Models of sequential data

20th February, 2024

Alexey Zaytsev,

Assistant professor, Skoltech

Skoltech

Common ways for classic ML application for time series data

1. Take input data including history for the target variable
2. Add differences, combinations, rolling means, medians, etc.
3. Add one-hot-encoding for important categorical features (day of week, holiday or not)

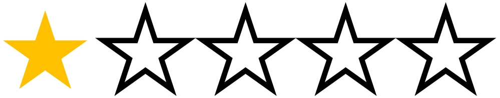
- Now we have input features for all points

- Let's apply our favorite ML regression algorithm

dmlc XGBoost

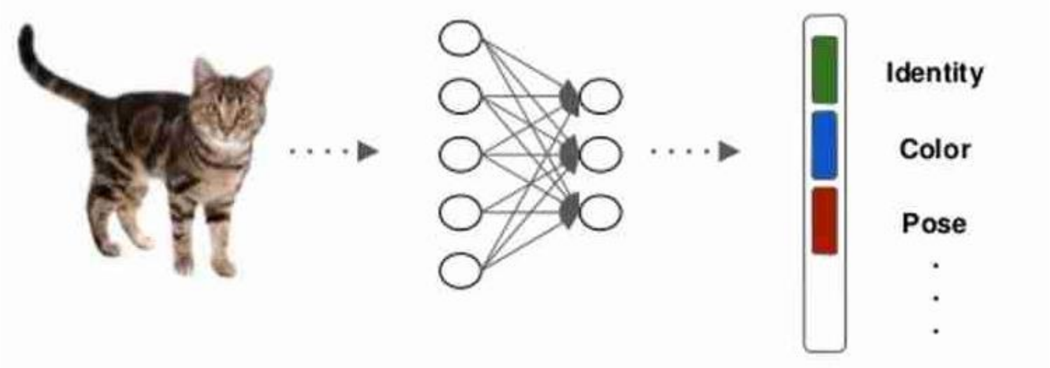
Skoltech

Deep Learning problems with sequential data: we need representations

	Input		Output
Speech recognition		→	“The quick brown fox jumped over the lazy dog.”
Sentiment classification	“There is nothing to like in this movie.”	→	
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running

Adopted from Andrew Ng

Semi-structured data processing: why do we need NNs?



Skoltech

Textbook example: next word prediction

The most complicated and difficult part of it was only just beginning.

Adopted from Ava Soleimany, MIT

Textbook example: next word prediction

<u>Idea 1:</u> use previous word(s)

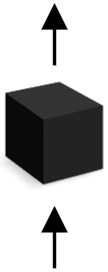
<u>Problem 1:</u> long-term dependencies

"**France** is where I grew up, but I now live in Boston. I speak fluent ___."

The most complicated and difficult part of it was only just beginning.

Feature representation: [0, 0, 0, 1, 0, 0]

Output $\hat{y}^t$



Input x^t

Adopted from Ava Soleimany, MIT

Textbook example: next word prediction

Idea 1: use previous word(s)
Idea 2: use bag of words model

Problem 1: long-term dependencies

The most complicated and difficult part of it was only just beginning.

Feature representation: [0, 3, 0, 2, 0, 0]

Bag of words: number of occurrences of each word

Skoltech

Adopted from Ava Soleimany, MIT

Textbook example: next word prediction

Idea 1: use previous word(s)
Idea 2: use bag of words model

Problem 1: long-term dependencies
Problem 2: order preservation

The food was good, not bad at all.

vs.

The food was bad, not good at all.

The most complicated and difficult part of it was only just beginning.

Feature representation: [0, 3, 0, 2, 0, 0]

Bag of words: number of occurrences of each word
(see also TF-IDF features)

Skoltech

Adopted from Ava Soleimany, MIT

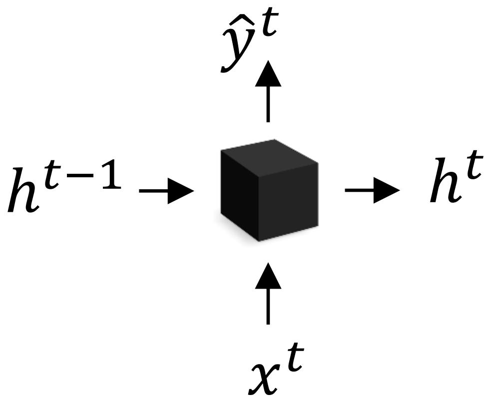
Model Design Criteria

1. Variable-length sequences processing

2. Long-term memory

3. Maintain order information

4. Natural preprocessing



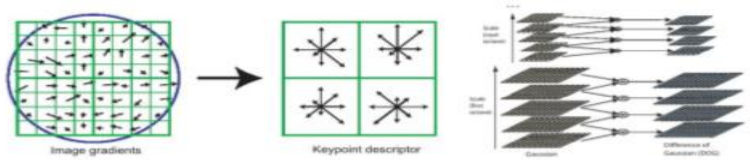
Recurrent Neural Networks are the solution!

Adopted from Ava Soleimany, MIT

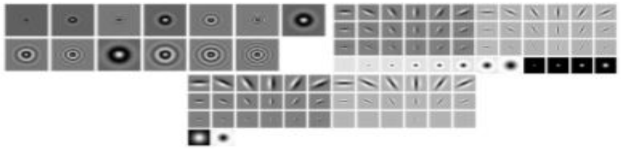
Sequence processing with classic ML models

1. Variable-length sequences processing — **YES (if one to one)**
2. Long-term memory — **NO**
3. Maintain order information — **NO**
4. Natural preprocessing — **a kind of**

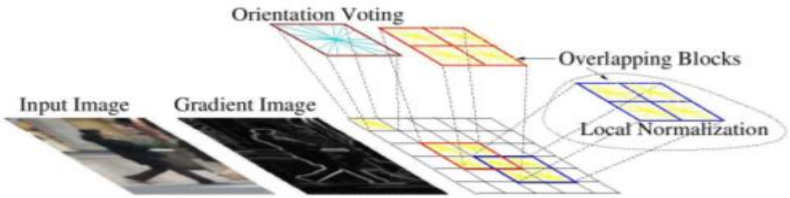
An art of feature construction



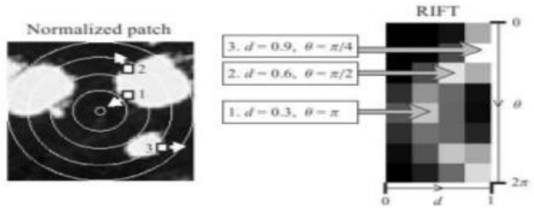
SIFT



Textons

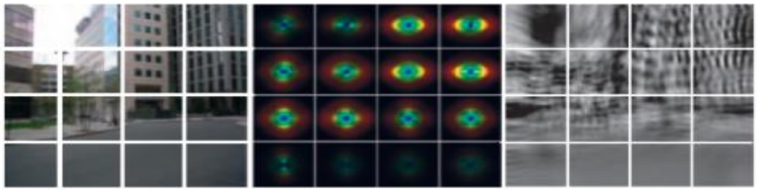


HoG

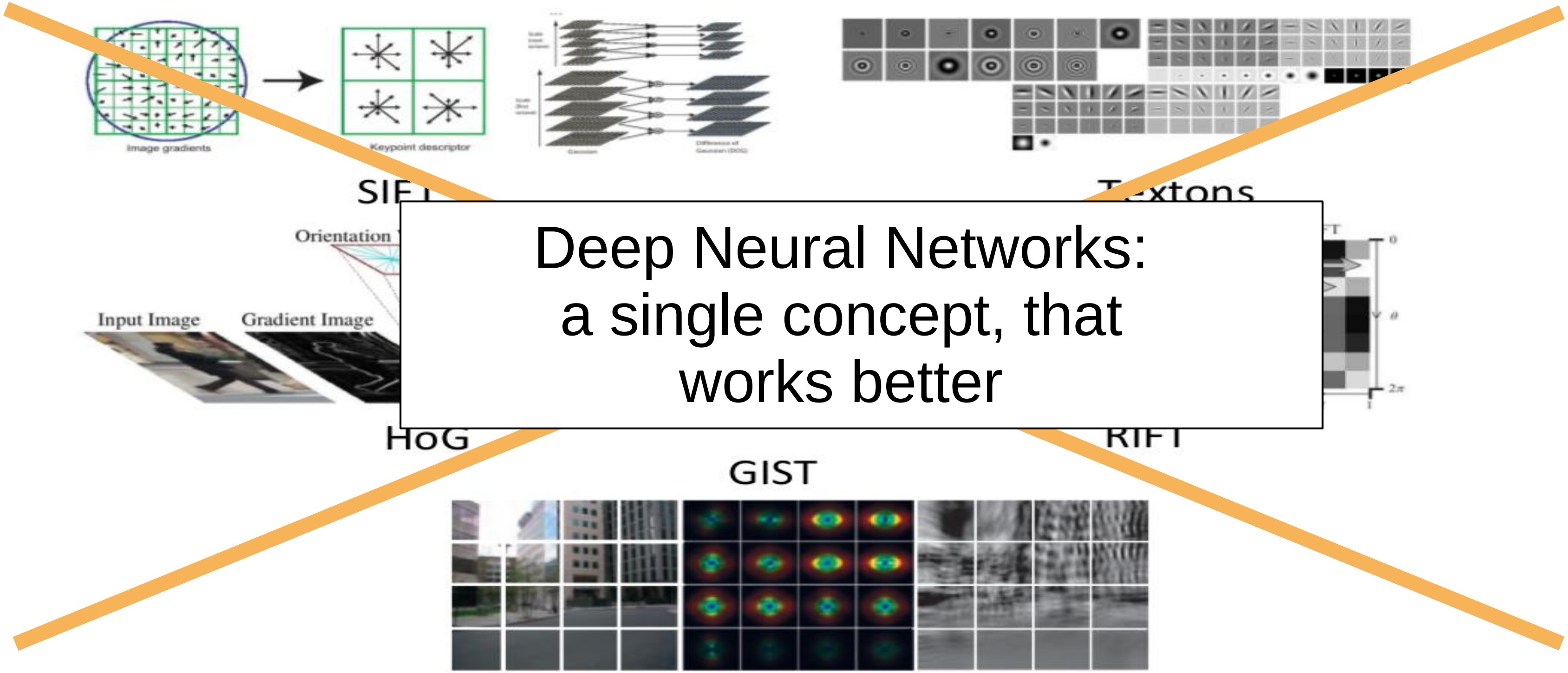


RIFT

GIST



No art now, just engineering



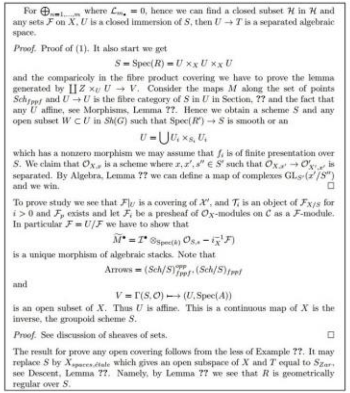
Deep Neural Networks:
a single concept, that
works better

Sequence data examples

		Large data set	Semi-structured data
Speech recognition		+	+
Sentiment classification	"There is nothing to like in this movie."	+	+
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	+	+
Machine translation	Voulez-vous chanter avec moi?	+	+
Video activity recognition		+	+

Examples of texts generated by LSTM (Long Short Term Memory NN)

- Shakespeare
- Wiki
- Algebraic geometrics articles
- Linux Source Code
- Dinosaurs names



For $\bigoplus_{n=1,\dots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) =$ $\mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\dots,0}$.

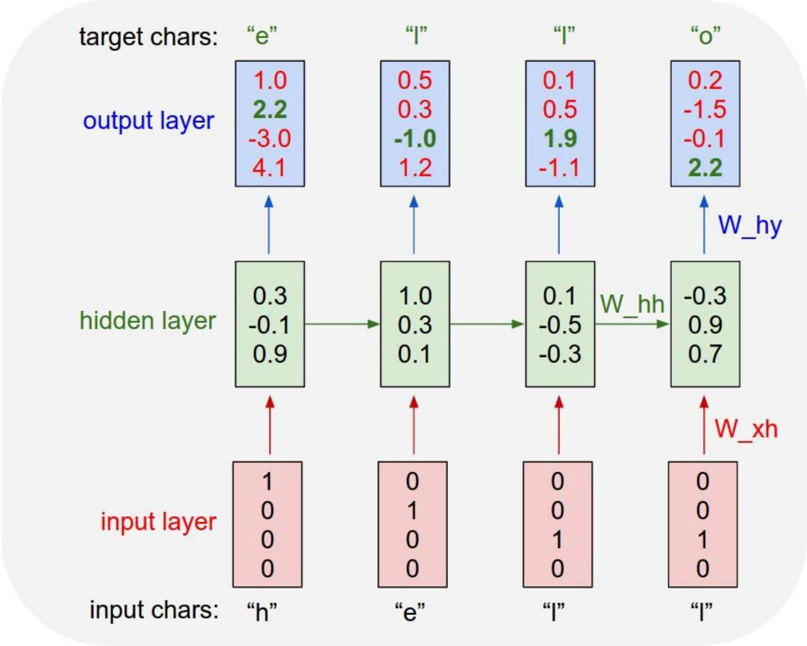
Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □



Skoltech

Conclusions

- Classic ML can’t handle semi-structured data common in sequential data processing
- We can *learn representations* via Neural Networks
- Results are nice even for relatively simple models

Skoltech

Machine translation: application example

Dog Translation Machine

Skoltech

Machine translation, 50-s

Cold war child: translator from Russian to English IBM 701 Translator

Doctor Dostert predicted that “five, perhaps three years hence, interlingual meaning conversion by electronic process in important functional areas of several languages may well be an accomplished fact.” (1954)

Rule-based approach that uses English-Russian dictionary

Project shut-down in 5 years: no significant progress

https://www.ibm.com/ibm/history/exhibits/701/701_translator.html
https://youtu.be/8ZtdVUB007A

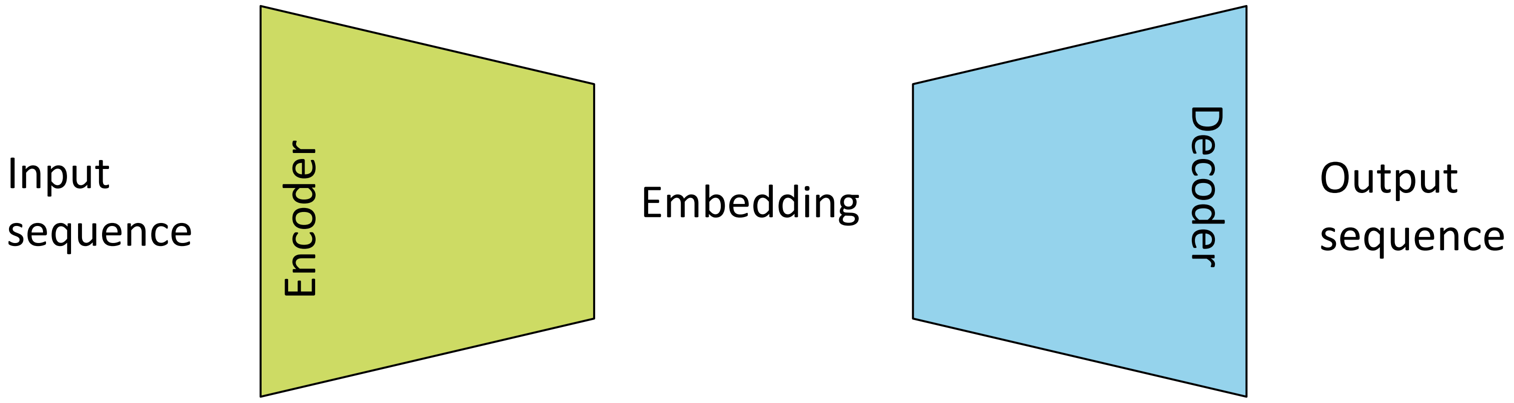
Skoltech

Statistical approach – the leading one before 2014

- Complicated and heavy model
- Many separate components
- Complex generation of inputs
- Support of a sophisticated system
- Quality is not great

