Recurrent Neural Networks

15th November, 2022

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)

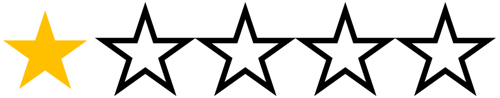
- Now we have input features for all points

- Let’s apply our favorite ML regression algorithm

dmlc XGBoost

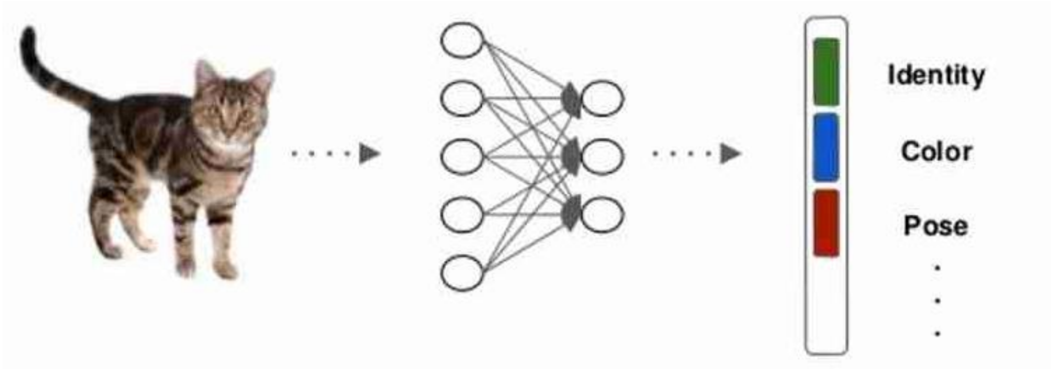
Skoltech

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

One example problem: why do we need Neural networks?



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

Idea 1: use previous word(s)

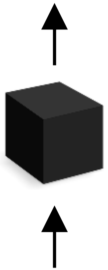
Problem 1: long-term dependencies

"**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

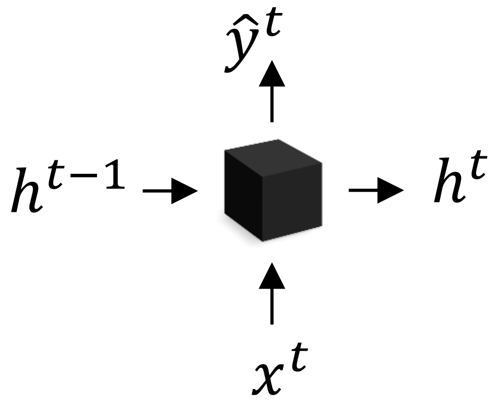
Model Design Criteria

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
2. Long-term memory
3. Maintain order information
4. Natural preprocessing

YES (if one to one)

NO 🙁

NO 🙁

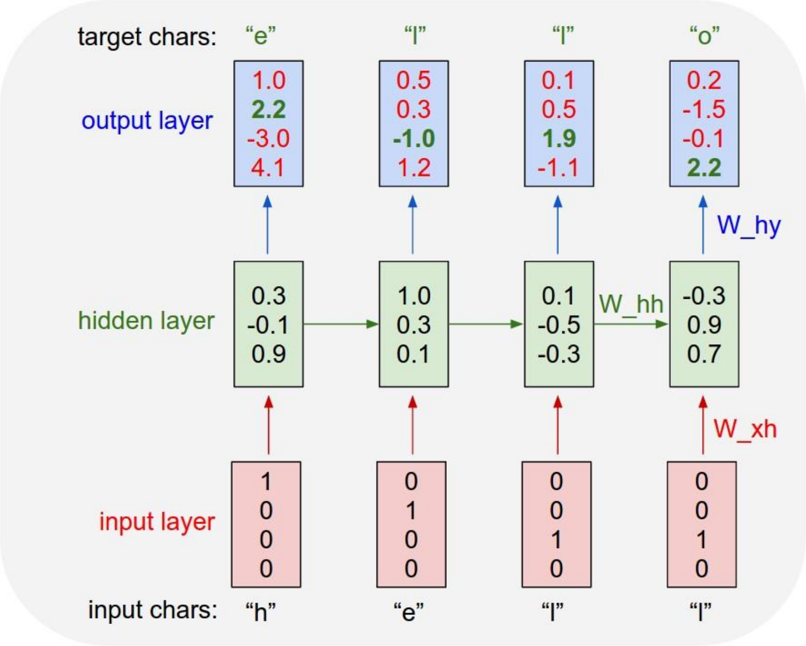
a kind of

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. □

Conclusions

- Classic ML can’t handle semi-structured data common in sequential data processing

- We should *learn representations* via Neural Networks

- Results are nice even for relatively simple models

Skoltech

Problem statements

We want to model the conditional distribution:

$$p(y_t | y_1, \dots, y_{t-1}, x_1, \dots, x_t)$$

Our goal would be to propose such model

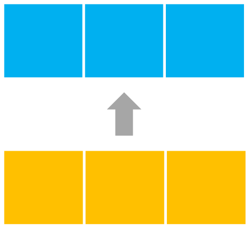
Problem statements

We also look at similar models and problems.

(a) Many-to-one

Classification

(b) One-to-one



Anomaly detection

(c) Many-to-many

Machine translation

Skoltech

Model Design Criteria

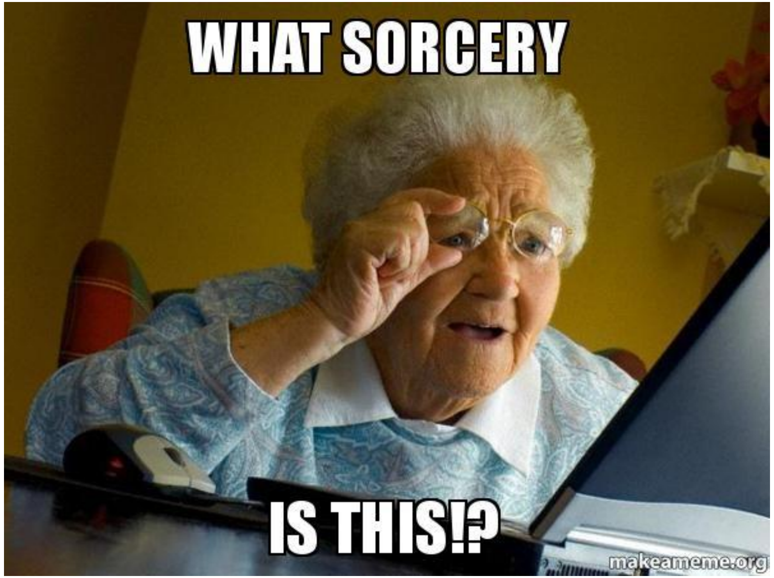
1. Variable-length sequences processing

2. Long-term memory

3. Maintain order information

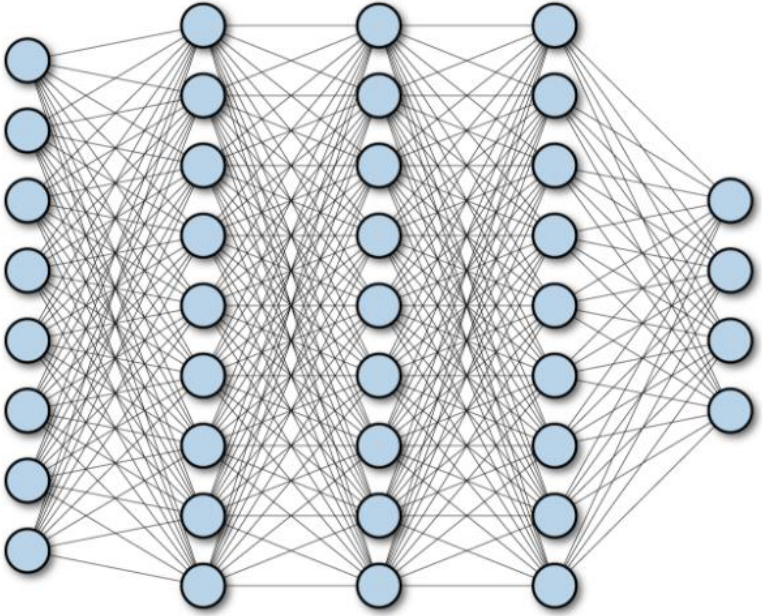
4. Natural preprocessing

5. **General end2end approach**

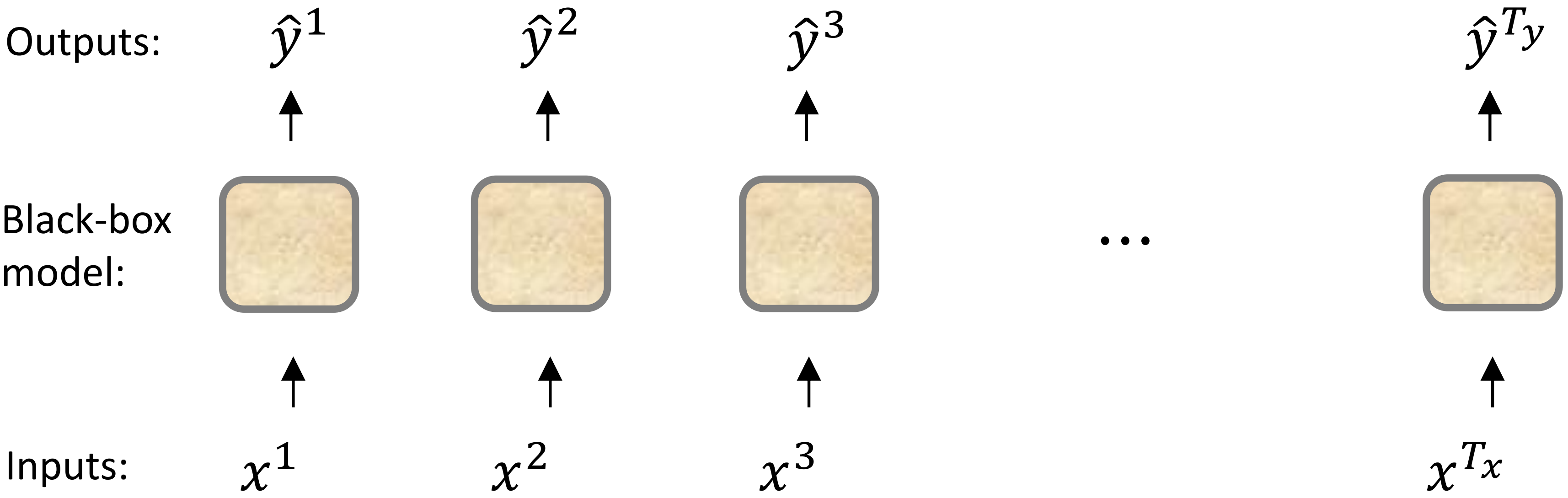


Skoltech

Fully connected neural networks?



Separate Fully-Connected Neural Networks or other separate models



We model $p(y_t|x_t)$ instead of $p(y_t|y_1, \dots, y_{t-1}, x_t)$

Why not a standard fully-connected network?

- Inputs, outputs can be different lengths in different examples
- Huge number of parameters (for length 512 it is about 256 000)
- Doesn’t share features learned across different positions of text
- Not easy to separate contribution from different *y*-s …*think about it as a problem to think about e.g. p(y*

Adopted from Andrew Ng

SOTA options

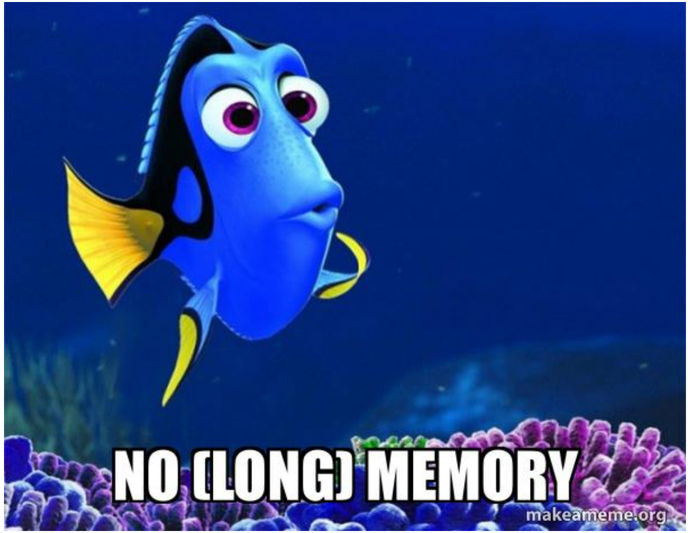
Recurrent Neural Networks	Today	SOTA in some problem
One dimensional convolutions (1D CNN)	Next lecture (also see ROCKET)	SOTA in another problems
Transformers	After them	SOTA in most NLP problems

SOTA results for scoring problem

		Micro Precision
Two RNN architectures	LSTM	<u>0.762</u>
	GRU	**0.765**
	1D CNN	0.745
	Transformer	0.755
	Gradient boosting	0.697

