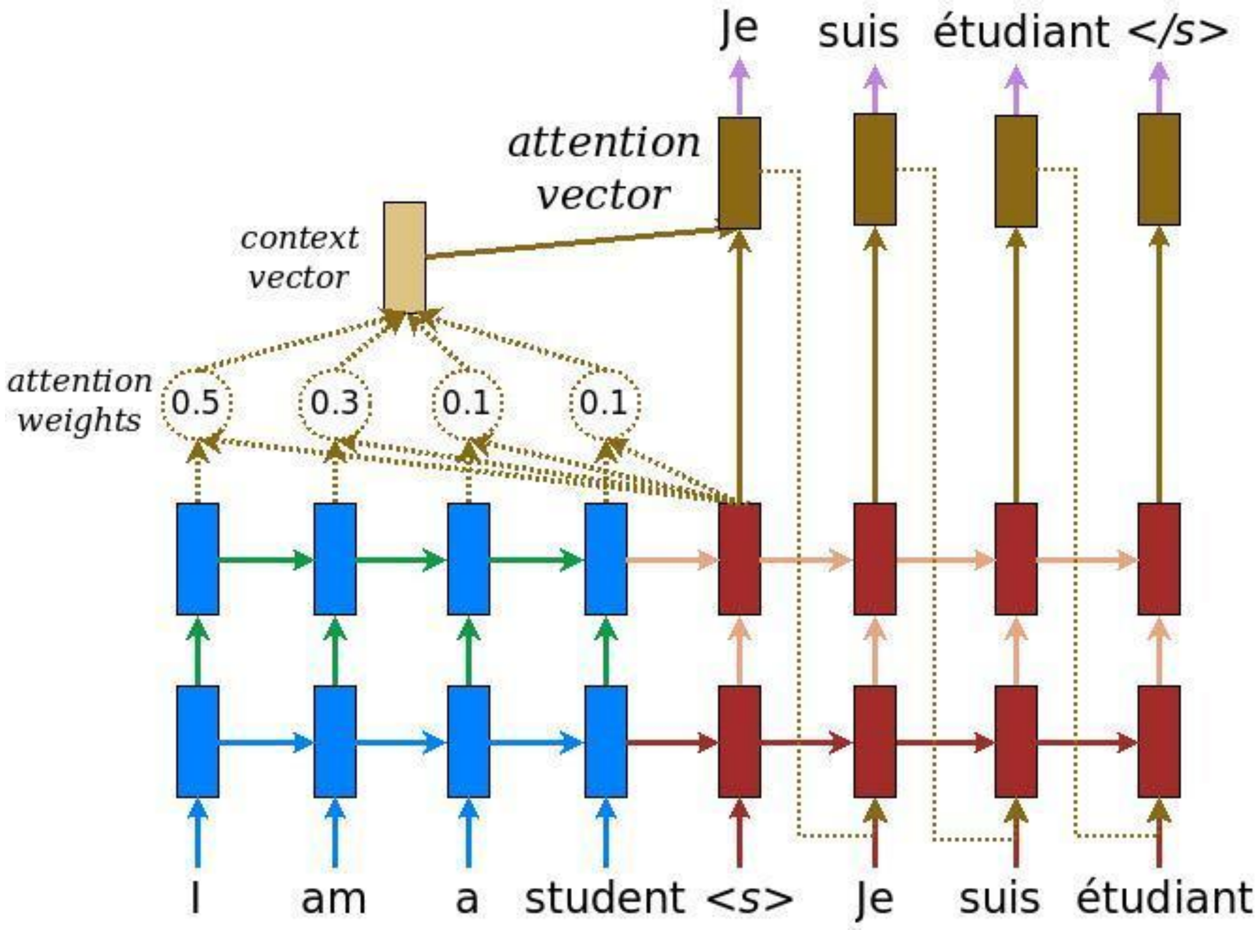
Attention mechanism

Attention mechanism

Problems with fixed-length encoded vectors (latent representations):

- input and output sequences may have different length;
- order of words in sequence may differ (not very important for French and even Chineese, but critical for Japanese, for example);
- complex relationships between input and output sequences: not necessarily one-to-one, may be many-to one, one-to-many, many-to-many.

Solution: attention mechanism



- We introduce attention mechanism, at each time step k we have a set of "importance weights" $\mathbf{a}(k)$ for the whole sequence at encoder.
- Decoder uses weighted sum of all hidden states of encoder at each time step instead the only last.

Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate // ICLR 2015
Luong M., Pham H., Manning C. Effective approaches to attention-based neural machine translation // arXiv:1508.04025.

Attention algorithm

1. We have source sequence $\boldsymbol{x} = [x_1, \ldots, x_n]$ and target output sequence $\boldsymbol{y} = [y_1, \ldots, y_m]$.
2. The decoder network has hidden state

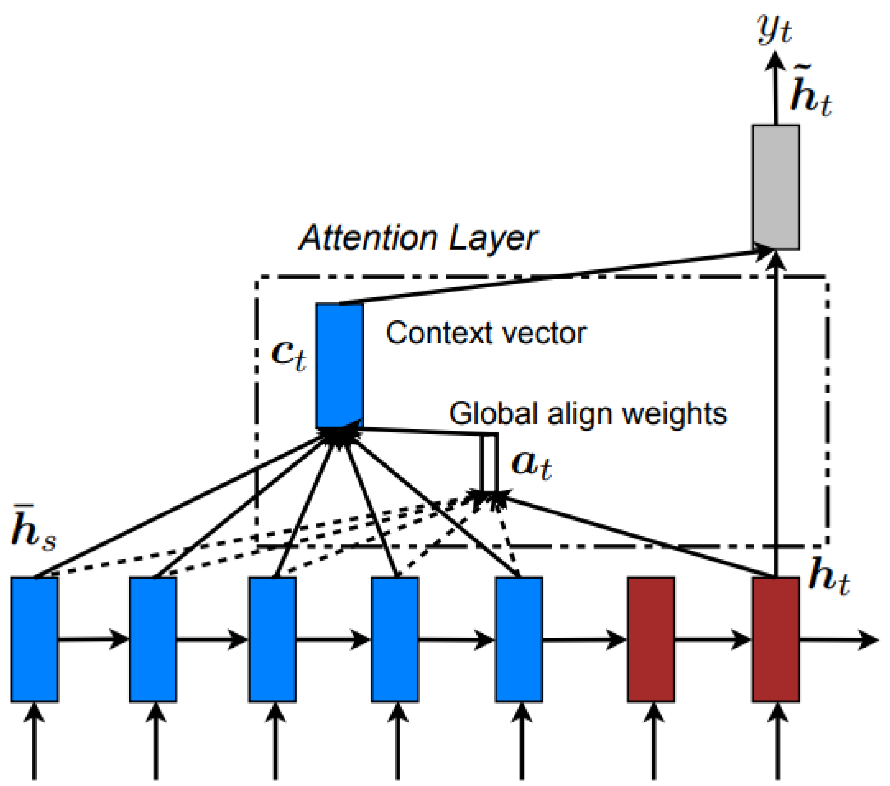
$$\boldsymbol{s_t} = f(\boldsymbol{s_{t-1}}, y_{t-1}, \boldsymbol{c_t})$$

3. We calculate:

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$ Context vector

$$\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}$$ Alignment score

$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$ Trainable alignment model



Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#definition

Attention algorithm

1. We have source sequence $\boldsymbol{x} = [x_1, \dots, x_n]$ and target output sequence $\boldsymbol{y} = [y_1, \dots, y_m]$.
2. The decoder network has hidden state
$$\boldsymbol{s_t} = f(\boldsymbol{s_{t-1}}, y_{t-1}, \boldsymbol{c_t})$$
3. We calculate:

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$

Context vector

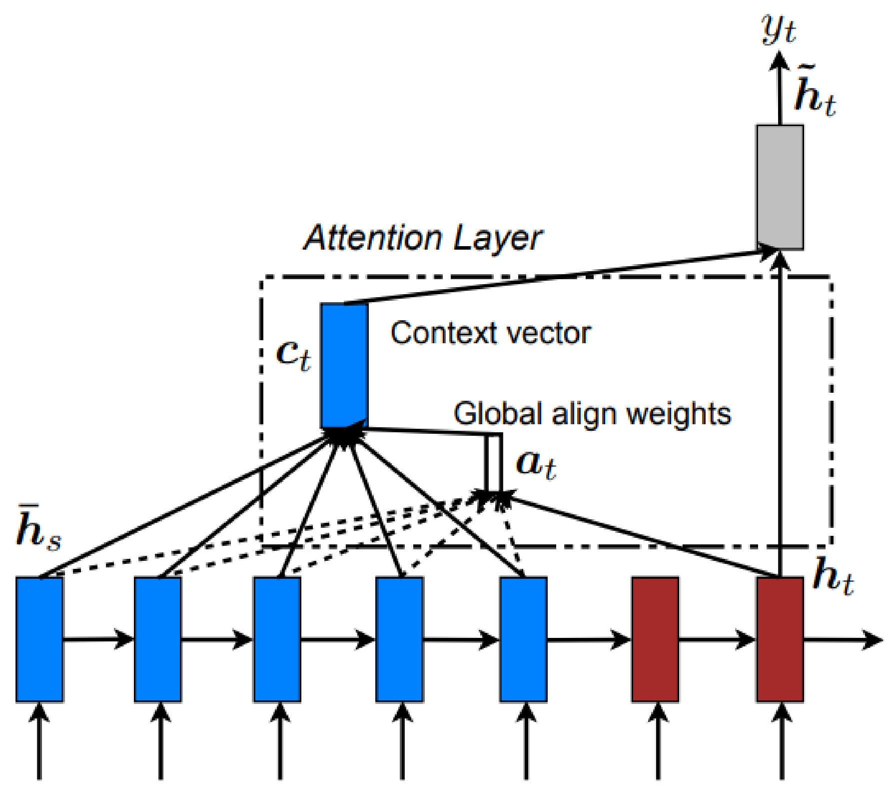
$$\alpha_{t,i} = \text{align}(y_t, x_i)$$

Alignment score

$$= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}$$

$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$

Trainable alignment model



Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#definition

Attention algorithm

1. We have source sequence $\boldsymbol{x} = [x_1, \dots, x_n]$ and target output sequence $\boldsymbol{y} = [y_1, \dots, y_m]$.
2. The decoder network has hidden state

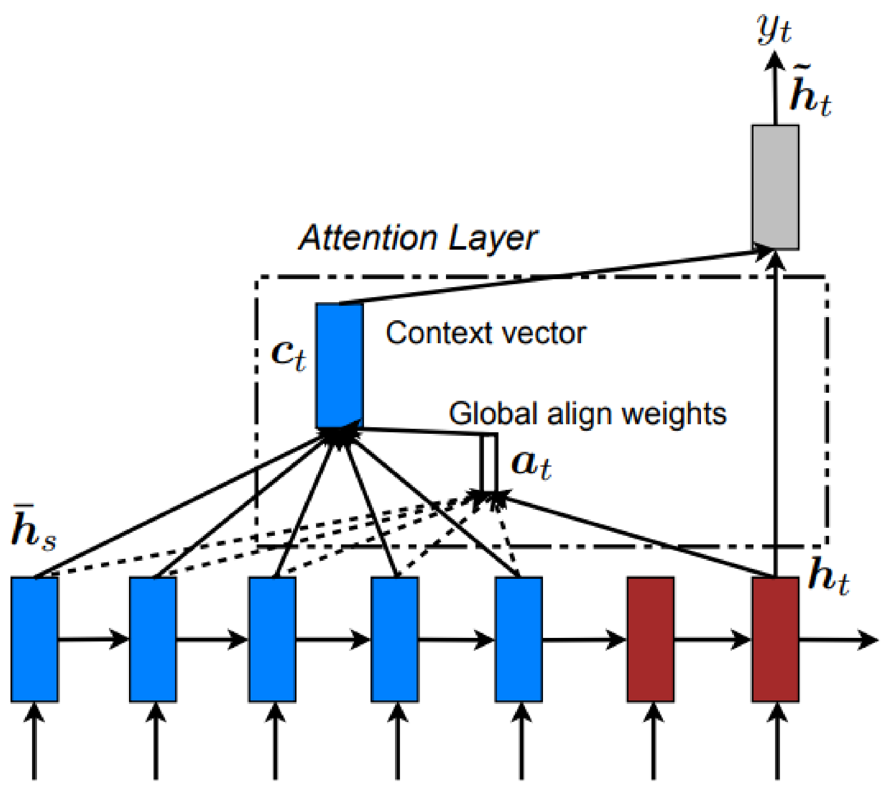
$$\boldsymbol{s_t} = f(\boldsymbol{s_{t-1}}, y_{t-1}, \boldsymbol{c_t})$$

3. We calculate:

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$ ← Context vector

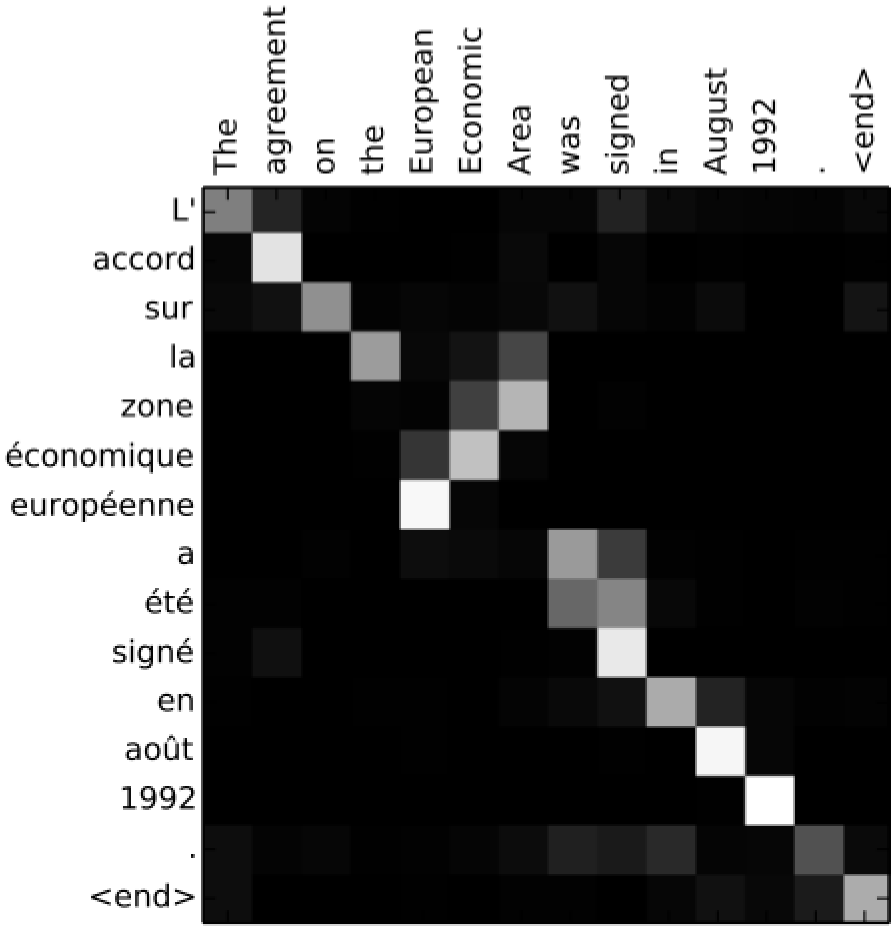
$$\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}$$ ← Alignment score

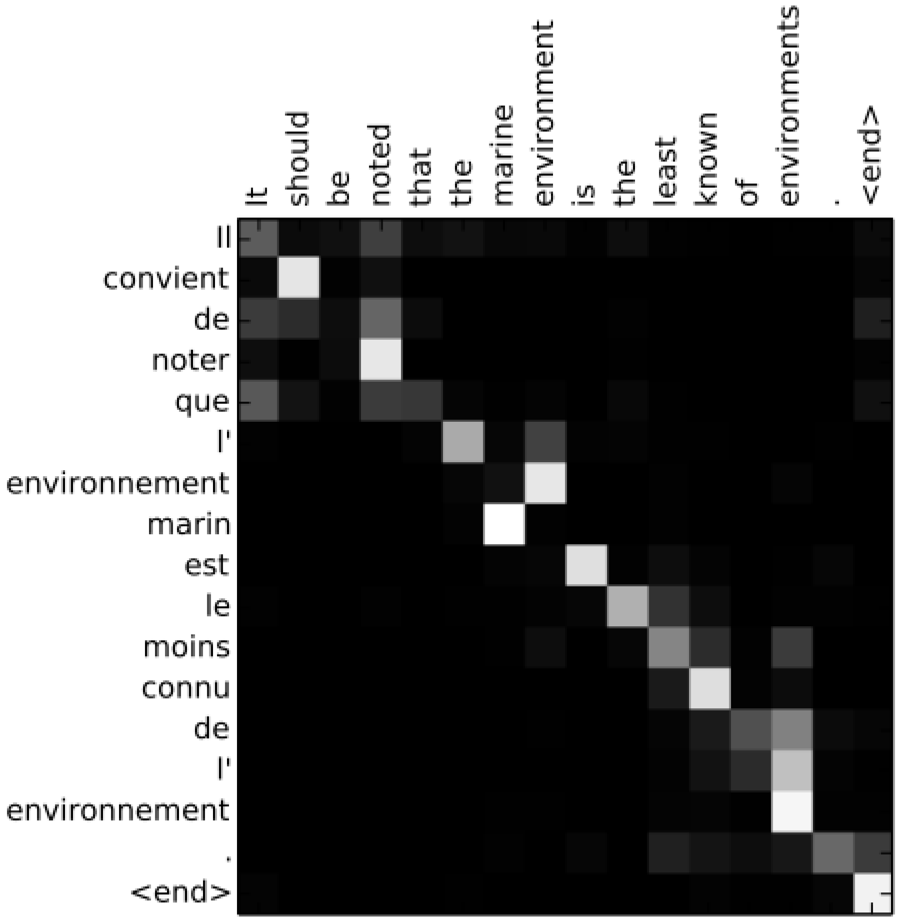
$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$ ← Trainable alignment model



Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#definition

Attention visualization





Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate // ICLR 2015

Family of Attentions (alignment models)

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

Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Attention algorithm – Query, Key, Value

source sequence $\boldsymbol{x} = [x_1, \dots, x_n]$ ~ **<Key, Value>**

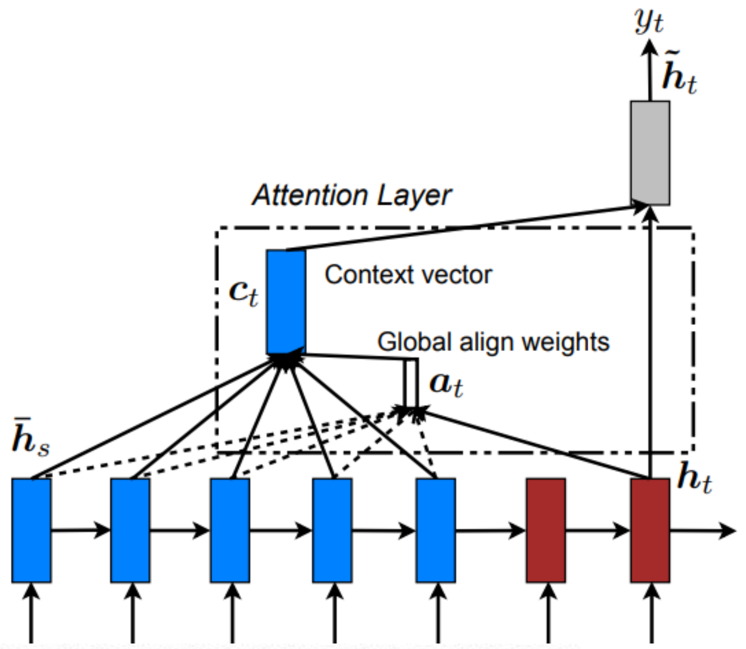
target sequence $\boldsymbol{y} = [y_1, \dots, y_m]$ ~ **<Query>**

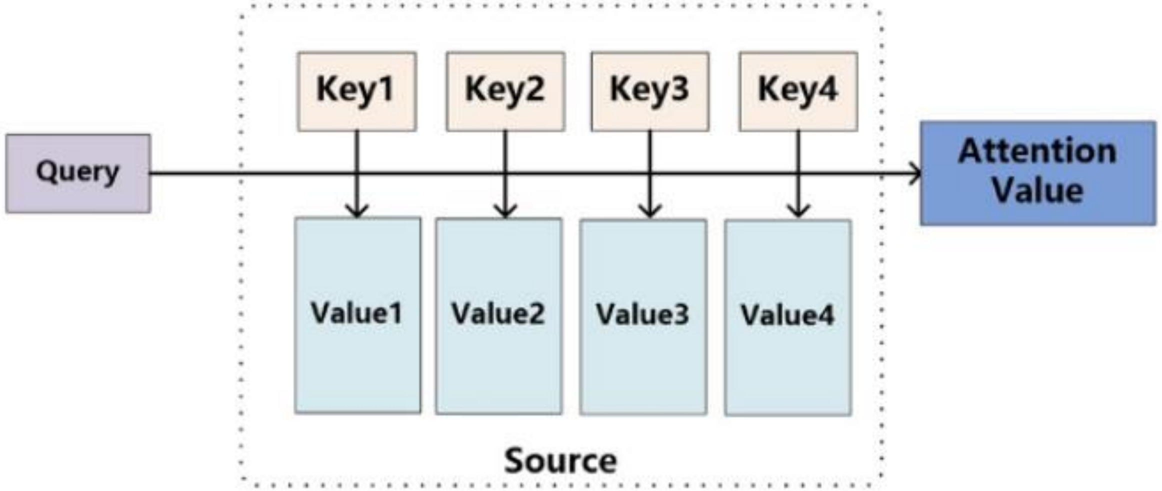
$\boldsymbol{h}_i$ - encoder's hidden state.

$$\boldsymbol{c}_t = \sum_{i=1}^{n} \alpha_{t,i}\, \boldsymbol{h}_i$$

Context vector

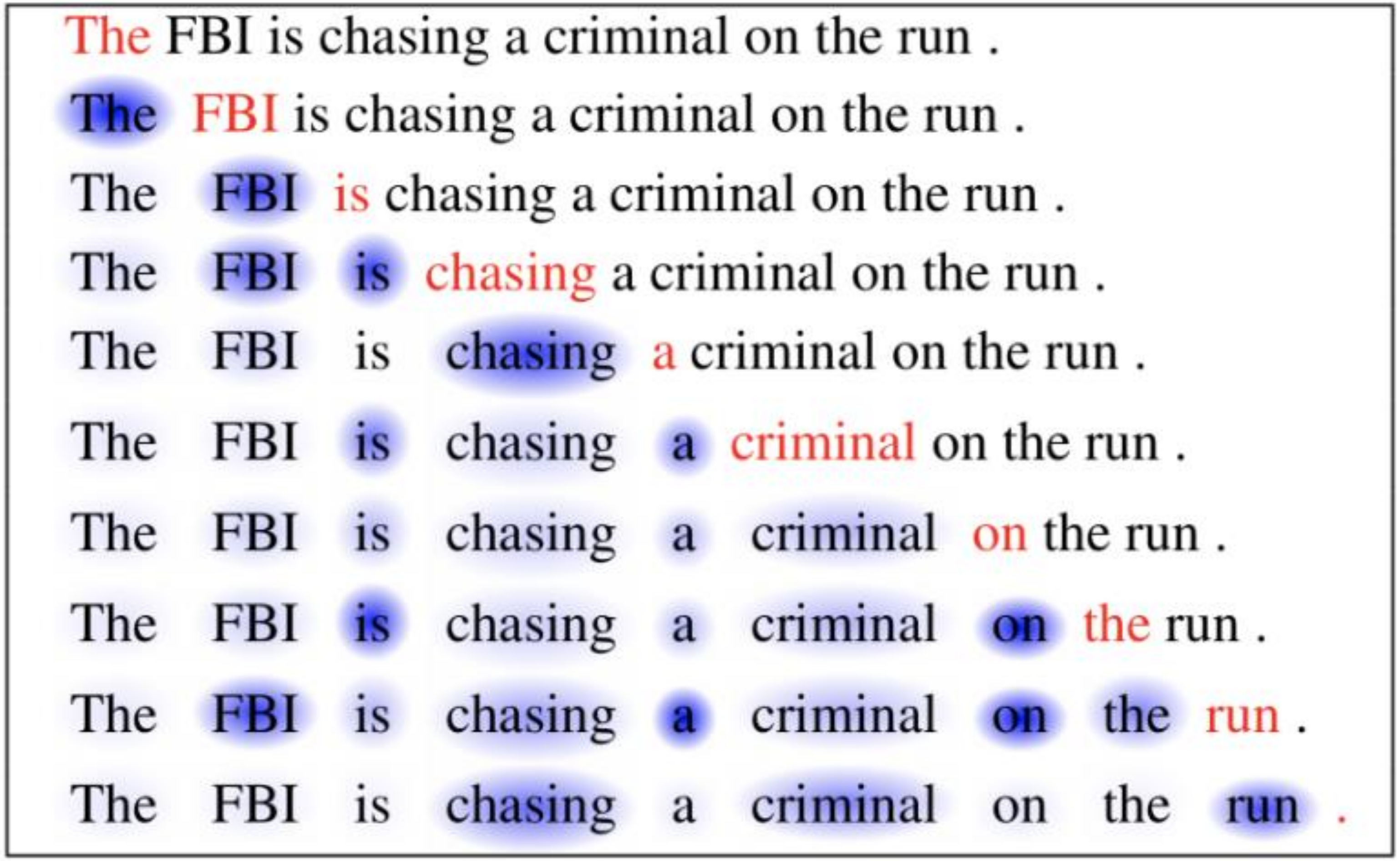
$$Attention(Query_t, Source) = \sum_{i=1}^{n} \alpha_{t,i}\, Value_i$$