Skoltech

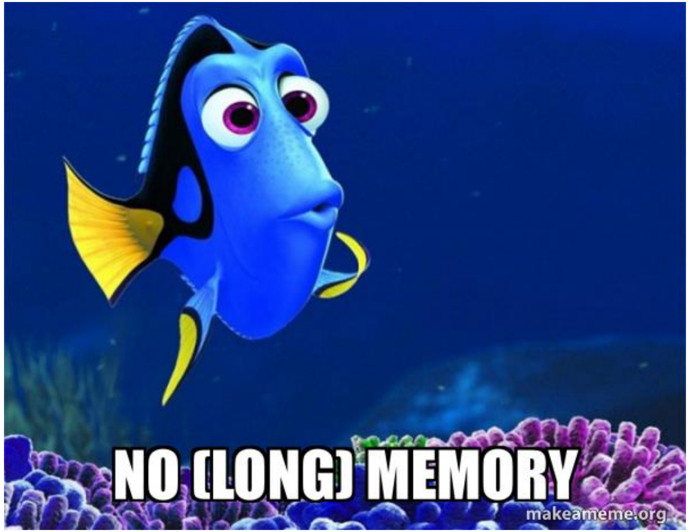
Neural network for machine translation



seq2seq (sequence to sequence) architecture

Skoltech

Vanilla Recurrent Neural Network



Skoltech

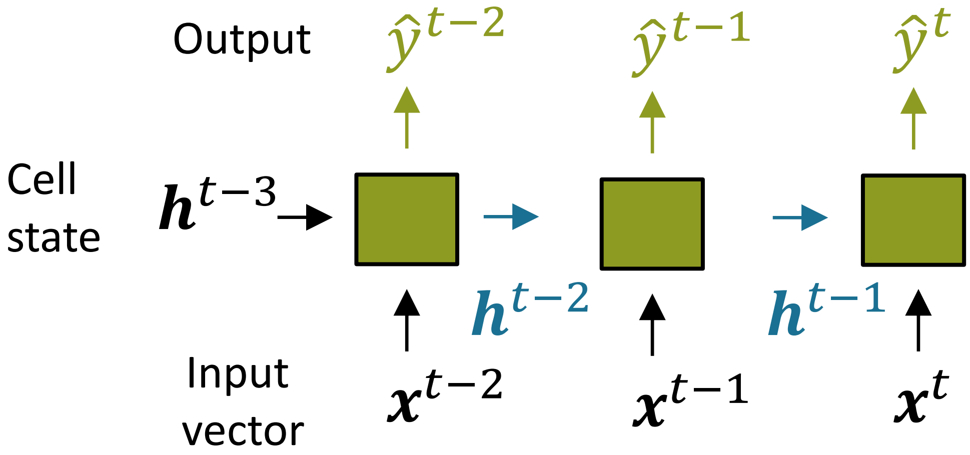
Simple RNN model

At each step:

$\hat{y}^t$ - output / model prediction

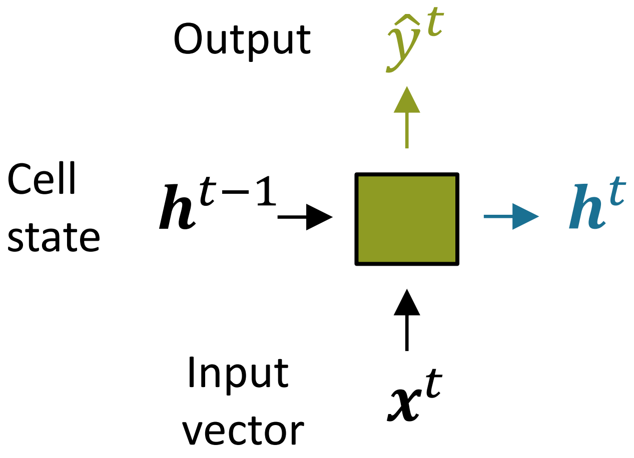
$\boldsymbol{x}^t$ - input vector / new information

$\boldsymbol{h}^t$ - cell / hidden state



Skoltech

Simple RNN model block



$$\boldsymbol{h}^t = f_h(\boldsymbol{x}^t, \boldsymbol{h}^{t-1})$$

$$\boldsymbol{h}^t = \tanh(V\boldsymbol{x}^t + W\boldsymbol{h}^{t-1} + b_h)$$

$$\hat{y}^t = f_y(\boldsymbol{h}^t)$$

$$\hat{y}^t = \mathrm{softmax}(U\boldsymbol{h}^t + b_y)$$

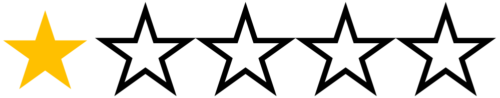
Sequence processing with Vanilla RNN

• Long-term memory	**NO** 😟
• Maintain order information	**YES**
• Natural preprocessing	**a kind of**
• Variable-length sequences processing	**YES (if one to one)**

Another semi-structured data problems

Credit scoring:
default prediction

Money
Transactions data

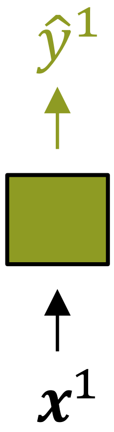


Fraud detection
in healthcare insurance

A history of visits

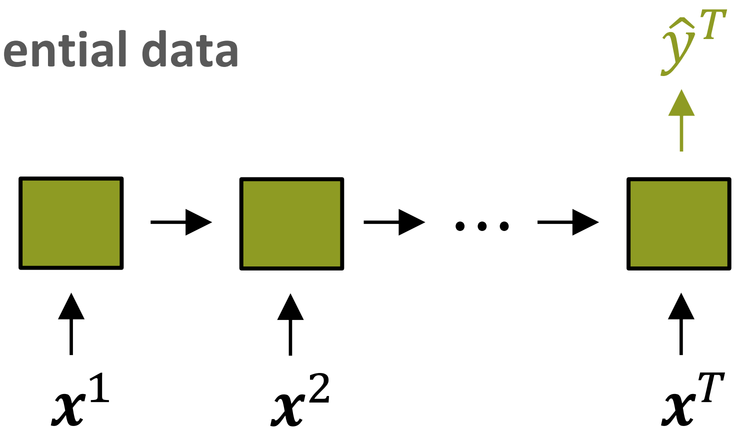
Is there a fraud?