Skoltech

A vanilla Recurrent Neural Network



Skoltech

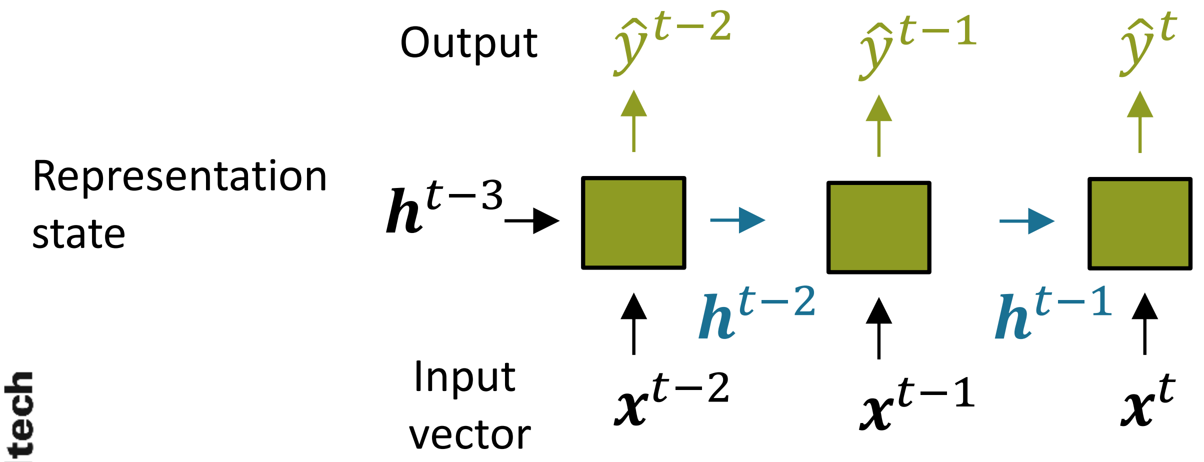
Simple RNN model

At each step:

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

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

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



Main idea:
The model inside dark green block is the same for all time moments!

$$p(y_t|y_1,\ldots,y_{t-1},x_1,\ldots,x_t) = p(y_t|\boldsymbol{h}_{t-1},x_t) = p(y_t|\boldsymbol{h}_t)$$

Skoltech

Simple RNN model block

General form:

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

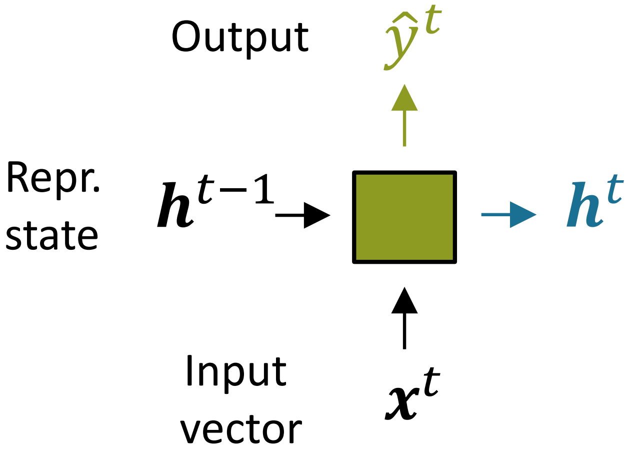
$$\hat{y}^t = f_y(\boldsymbol{h}^t)$$

Vanilla RNN model for classification:

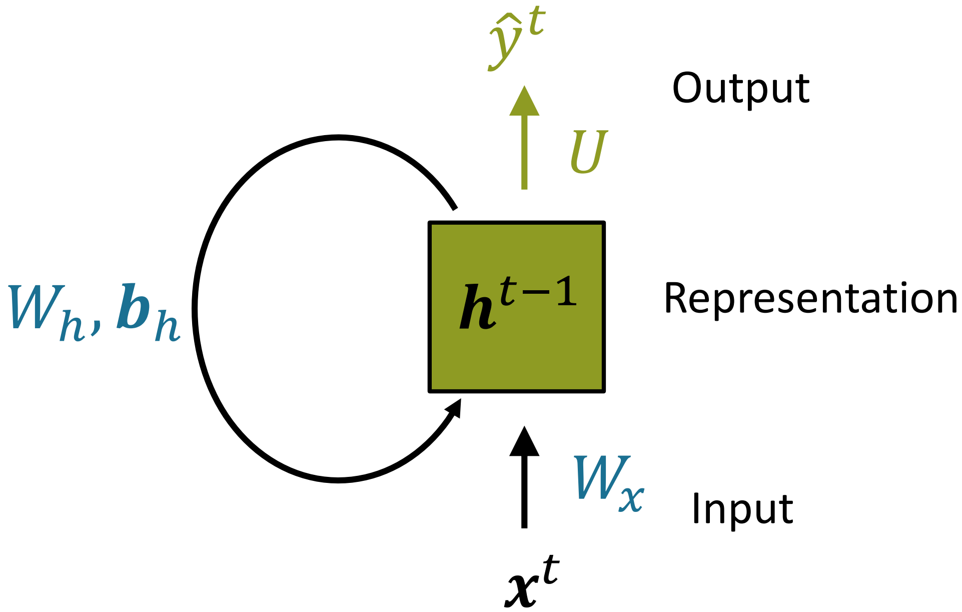
$$\boldsymbol{h}^t = \tanh(V\boldsymbol{x}^t + W\boldsymbol{h}^{t-1} + \boldsymbol{b}_h)$$

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

Parameters: V, W, U



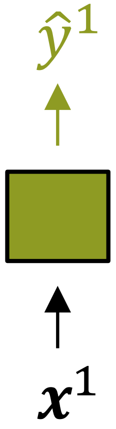
Compressed form of RNN



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

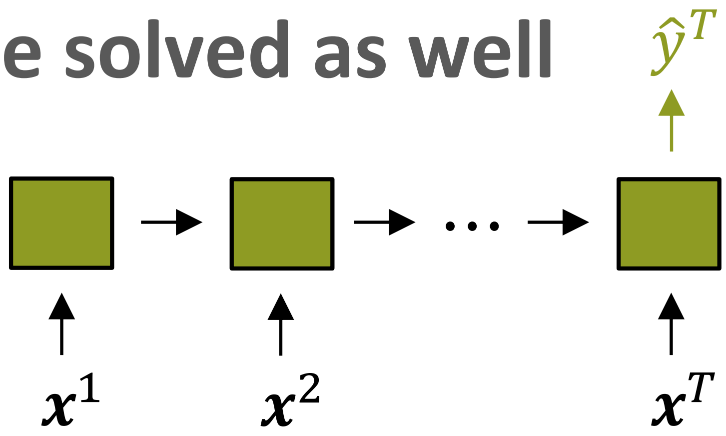
$$\hat{y}^t = \mathrm{softmax}(U\boldsymbol{h}^t + b_y)$$

Other problems are solved as well

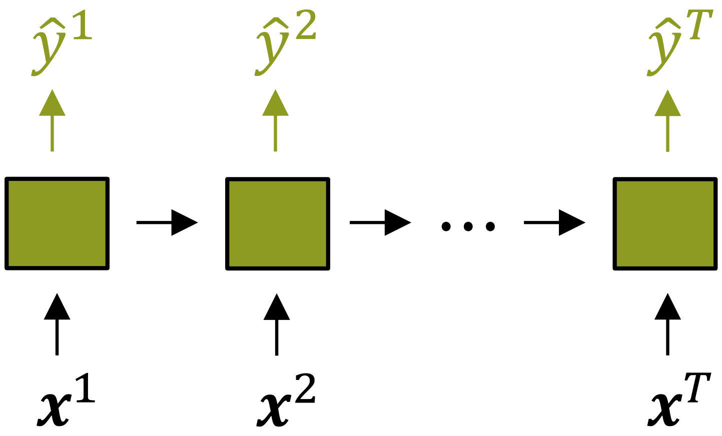


No recurrent structure

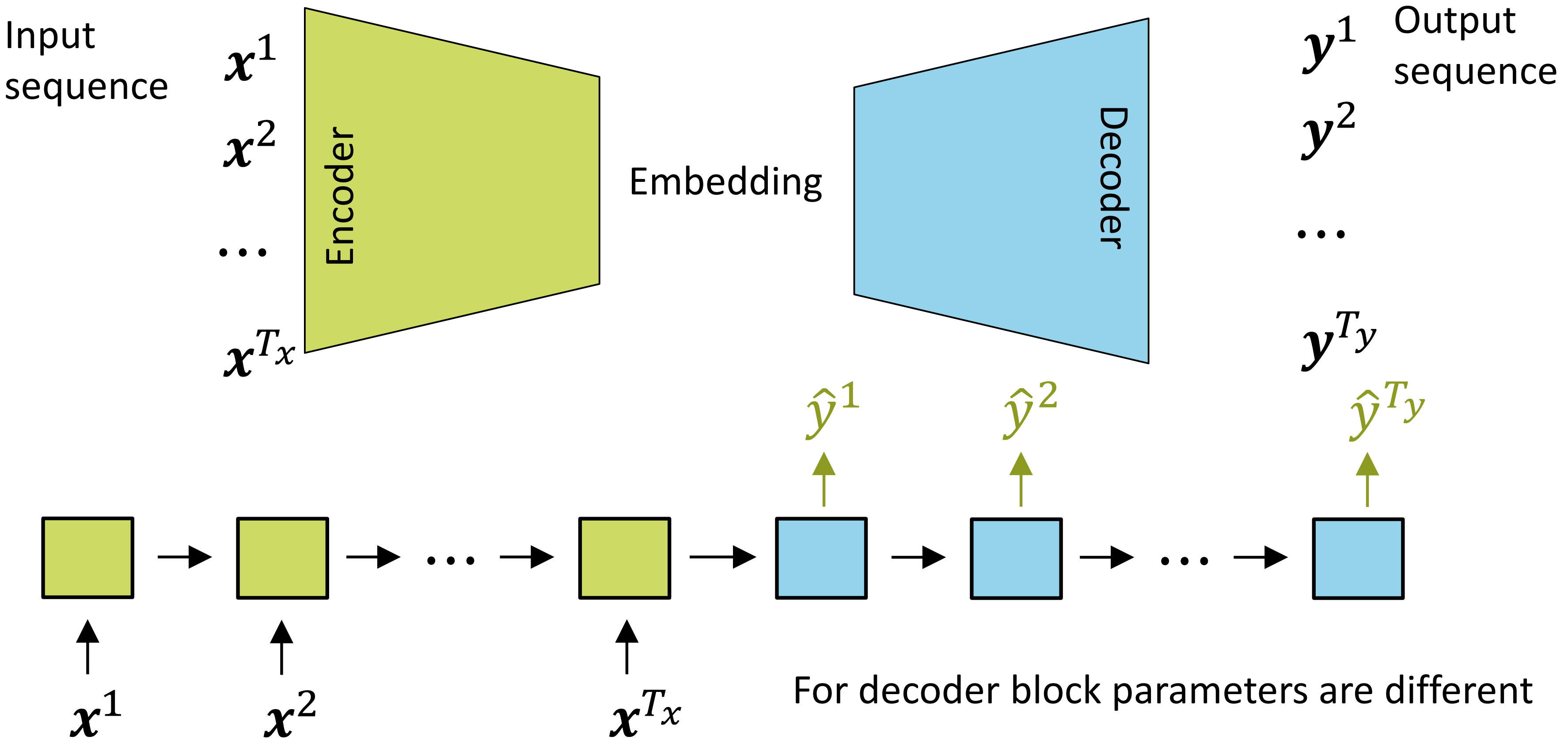
Sequence to a single output



Sequence to a sequence of outputs



A sequence to sequence model



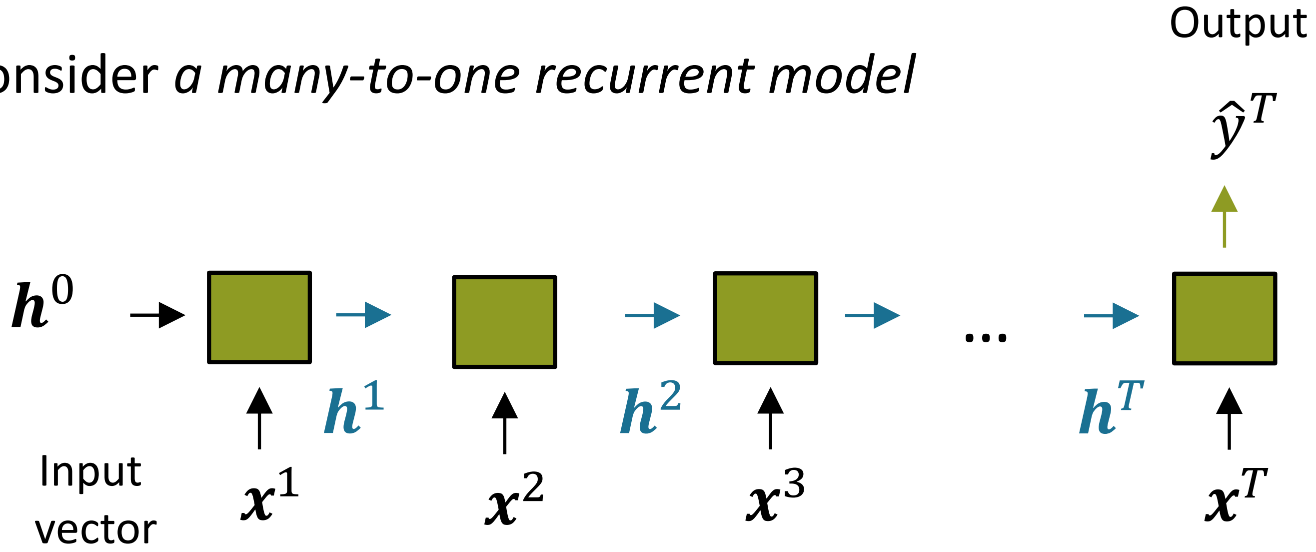
For decoder block parameters are different