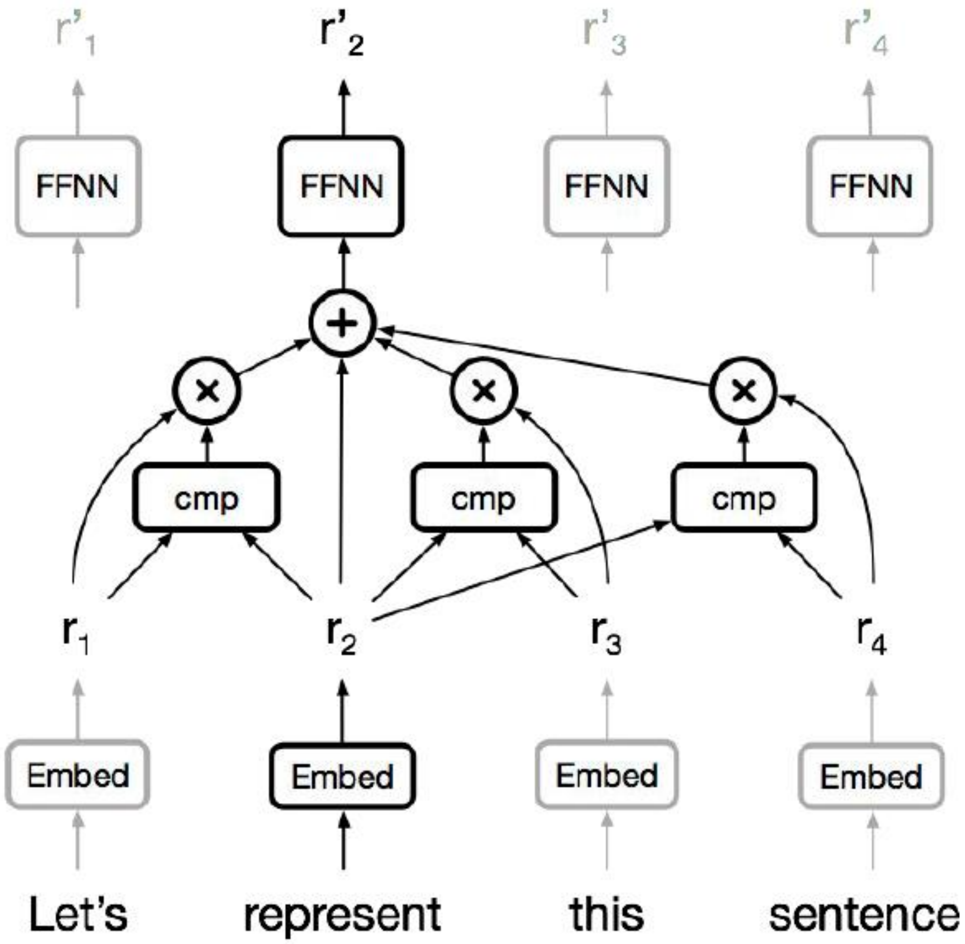
Source: https://medium.com/@bgg/seq2seq-pay-attention-to-self-attention-part-2-cf81bf32c73d

Self-attention



Jianpeng Cheng, Li Dong, Mirella Lapata. Long Short-Term Memory-Networks for Machine Reading // EMNLP 2016

Self-attention



Source: Stanford course CS224d, Deep Learning for Natural Language Processing, Lecture 9.

Numerical Complexity

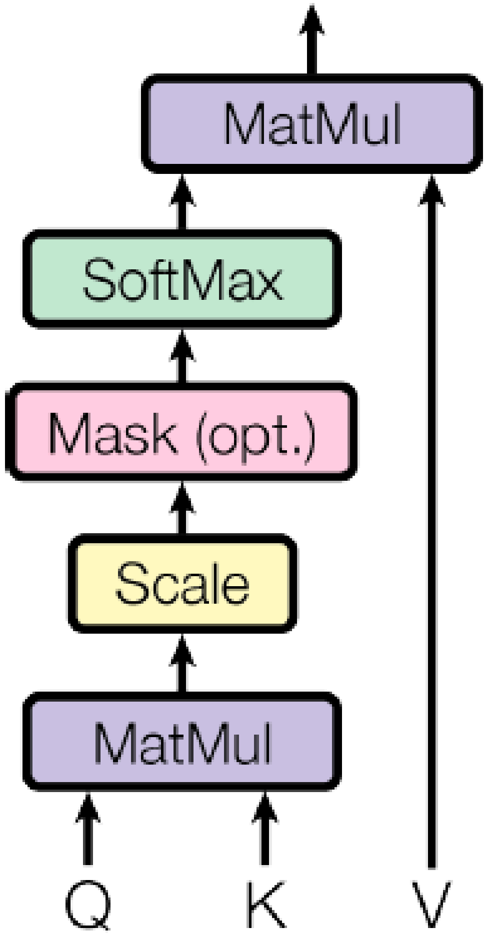
Model	FLOPs
RNN	$O(length \cdot dim^2)$
1D ConvNet	$O(length \cdot dim^2 \cdot K)$
Self-attention	$O(length^2 \cdot dim)$

Numerical Complexity

Model	FLOPs
RNN	$O(length \cdot dim^2)$
1D ConvNet	$O(length \cdot dim^2 \cdot K)$
Self-attention	$O(length^2 \cdot dim)$

length ~ 100, dim ~ 1000

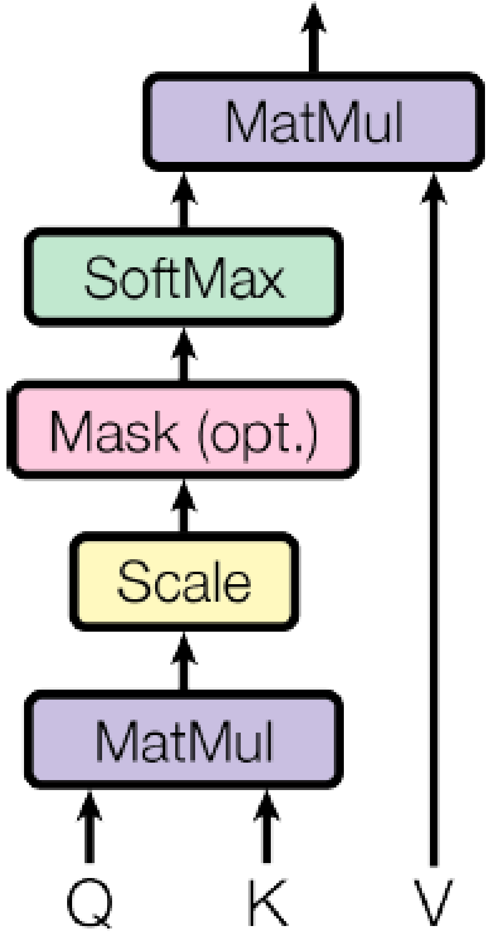
Scaled Dot-Product Attention



$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Query Key Value

Scaled Dot-Product Attention

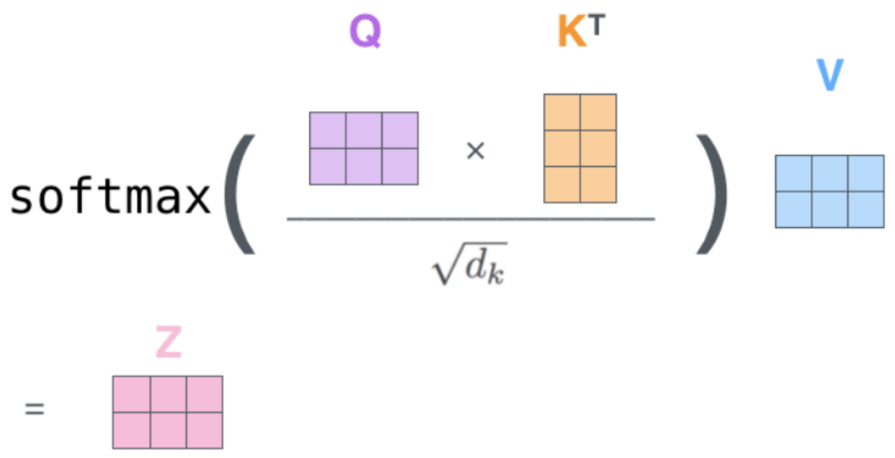


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Query

Key = Value

In matrix form:



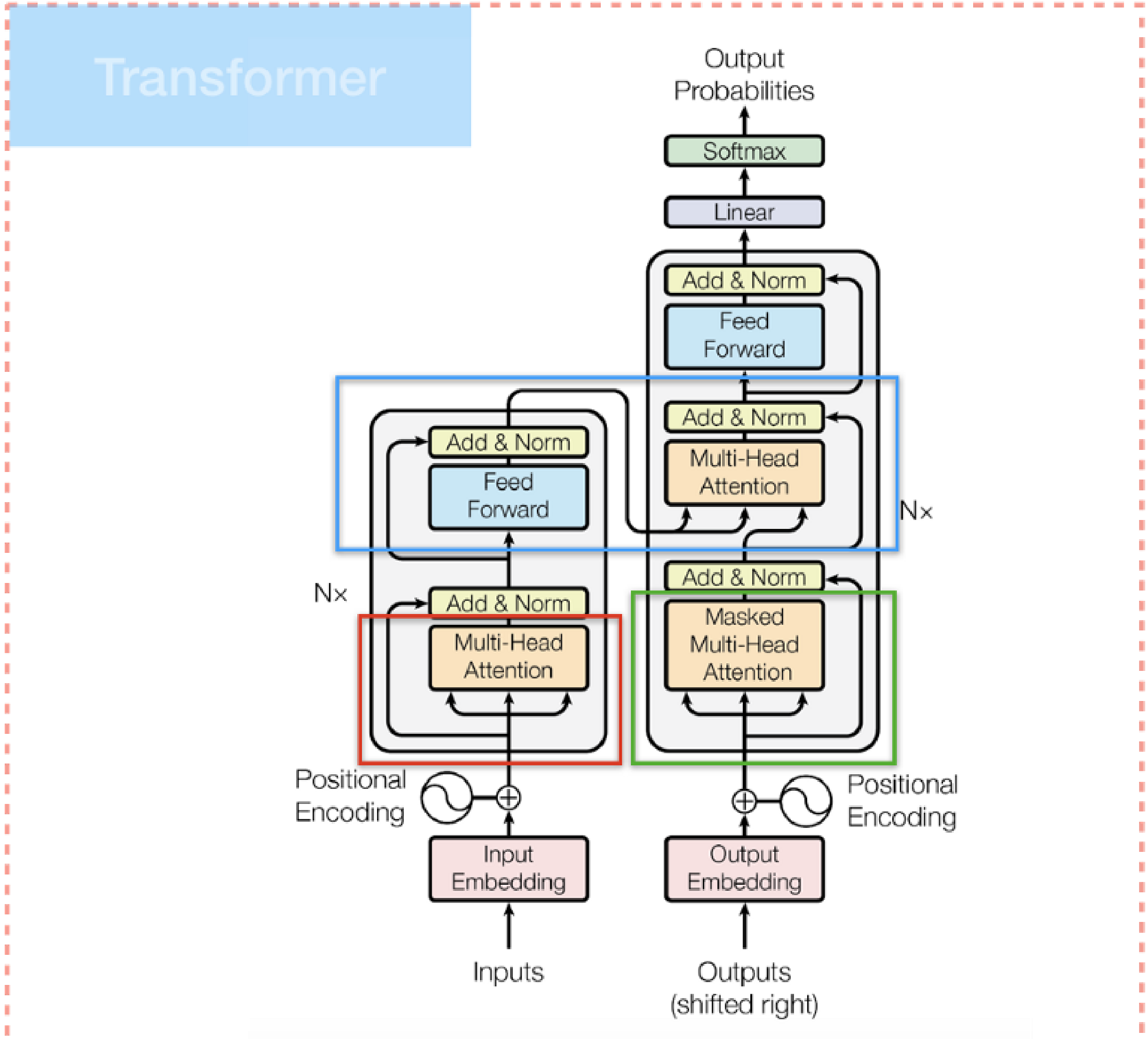


Figure 1: The Transformer - model architecture.

encoder self attention

1. Multi-head Attention
2. Query=Key=Value

decoder self attention

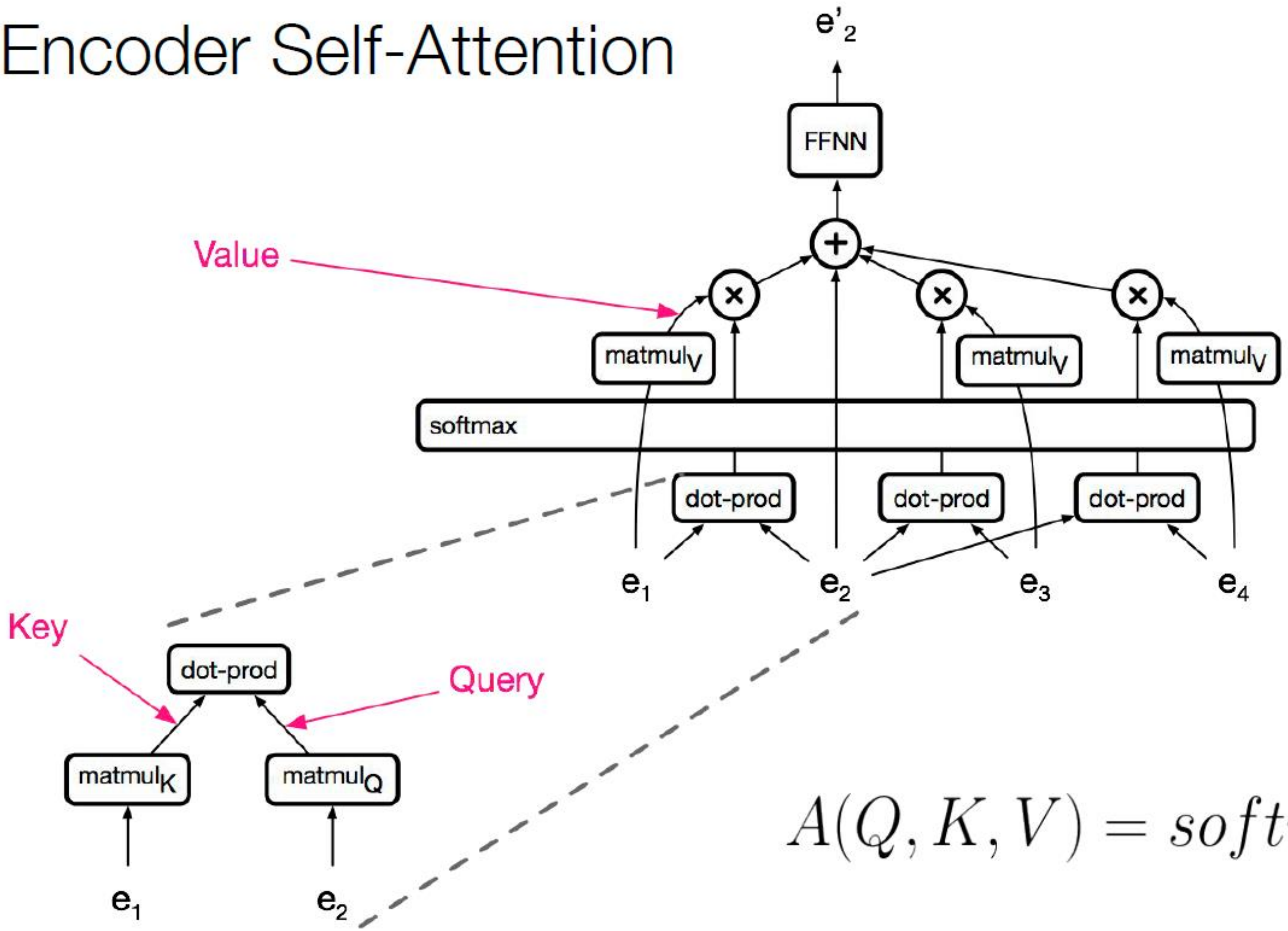
1. Masked Multi-head Attention
2. Query=Key=Value