Problem statements for sequential data

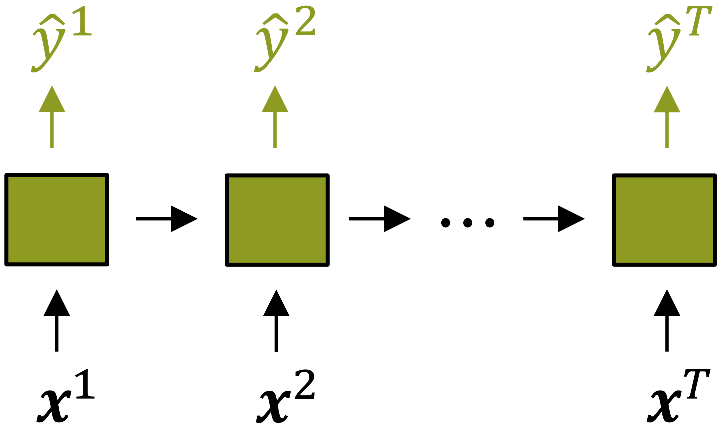


No recurrent structure

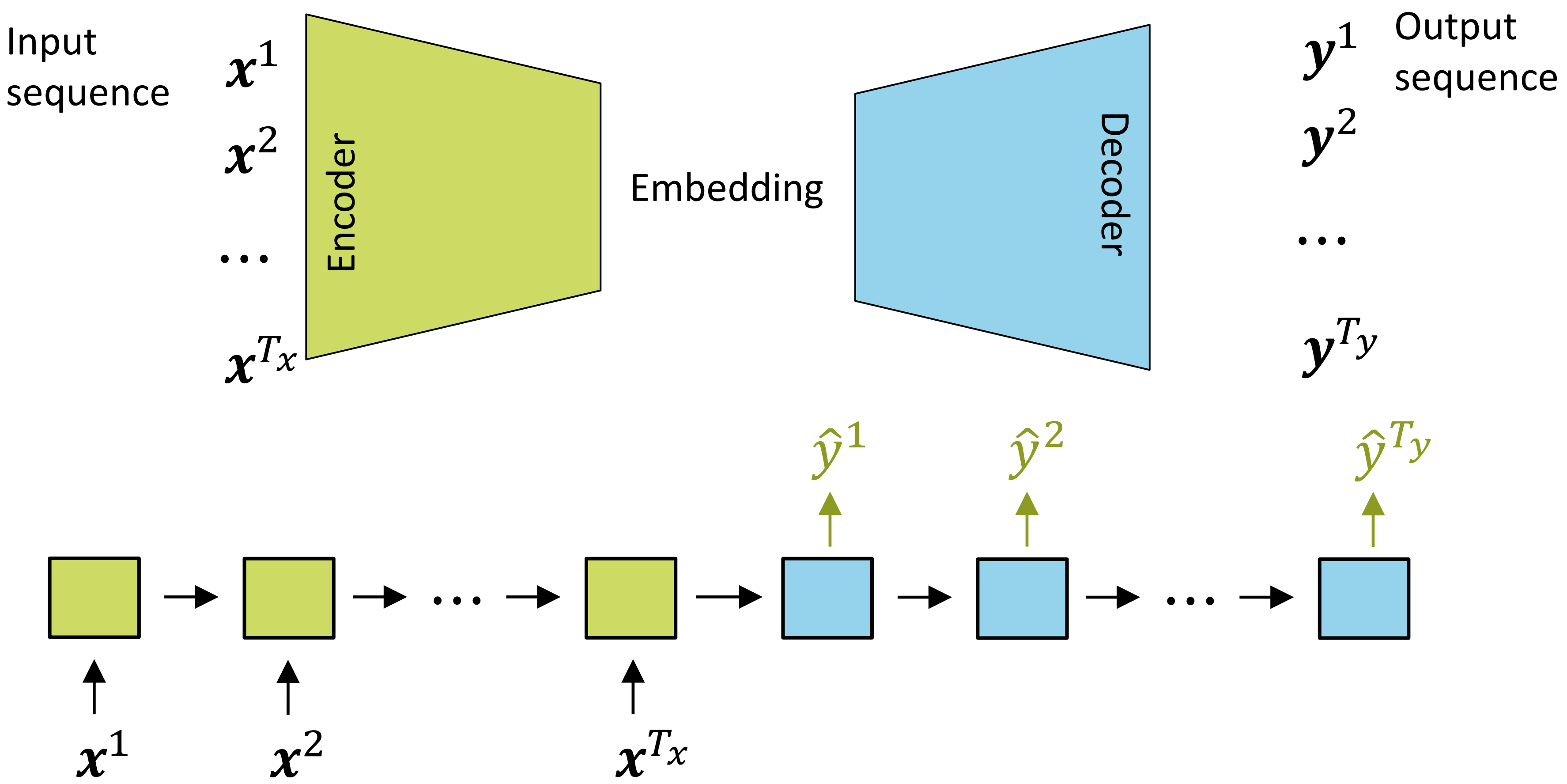
Sequence to a single output



Sequence to a sequence of outputs



A sequence to sequence problem

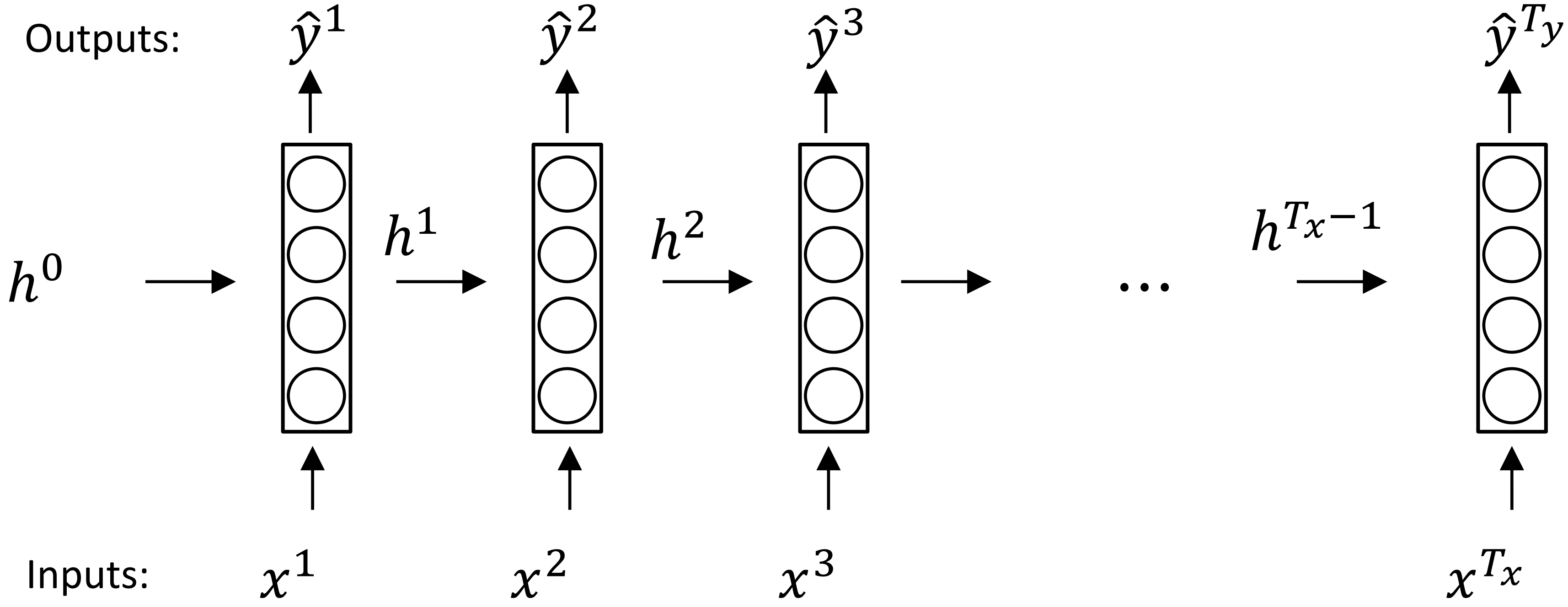


Skoltech

More on Recurrent Neural Networks

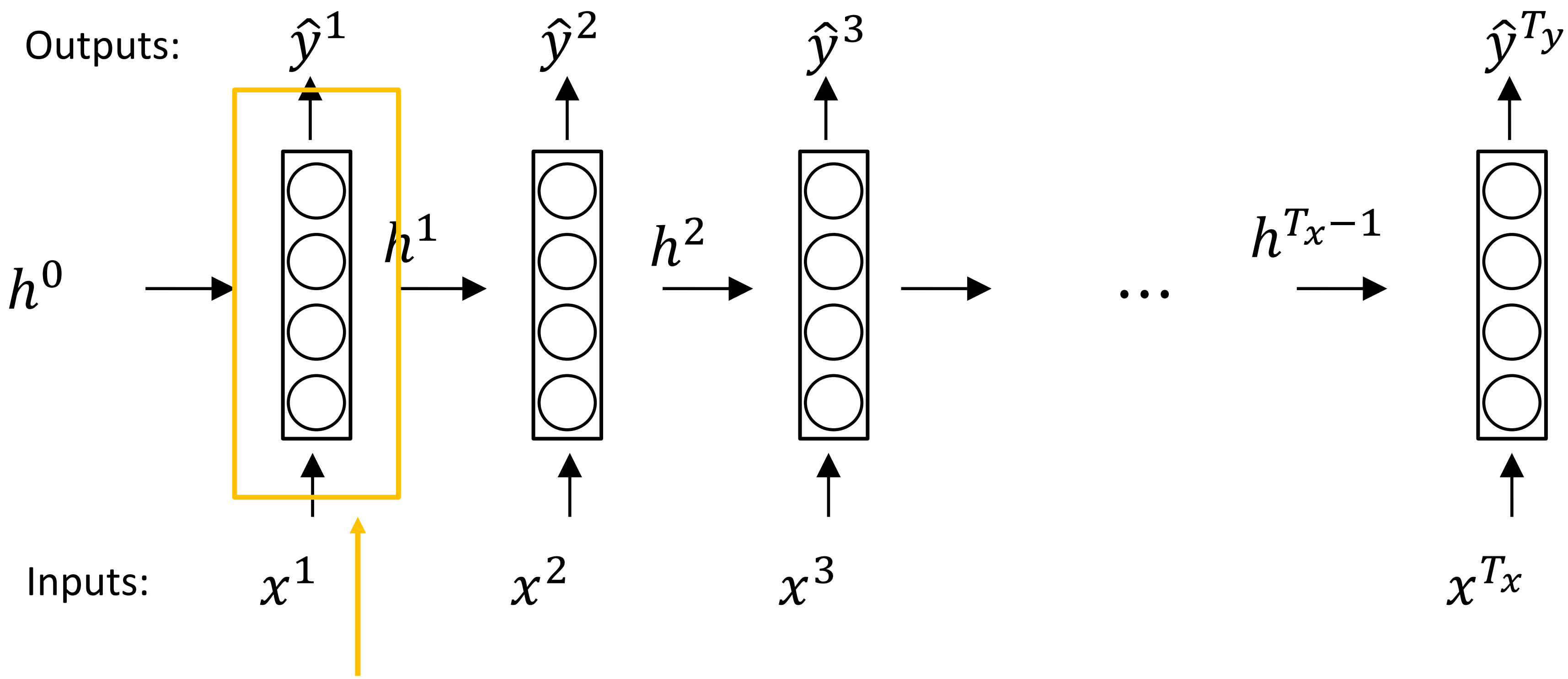
Skoltech

Forward propagation through Recurrent Neural Network



Adopted from Andrew Ng

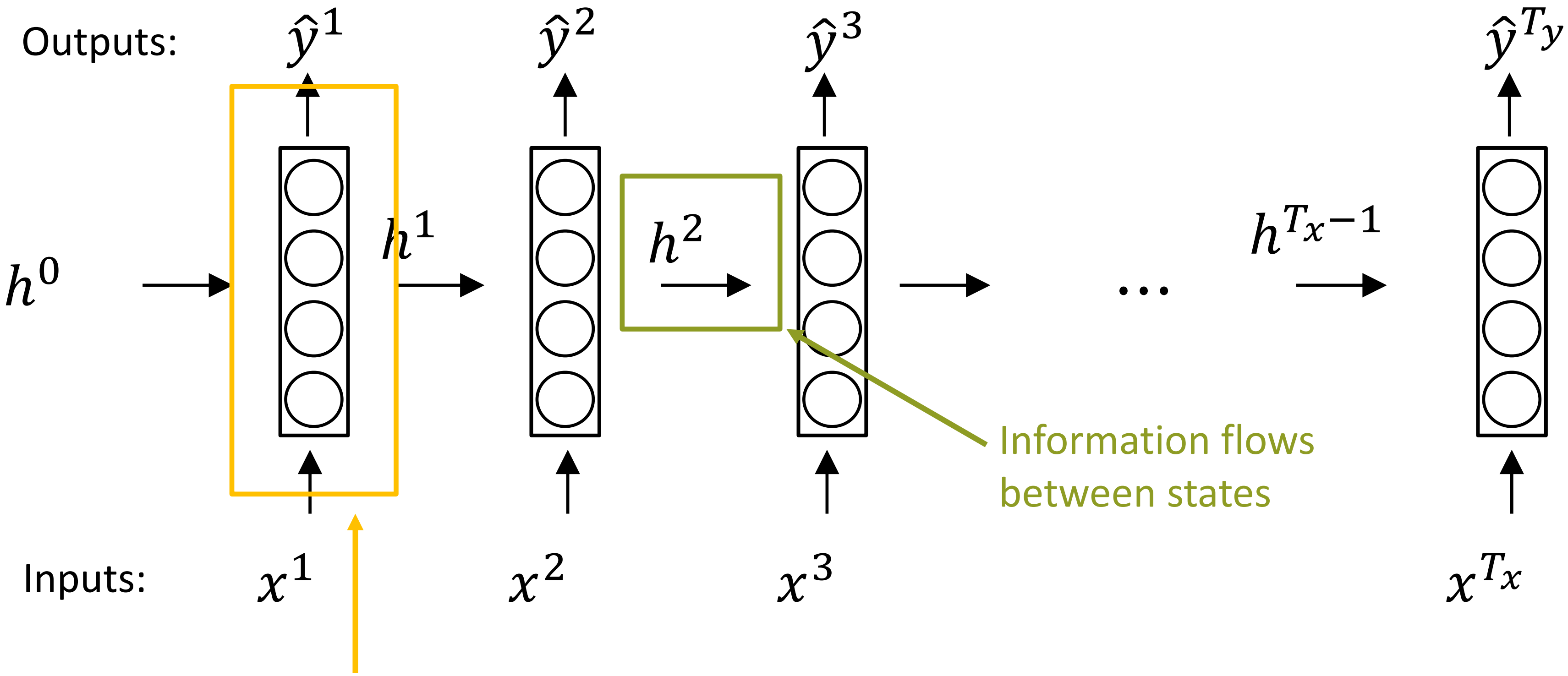
Forward propagation through Recurrent Neural Network



Processing unit is the same for all time moments.
Units has parameters we want to learn!

Adopted from Andrew Ng

Forward propagation through Recurrent Neural Network

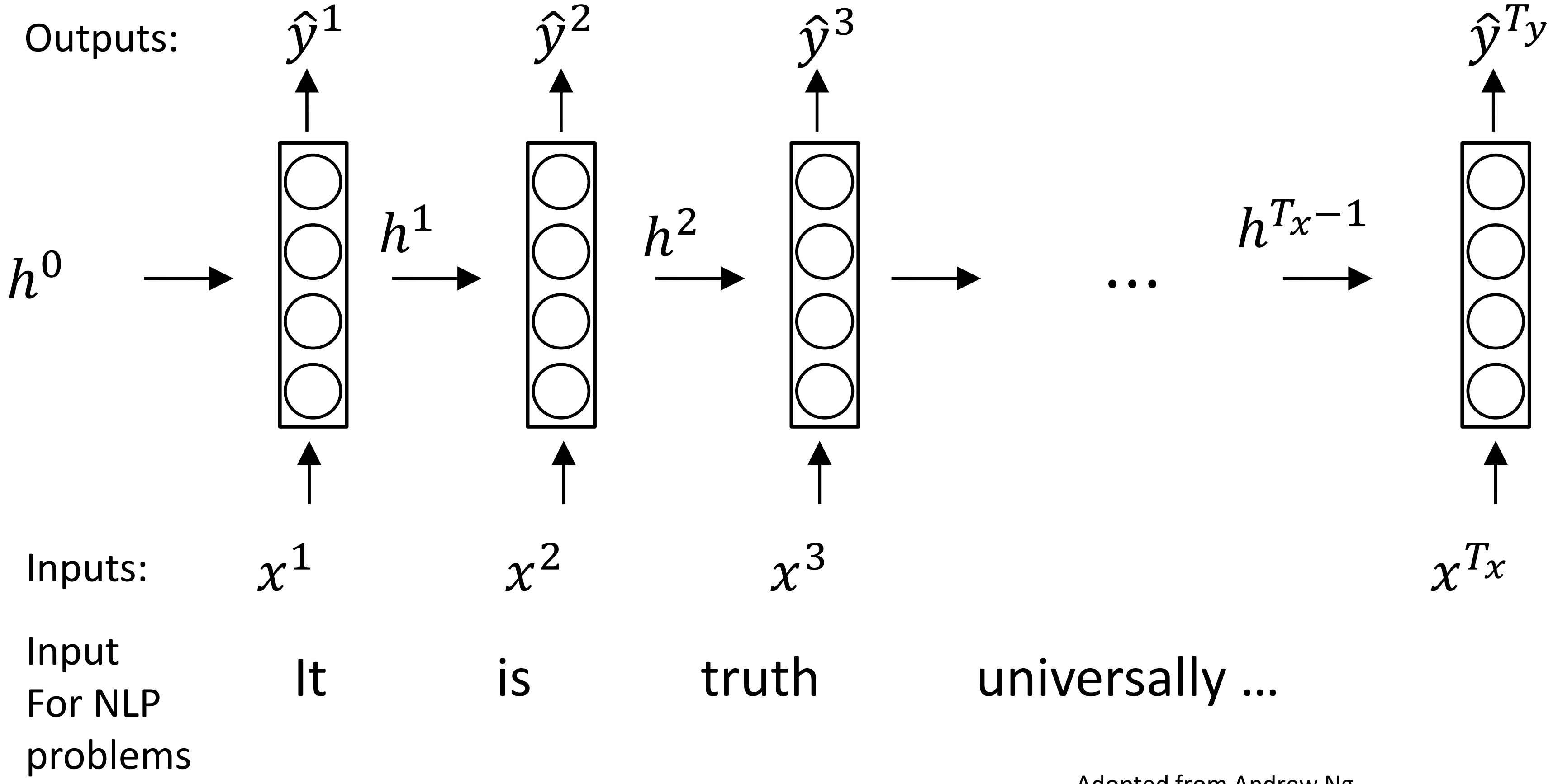


Processing unit is the same for all time moments.
Units has parameters we want to learn!

Skoltech

Adopted from Andrew Ng

Forward propagation through Recurrent Neural Network



Adopted from Andrew Ng

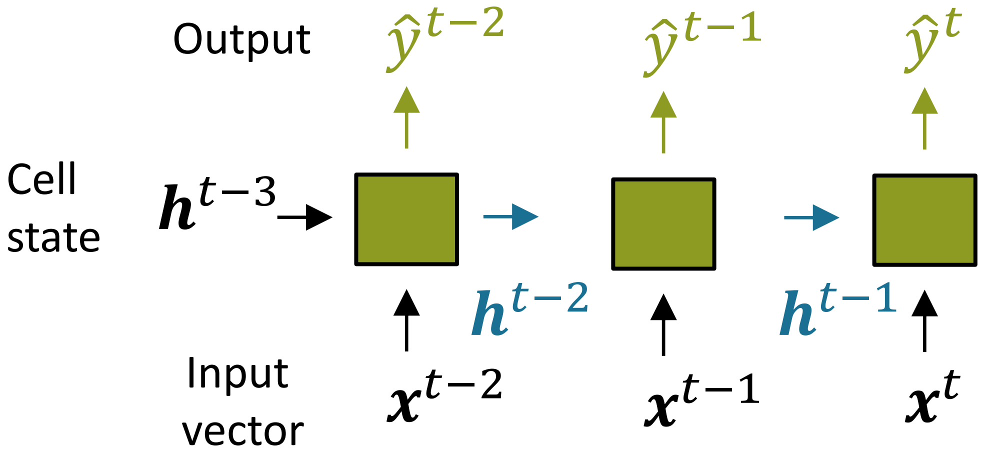
Simple RNN model

At each step:

$\hat{y}^t$ - output / model prediction

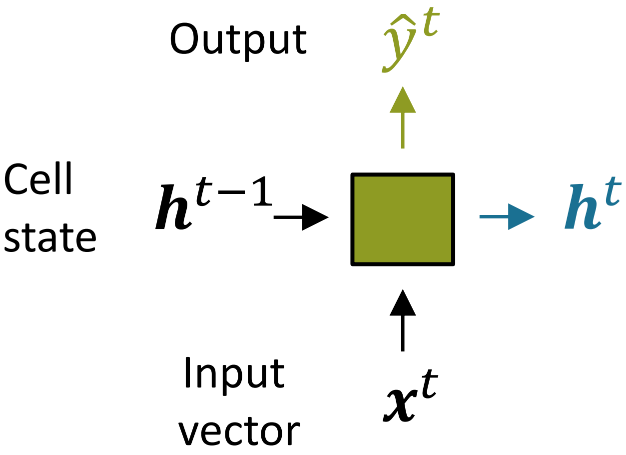
$\boldsymbol{x}^t$ - input vector / new information

$\boldsymbol{h}^t$ - cell / hidden state



Skoltech

Simple RNN model block



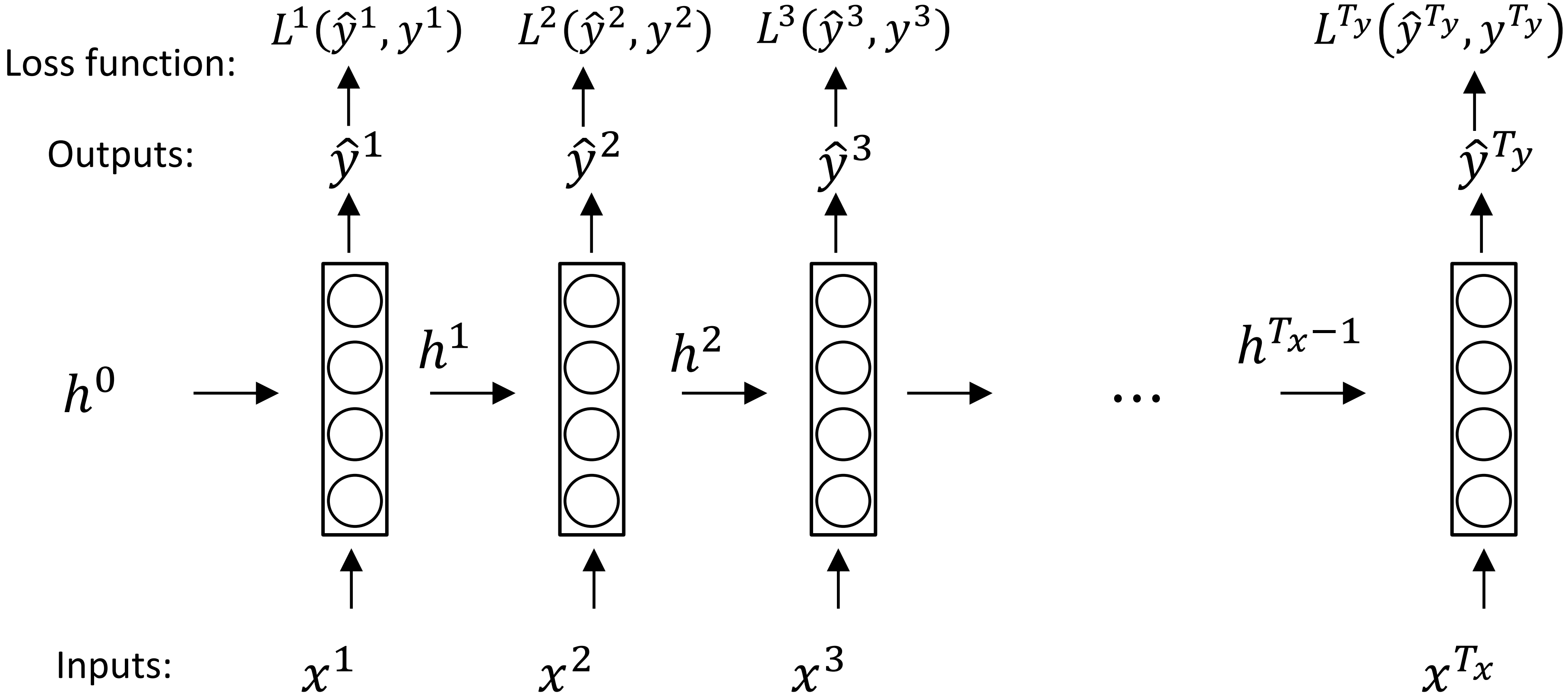
$$\boldsymbol{h}^t = f_h(\boldsymbol{x}^t, \boldsymbol{h}^{t-1})$$

$$\boldsymbol{h}^t = \tanh(V\boldsymbol{x}^t + W\boldsymbol{h}^{t-1} + b_h)$$

$$\hat{y}^t = f_y(\boldsymbol{h}^t)$$

$$\hat{y}^t = \mathrm{softmax}(U\boldsymbol{h}^t + b_y)$$

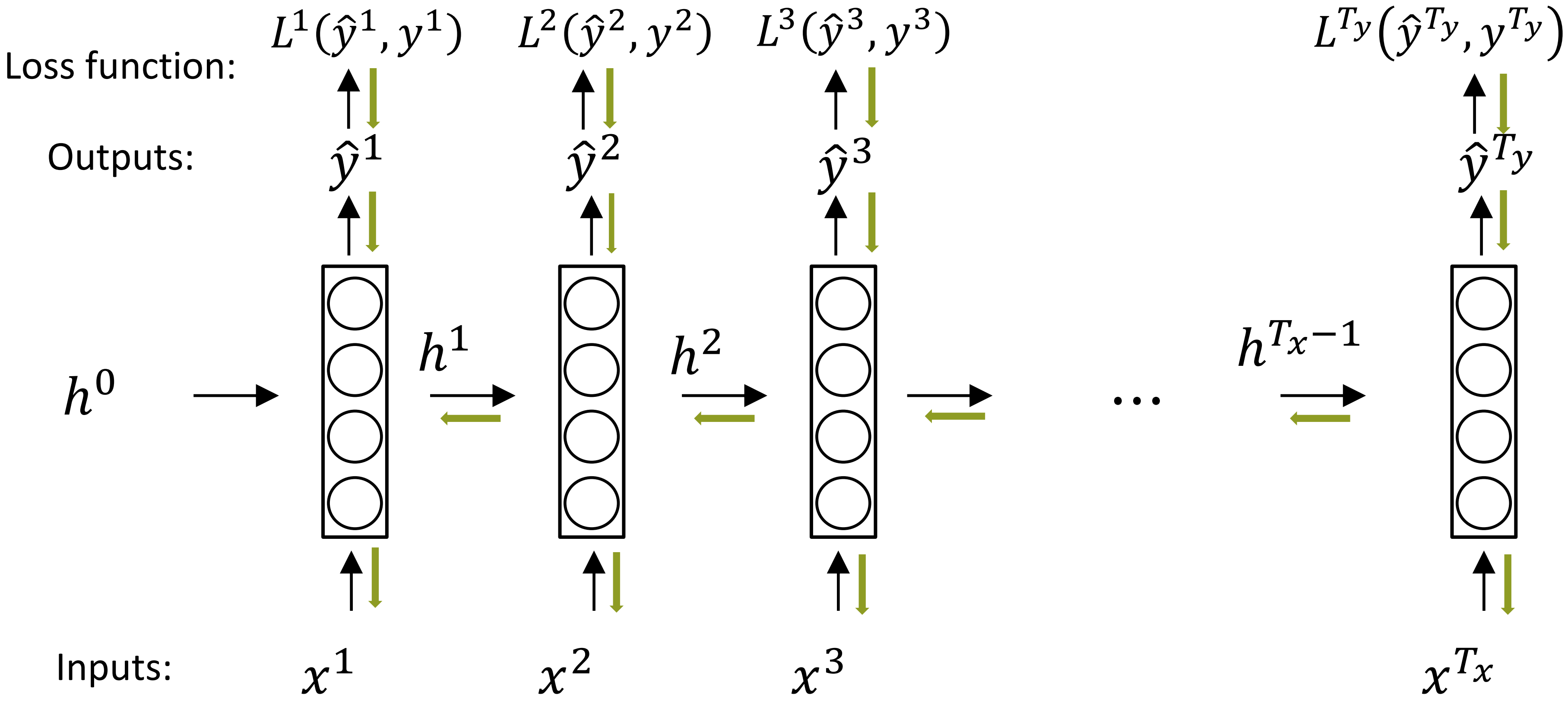
Backward propagation through Recurrent Neural Network