Details on Recurrent Neural Networks

Skoltech

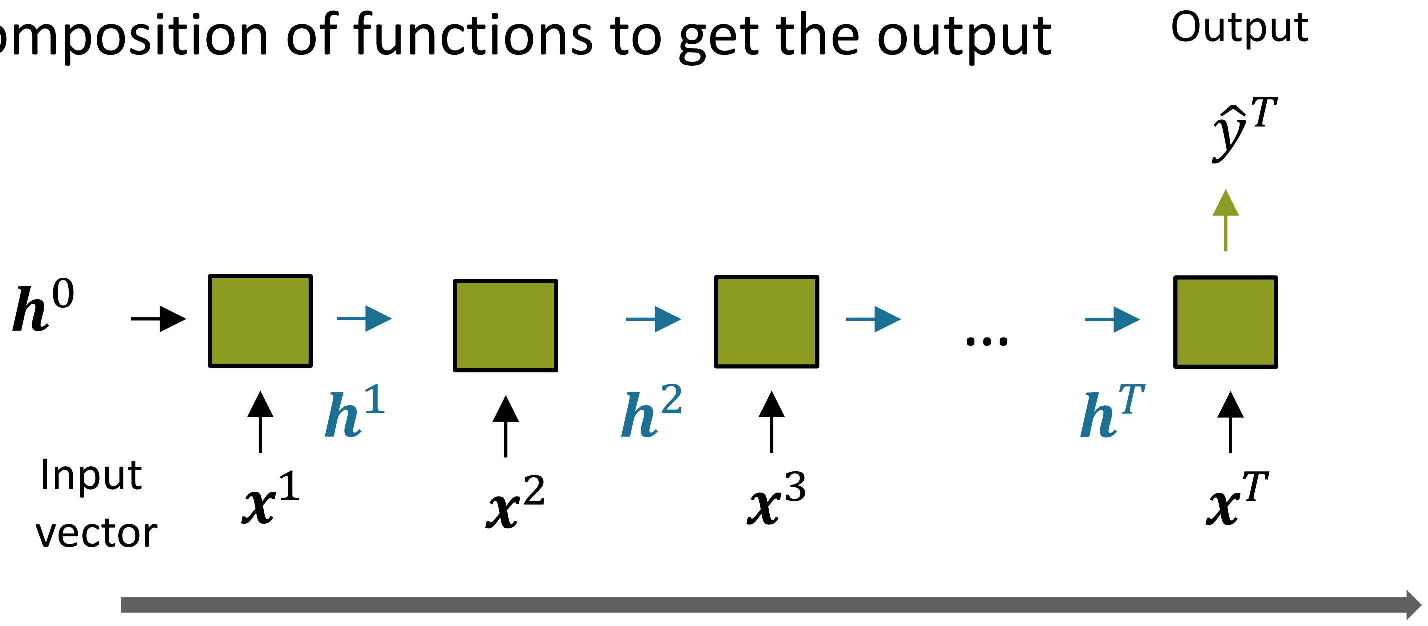
An RNN model

Let us consider *a many-to-one recurrent model*



Forward propagation through RNN

Composition of functions to get the output

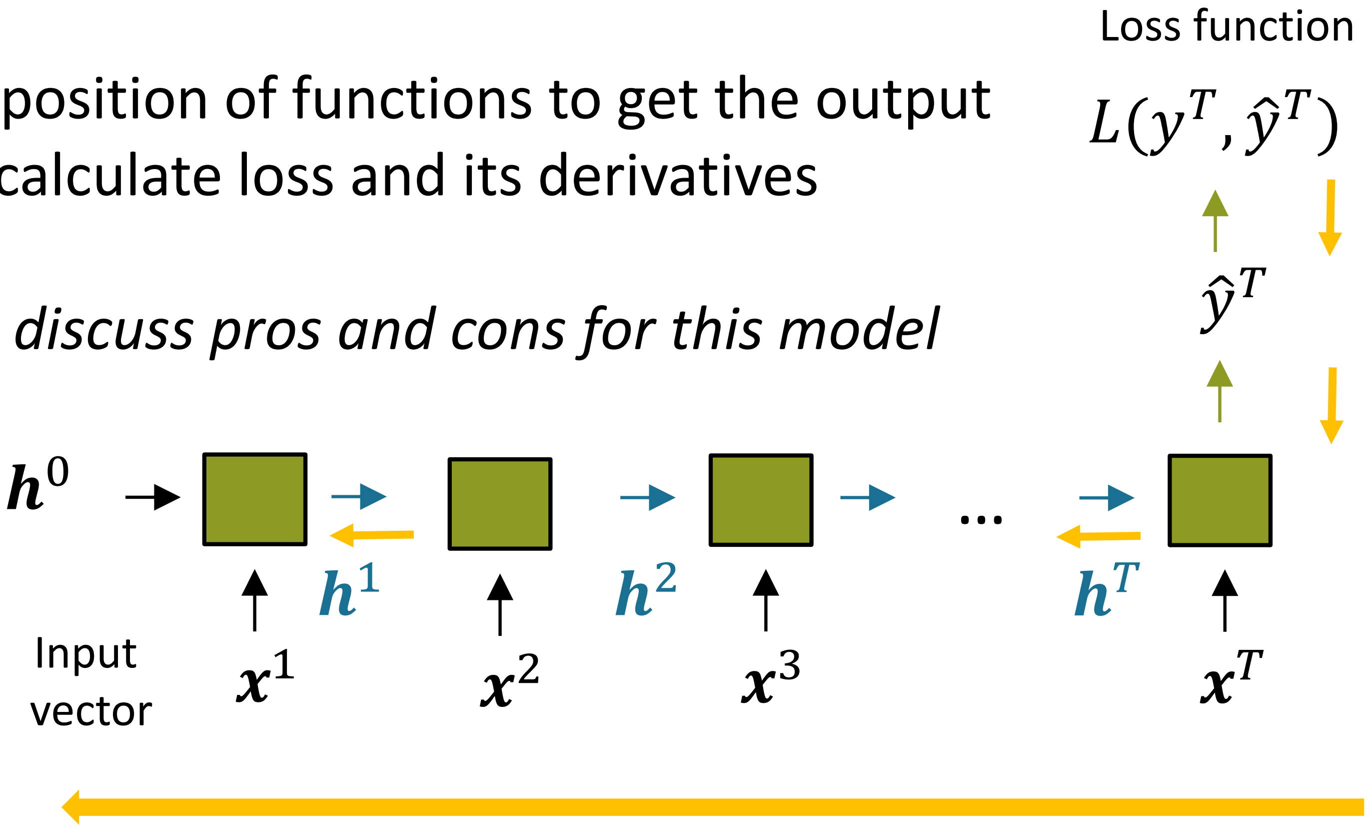


Processing block is the same for all time moments.
The processing time is linear in T: it is O(T).

Backward propagation through RNN

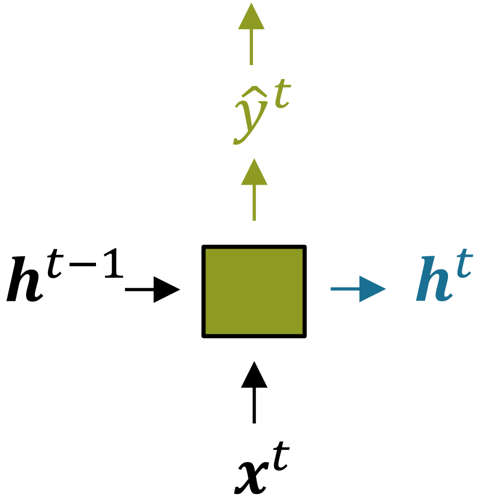
Composition of functions to get the output and calculate loss and its derivatives

Let's discuss pros and cons for this model



Backpropagation w.r.t. U

$$L_t = L^t(\hat{y}^t, y^t)$$

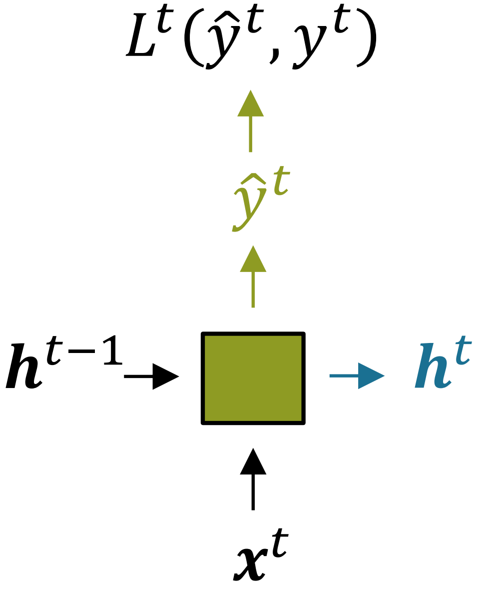


$$\boxed{\frac{\partial L_T}{\partial U} = \frac{\partial L_T}{\partial \hat{y}^T} \frac{\partial \hat{y}^T}{\partial U}}$$

Recall the formula for the output

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

Backpropagation w.r.t. W



$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial \hat{y}^T}\frac{\partial \hat{y}^T}{\partial W}$$

$$\frac{\partial L_t}{\partial \hat{y}^t}\frac{\partial \hat{y}^t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}^t}\frac{\partial \hat{y}^t}{\partial h^t}\left(\frac{\partial h^t}{\partial W} + \frac{\partial h^t}{\partial h^{t-1}}\frac{\partial h^{t-1}}{\partial W} + \cdots\right)$$

$$\frac{\partial L_t}{\partial \hat{y}^t}\frac{\partial \hat{y}^t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}^t}\frac{\partial \hat{y}^t}{\partial h^t}\sum_{i=0}^{t}\frac{\partial h^i}{\partial W}\left(\prod_{j=i+1}^{t}\frac{\partial h^j}{\partial h^{j-1}}\right)$$

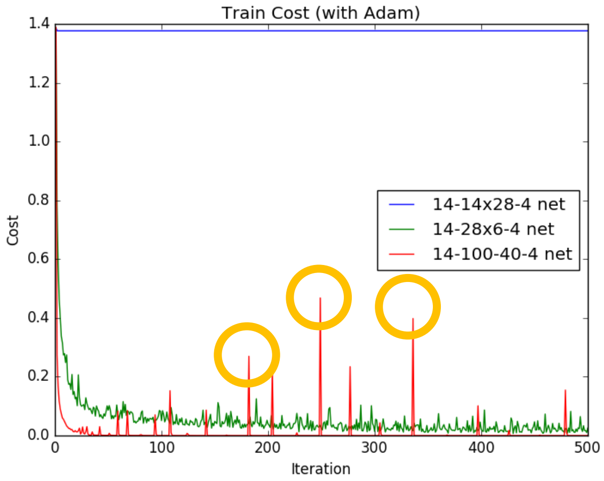
We have a product of derivatives
It can be a problem.

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

Problem: gradient explosion

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}^t}\frac{\partial \hat{y}^t}{\partial h^t}\sum_{i=0}^{t}\frac{\partial h^i}{\partial W}\left(\prod_{j=i+1}^{t}\frac{\partial h^j}{\partial h^{j-1}}\right)$$

$\left\|\frac{\partial h^j}{\partial h^{j-1}}\right\| > 1$, nonstationary (c.t. ARIMA lecture)



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

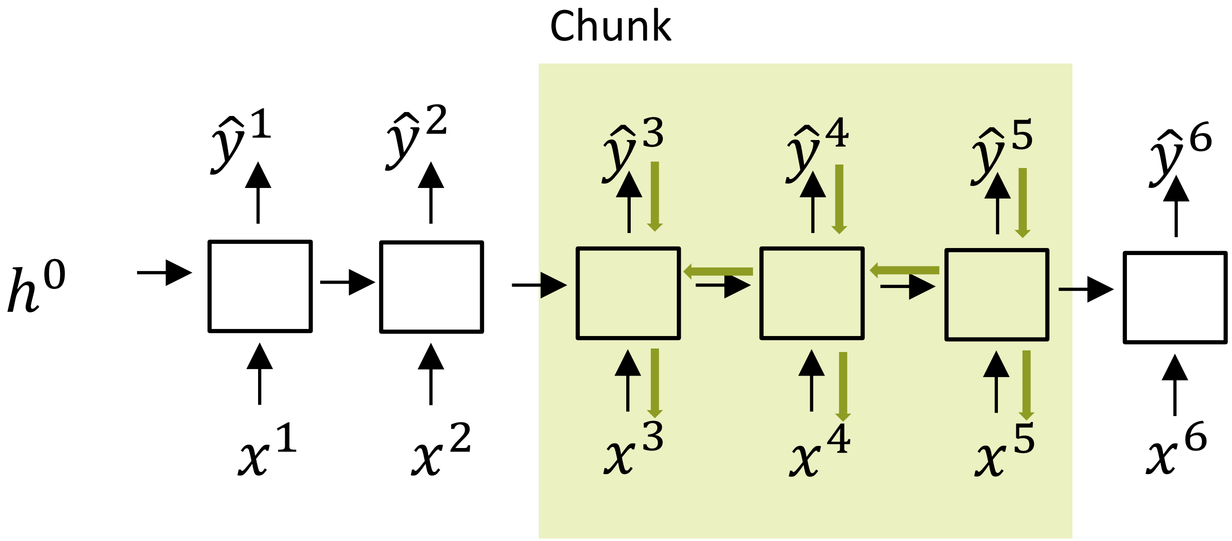
Skoltech

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

Problem: gradient explosion

Solutions:

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



Skoltech

Problems of classic RNN: gradient vanishing

A more serious problem

Many values < 1

Product << 1

$$\left\|\frac{\partial h^j}{\partial h^{j-1}}\right\| < 1$$

Can't learn long range dependences

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
- Other?

