

# Hluboké učení pro zpracování jazyka

Aleš Horák

E-mail: [hales@fi.muni.cz](mailto:hales@fi.muni.cz)

[http://nlp.fi.muni.cz/nlp\\_intro/](http://nlp.fi.muni.cz/nlp_intro/)

Obsah:

- Od klasických k hlubokým neuronovým sítím
- Neurální jazykové modely
- Architektura Long Short-Term Memory (LSTM)
- Architektura Transformer

# Klasické neuronové sítě a text

## Neuronové sítě:

- od 1943 – McCulloch & Pitts – matematický model neuronu
- 1965 – první praktická vícevrstvá dopředná síť
- 1982 – praktická implementace zpětného šíření chyby pro trénování vícevrstevných sítí

praktické využití – klasifikační úlohy

dvouvrstvá

jednovrstvá

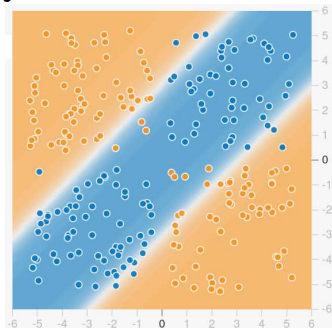
# Klasické neuronové sítě a text

## Neuronové sítě:

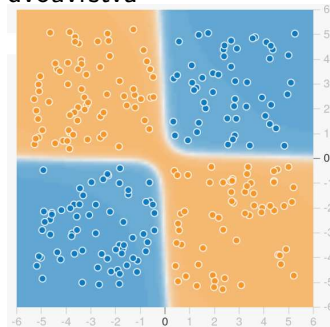
- od 1943 – McCulloch & Pitts – matematický model neuronu
- 1965 – první praktická vícevrstvá dopředná síť
- 1982 – praktická implementace zpětného šíření chyby pro trénování vícevrstevných sítí

praktické využití – klasifikační úlohy

jednovrstvá



dvouvrstvá



# Klasické neuronové sítě a text

Neuronová síť pracuje **pouze s čísly** – jak zadat text?

## 1. slova jako **prvky vstupu**:

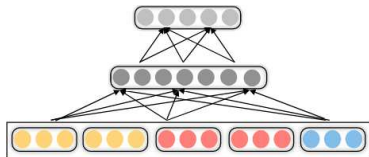
- pevně daný **slovník**  $n$  slov

*one-hot* kódování:

$\langle 1, 0, 0, 0, \dots \rangle$

$\langle 0, 1, 0, 0, \dots \rangle$

- pevně daná (maximální) **délka vstupu**  $m$
- vstup síť –  $m \times n$
- **není vhodné** pro velké slovníky



## 2. slova jako **slovní vektory** (*word embeddings*):

- stanovení **pevné dimenze**
- **předpočítání/předtrénování** na velmi velkých neanotovaných textech  
⇒ **neurální jazykové modely**
- zachycení **sémantiky** – podobná slova síť zpracuje podobně
- univerzálnější – **vektory částí slov** (*subword/character embeddings*)
- jen výměnou modelu můžeme **zpřesnit výsledky**

# Klasické neuronové sítě a text

Neuronová síť pracuje **pouze s čísly** – jak zadat text?

## 1. slova jako **prvky vstupu**:

- pevně daný **slovník**  $n$  slov  
*one-hot* kódování:

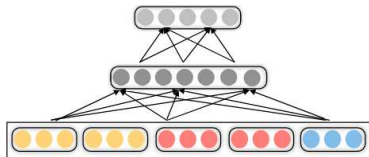
$\langle 1, 0, 0, 0, \dots \rangle$

$\langle 0, 1, 0, 0, \dots \rangle$

- pevně daná (maximální) **délka vstupu**  $m$
- vstup síť –  $m \times n$
- **není vhodné** pro velké slovníky

## 2. slova jako **slovní vektory** (*word embeddings*):

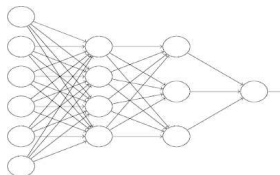
- stanovení **pevné dimenze**
- **předpočítání**/**předtrénování** na velmi velkých neanotovaných textech  
⇒ **neurální jazykové modely**
- zachycení **sémantiky** – podobná slova síť zpracuje podobně
- univerzálnější – **vektory částí slov** (*subword/character embeddings*)
- jen výměnou modelu můžeme **zpřesnit výsledky**



# Hluboké učení

## Hluboké neuronové síť:

- dopředné síť – plně propojené vrstvy

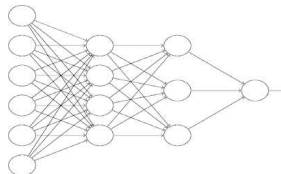


- cca od r. 2000 – metody a HW (GPU karty) pro učení sítí, které se skládají z mnoha (až desítek) heterogenních vrstev: konvoluční, sdružující (*max pooling*), rekurentní, klasifikační (*soft max*), ...