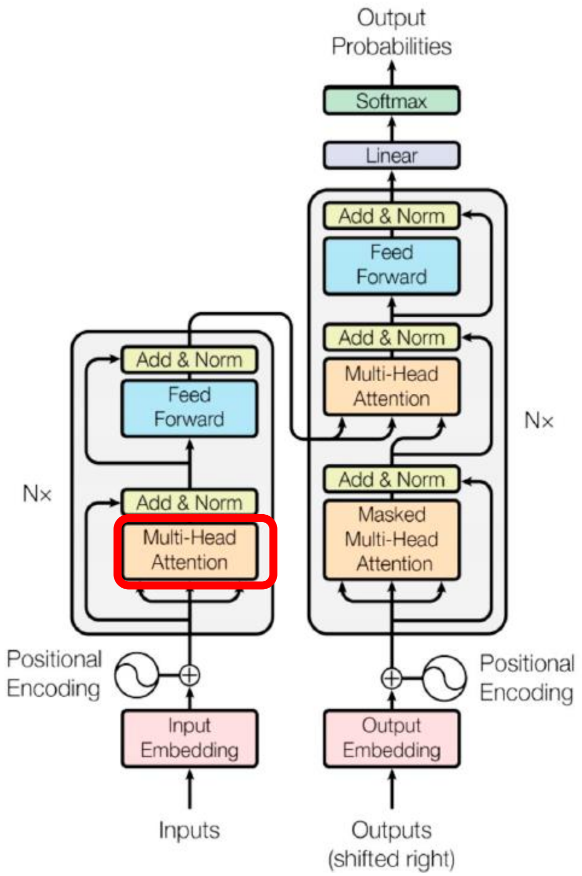
encoder-decoder attention

1. Multi-head Attention
2. Encoder Self attention=Key=Value
3. Decoder Self attention=Query

Source: https://medium.com/@bgg/seq2seq-pay-attention-to-self-attention-part-2-cf81bf32c73d

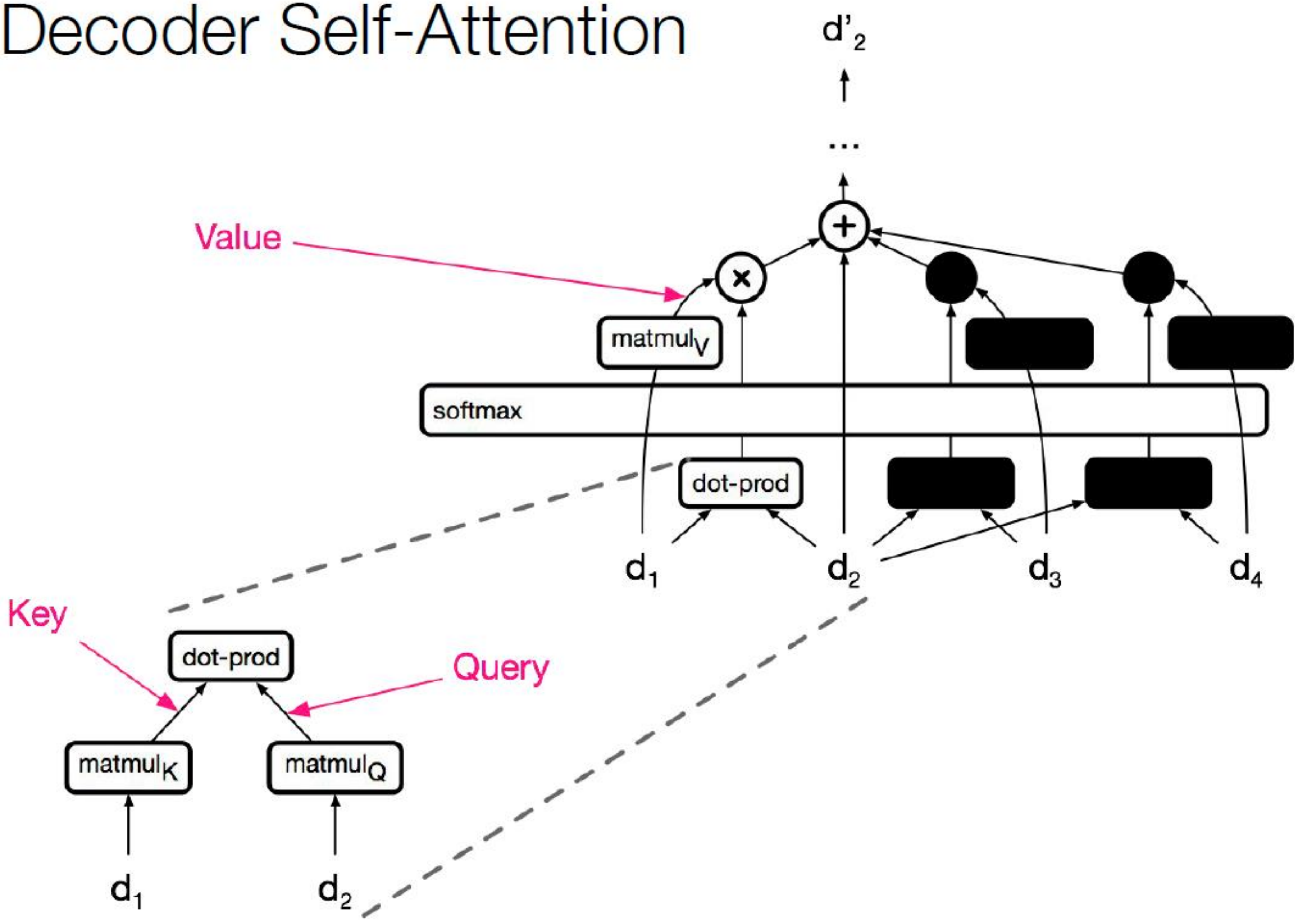
Encoder Self-Attention

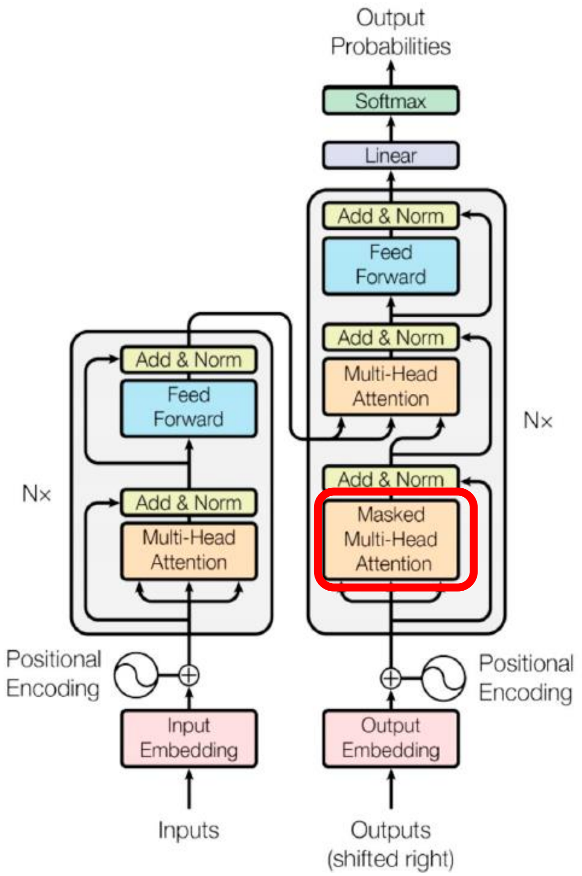




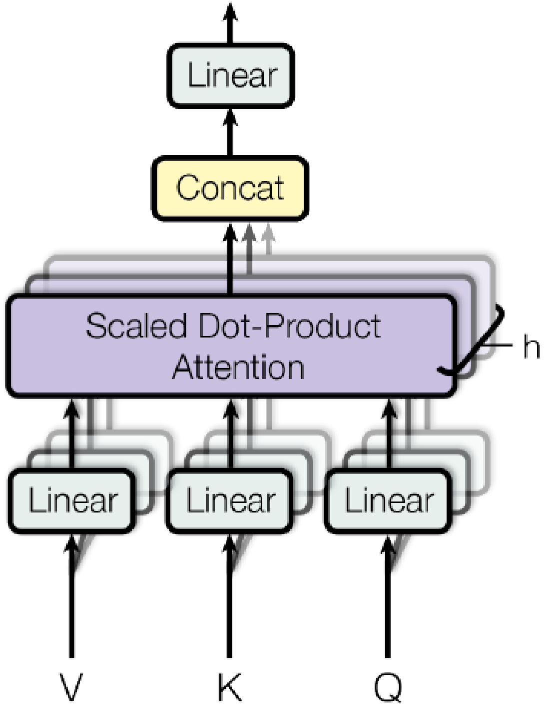
$$A(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Decoder Self-Attention