Selection of RNN architecture

LSTM

GRU

Better RNN units: LSTM and GRU

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

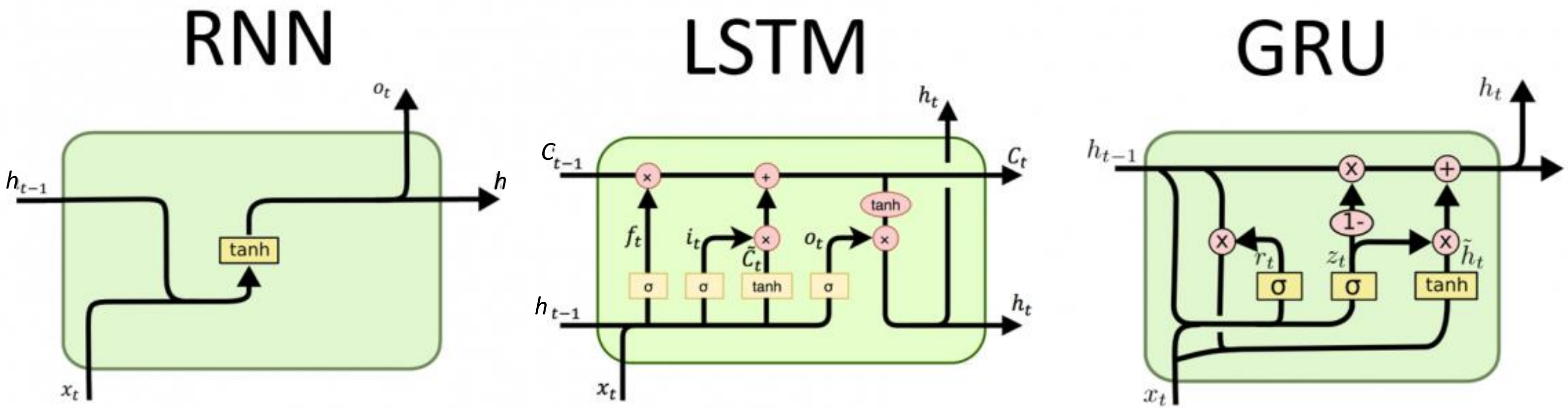


Image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

1. Schmidhuber, J., & Hochreiter, S. Long short-term memory. Neural Computations. 1997.
2. Cho, K., Van Merriënboer et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. EMNLP. 2014.

Details on how LSTM works

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

Temporal representation =
Cell representation (long term) +
Hidden state (short term)

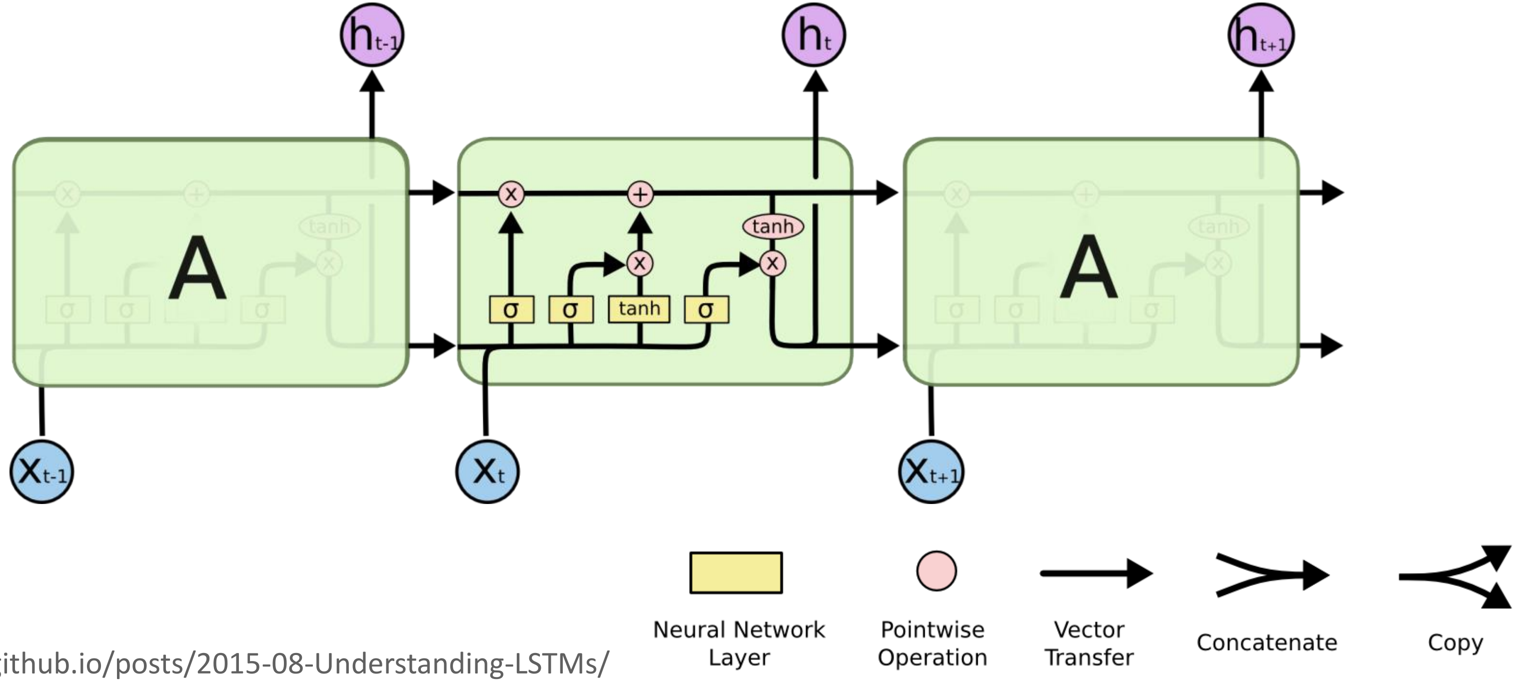
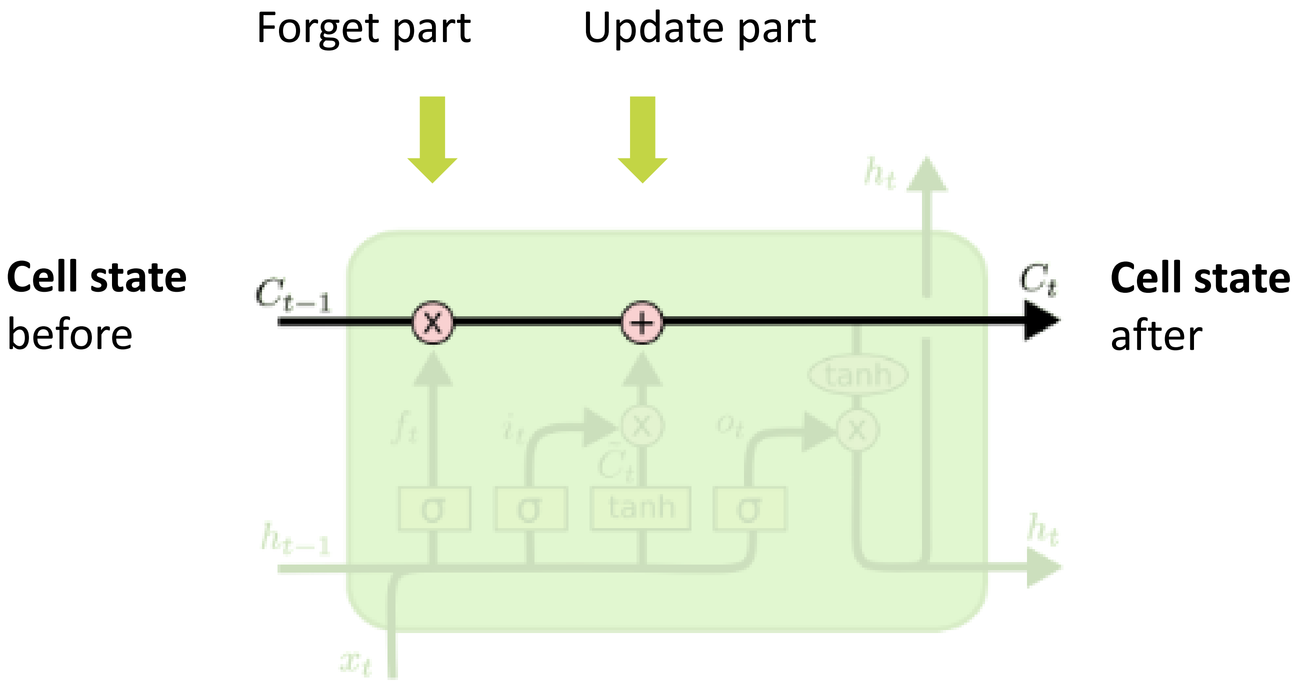


Image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Skoltech

Long term memory part – Cell state



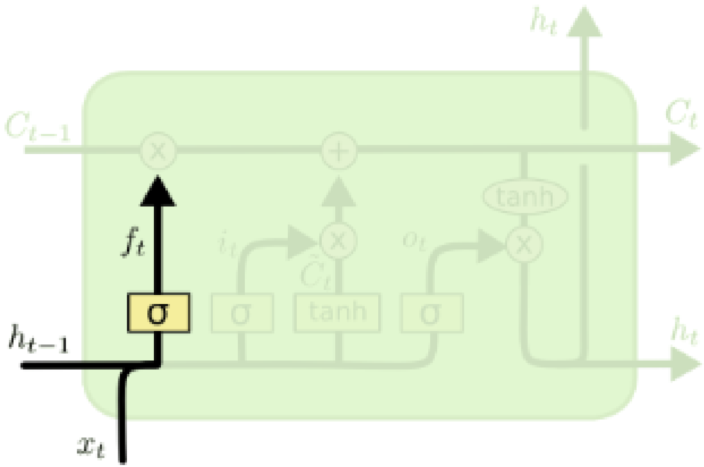
Skoltech

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

Forget part

Identify how much should we forget

hidden state



inputs

$$f_t = \sigma(W_f[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_f)$$

$[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t]$ is the concatenation

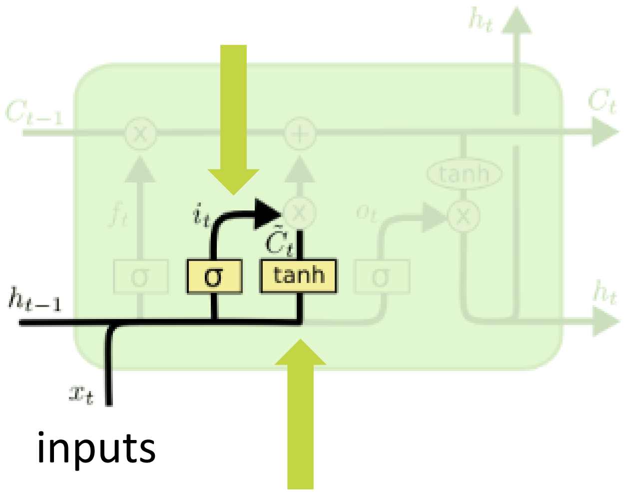
$W_f, \boldsymbol{b}_f$ are forget parameters

σ is the sigmoid function, so the output is between 0 and 1

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

Update part

Identify how much we should update



hidden state (a)

inputs

What update we do

$$\boldsymbol{i}_t = \sigma(W_i[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_i)$$

$$\tilde{\boldsymbol{c}}_t = \tanh\,(W_c[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_c)$$

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

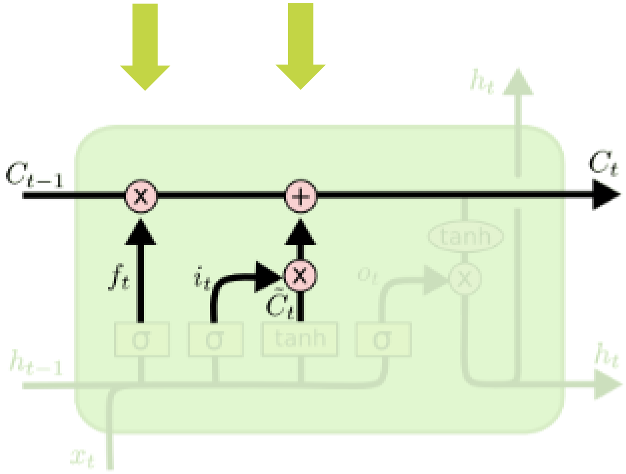
Skoltech

Long term memory part – Cell state

Forget part

Update part

Cell state before



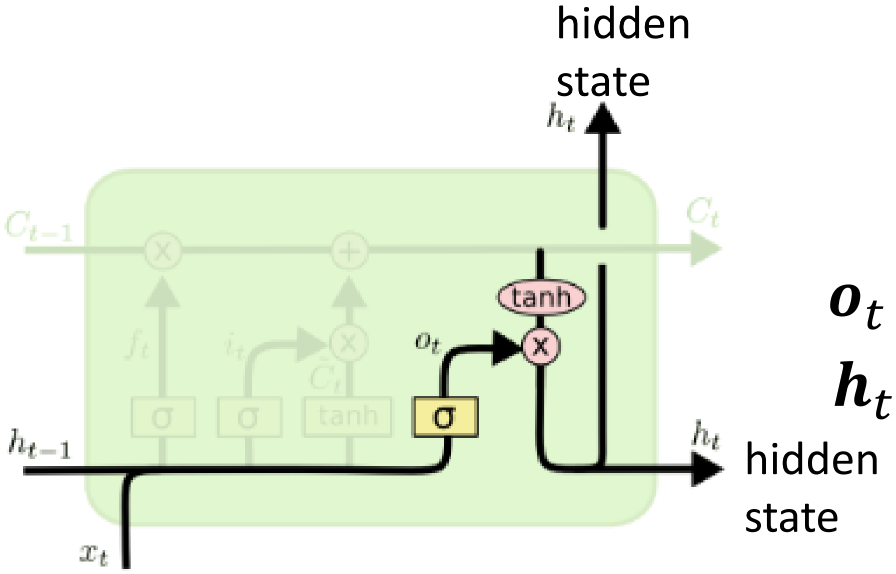
Cell state after

$$\boldsymbol{c}_t = \boldsymbol{f}_t \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \tilde{\boldsymbol{c}}_t$$