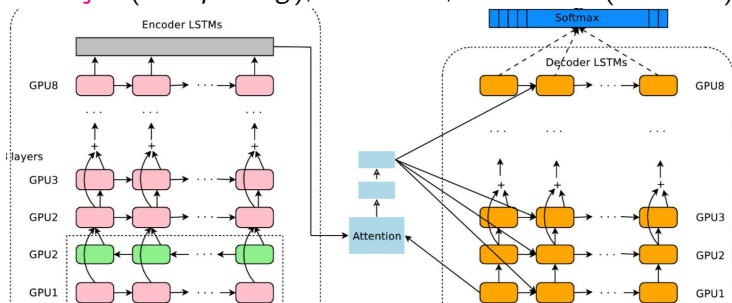
# Hluboké učení

## Hluboké neuronové sítě:

- dopředné sítě – plně propojené vrstvy



- cca od r. 2000 – metody a HW (GPU karty) pro učení sítí, které se skládají z **mnoha** (až desítek) **heterogenních vrstev**: **konvoluční**, **sdužující** (*max pooling*), **rekurentní**, **klasifikační** (*soft max*), ...



# Obsah

- 1 Od klasických k hlubokým neuronovým sítím
  - Klasické neuronové sítě a text
  - Hluboké učení
- 2 Neurální jazykové modely
  - S pevným kontextem
  - Rekurentní jazykový model
  - Praktické využití rekurentních sítí
- 3 Architektura Long Short-Term Memory (LSTM)
- 4 Architektura Transformer
  - Mechanismus attention
  - Architektura Transformer
  - Pokročilé jazykové modely



# Neurální jazykový model

připomínka – **jazykový model**:

SLiDo

vstup: začátek textu jako řetězec slov  $W = w_1 w_2 w_3 \dots w_{i-1}$

výstup: pravděpodobnostní distribuce dalšího slova  $P(w_i | w_1 w_2 \dots w_{i-1})$

základní **neurální jazykový model s pevným kontextem** (*fixed-window*)

výstupní distribuce

$$\vec{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$$

skrytá vrstva

$$h = f(We + b_1)$$

řetěžené vektory slov

$$e = [e_1; e_2; e_3; e_4]$$

slova na vstupu

$$w_1 w_2 w_3 w_4$$

$w_1$

$w_2$

$w_3$

$w_4$

# Neurální jazykový model

připomínka – **jazykový model**:

vstup: začátek textu jako řetězec slov  $W = w_1 w_2 w_3 \dots w_{i-1}$

výstup: pravděpodobnostní distribuce dalšího slova  $P(w_i | w_1 w_2 \dots w_{i-1})$

základní **neurální jazykový model s pevným kontextem** (*fixed-window*)

výstupní distribuce

$$\vec{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$$

skrytá vrstva

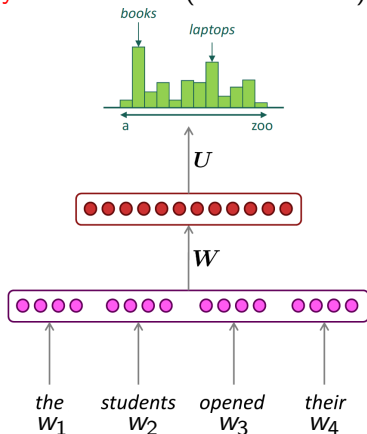
$$h = f(We + b_1)$$

řetěžené vektory slov

$$e = [e_1; e_2; e_3; e_4]$$

slova na vstupu

$$w_1 w_2 w_3 w_4$$



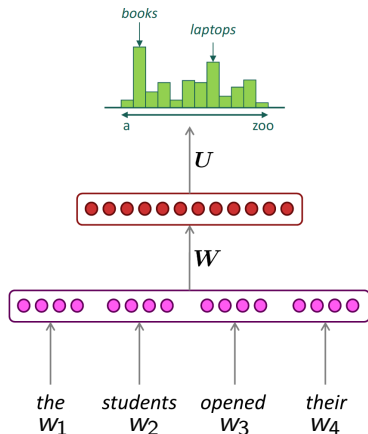
# Neurální jazykový model s pevným kontextem

**výhody** proti  $n$ -gramovému modelu:

- není problém s **nenalezenými  $n$ -gramy**
- nemusíme počítat a ukládat **velké seznamy  $n$ -gramů**

**problémy:**

- (malá) šířka kontextu
- rozšíření kontextu – zvětšuje  $W$
- ideální kontext je příliš velký
- váhy  $W$  závisí na pořadí slov –  $w_1$  má jiné váhy než  $w_2$



obr. z Stanford.Uni.

je potřebná neurální architektura pro **libovolně dlouhý vstup**

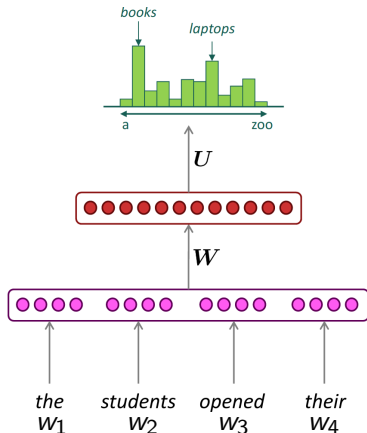
# Neurální jazykový model s pevným kontextem

**výhody** proti  $n$ -gramovému modelu:

- není problém s **nenalezenými  $n$ -gramy**
- nemusíme počítat a ukládat **velké seznamy  $n$ -gramů**

**problémy:**

- (malá) šířka kontextu
- rozšíření kontextu – zvětšuje  $W$
- ideální kontext je příliš velký
- váhy  $W$  závisí na pořadí slov –  $w_1$  má jiné váhy než  $w_2$



obr. z Stanford.Uni.