Skoltech

Adopted from Andrew Ng

Backward propagation through Recurrent Neural Network



Adopted from Andrew Ng

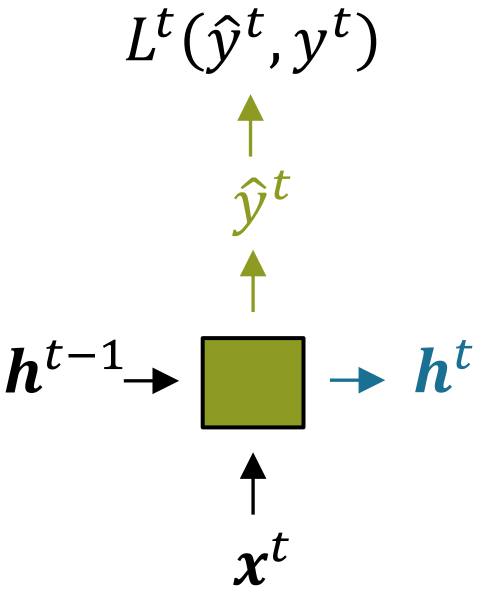
Skoltech

Backpropagation w.r.t. U

$$L = \sum_{i=1}^{T_y} L^i(\hat{y}^i, y^i)$$

$$\frac{\partial L}{\partial U} = \sum_{i=1}^{T_y} \frac{\partial L_i}{\partial U} = \sum_{i=1}^{T_y} \frac{\partial L_i}{\partial \hat{y}^i} \frac{\partial \hat{y}^i}{\partial U}$$

$$\hat{y}^t = \mathrm{softmax}(U\boldsymbol{h}^t + b_y)$$



Backpropagation w.r.t. W

$$\hat{y}^t = \mathrm{softmax}(U\boldsymbol{h}^t + b_y)$$

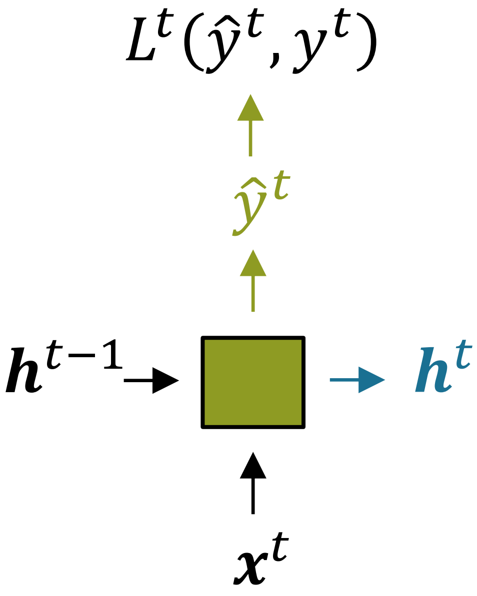
$$L = \sum_{i=1}^{T_y} L^i(\hat{y}^i, y^i)$$

$$\frac{\partial L}{\partial W} = \sum_{i=1}^{T_y} \frac{\partial L_i}{\partial W} = \cancel{\sum_{i=1}^{T_y} \frac{\partial L_i}{\partial \hat{y}^i}\frac{\partial \hat{y}^i}{\partial W}}$$

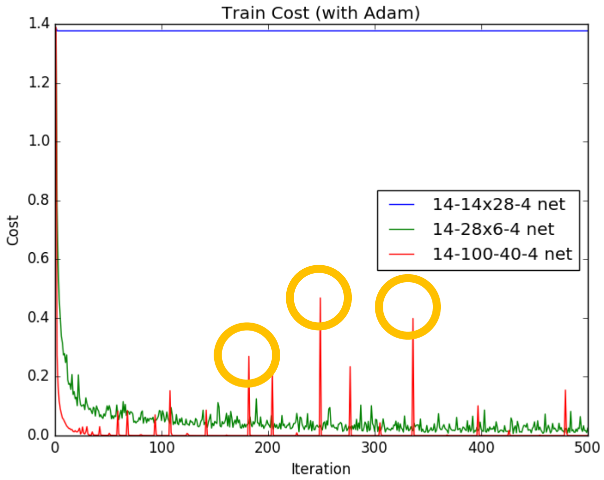
$$\boldsymbol{h}^t = \tanh(V\boldsymbol{x}^t + W\boldsymbol{h}^{t-1} + b_h)$$

$$\frac{\partial L_i}{\partial \hat{y}^i}\frac{\partial \hat{y}^i}{\partial W} = \frac{\partial L_i}{\partial \hat{y}^i}\frac{\partial \hat{y}^i}{\partial h_t}\left(\frac{\partial h_t}{\partial W} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W} + \right)$$

$$\frac{\partial L_i}{\partial \hat{y}^i}\frac{\partial \hat{y}^i}{\partial W} = \frac{\partial L_i}{\partial \hat{y}^i}\frac{\partial \hat{y}^i}{\partial h_t}\Sigma_{i=0}^{T_y}\left(\prod_{j=i+1}^{T_y}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_i}{\partial W}$$



Problems of classic RNN



Solution:

- **Gradient clipping to scale big gradients**

1. $g = \frac{\partial L}{\partial W}$
2. If $g > t$ for some threshold t:

$$g = \frac{t}{\|g\|}\, g$$

Threshold t is selected given the dynamic of loss function over iterations

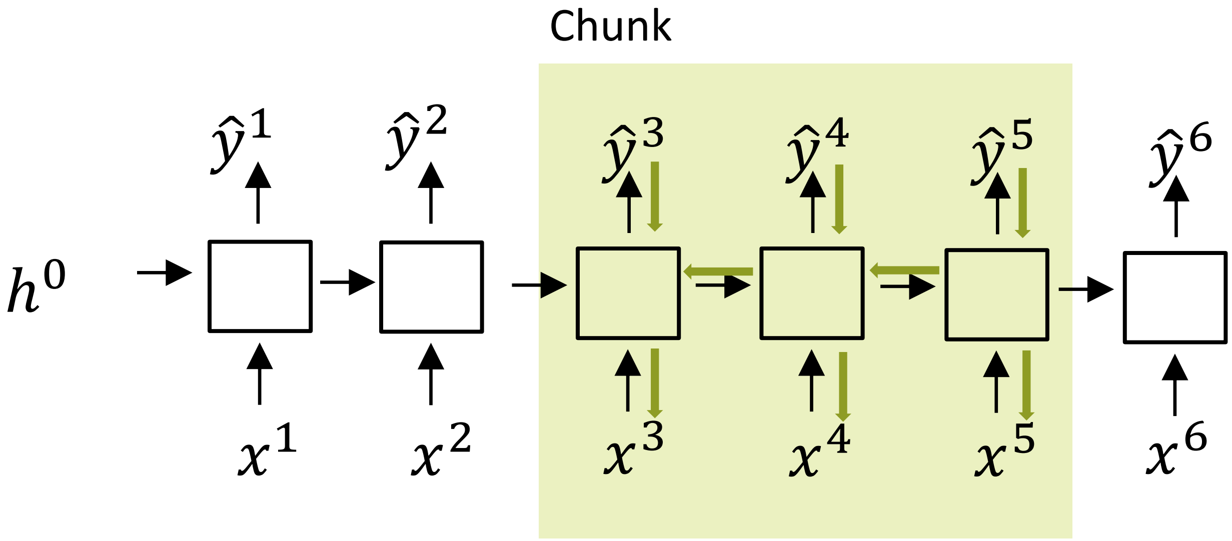
Image from https://stats.stackexchange.com/questions/303857/explanation-of-spikes-in-training-loss-vs-iterations-with-adam-optimizer

Skoltech

Problems of classic RNN

Solution:

- Gradient clipping to scale big gradients
- Truncated backpropagation through time



Skoltech

Problems of classic RNN

Gradients vanishing is a more serious problem

Many values < 1

Product << 1

Bias parameters to capture long-term dependencies

Hard to detect!

Tricks:

Activation functions

- Use ReLU

Parameter initialization

- Initialize weights to identity matrix
- Initialize biases to zero

Skoltech

Another big problem of classic RNN

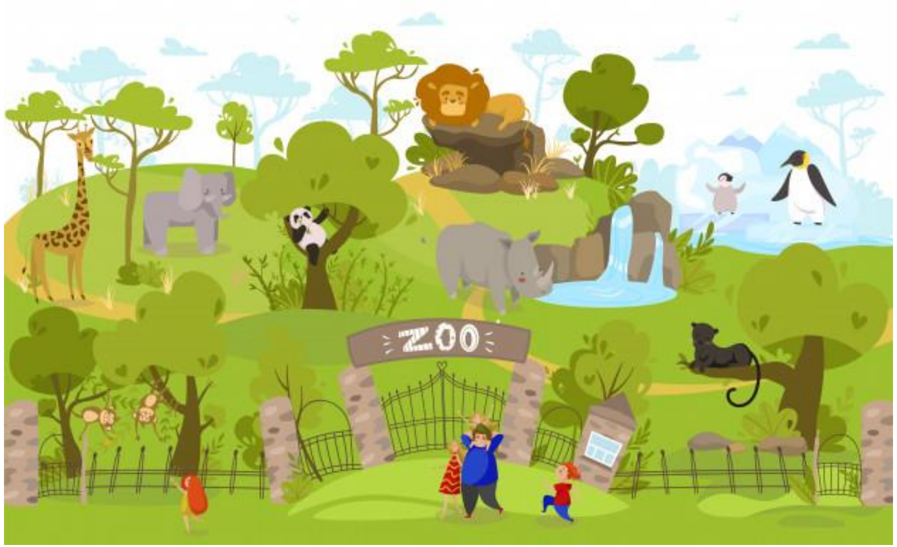
Problem: Neural networks forget fast, and it is hard to learn long-term dependencies

Solution: Gated architectures

More complex recurrent units with gates to control what information is passed through

- GRU (Gated Recurrent Unit)
- LSTM (Long-Short Term Memory)

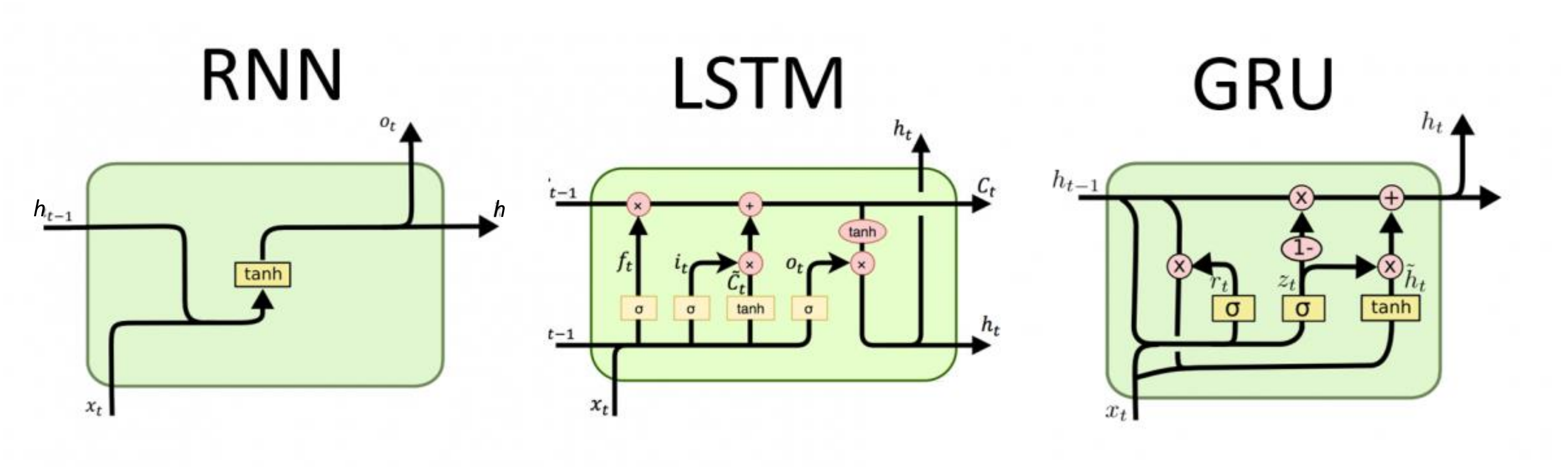
Selection of RNN architecture



Skoltech

Better RNN units: LSTM and GRU

- LSTM: long short term memory [1]
- GRU: Gated recurrent unit [2]



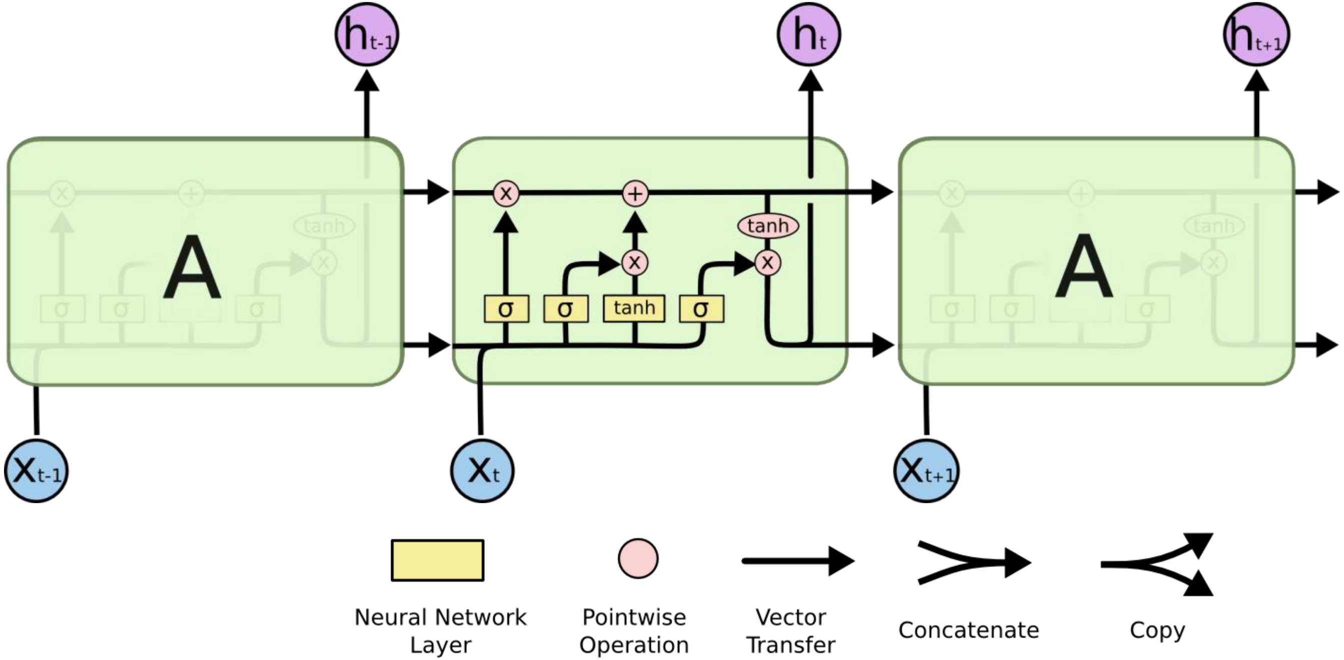
1. Schmidhuber, J., & Hochreiter, S. (1997). Long short-term memory. *Neural Comput*, *9*(8), 1735-1780.
2. Cho, K., Van Merriënboer et al. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Skoltech

Details on how LSTM works

Remembering information for long periods of time is practically the default behavior of LSTM