Skoltech

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

Update everything else



$$\boldsymbol{o}_t = \sigma(W_o[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_o)$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \tanh(\boldsymbol{c}_t)$$

Skoltech

Image source: 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

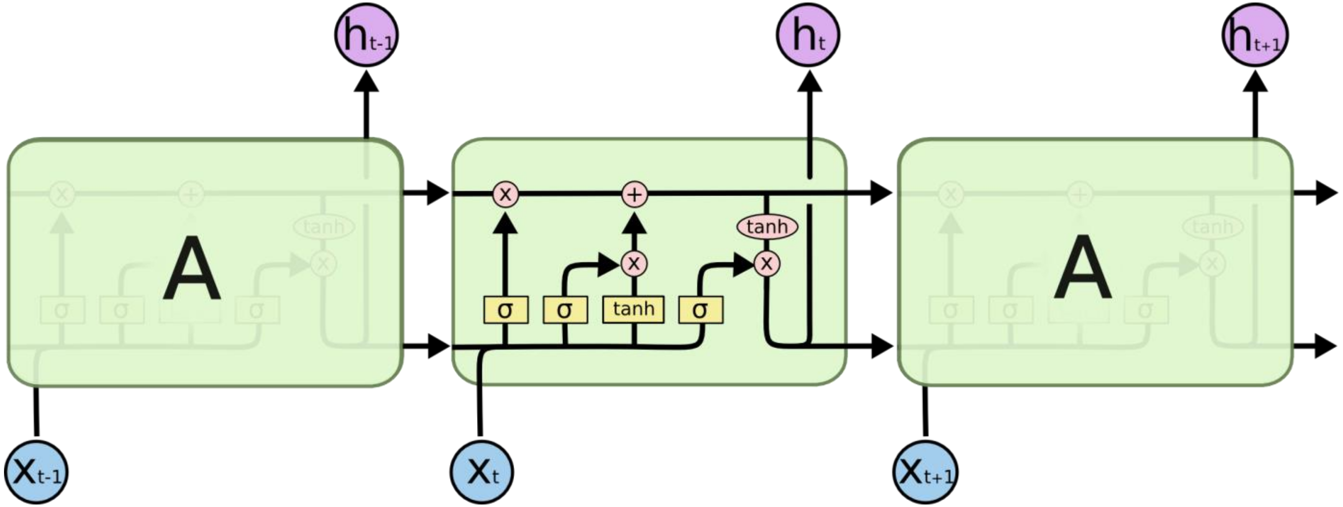


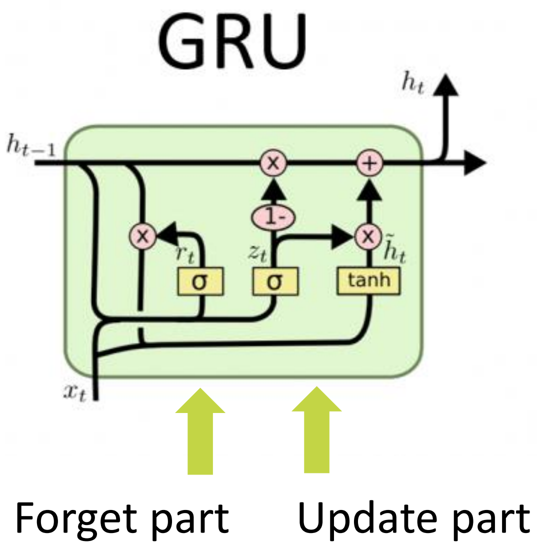
Image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

GRU – Gated Recurrent Unit

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

$$\boldsymbol{r}^t = \sigma(W_{xr}\boldsymbol{x}^t + W_{hr}\boldsymbol{h}^{t-1} + b_r)$$
$$\boldsymbol{z}^t = \sigma(W_{xz}\boldsymbol{x}^t + W_{hz}\boldsymbol{h}^{t-1} + b_z)$$

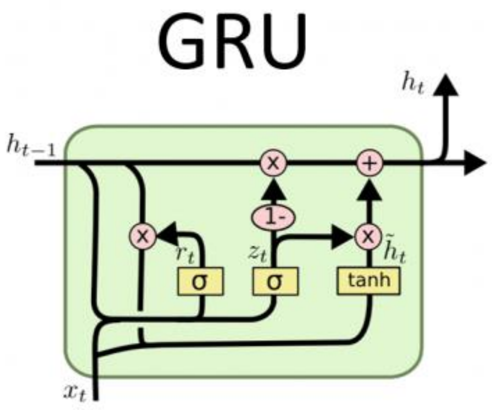
$$\widetilde{\boldsymbol{h}}^t = \tanh(W_{xh}\boldsymbol{x}^t + W_{hr}(\boldsymbol{r}^t \odot \boldsymbol{h}^{t-1}) + b_h)$$
$$\boldsymbol{h}^t = \boldsymbol{z}^t \odot \boldsymbol{h}^{t-1} + (1 - \boldsymbol{z}^t) \odot \widetilde{\boldsymbol{h}}^t$$



Skoltech

Cho, Kyunghyun et al. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.

GRU – Gated Recurrent Unit



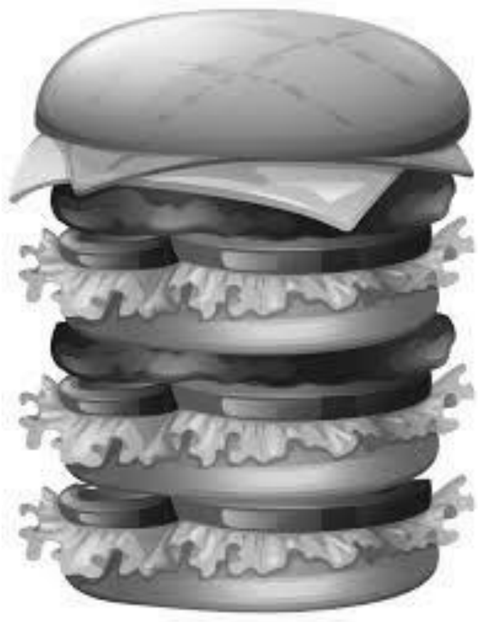
- Update gate – what to pay attention to
- Reset gate – what to forget

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

Skoltech

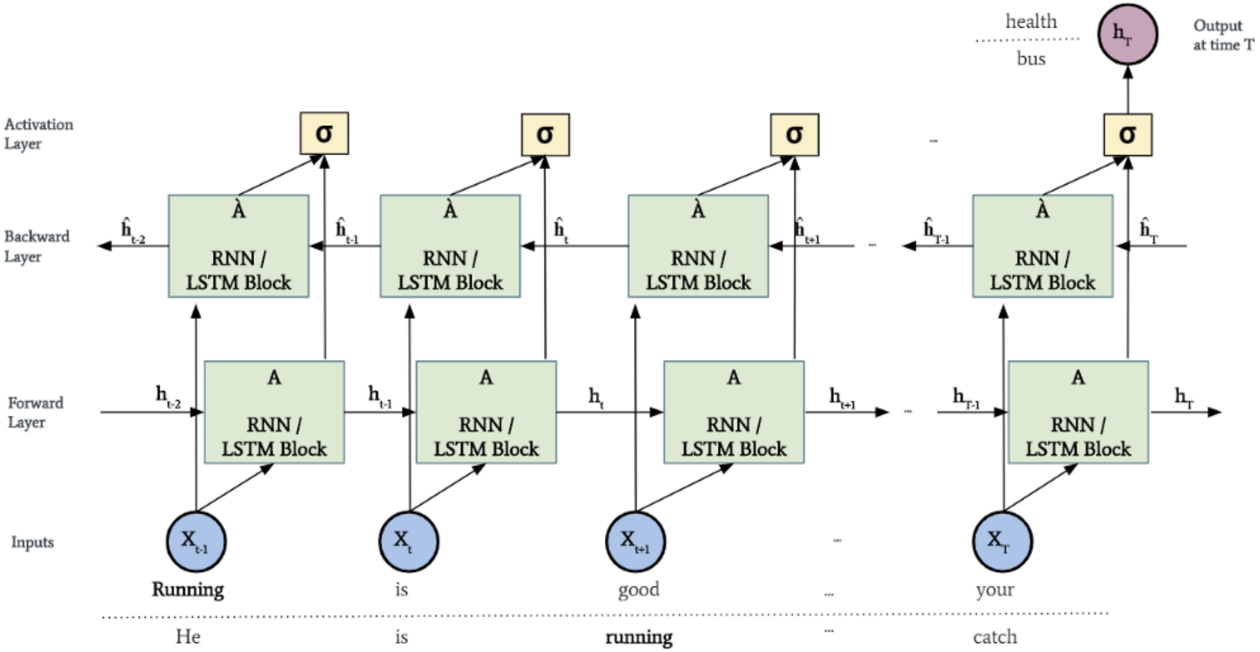
Cho, Kyunghyun et al. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.

Multilayer architectures



Skoltech

Other architectures: bidirectional LSTM



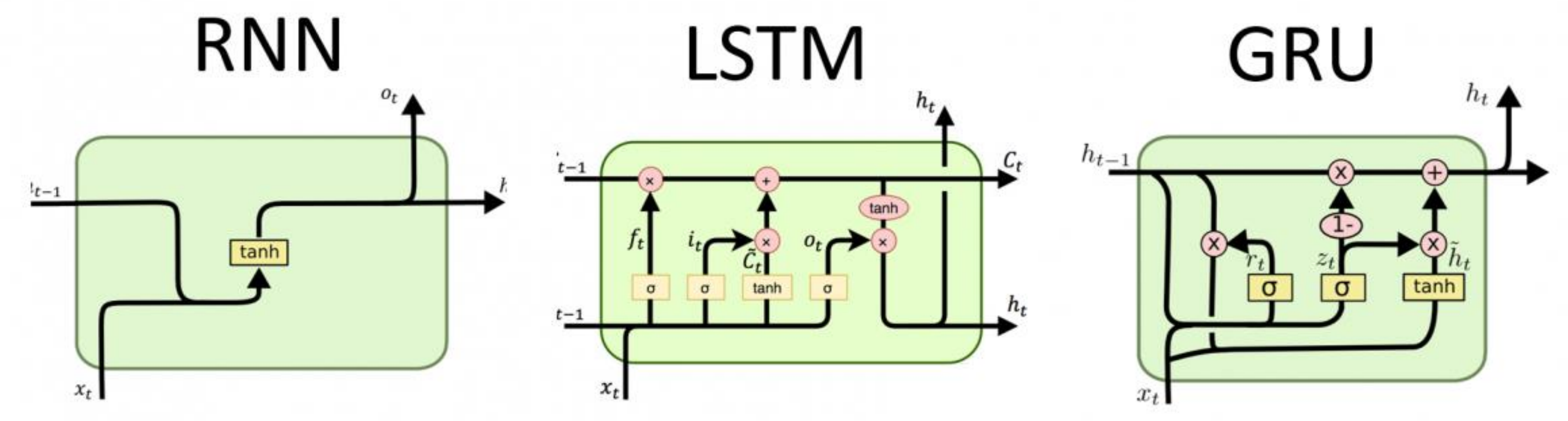
Skoltech

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:

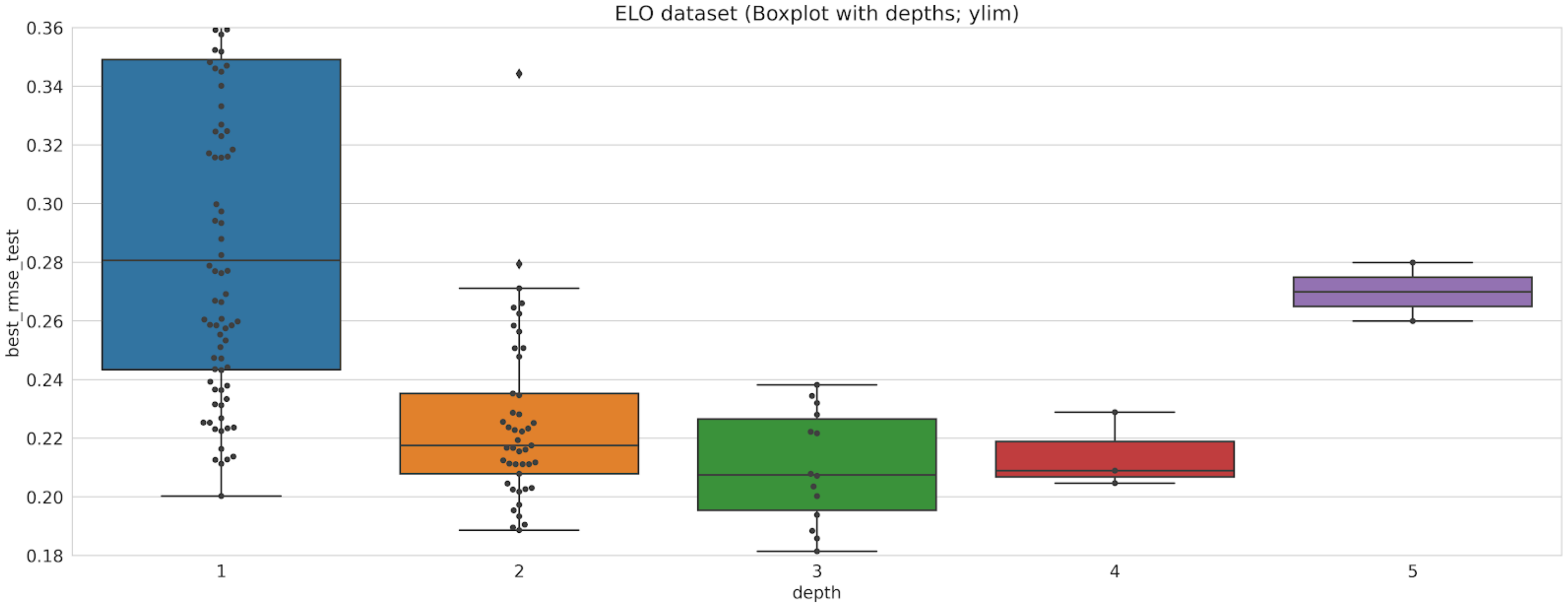
- Handwriting Recognition
- Speech Recognition
- Protein Structure Prediction (Bioinformatics)



Skoltech

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

Multiple layers RNN performance



Skoltech

Does a better recurrent block exist?