je potřebná neurální architektura pro **libovolně dlouhý vstup**

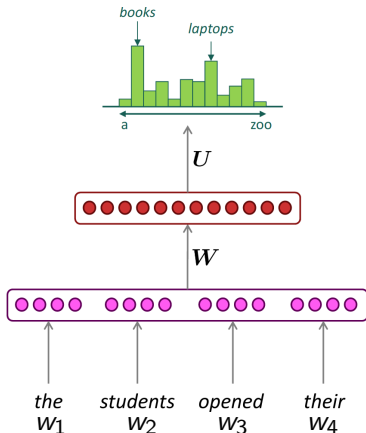
# Neurální jazykový model s pevným kontextem

**výhody** proti  $n$ -gramovému modelu:

- není problém s **nenalezenými  $n$ -gramy**
- nemusíme počítat a ukládat **velké seznamy  $n$ -gramů**

**problémy:**

- (malá) **šířka kontextu**
- rozšíření kontextu – zvětšuje  $W$
- ideální kontext je **příliš velký**
- váhy  $W$  závisí na **pořadí slov** –  $w_1$  má jiné váhy než  $w_2$



obr. z Stanford.Uni.

je potřebná neurální architektura pro **libovolně dlouhý vstup**

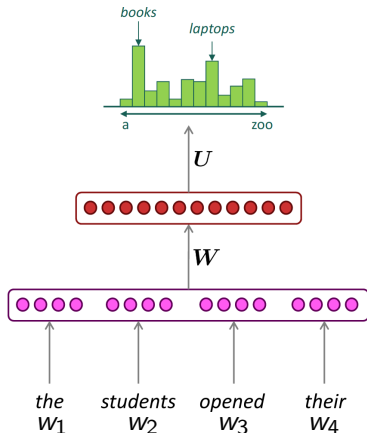
# Neurální jazykový model s pevným kontextem

**výhody** proti  $n$ -gramovému modelu:

- není problém s **nenalezenými  $n$ -gramy**
- nemusíme počítat a ukládat **velké seznamy  $n$ -gramů**

**problémy:**

- (malá) **šířka kontextu**
- rozšíření kontextu – zvětšuje  $W$
- ideální kontext je **příliš velký**
- váhy  $W$  závisí na **pořadí slov** –  $w_1$  má jiné váhy než  $w_2$



obr. z Stanford.Uni.

je potřebná neurální architektura pro **libovolně dlouhý vstup**

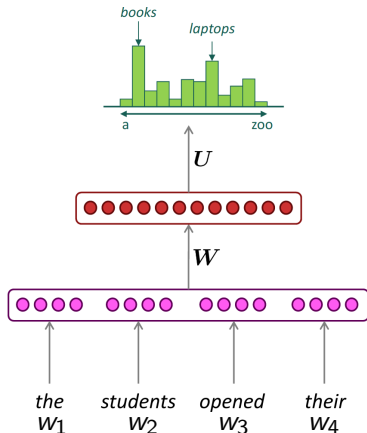
# Neurální jazykový model s pevným kontextem

**výhody** proti  $n$ -gramovému modelu:

- není problém s **nenalezenými  $n$ -gramy**
- nemusíme počítat a ukládat **velké seznamy  $n$ -gramů**

**problémy:**

- (malá) **šířka kontextu**
- rozšíření kontextu – zvětšuje  $W$
- ideální kontext je **příliš velký**
- váhy  $W$  závisí na **pořadí slov** –  $w_1$  má jiné váhy než  $w_2$



obr. z Stanford.Uni.

je potřebná neurální architektura pro **libovolně dlouhý vstup**

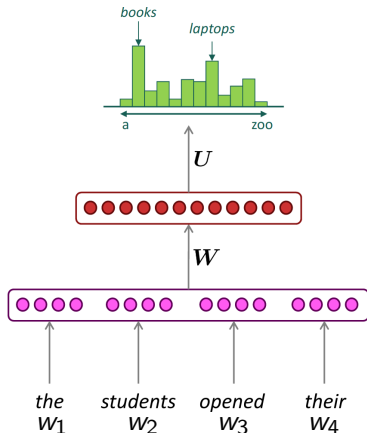
# Neurální jazykový model s pevným kontextem

**výhody** proti  $n$ -gramovému modelu:

- není problém s **nenalezenými  $n$ -gramy**
- nemusíme počítat a ukládat **velké seznamy  $n$ -gramů**

**problémy:**

- (malá) **šířka kontextu**
- rozšíření kontextu – zvětšuje  $W$
- ideální kontext je **příliš velký**
- váhy  $W$  závisí na **pořadí slov** –  $w_1$  má jiné váhy než  $w_2$



obr. z Stanford.Uni.

je potřebná neurální architektura pro **libovolně dlouhý vstup**



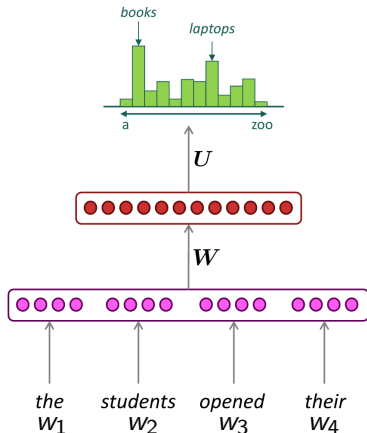
# Neurální jazykový model s pevným kontextem