LSTM was proposed by J. Schmidhuber group in 1991



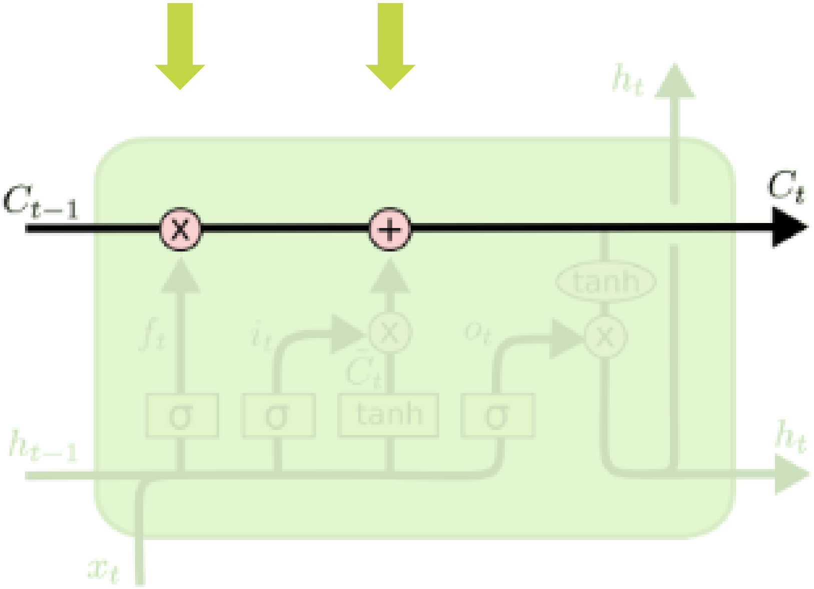
Skoltech

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

Long term memory part – Cell state

Forget part Update part

Cell state before



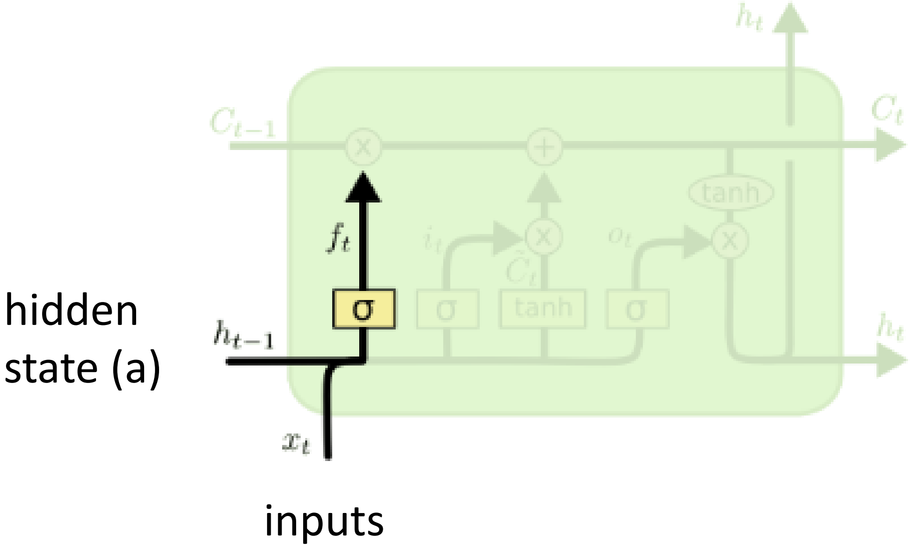
Cell state after

Skoltech

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

Forget part

Identify how much should we forget: sigmoid returns value between 0 and 1



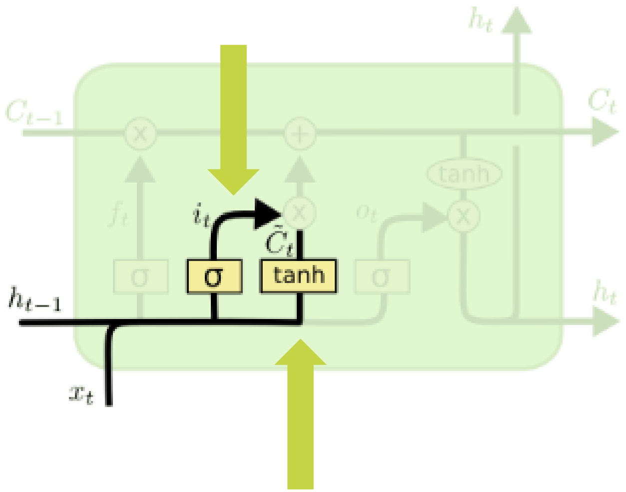
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

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

Update part

Identify how much should we update:
sigmoid returns value between 0 and 1

hidden state (a)



inputs

What update we do

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Skoltech

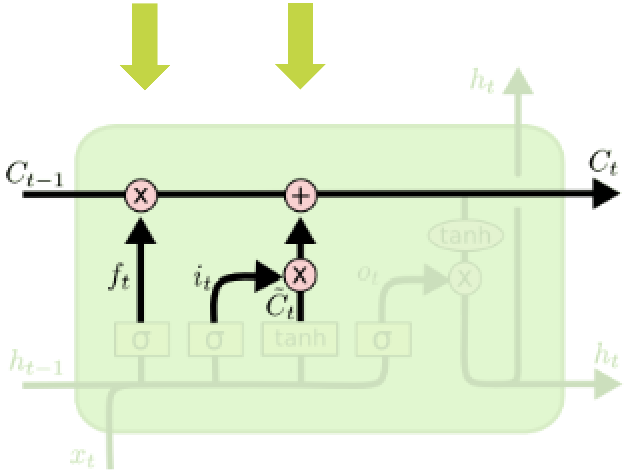
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long term memory part – Cell state

Forget part

Update part

Cell state
before



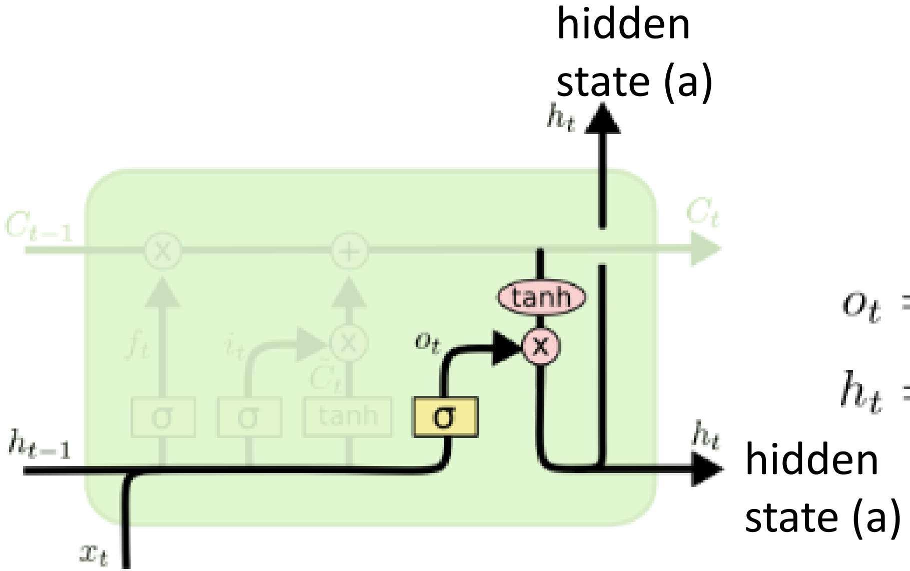
Cell state
after

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Skoltech

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

Update everything else



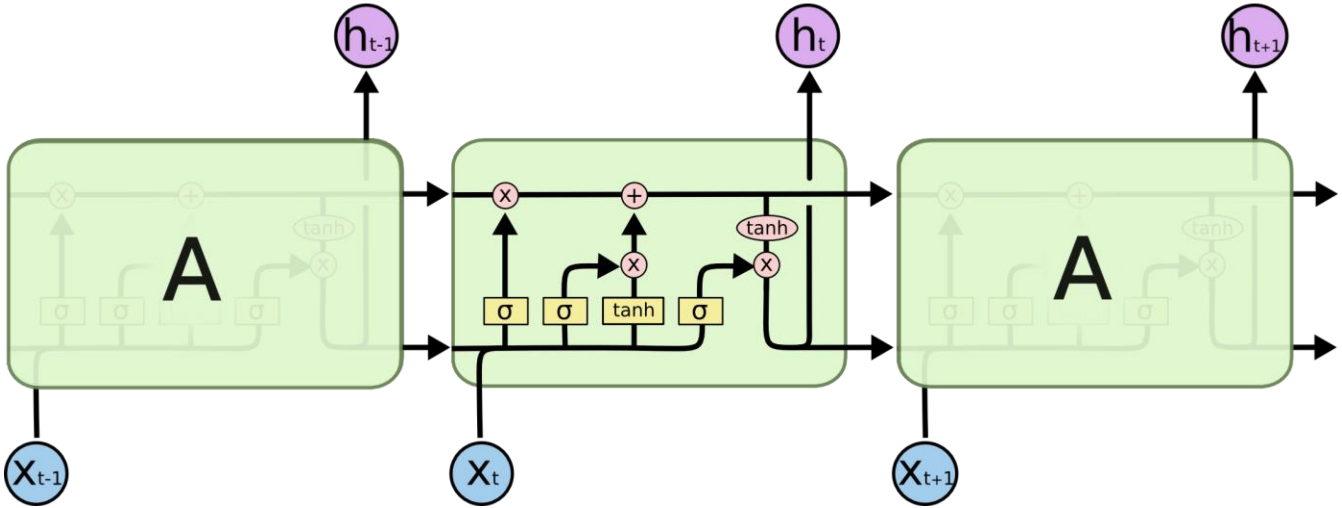
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

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

Details on how LSTM works

- There are cell and hidden (activation) states
- LSTM block forgets and updates cell state during processing at one block



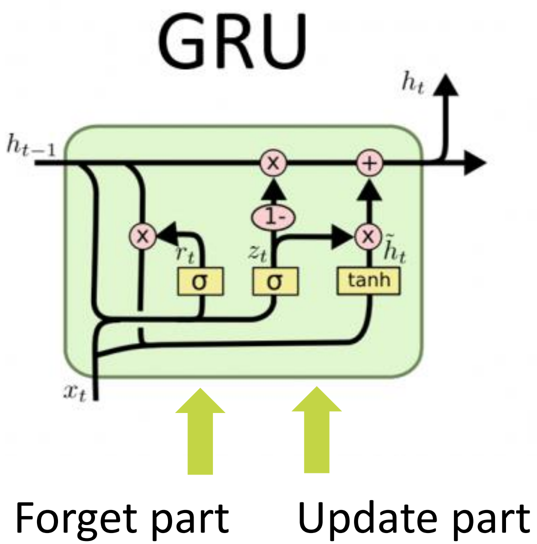
Skoltech

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

GRU – Gated Recurrent Unit

- Update gate – what to pay attention to
- Reset gate – what to forget

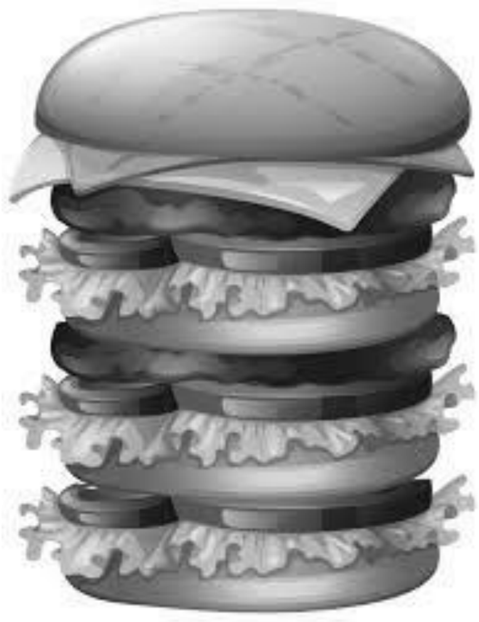
- Slightly worse than LSTM for NLP – but not in all problems
- Simpler and cheaper than LSTM



Skoltech

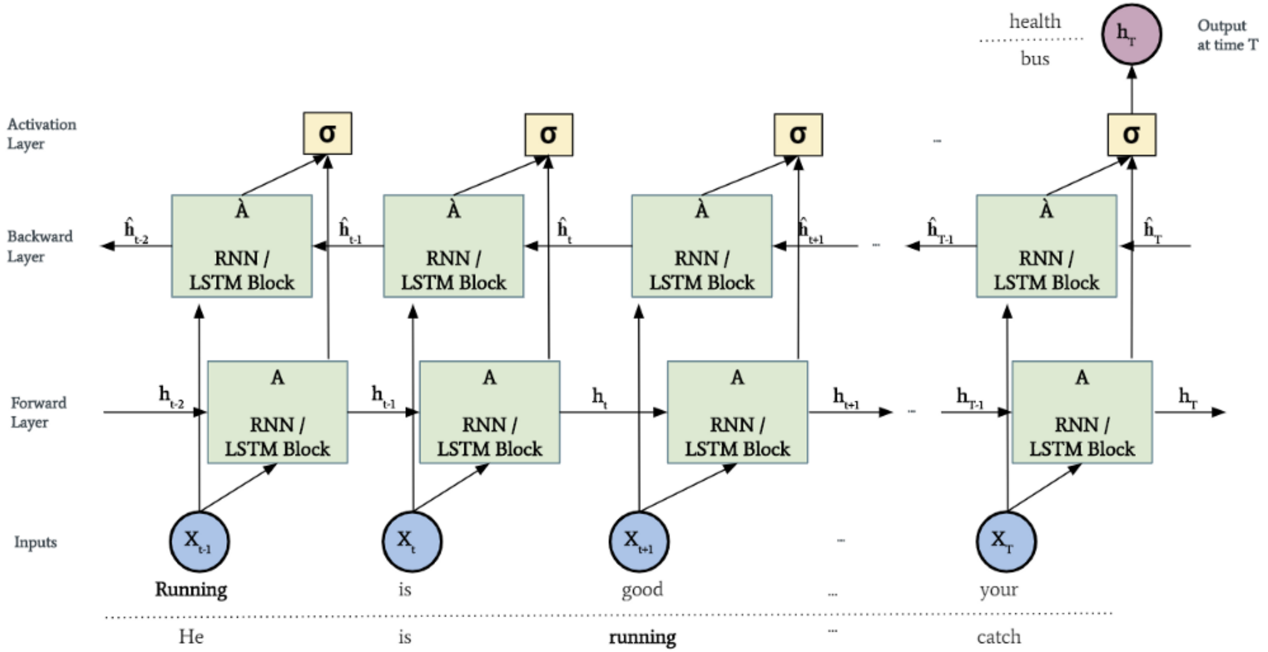
Cho, Kyunghyun et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. EMNLP.=-098]-[9o08i=-89-p9o0i8o0-9-9-980876-=-=-=-=-=-=-=[]jnhg765432098765438 75r4e6666667y5yt2014.

Multilayer architectures



Skoltech

Other architectures: bidirectional LSTM



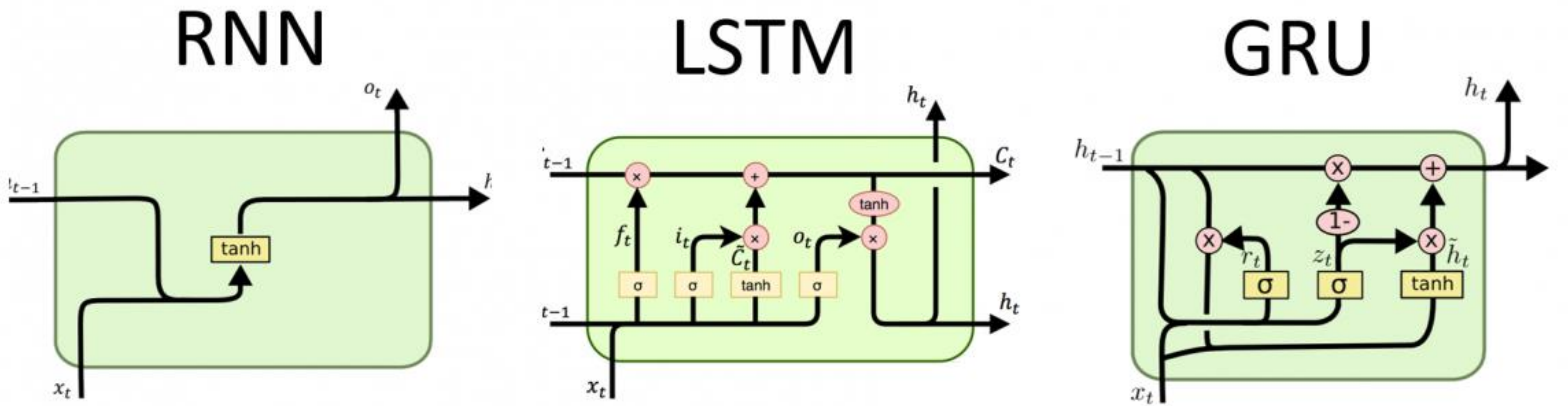
Skoltech

https://mc.ai/review-generation-using-bidirectional-long-short-term-memorylstm/

Other architectures: bidirectional LSTM

Bidirectional LSTM are useful when we benefit from the future data or can use it:

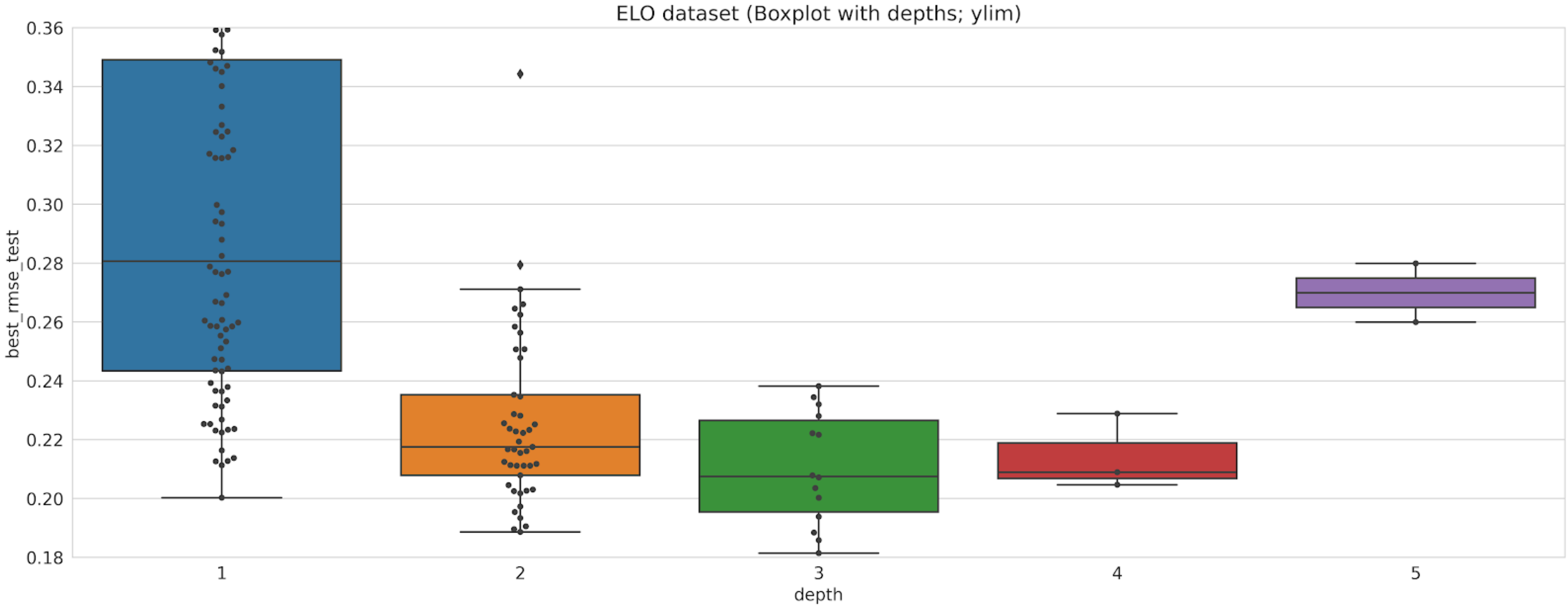
- Handwriting Recognition
- Speech Recognition
- Protein Structure Prediction (Bioinformatics)