Towards a better recurrent block

- LSTM architecture is ad-hoc and has a substantial number of components whose purpose is not immediately apparent
- Like the LSTM, it is hard to tell, at a glance, which part of the GRU is essential for its functioning.

- Let's compare 10 000 different architectures on 3 problems with 1 000 of them pass the initial filtering stage: genetic algorithm
- Each architecture has been evaluated on about 220 hyperparameter settings.
- 230 000 hyperparameter configurations in total!

Jozefowicz, R., Zaremba, W., & Sutskever, I. 2015. An empirical exploration of recurrent network architectures. ICML

Best found architectures MUTx are close to GRU

GRU:

$$
\begin{aligned}
r_t &= \mathrm{sigm}\left(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_{t-1} + b_{\mathrm{r}}\right) \\
z_t &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} h_{t-1} + b_{\mathrm{z}}) \\
\tilde{h}_t &= \tanh(W_{\mathrm{xh}} x_t + W_{\mathrm{hh}}(r_t \odot h_{t-1}) + b_{\mathrm{h}}) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t
\end{aligned}
$$

Arch.	N	N-dropout	P
Tanh	3.612	3.267	6.809
LSTM	3.492	3.403	6.866
LSTM-f	3.732	3.420	6.813
LSTM-i	3.426	**3.252**	6.856
LSTM-o	3.406	**3.253**	6.870
LSTM-b	3.419	3.345	6.820
GRU	3.410	3.427	6.876
MUT1	**3.254**	3.376	**6.792**
MUT2	3.372	3.429	6.852
MUT3	3.337	3.505	6.840

Table 2. Negative Log Likelihood on the music datasets. N stands for Nottingham, N-dropout stands for Nottingham with nonzero dropout, and P stands for Piano-Midi.

MUT1:

$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + \tanh(x_t) + b_{\mathrm{h}}) \odot z \\
&+ h_t \odot (1 - z)
\end{aligned}
$$

MUT2:

$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} h_t + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh} x_t + b_{\mathrm{h}}) \odot z \\
&+ h_t \odot (1 - z)
\end{aligned}
$$

MUT3:

$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} \tanh(h_t) + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh} x_t + b_{\mathrm{h}}) \odot z \\
&+ h_t \odot (1 - z)
\end{aligned}
$$

Jozefowicz, R., Zaremba, W., & Sutskever, I. 2015. An empirical exploration of recurrent network architectures. ICML

Towards a better recurrent block

- LSTM architecture is ad-hoc and has a substantial number of components whose purpose is not immediately apparent
- Like the LSTM, it is hard to tell, at a glance, which part of the GRU is essential for its functioning.

- Let’s compare 8 LSTM variants and hope for the best by search over the space of hyperparameters with 5400 runs in total
- No significant improvement over the common LSTM
- Some advices on hyperparameters selection

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. 2016. LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems

Neural architecture search for a better Recurrent block

- Linear: $f(x_1, .., x_n) = W_1 x_1 + .. + W_n x_n + b$,
- Blending (element wise): $f(z, x, y) = z \odot x + (1 - z) \odot y$,
- Element wise product and sum,
- Activations: Tanh, Sigmoid, and LeakyReLU.

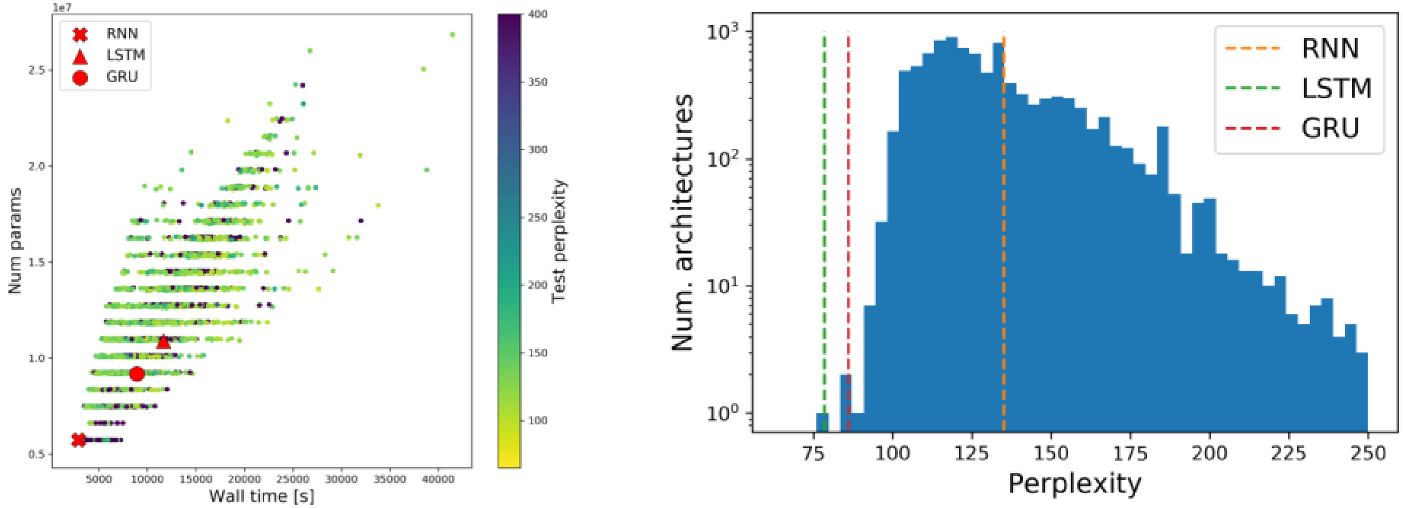
- Number of nodes < 25
- Number of hidden states < 4
- Number of linear input vectors < 4

(a) Simple RNN Cell (b) LSTM Cell (c) GRU Cell

Figure 2: Examples of conventional RNN cells. Colors of nodes highlight the corresponding previous and new hidden states, green color also highlights the input vector. Black dashed, blue and red edges indicate blending arguments z, x and y respectively.

Klyuchnikov, N., Trofimov, I., Artemova, E., Salnikov, M., Fedorov, M., & Burnaev, E. (2020). NAS-bench-NLP: Neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*.

Neural architecture search for a better Recurrent block



(a) Joint distribution of metrics.

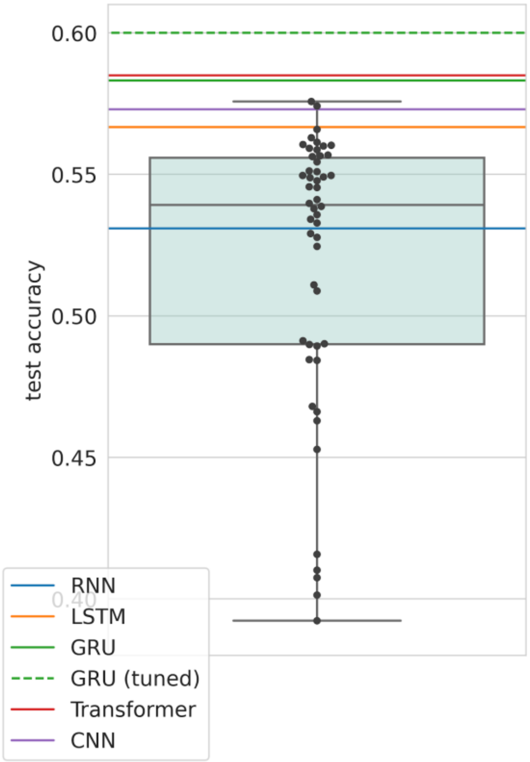
(b) Best test perplexity distribution.

Figure 4: Architectures metrics on PTB.

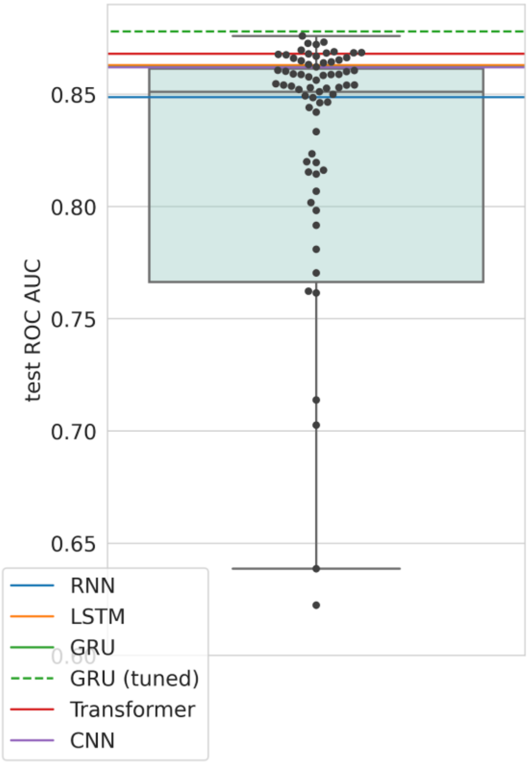
Klyuchnikov, N., Trofimov, I., Artemova, E., Salnikov, M., Fedorov, M., & Burnaev, E. (2020). Nas-bench-NLP: Neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*.