Multi-Head Attention



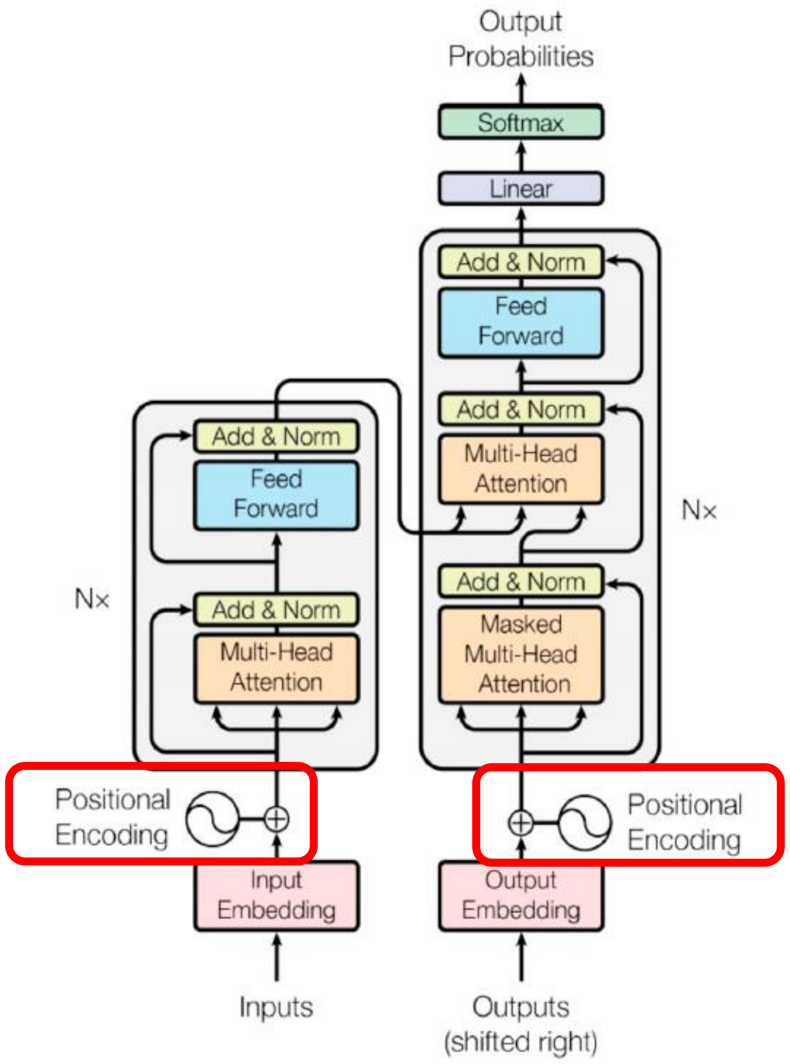
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$

$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Positional Encodings

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Goal: add to input embeddings information about the place and order of inputs

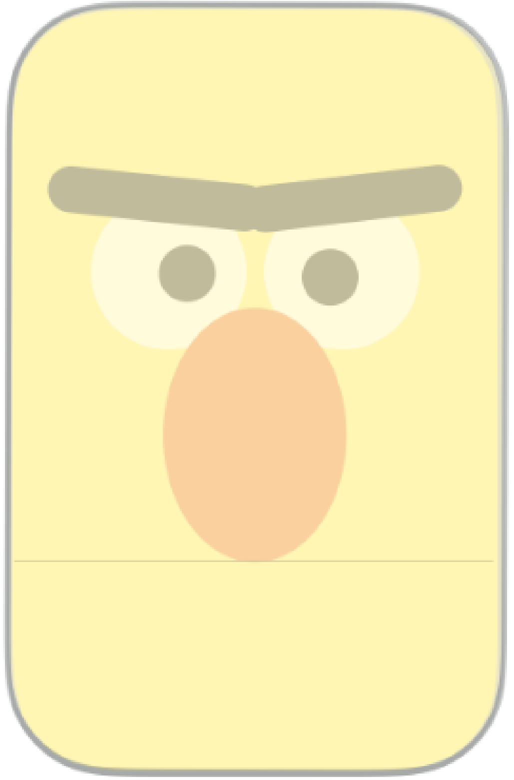


BLEU scores on English-to-German and English-to-French news test2014

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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

BERT

BERT



- BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers
- pre-trained BERT representations can be fine-tuned with just one additional output layer to create SOTA models for a wide range of tasks

Pre-training Data

- BooksCorpus (800M words) (Zhu et al., 2015)
- English Wikipedia (2,500M words).

- use a document-level corpus (for getting sentence context)
- extract only the text passages and ignore lists, tables, and headers

	Training Compute + Time	Usage Compute
$BERT_{BASE}$	4 Cloud TPUs, 4 days	1 GPU
$BERT_{LARGE}$	16 Cloud TPUs, 4 days	1 TPU

Setup details

- BERT's model architecture is a multi-layer bidirectional Transformer encoder.
- Parameters: the number of layers (i.e., Transformer blocks) as **L**, the hidden size as **H**, and the number of self-attention heads as **A.**

BERT_BASE: L=12, H=768, A=12
Total Parameters=110M

BERT_LARGE: L=24, H=1024, A=16
Total Parameters=340M

Pre-training task: Masked LM

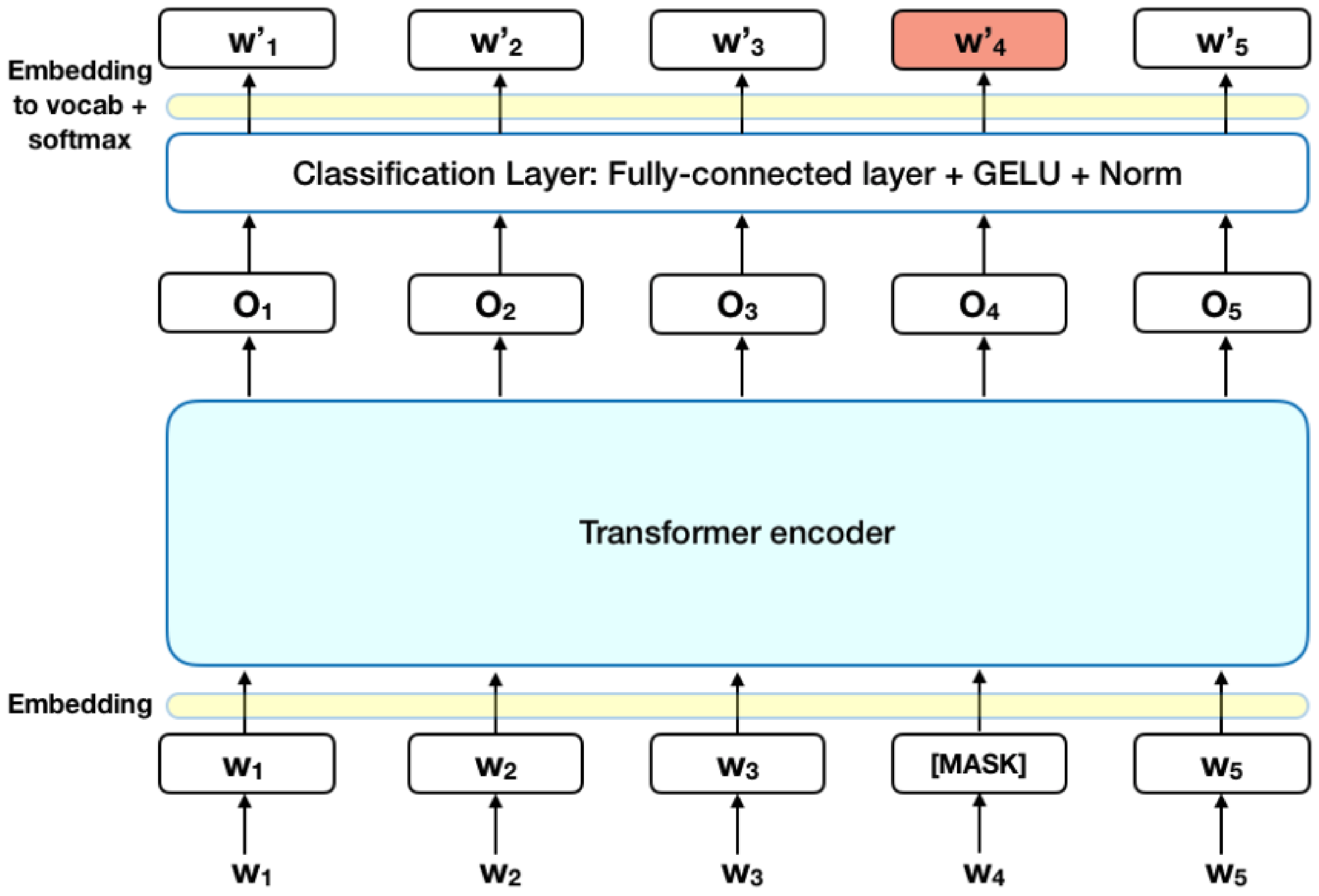
- Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token.