**výhody** proti  $n$ -gramovému modelu:

- není problém s **nenalezenými  $n$ -gramy**
- nemusíme počítat a ukládat **velké seznamy  $n$ -gramů**

**problémy:**

- (malá) **šířka kontextu**
- rozšíření kontextu – zvětšuje  $W$
- ideální kontext je **příliš velký**
- váhy  $W$  závisí na **pořadí slov** –  $w_1$  má jiné váhy než  $w_2$



obr. z Stanford.Uni.

je potřebná neurální architektura pro **libovolně dlouhý vstup**

# Rekurentní jazykový model

výstupní distribuce

$$\vec{y}^{(t)} = \text{softmax}(U\mathbf{h}^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

skryté stavy

$$\mathbf{h}^{(t)} = \sigma(W_h \mathbf{h}^{(t-1)} + W_e \mathbf{e}^{(t)} + \mathbf{b}_1)$$

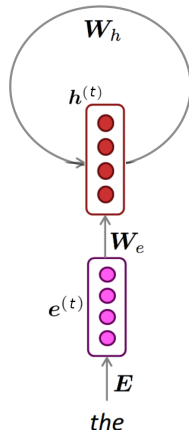
váhy  $W_h$  a  $W_e$  se aplikují opakovaně

jednotlivé vektory slov

$$\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \mathbf{e}^{(3)}, \mathbf{e}^{(4)}, \dots$$

slova na vstupu

$$w_1 w_2 w_3 w_4$$



# Rekurentní jazykový model

výstupní distribuce

$$\vec{y}^{(t)} = \text{softmax}(Uh^{(t)} + b_2) \in \mathbb{R}^{|V|}$$

skryté stavy

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$

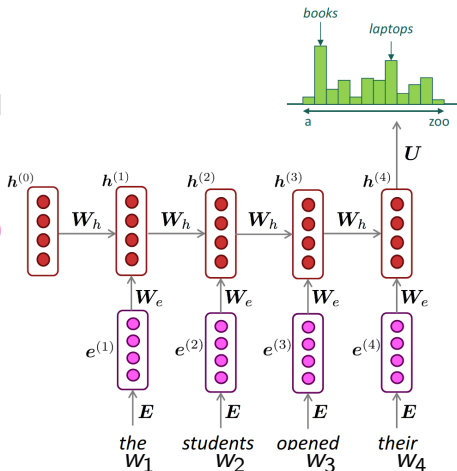
váhy  $W_h$  a  $W_e$  se aplikují opakovaně

jednotlivé vektory slov

$$e^{(1)}, e^{(2)}, e^{(3)}, e^{(4)}, \dots$$

slova na vstupu

$$w_1 w_2 w_3 w_4$$



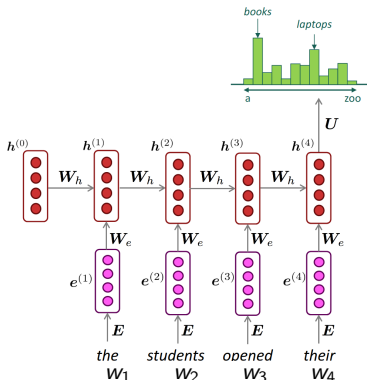
# Rekurentní jazykový model

**výhody** rekurentního modelu:

- může zpracovat vstup libovolné délky (v praxi věta)
- výpočet je založen na celé sekvenci
- váhy jsou stejné – nezávislost na pozici

**nevýhody:**

- rekurentní výpočet je pomalý
- s délkou sekvence se informace rozměňují



vyzkoušejte – [https://muni.cz/go/ib030\\_text\\_gener](https://muni.cz/go/ib030_text_gener)

# Rekurentní jazykový model

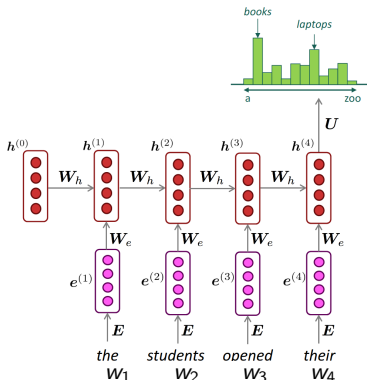
**výhody** rekurentního modelu:

- může zpracovat vstup libovolné délky (v praxi věta)
- výpočet je založen na celé sekvenci
- váhy jsou stejné – nezávislost na pozici

**nevýhody:**

- rekurentní výpočet je pomalý
- s délkou sekvence se informace rozměňují

vyzkoušejte – [https://muni.cz/go/ib030\\_text\\_gener](https://muni.cz/go/ib030_text_gener)



# Rekurentní jazykový model

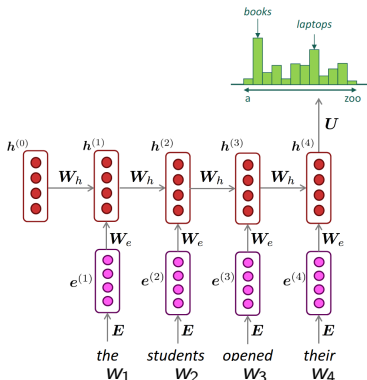
**výhody** rekurentního modelu:

- může zpracovat vstup libovolné délky (v praxi věta)
- výpočet je založen na celé sekvenci
- váhy jsou stejné – nezávislost na pozici

**nevýhody:**

- rekurentní výpočet je pomalý
- s délkou sekvence se informace rozměňují

vyzkoušejte – [https://muni.cz/go/ib030\\_text\\_gener](https://muni.cz/go/ib030_text_gener)



# Rekurentní jazykový model

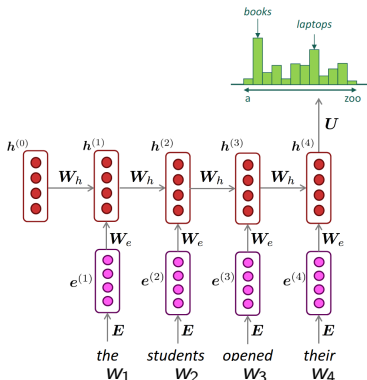
**výhody** rekurentního modelu:

- může zpracovat vstup libovolné délky (v praxi věta)
- výpočet je založen na celé sekvenci
- váhy jsou stejné – nezávislost na pozici

**nevýhody:**

- rekurentní výpočet je pomalý
- s délkou sekvence se informace rozměšuje

vyzkoušejte – [https://muni.cz/go/ib030\\_text\\_gener](https://muni.cz/go/ib030_text_gener)



# Rekurentní jazykový model

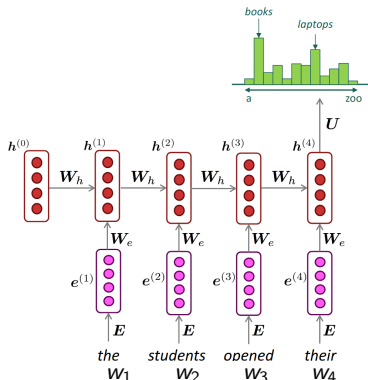
**výhody** rekurentního modelu:

- může zpracovat vstup libovolné délky (v praxi věta)
- výpočet je založen na celé sekvenci
- váhy jsou stejné – nezávislost na pozici

**nevýhody:**

- rekurentní výpočet je pomalý
- s délkou sekvence se informace rozměňuje

vyzkoušejte – [https://muni.cz/go/ib030\\_text\\_gener](https://muni.cz/go/ib030_text_gener)





# Kvalita jazykového modelu

základní **srovnávací metrika** – **perplexita** (“zmatenost”):

- srovnání na **vybraném testovacím textu**
- poměrově vyjadřuje **z kolika slov** se vybírá predikce
- **nižší hodnota** = **lepší perplexita**

Rekurentní modely → zlepšení perplexity

*n*-gramový

model →

rekur. ↓

modely

Model	Perplexita
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	<b>67.6</b>
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + Blackout sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 Oozefowicz et al., 2016)	43.7
2-layer LSTM-8192 Oozefowicz et al., 2016)	<b>30.0</b>

zdroj [research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words](https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words)

# Kvalita jazykového modelu

základní **srovnávací metrika** – **perplexita** (“zmatenost”):

- srovnání na **vybraném testovacím textu**
- poměrově vyjadřuje **z kolika slov** se vybírá predikce
- **nižší hodnota** = **lepší perplexita**

Rekurentní modely → zlepšení perplexity

*n*-gramový  
model →  
rekur. ↓  
modely

Model	Perplexita
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	<b>67.6</b>
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + Blackout sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 Oozefowicz et al., 2016)	43.7
2-layer LSTM-8192 Oozefowicz et al., 2016)	<b>30.0</b>

zdroj [research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words](https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words)

# Kvalita jazykového modelu

základní **srovnávací metrika** – **perplexita** (“zmatenost”):

- srovnání na **vybraném testovacím textu**
- poměrově vyjadřuje **z kolika slov** se vybírá predikce
- **nižší hodnota** = **lepší perplexita**

Rekurentní modely → zlepšení perplexity

*n*-gramový  
model →  
rekur. ↓  
modely

Model	Perplexita
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	<b>67.6</b>
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + Blackout sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 Oozefowicz et al., 2016)	43.7
2-layer LSTM-8192 Oozefowicz et al., 2016)	<b>30.0</b>

zdroj [research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words](https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words)

# Kvalita jazykového modelu

základní **srovnávací metrika** – **perplexita** (“zmatenost”):

- srovnání na **vybraném testovacím textu**
- poměrově vyjadřuje **z kolika slov** se vybírá predikce
- **nižší hodnota** = **lepší perplexita**

**Rekurentní modely** → **zlepšení perplexity**

*n*-gramový  
model →  
rekur. ↓  
modely

Model	Perplexita
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	<b>67.6</b>
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + Blackout sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 Oozefowicz et al., 2016)	43.7
2-layer LSTM-8192 Oozefowicz et al., 2016)	<b>30.0</b>

zdroj [research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words](https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words)

# Hyperparametry hlubokého učení

Hluboké učení – kromě **architektury sítě** i množství **hyperparametrů**:

- **regularizace** (např. L2-regularizace) – brání **přeučení sítě** ke **ztrátové funkci** (*loss function*) přidává “pokutu”  $\lambda \sum_k w_k^2$
- **dropout** – cíleně způsobuje **náhodné výpadky** vstupních hodnot simuluje **šum v datech** – nutí síť ke **generalizaci**
- **dimenze skryté vrstvy** (*hidden layer dimension*) vyšší hodnoty umožní síti více si **pamatovat**, ale prodlužují výpočet
- **optimalizační funkce** (*optimizer*) základní – *Stochastic Gradient Descent*, *SGD* pro různé úlohy sofistikované varianty *Adagrad*, *RMSprop*, *Adam*, ...
- **míra učení** (*learning rate*) – řídí **rychlost učení** obvykle **funkce**, jejíž hodnota se snižuje, např.  $lr_0 e^{-kt}$  hodně závisí na optimalizační funkci



**Hodnoty hyperparametrů** se určují **podle zkušeností** nebo **experimentálně**

# Hyperparametry hlubokého učení

Hluboké učení – kromě **architektury sítě** i množství **hyperparametrů**:

- **regularizace** (např. L2-regularizace) – brání **přeučení sítě** ke **ztrátové funkci** (*loss function*) přidává “pokutu”  $\lambda \sum_k w_k^2$
- **dropout** – cíleně způsobuje **náhodné výpadky** vstupních hodnot simuluje **šum v datech** – nutí síť ke **generalizaci**
- **dimenze skryté vrstvy** (*hidden layer dimension*) vyšší hodnoty umožní síti více si **pamatovat**, ale prodlužují výpočet
- **optimalizační funkce** (*optimizer*) základní – *Stochastic Gradient Descent*, *SGD* pro různé úlohy sofistikované varianty *Adagrad*, *RMSprop*, *Adam*, ...
- **míra učení** (*learning rate*) – řídí **rychlost učení** obvykle **funkce**, jejíž hodnota se snižuje, např.  $lr_0 e^{-kt}$  hodně závisí na optimalizační funkci



**Hodnoty hyperparametrů** se určují **podle zkušeností** nebo **experimentálně**

# Hyperparametry hlubokého učení

Hluboké učení – kromě **architektury sítě** i množství **hyperparametrů**:

- **regularizace** (např. L2-regularizace) – brání **přeučení sítě** ke **ztrátové funkci** (*loss function*) přidává “pokutu”  $\lambda \sum_k w_k^2$
- **dropout** – cíleně způsobuje **náhodné výpadky** vstupních hodnot simuluje **šum v datech** – nutí síť ke **generalizaci**
- **dimenze skryté vrstvy** (*hidden layer dimension*) vyšší hodnoty umožní síti více si **pamatovat**, ale prodlužují výpočet
- **optimalizační funkce** (*optimizer*) základní – *Stochastic Gradient Descent*, *SGD* pro různé úlohy sofistikované varianty *Adagrad*, *RMSprop*, *Adam*, ...
- **míra učení** (*learning rate*) – řídí **rychlost učení** obvykle **funkce**, jejíž hodnota se snižuje, např.  $lr_0 e^{-kt}$  hodně závisí na optimalizační funkci



**Hodnoty hyperparametrů** se určují **podle zkušeností** nebo **experimentálně**

# Hyperparametry hlubokého učení

Hluboké učení – kromě **architektury sítě** i množství **hyperparametrů**:

- **regularizace** (např. L2-regularizace) – brání **přeučení sítě** ke **ztrátové funkci** (*loss function*) přidává “pokutu”  $\lambda \sum_k w_k^2$
- **dropout** – cíleně způsobuje **náhodné výpadky** vstupních hodnot simuluje **šum v datech** – nutí síť ke **generalizaci**
- **dimenze skryté vrstvy** (*hidden layer dimension*) vyšší hodnoty umožní síti více si **pamatovat**, ale prodlužují výpočet
- **optimalizační funkce** (*optimizer*)  
základní – *Stochastic Gradient Descent*, *SGD*  
pro různé úlohy sofistikované varianty *Adagrad*, *RMSprop*, *Adam*, ...
- **míra učení** (*learning rate*) – řídí **rychlost učení**  
obvykle **funkce**, jejíž hodnota se snižuje, např.  $lr_0 e^{-kt}$   
hodně závisí na optimalizační funkci



**Hodnoty hyperparametrů** se určují **podle zkušeností** nebo **experimentálně**



# Hyperparametry hlubokého učení

Hluboké učení – kromě **architektury sítě** i množství **hyperparametrů**:

- **regularizace** (např. L2-regularizace) – brání **přeučení sítě** ke **ztrátové funkci** (*loss function*) přidává “pokutu”  $\lambda \sum_k w_k^2$
- **dropout** – cíleně způsobuje **náhodné výpadky** vstupních hodnot simuluje **šum v datech** – nutí síť ke **generalizaci**
- **dimenze skryté vrstvy** (*hidden layer dimension*) vyšší hodnoty umožní síti více si **pamatovat**, ale prodlužují výpočet
- **optimalizační funkce** (*optimizer*) základní – *Stochastic Gradient Descent*, *SGD* pro různé úlohy sofistikované varianty *Adagrad*, *RMSprop*, *Adam*, ...
- **míra učení** (*learning rate*) – řídí **rychlost učení** obvykle **funkce**, jejíž hodnota se snižuje, např.  $lr_0 e^{-kt}$  hodně závisí na optimalizační funkci



Hodnoty **hyperparametrů** se určují **podle zkušeností** nebo **experimentálně**