Skoltech

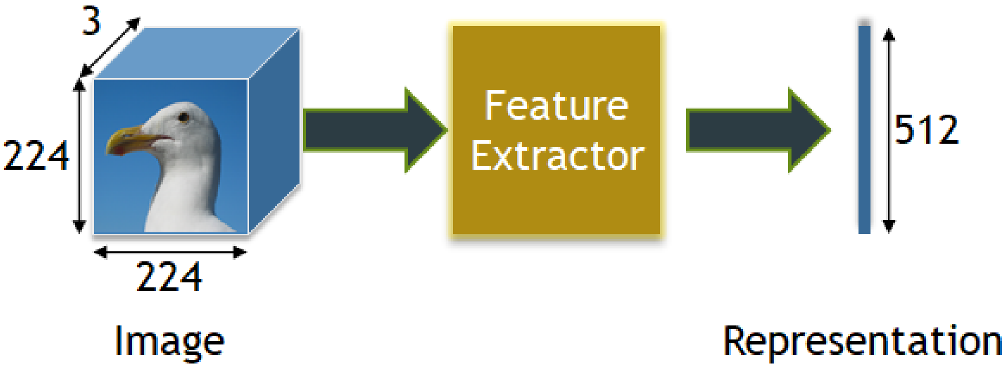
https://mc.ai/review-generation-using-bidirectional-long-short-term-memorylstm/

Multiple layers RNN performance



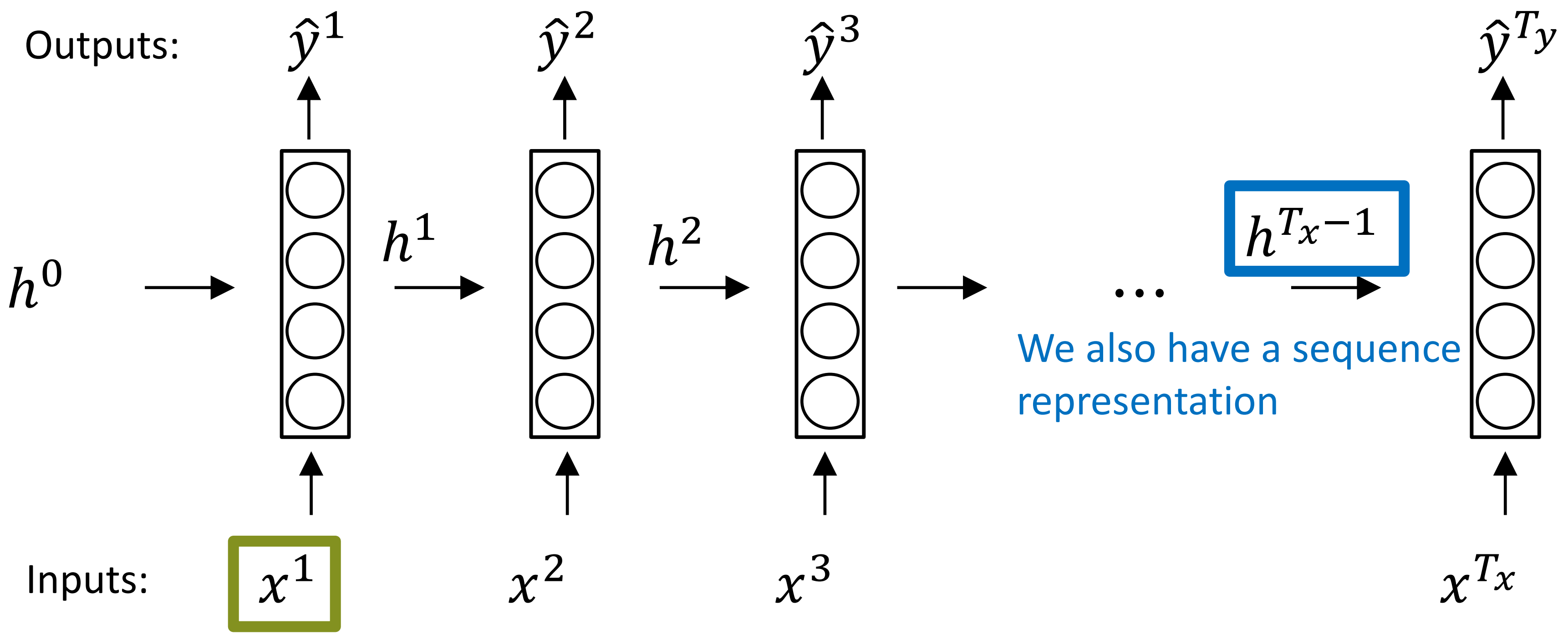
Skoltech

Representation learning is the core feature of Neural Networks



Skoltech

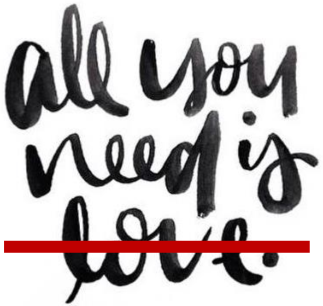
Representation learning is still here for Recurrent Neural Networks



Most of the time we also learn representations of objects in an end2end manner with backpropagation

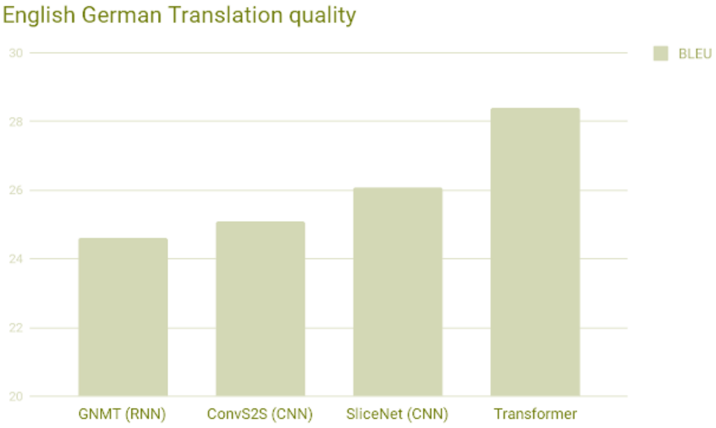
Attention mechanism

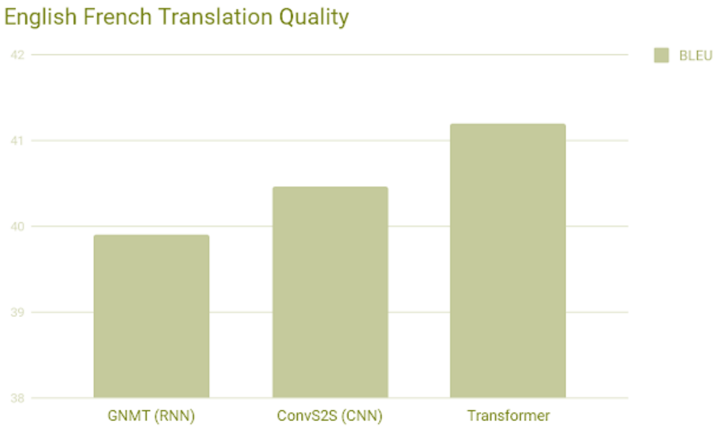
New state of the art:
attention is all we need



attention

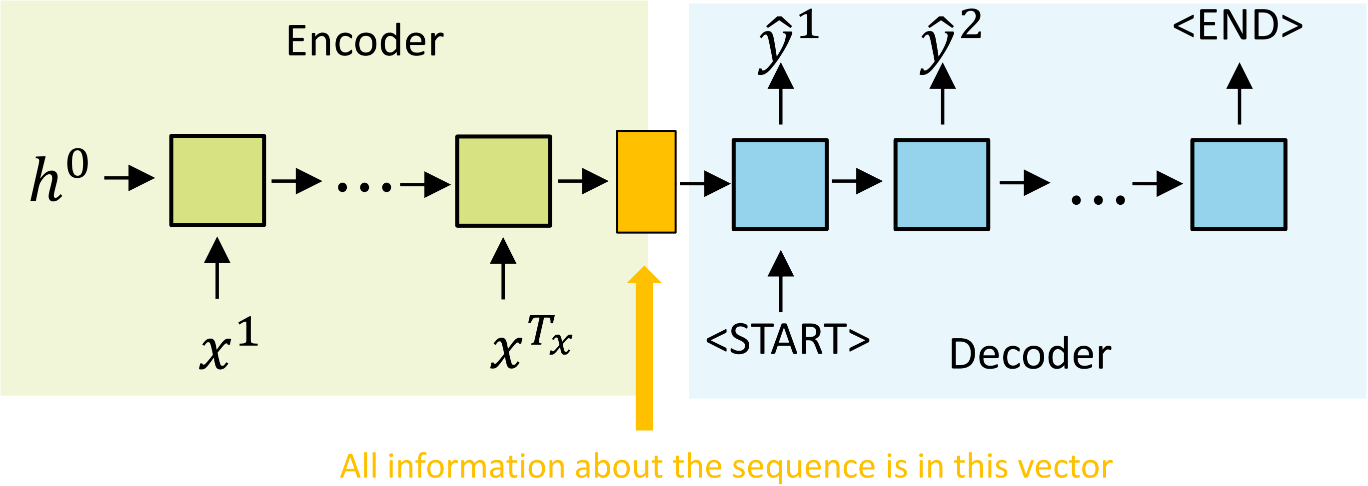
Higher BLEU scores are better





https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Bottleneck in seq2seq models



Skoltech

Attention

- Solution to the bottleneck problem
- Direction connection between parts of input and output sequence