Best architecture for Transactions data

Age prediction

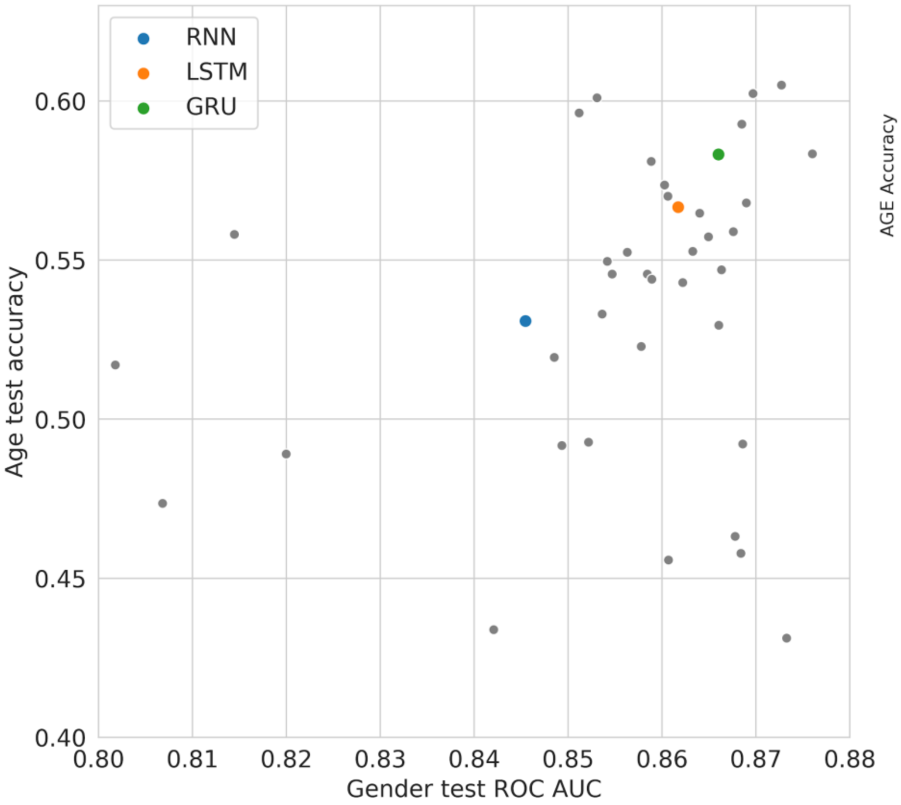


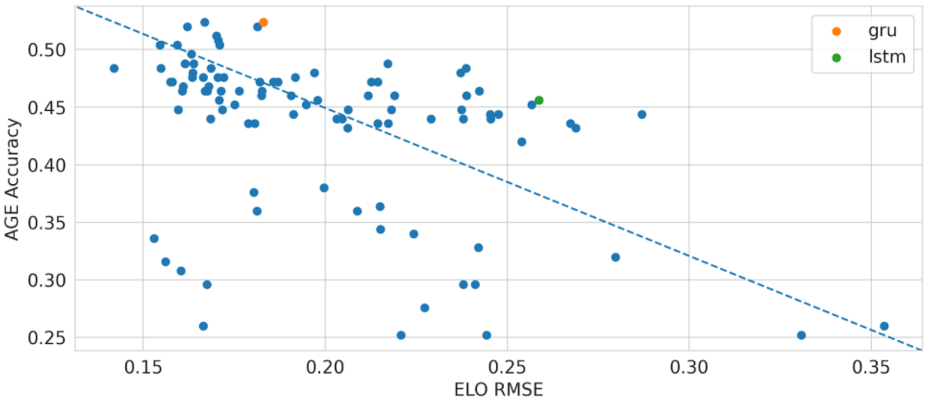
Gender prediction



Skoltech

Architecture performance transferability





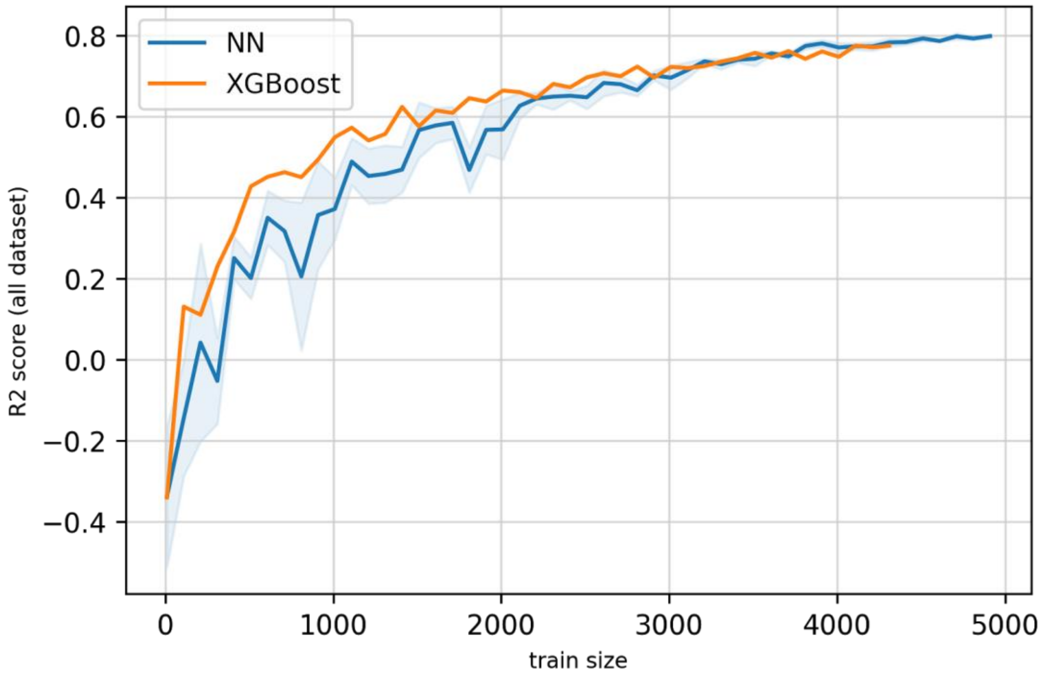
Skoltech

Can we predict the performance of an architecture?

Skoltech

We can, but we need large samples

For path features and NLP NAS bench with 11000 graph2vec features

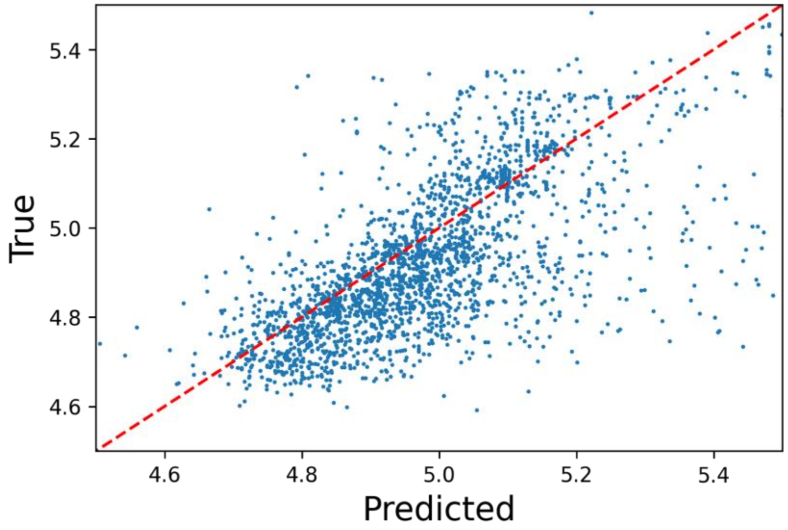


Skoltech

A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs. arXiv preprint arXiv:1707.05005, 2017.

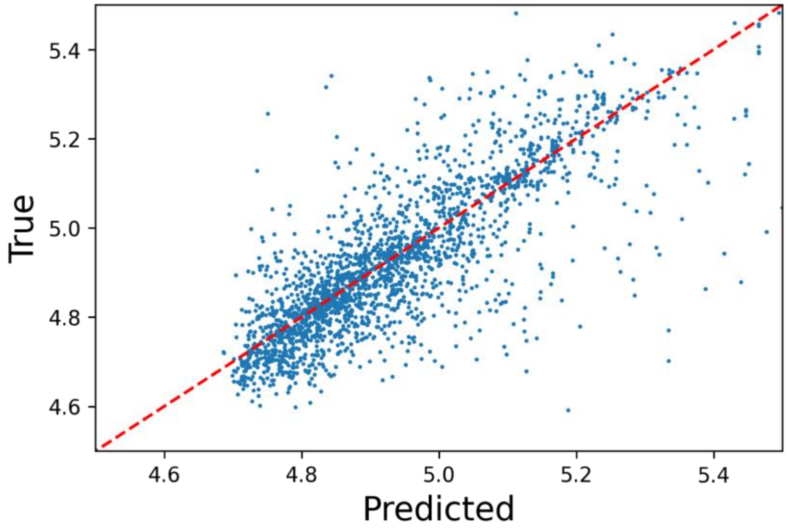
We can, but we need large samples

For path features and NLP NAS bench with 11000 features



$R^2 = 0.7455$

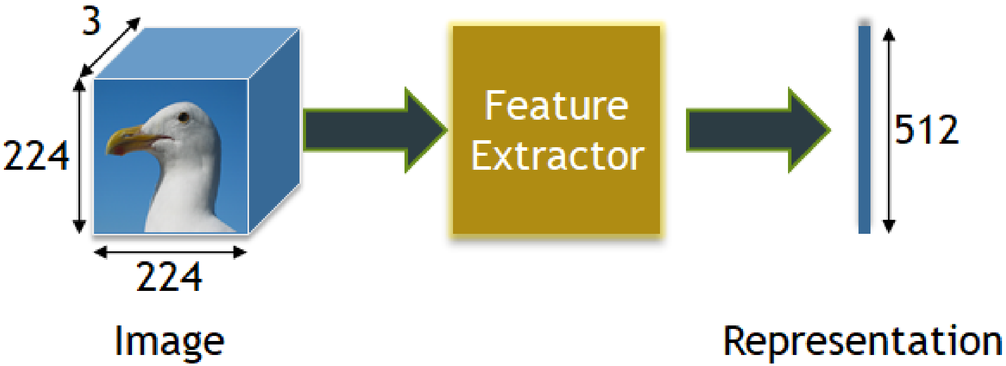
XGBRegressor 1000 estimators,
lr=0.05, default parameters



$R^2 = 0.7866$

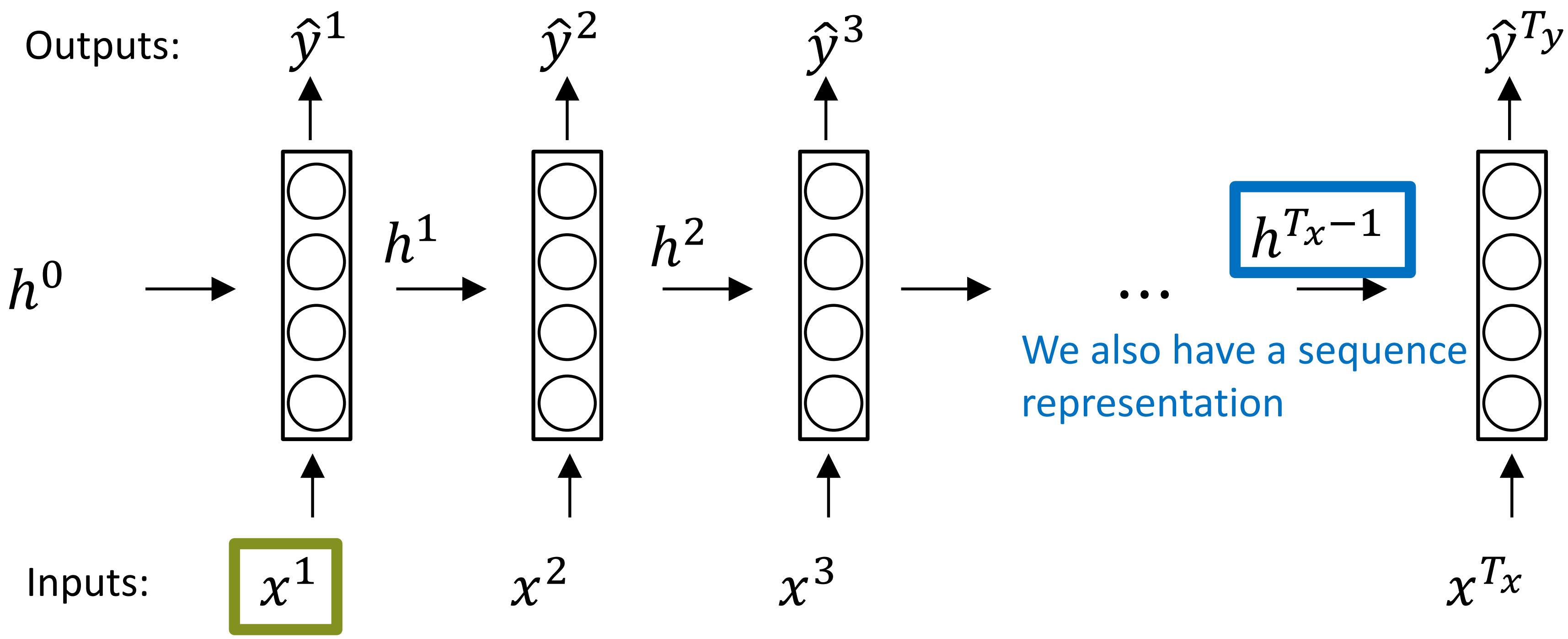
Dense NN, 10 layers with dim=20,
ELU lr=0.005, 120 epochs

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

Skoltech

Take-home messages

- For some types of data classic methods fail:
  we need to learn a representation i.e. extract features automatically

- Neural Networks provide enough flexibility for this problem for various data types

- The basic architecture is Recurrent Neural Network (RNN)

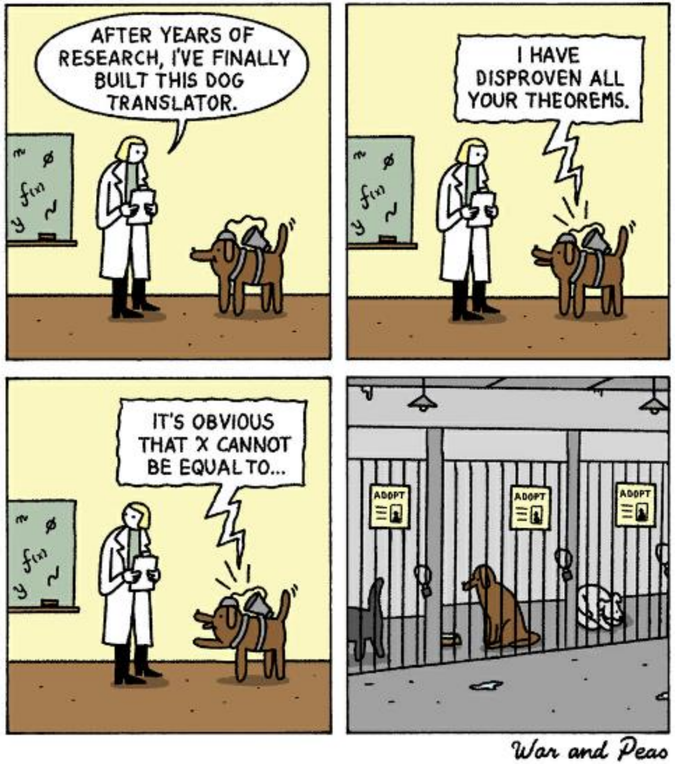
- But we can do better in terms of keeping the necessary information with LSTM and GRU blocks/architectures

Sources

- Recurrent Neural Networks | in Deep learning course by MIT 6.S191
- Coursera course on Sequence models
  https://www.coursera.org/learn/nlp-sequence-models
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Skoltech