# Hyperparametry hlubokého učení

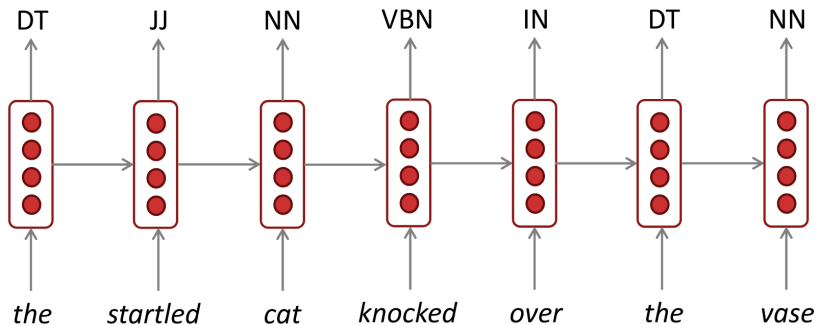
Hluboké učení – kromě **architektury sítě** i množství **hyperparametrů**:

- **regularizace** (např. L2-regularizace) – brání **přeučení sítě** ke **ztrátové funkci** (*loss function*) přidává “pokutu”  $\lambda \sum_k w_k^2$
- **dropout** – cíleně způsobuje **náhodné výpadky** vstupních hodnot simuluje **šum v datech** – nutí síť ke **generalizaci**
- **dimenze skryté vrstvy** (*hidden layer dimension*) vyšší hodnoty umožní síti více si **pamatovat**, ale prodlužují výpočet
- **optimalizační funkce** (*optimizer*) základní – *Stochastic Gradient Descent*, *SGD* pro různé úlohy sofistikované varianty *Adagrad*, *RMSprop*, *Adam*, ...
- **míra učení** (*learning rate*) – řídí **rychlost učení** obvykle **funkce**, jejíž hodnota se snižuje, např.  $lr_0 e^{-kt}$  hodně závisí na optimalizační funkci



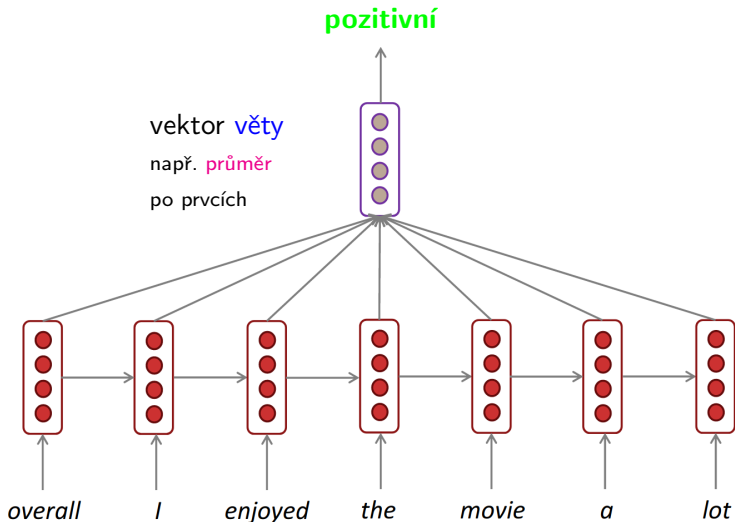
**Hodnoty hyperparametrů** se určují **podle zkušeností** nebo **experimentálně**

# Využití rekurentních sítí – značkování

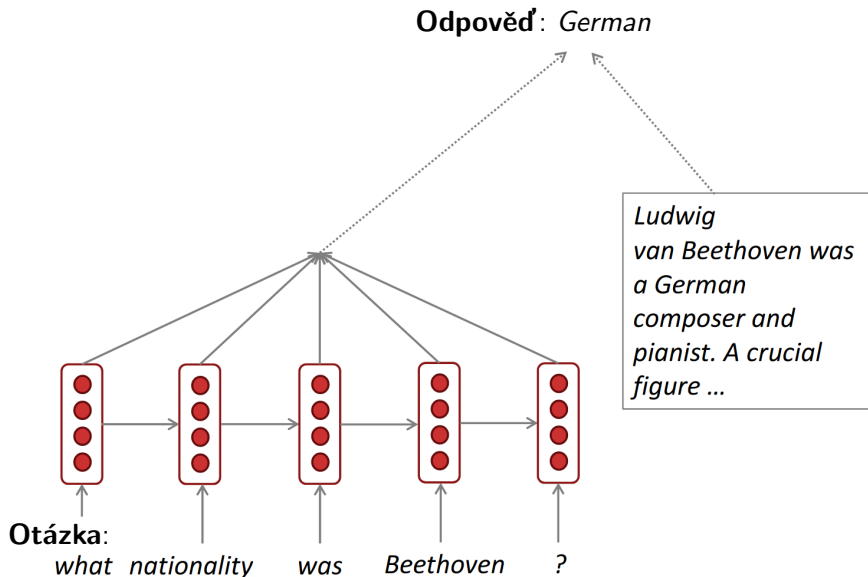


# Využití rekurentních sítí – klasifikace vět

např. **analýza sentimentu**

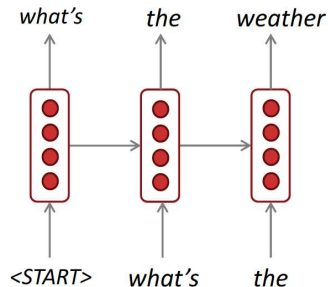
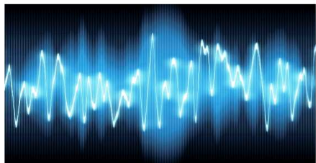