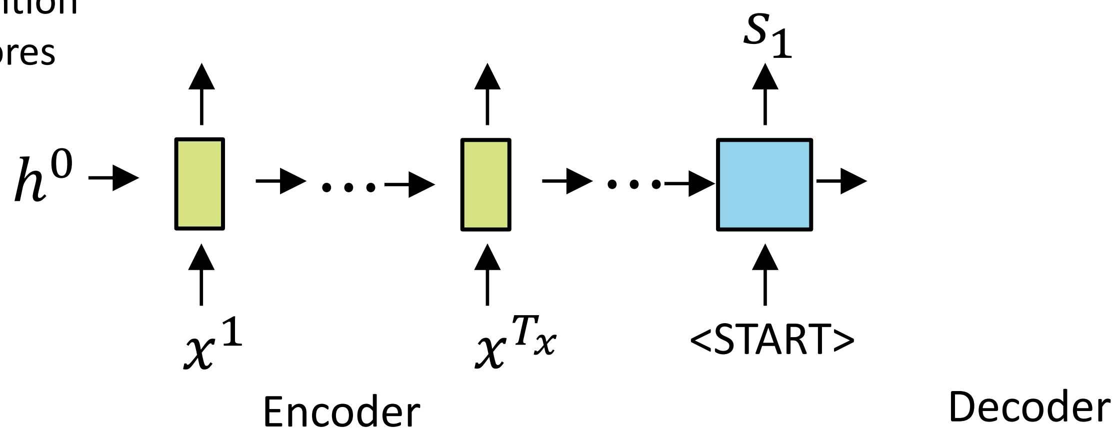
Skoltech

Sequence 2 sequence with attention

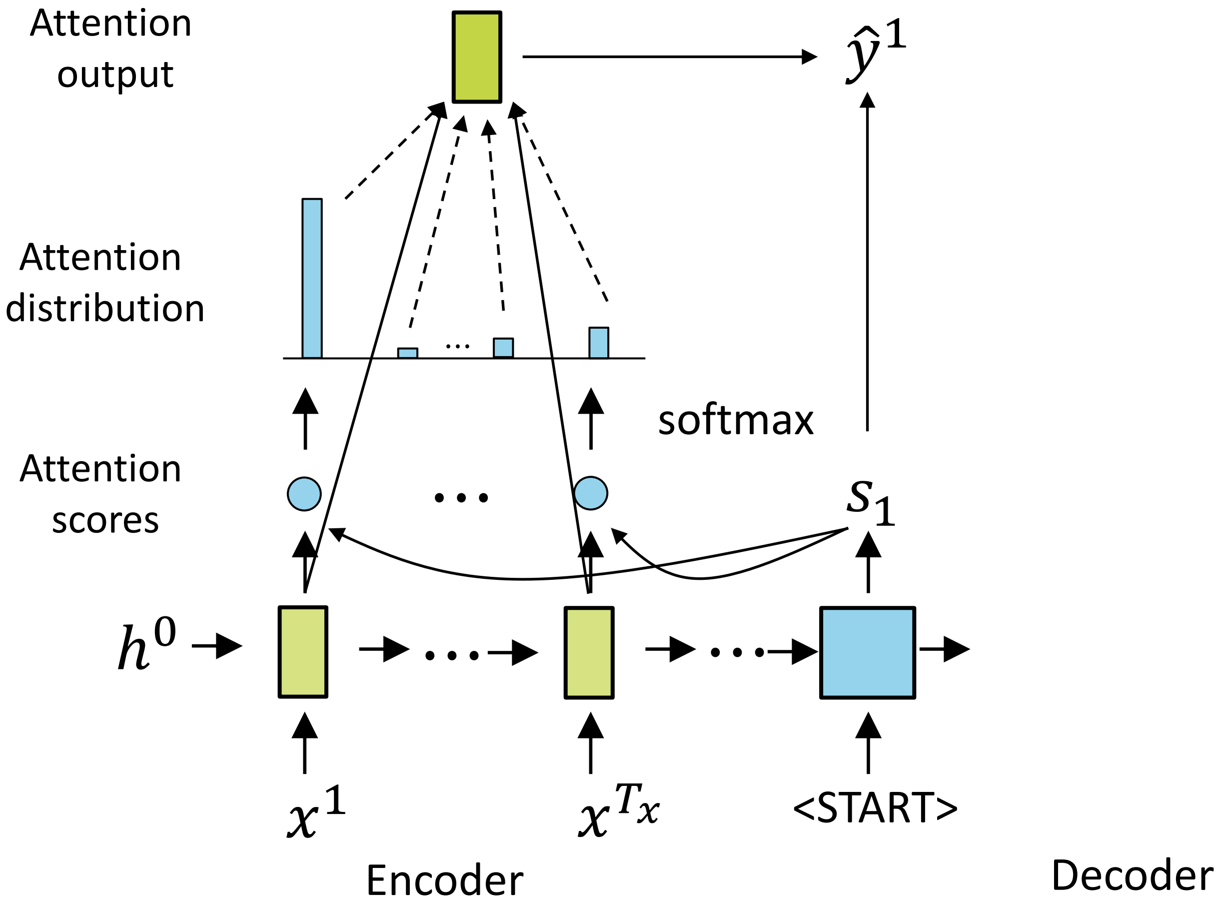
Attention output

Attention distribution

Attention scores

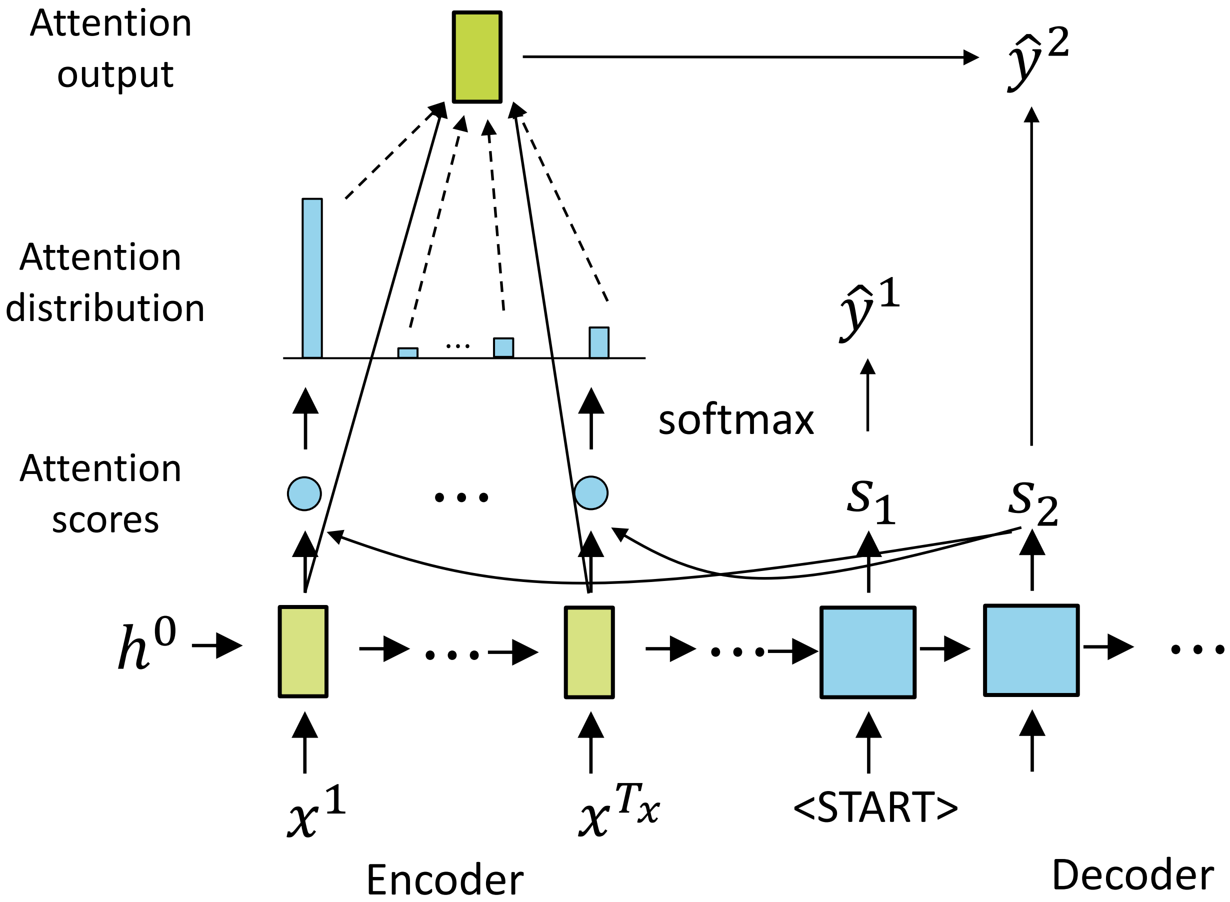


Sequence 2 sequence with attention



Skoltech

Sequence 2 sequence with attention



Skoltech

Attention: formulas

- First RNN produces encoder hidden states $\boldsymbol{h}_1, \dots, \boldsymbol{h}_{T_x} \in \mathbb{R}^h$
- Decoder hidden state $\boldsymbol{s}_t \in \mathbb{R}^h$ at time step t
- Attention scores for step t:
  $\boldsymbol{e^t} = [\boldsymbol{s}_t^{\boldsymbol{T}} \boldsymbol{h}_1, \dots, \boldsymbol{s}_t^{\boldsymbol{T}} \boldsymbol{h}_{T_x}] \in \mathbb{R}^{T_x}$
- Softmax to get attention distribution: all values are positive, sum of all values is 1:
  $\boldsymbol{\alpha^t} = \mathrm{softmax}(\boldsymbol{e^t}) \in \mathbb{R}^{T_x}$
- Attention output $\boldsymbol{a}_t$ is a weighted sum of hidden states:

$$\boldsymbol{a}_t = \sum_{i=1}^{T_x} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- We concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state $\boldsymbol{s}_t$ and proceed to the non-attention part of our seq2seq model

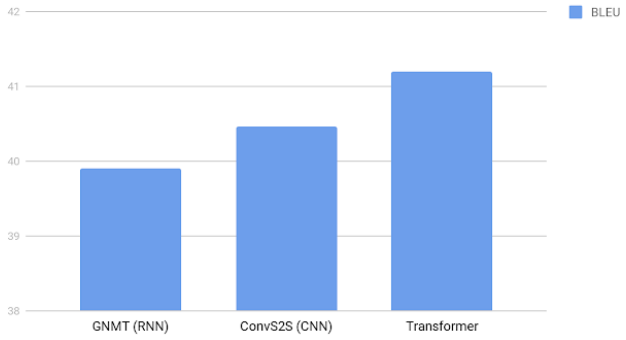
$$[\boldsymbol{a}_t, \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is just great

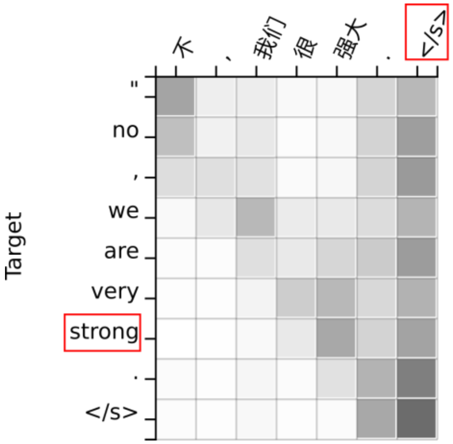
- Significantly improves performance of NMT
- Solves the bottleneck problem
  - All encoder tokens are connected to all decoder tokens
- No more vanishing gradients
  - All to All connection
- Provides some interpretability
  - see alignment figure

- Similar to RNN seq2seq, but greater!

English French Translation Quality



BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to French translation benchmark.



Skoltech

Attention is a general deep learning idea

We can use attention in many architectures and many tasks

- Other NLP problems
- Sequential data processing
- Graph Neural Networks

Key value interpretation:

s_i - query to a database — Hidden state of the decoder

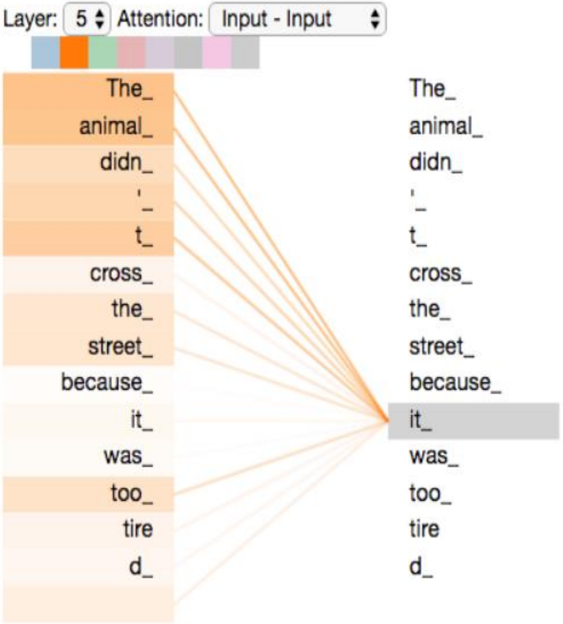
k_i - keys in the database — Hidden state of the encoder

h_i - values in the database — Hidden state of the encoder

- Calculate correspondence $e(s_i, k_i)$
- Calculate weights on the base of correspondence values
- Extract information as weighted sum of values $\sum_{i=1}^{n} \alpha_i \boldsymbol{h}_i$

Skoltech

Interpretability of attention



Remove a token i^* with max attention…

Remove random: Decision flip? (columns) / **Remove i^*: Decision flip?** (rows)

	Yahoo: Yes	Yahoo: No	IMDB: Yes	IMDB: No
Yes	0.5	8.7	2.2	12.2
No	1.3	89.6	1.4	84.2

	Amazon: Yes	Amazon: No	Yelp: Yes	Yelp: No
Yes	2.7	7.6	1.5	8.9
No	2.7	87.1	1.9	87.7

Figure source:
https://jalammar.github.io/illustrated-transformer/,
https://github.com/jessevig/bertviz

Serrano, Sofia, and Noah A. Smith. "Is attention interpretable?." arXiv preprint arXiv:1906.03731 (2019). ACL 2019.

General attention idea

Skoltech

Attention

We need a representation of an object.

A “coarse” description of an object is available.

For other objects we have more detailed stored information.

We use attention to extract these information

Skoltech

Single-object key value interpretation

$\boldsymbol{q}_i$ - query to a database

$\boldsymbol{k}_j$ - keys in the database

$\boldsymbol{v}_j$ - values in the database

We calculate attention scores

$\alpha_j = e(\boldsymbol{q}_i, \boldsymbol{k}_j) = \boldsymbol{s}_i^T \boldsymbol{k}_j$ (other distances also possible)

$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{\alpha})$

Then we extract the information as weighted sum of values

$\mathbf{a}_\mathrm{i} = \sum_{j=1}^{T_x} \alpha_j \boldsymbol{v}_j$

Skoltech

Matrix key value interpretation

$\boldsymbol{q}_i$ - query to a database

$\boldsymbol{k}_j$ - keys in the database

$\boldsymbol{v}_j$ - values in the database

We calculate correspondences

$$A(q, K, V) = \sum_i \frac{\exp(\boldsymbol{q}_i^T \boldsymbol{k}_j)}{\sum_l \exp(\boldsymbol{q}_i^T \boldsymbol{k}_l)} \boldsymbol{v}_j$$

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

Skoltech

“Databases” for the attention

- Nodes in a graph
- Tokens in a sequence (we need to specify the position, as the order is important)
- Sets of objects

Single-object key value interpretation

$\boldsymbol{q}_i$ - query to a database	Hidden state of the *decoder*
$\boldsymbol{k}_j$ - keys in the database	Hidden state of the *encoder*
$\boldsymbol{v}_j$ - values in the database	Hidden state of the *encoder*

We calculate attention scores

$\alpha_j = e(\boldsymbol{q}_i, \boldsymbol{k}_j) = \boldsymbol{s}_i^T \boldsymbol{k}_j$ (other distances also possible)

$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{\alpha})$

Then we extract the information as weighted sum of values

$\mathbf{a}_\mathrm{i} = \sum_{j=1}^{T_x} \alpha_j \boldsymbol{v}_j$

Skoltech

Scaled attention values

For large dimension of the space of keys d_k:

- Large variances dot products $\boldsymbol{q}_i^T \boldsymbol{k}_j$
- Softmax only pays attention to some keys
- Gradients are small, hard to learn

Old formula:

$$A(Q, K, V) = \mathrm{softmax}\left(\mathrm{QK^T}\right)\mathrm{V}$$

New scaled formula:

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

Transformers with self-attention

Skoltech

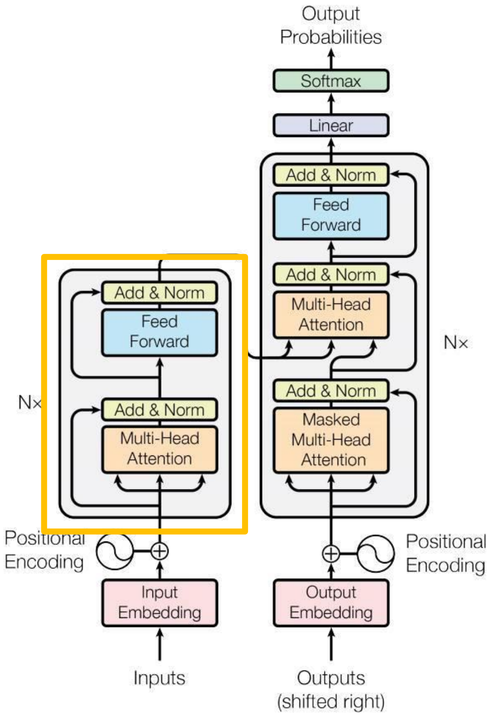
Transformer is based on the same idea

Now we completely drop RNN part

Also we repeat *self-attention* many times

Further we'll consider separate parts:

- Multi-head attention
- Feed Forward



Skoltech

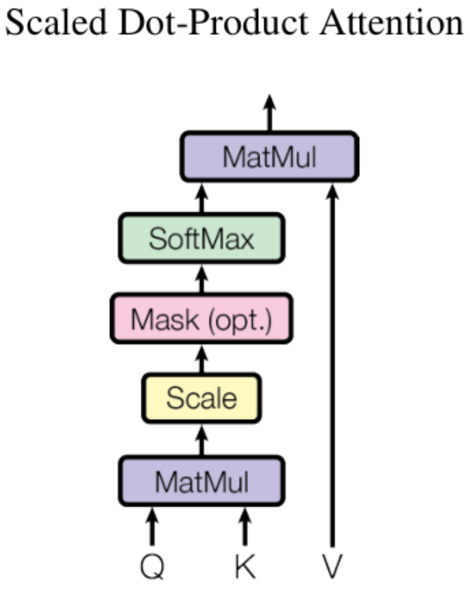
https://pytorch.org/tutorials/beginner/transformer_tutorial.html

Attention / Self-attention block

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

d_k is the dimension of query and key,
we scale to take control of large values of dot-product in high dimensions

A possible option is to replace scaled dot-product used here with additive attention: a single-hidden layer neural network.



query key value

“Attention is all you need” paper

Self-attention block

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

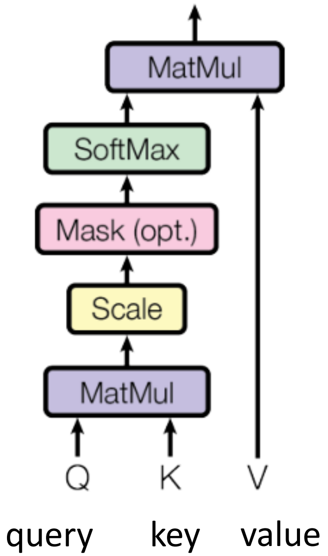
We produce queries, keys, and values using initial word embeddings

$$\mathrm{Q} = \mathrm{XW}^{\mathrm{Q}}, \dim\left(\mathrm{W}^{\mathrm{Q}}\right) = \mathrm{d}_{\mathrm{x}} \times \mathrm{d}_{q},$$

$$\mathrm{K} = \mathrm{XW}^{K}, \dim\left(\mathrm{W}^{\mathrm{Q}}\right) = \mathrm{d}_{\mathrm{x}} \times \mathrm{d}_{k},$$

$$\mathrm{V} = \mathrm{XW}^{V}, \dim\left(\mathrm{W}^{\mathrm{Q}}\right) = \mathrm{d}_{\mathrm{x}} \times \mathrm{d}_{v},$$

Scaled Dot-Product Attention

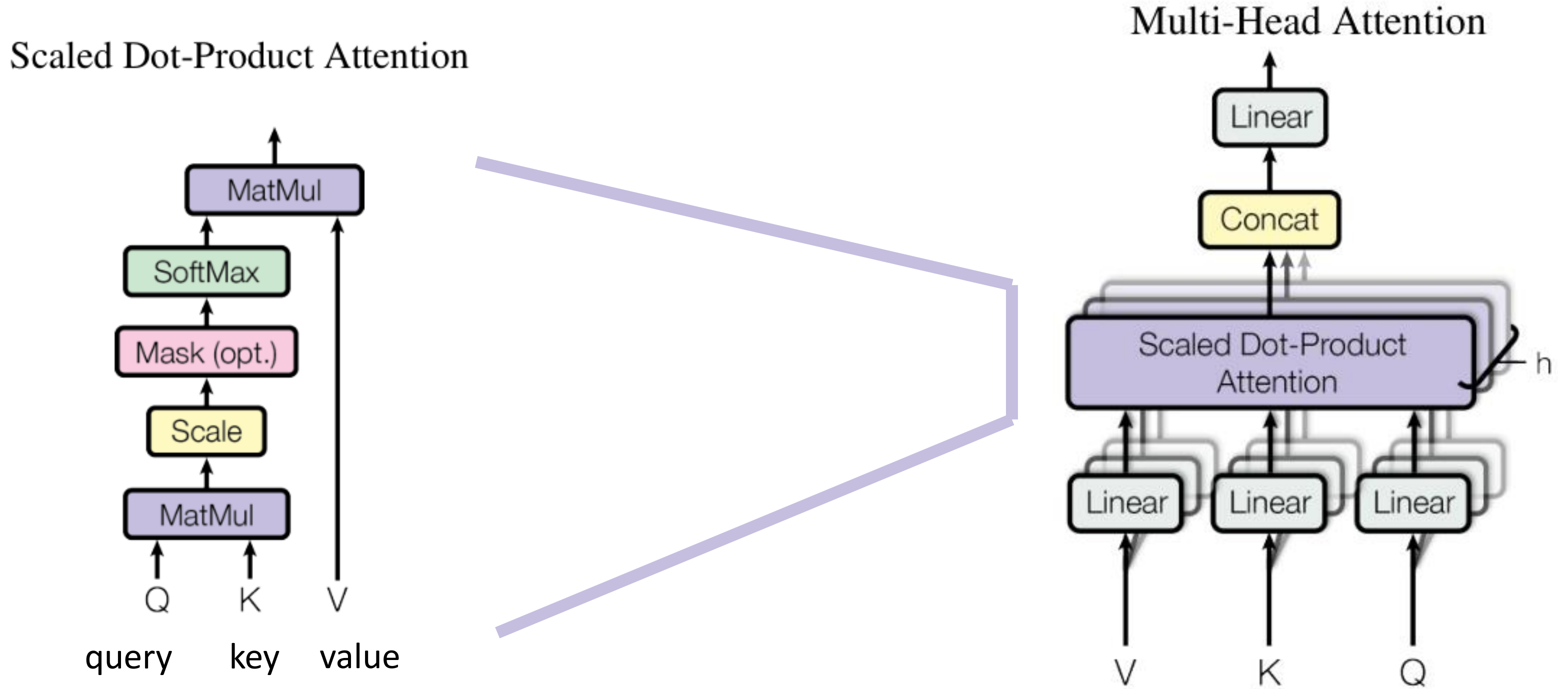


“Attention is all you need” paper

Multi-Head attention

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

h heads in total



“Attention is all you need” paper

Full block

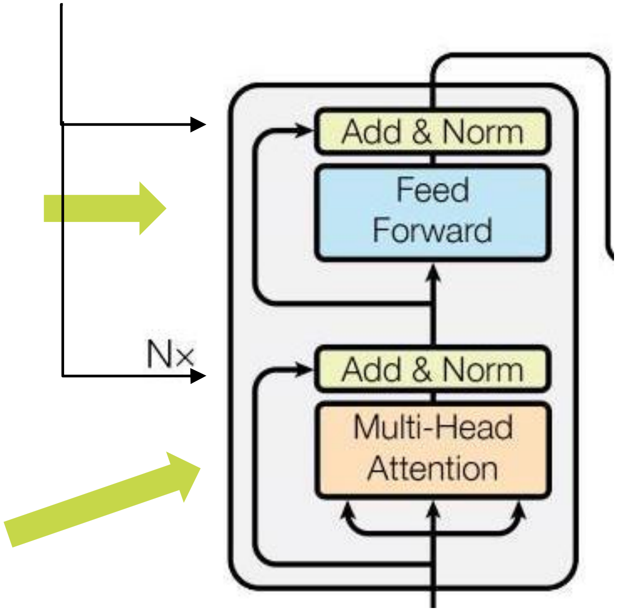
$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$