- 80% of the time: Replace the word with the [MASK] token,
  e.g., my dog is hairy → my dog is [MASK]

- 10% of the time: Replace the word with a random word,
  e.g., my dog is hairy → my dog is apple

- 10% of the time: Keep the word unchanged,
  e.g., my dog is hairy → my dog is hairy

Pre-training task: Masked LM



Pre-training task: Next Sentence Prediction

Input = `[CLS] the man went to [MASK] store [SEP]`

`he bought a gallon [MASK] milk [SEP]`

Label = `IsNext`

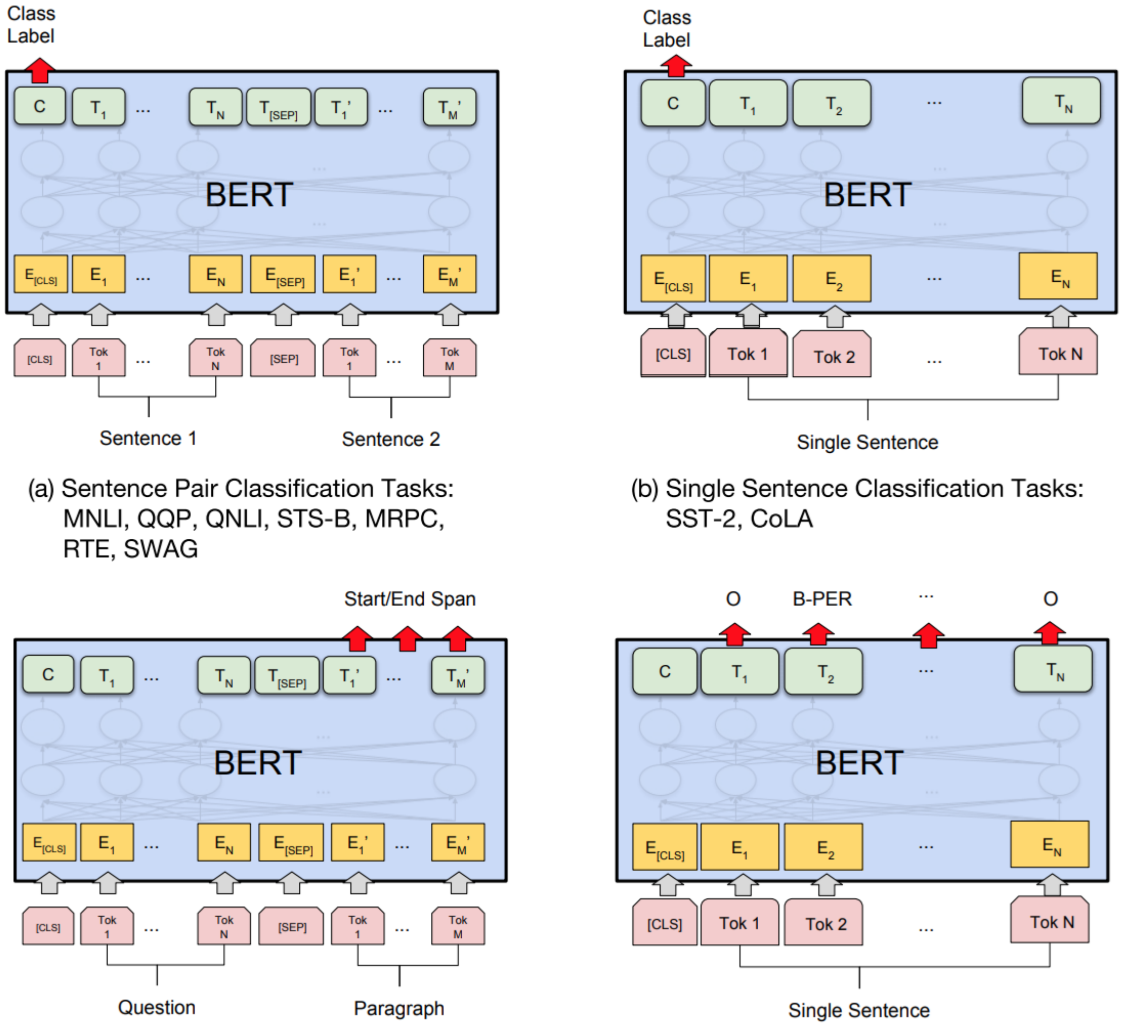
Input = `[CLS] the man [MASK] to the store [SEP]`

`penguin [MASK] are flight ##less birds [SEP]`

Label = `NotNext`

Model achieves 97%-98% accuracy on this task

Fine-tuning



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

- Use the final hidden state (which corresponds to [CLS]) as sentence representation

- Batch size: 16, 32
- Learning rate (Adam): 5e-5, 3e-5, 2e-5
- Number of epochs: 3, 4

General Language Understanding Evaluation (GLUE)

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); $\text{BERT}_{\text{BASE}}$ = (L=12, H=768, A=12); $\text{BERT}_{\text{LARGE}}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.

Standford Question Answering Dataset (SQuAD)

- **Input Question:**

  `Where do water droplets collide with ice crystals to form precipitation?`

- **Input Paragraph:**

  `... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. ...`

- **Output Answer:**

  `within a cloud`

System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Named Entity Recognition (NER)

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

Table 3: CoNLL-2003 Named Entity Recognition results. The hyperparameters were selected using the Dev set, and the reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

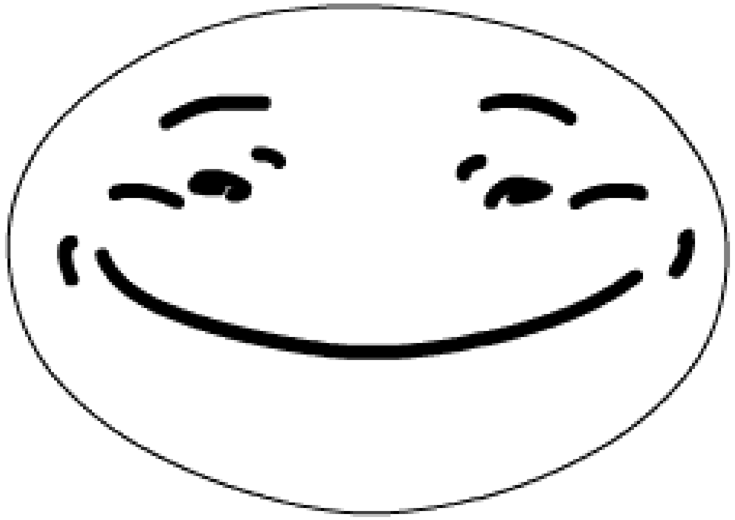
References

1. С. Николенко, А. Кадурин, Е. Архангельская. “Глубокое обучение: погружение в мир нейронных сетей”, 2018.
2. Stanford course CS224d, Deep Learning for Natural Language Processing.
3. https://github.com/tensorflow/nmt
4. Sebastian Ruder. NLP's ImageNet moment has arrived http://ruder.io/nlp-imagenet/
5. Attention? Attention. Lilian Weng blog. https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
6. Kostia Omelianchuk, BERT tutorial // Grammarly’s blog

Links

1. TransferNLP lib for Pytorch https://github.com/feedly/transfer-nlp
2. Google's Tensor2Tensor package(Tensorflow) https://github.com/tensorflow/tensor2tensor
3. The Annotated Transformer – Harward NLP http://nlp.seas.harvard.edu/2018/04/03/attention.html
4. Pytorch-pretrained-BERT (huggingface) https://github.com/huggingface/pytorch-pretrained-BERT
5. AllenNLP for PyTorch (BERT, ELMo, etc.) https://allennlp.org/

Thank you for attention!



artem.chernodub@grammarly.com