Machine translation: application example



Dog Translation Machine

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

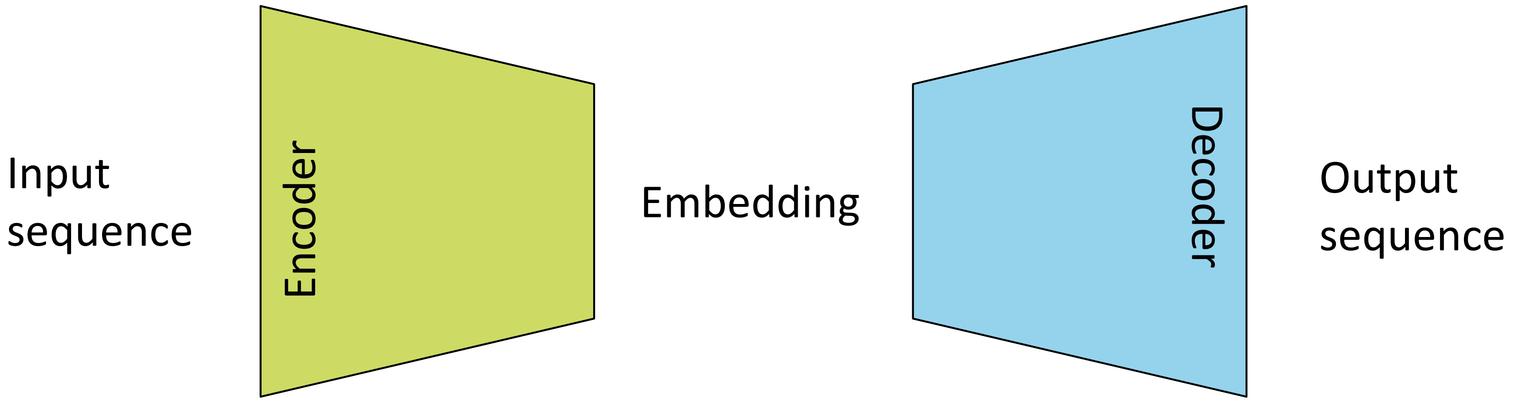
Project shut-down in 5 years: no significant progress

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

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

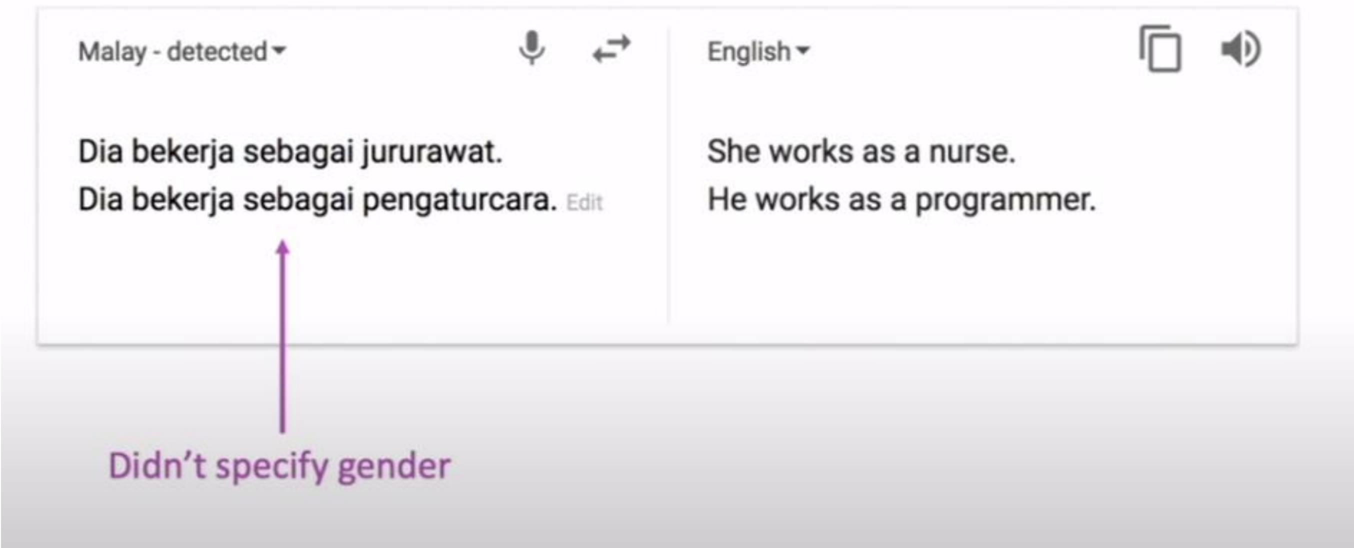
Neural network for machine translation



seq2seq (sequence to sequence) architecture

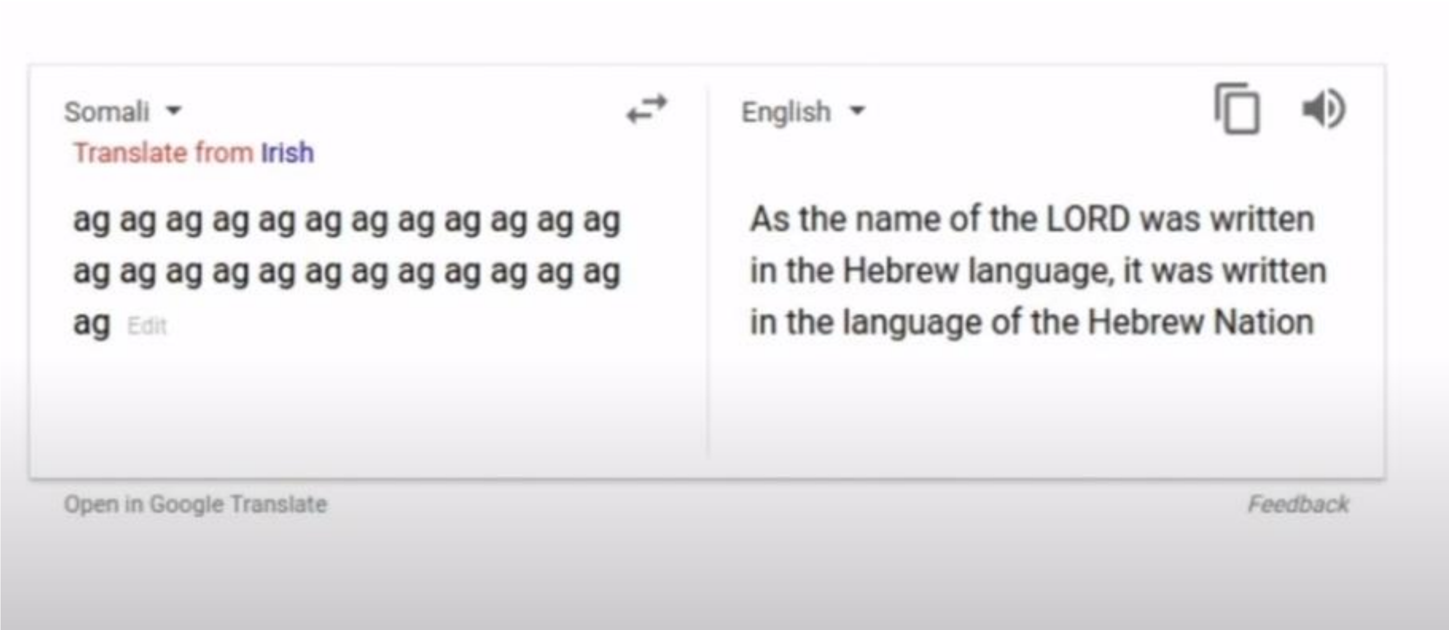
Not a perfect solution with little control over result

Contains biases from learning sample



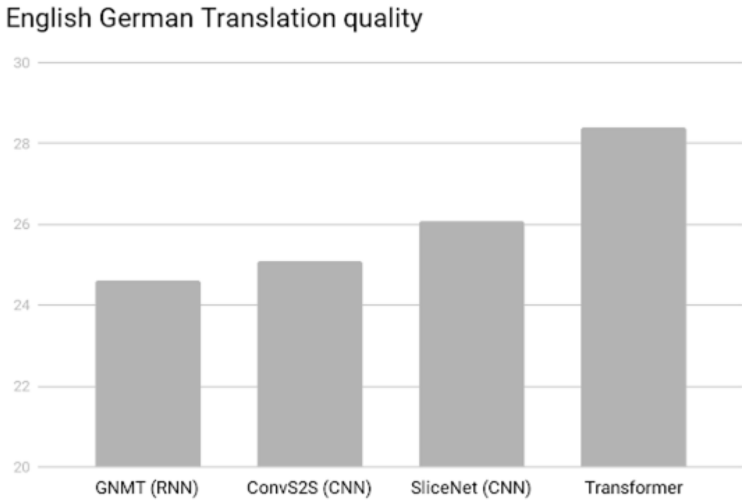
Skoltech

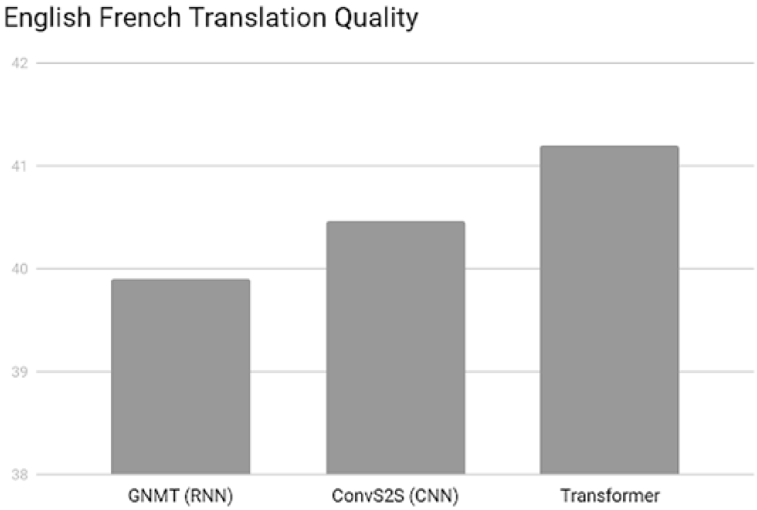
Output strange results



Skoltech

Attention/transformer – new state-of-the art for Machine translation

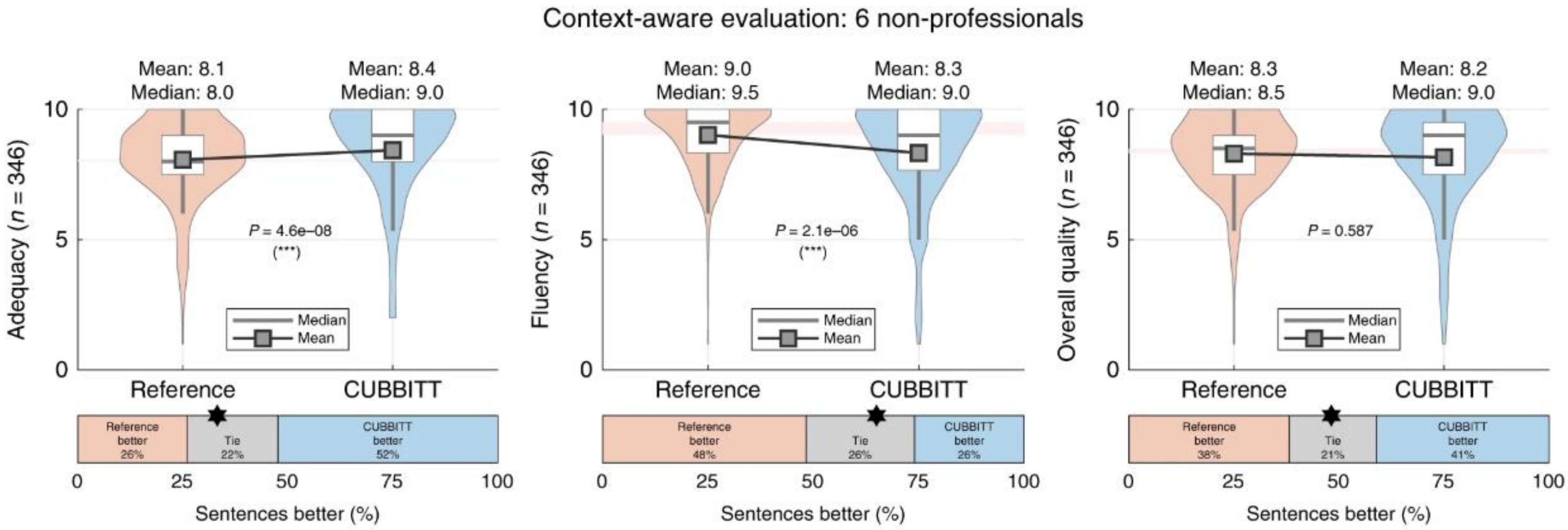




- Maximize BLEU translation quality metric
- The best model is **Transformer**

Skoltech

NLP deep learning models: CUBBITT is comparable to reference human translation for a selected dataset on News translation



Popel, Martin, et al. "Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals." *Nature communications* 11.1 (2020): 1-15.

Conclusions

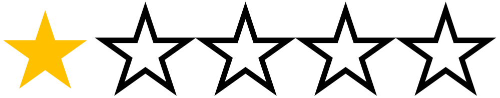
- Machine translation problem was attacked by many problems during a long period of time
- Right now the state-of-the-art is Neural Networks Transformers
- They can even improve over human-translation

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?