Two linear transformation with ReLU activation in between

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Multi-head attention

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

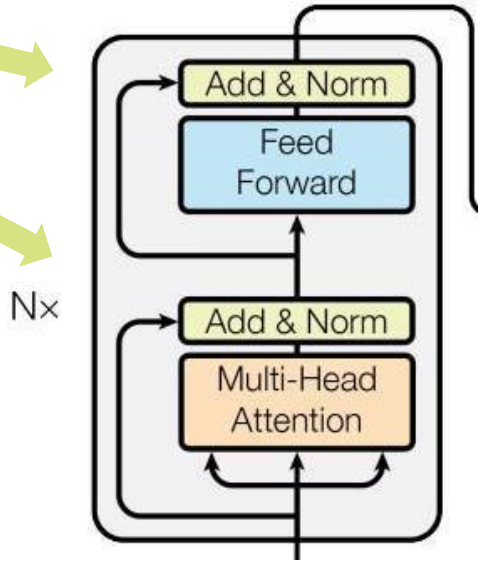


There are 6 consecutive *Full blocks* in the paper transformer architecture

"Attention is all you need" paper

Full block: normalization and residual connection

$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$$



Why? Speeds up training!

- Similar to Batch Normalization
- But can be used with batch size 1
- Can be used with RNNs and Transformers

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

Skoltech

Layer Normalization, https://arxiv.org/abs/1607.06450

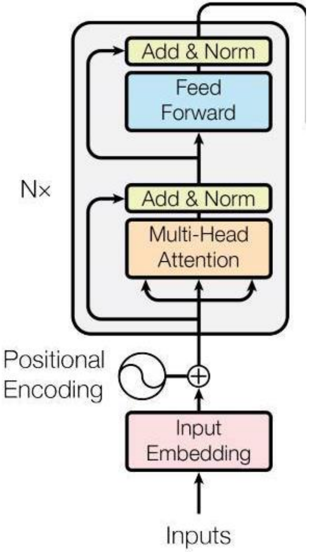
Position encoding

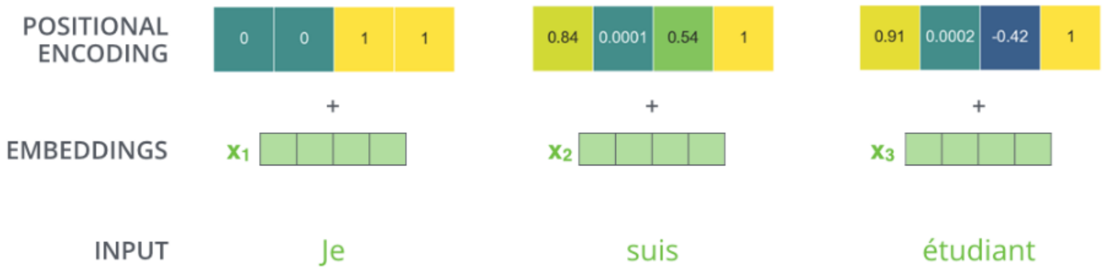
In addition to usual embeddings of inputs we use position encoding to capture position

They are not one-hot vectors, as we want to handle various-length sequences

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

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





Real example of positional encoding with a toy embedding size of 4

“Attention is all you need” paper http://jalammar.github.io/illustrated-transformer/

Skoltech

Transformer training

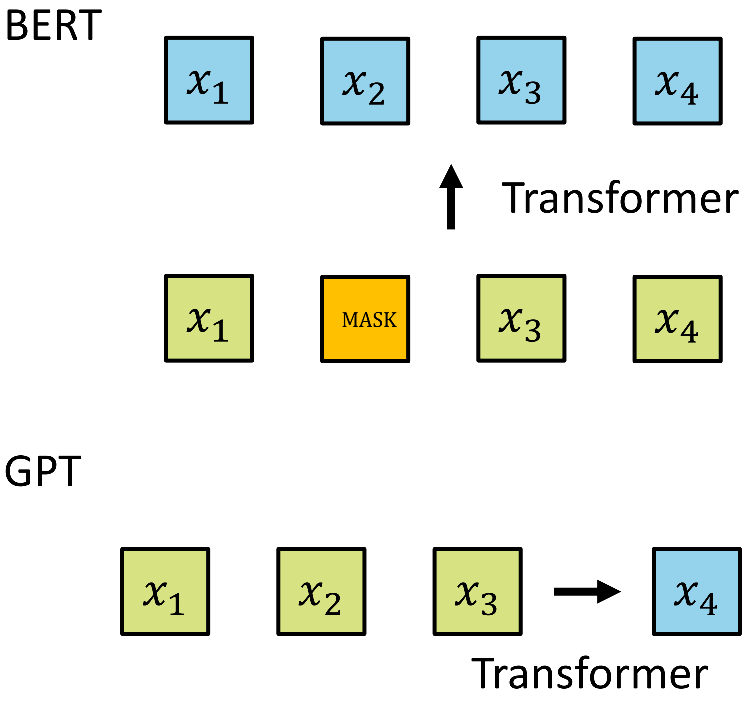
- Masked language model:

Replace random tokens with masks, try to reconstruct them using a Neural network

- Next token prediction

Predict next token

Self-supervised learning: We don't need labeled examples, we just create them



Efficiency of transformers

Brown, T. B., Mann, B., Ryder, N. et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

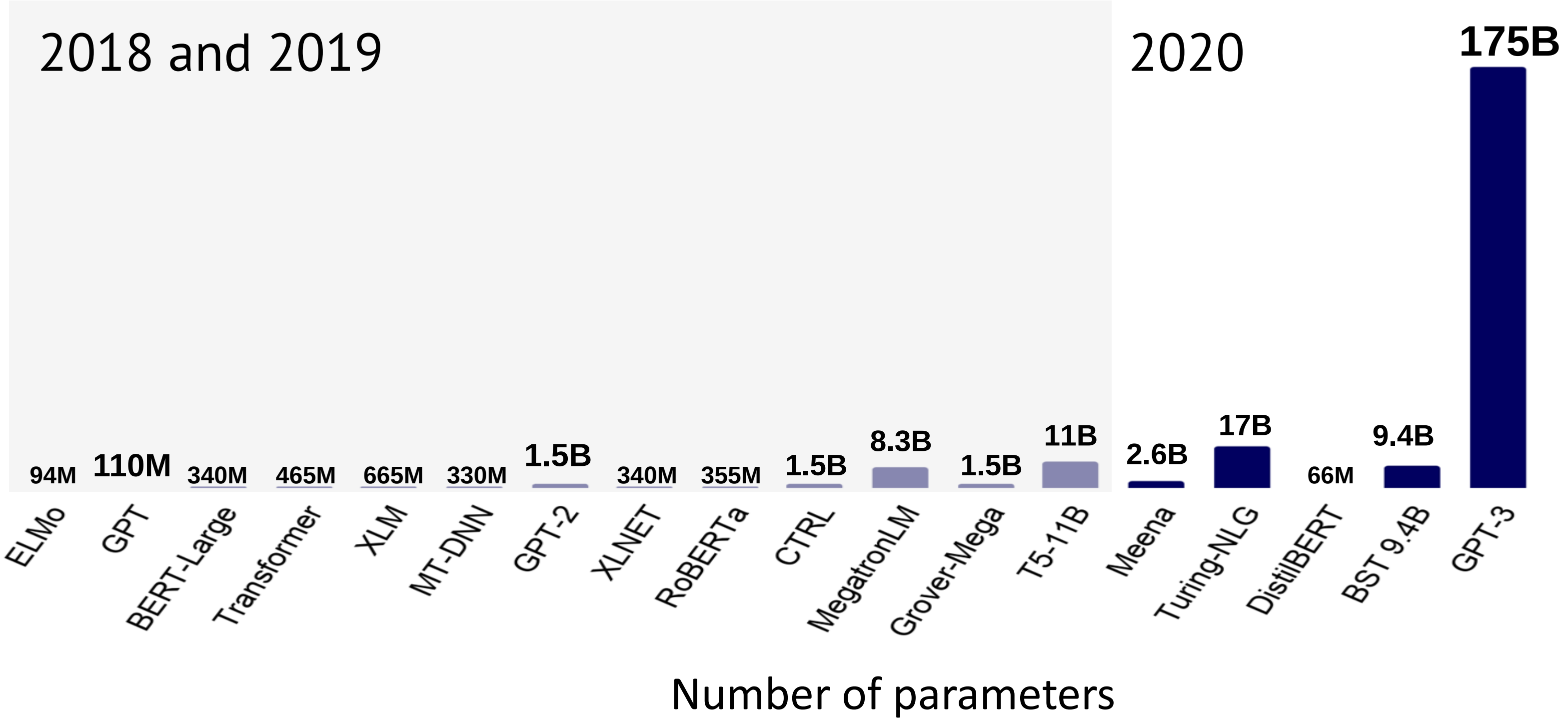
Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p-value)	"I don't know" assignments
Control (deliberately bad model)	86%	83%–90%	-	3.6 %
GPT-3 Small	76%	72%–80%	3.9 (2e-4)	4.9%
GPT-3 Medium	61%	58%–65%	10.3 (7e-21)	6.0%
GPT-3 Large	68%	64%–72%	7.3 (3e-11)	8.7%
GPT-3 XL	62%	59%–65%	10.7 (1e-19)	7.5%
GPT-3 2.7B	62%	58%–65%	10.4 (5e-19)	7.1%
GPT-3 6.7B	60%	56%–63%	11.2 (3e-21)	6.2%
GPT-3 13B	55%	52%–58%	15.3 (1e-32)	7.1%
GPT-3 175B	52%	49%–54%	16.9 (1e-34)	7.8%

Table 3.11: Human accuracy in identifying whether short (~200 word) news articles are model generated. We find that human accuracy (measured by the ratio of correct assignments to non-neutral assignments) ranges from 86% on the control model to 52% on GPT-3 175B. This table compares mean accuracy between five different models, and shows the results of a two-sample T-Test for the difference in mean accuracy between each model and the control model (an unconditional GPT-3 Small model with increased output randomness).

Transformer models are monstrous



Skoltech

Training cost estimation: 10-50 MLN US$ for GPT-3

Self-attention block complexity

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

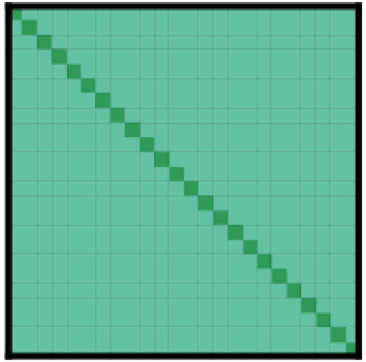
Memory complexity is $O(d_x^2)$
Computational complexity is $O(d_x^2)$

Max sequence size in popular models (e.g. BERT) is only $n = d_x = 512$ tokens

In modern models tokens are parts of words

Output

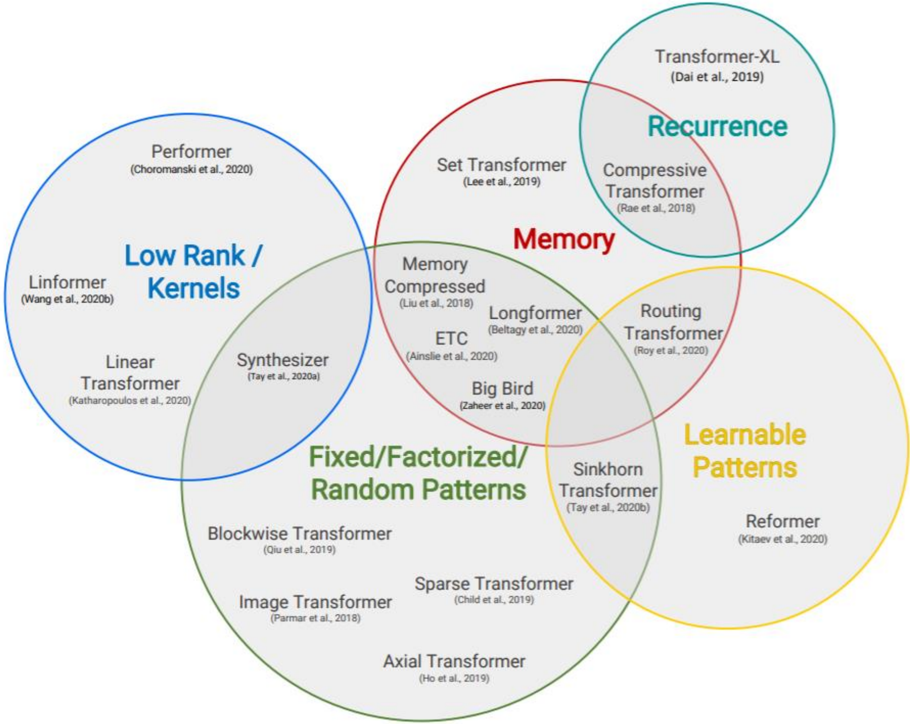
Input



Full n^2 attention

Transformer for long sequences

To work with sequences with significant length we should decrease memory consumption and computation complexity $O(n^2)$



Skoltech

Tay, Yi, et al. "Efficient transformers: A survey." *arXiv preprint arXiv:2009.06732* (2020).

BigBird approach

Memory requirements, n is the sequence length:

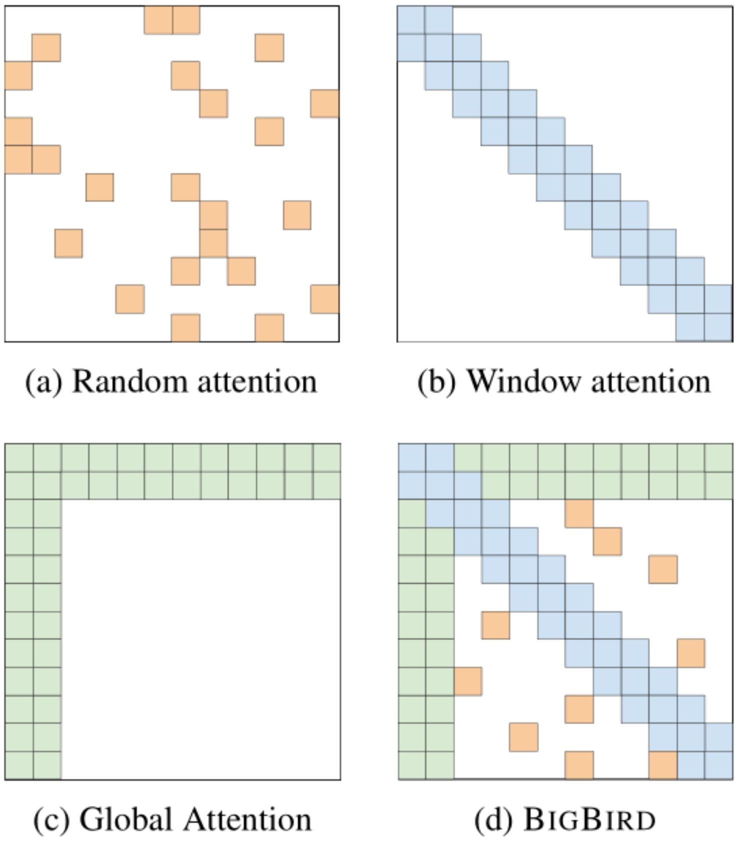
Random attention requires $O(r \cdot n)$

Sliding window requires $O(h \cdot n)$, h is the window size

Global requires $O(g \cdot n)$, g is the global tokens number

BigBird combines 3 types of attention mechanism. All of them have linear complexity.

Total: $\mathbf{O((r + h + g) \cdot n)}$



Zaheer, Manzil, et al. "Big bird: Transformers for longer sequences." *Advances in Neural Information Processing Systems* 33 (2020): 17283-17297.

Skoltech

Take-home messages

- Attention mechanism is for “information extraction”
- Transformer architecture is a multi-layer architecture based on the self attention layer