# Využití rekurentních sítí – odpovídání na otázky



# Využití rekurentních sítí – podmíněné generování

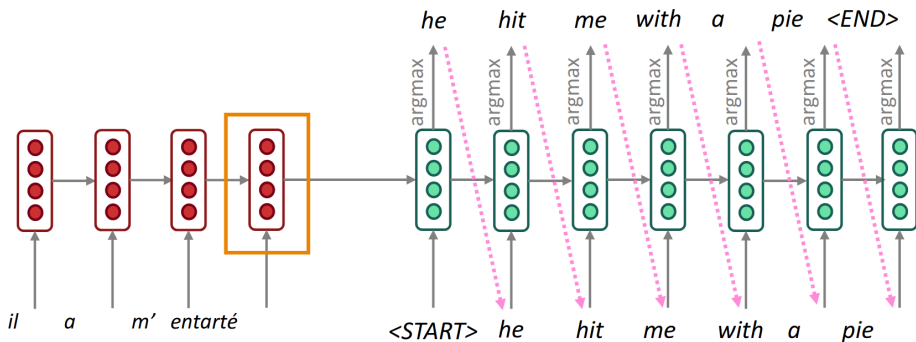
zapojení dalších **sekvenčních podmínek** – **syntéza řeči**, **strojový překlad**, **sumarizace**



# Využití rekurentních sítí – seq2seq

častá varianta – model **sequence-to-sequence** (seq2seq)

dvě rekurentní sítě – **enkodér** a **dekodér**



# Obsah

- 1 Od klasických k hlubokým neuronovým sítím
  - Klasické neuronové sítě a text
  - Hluboké učení
- 2 Neurální jazykové modely
  - S pevným kontextem
  - Rekurentní jazykový model
  - Praktické využití rekurentních sítí
- 3 Architektura Long Short-Term Memory (LSTM)
- 4 Architektura Transformer
  - Mechanismus attention
  - Architektura Transformer
  - Pokročilé jazykové modely



# Architektura Long Short-Term Memory (LSTM)

problém trénování velkých RNN – **mizející gradient** (násobení malých čísel  $\rightarrow 0$ )

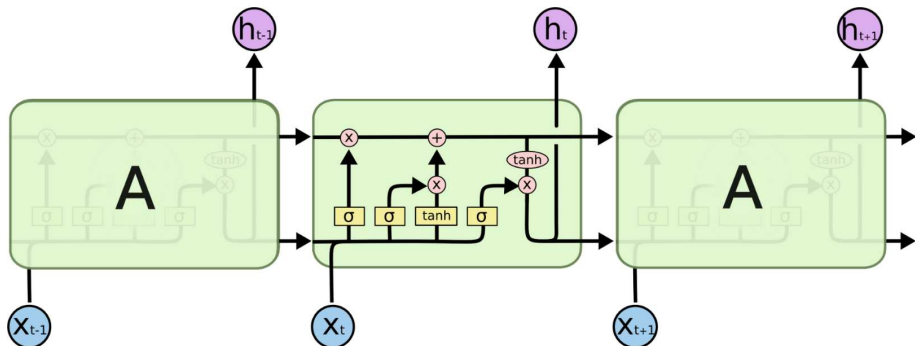
řešení – architektura Long Short-Term Memory, LSTM:

- buňka (*cell*)  $c_t$  – pomocná paměť
- 3 brány: vstupní, výstupní a zapomínací (*forget*) – regulace info do a z buňky

# Architektura Long Short-Term Memory (LSTM)

problém trénování velkých RNN – **mizející gradient** (násobení malých čísel  $\rightarrow 0$ )  
 řešení – **architektura Long Short-Term Memory, LSTM**:

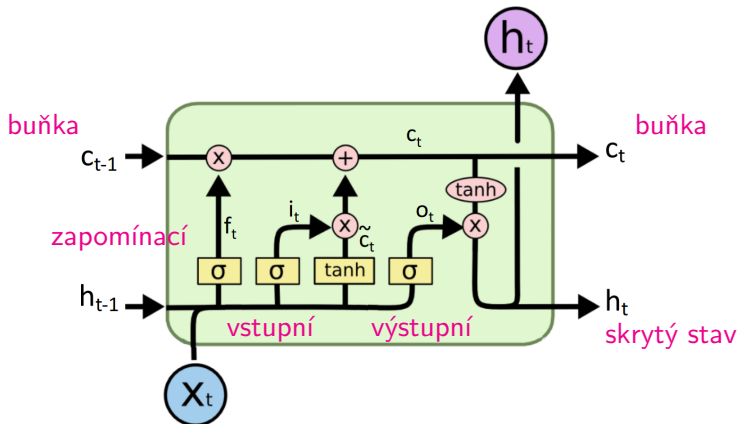
- **buňka** (*cell*)  $c_t$  – pomocná paměť
- 3 brány: **vstupní**, **výstupní** a **zapomínací** (*forget*) – regulace info do a z buňky



# Architektura Long Short-Term Memory (LSTM)

problém trénování velkých RNN – **mizející gradient** (násobení malých čísel  $\rightarrow 0$ )  
 řešení – **architektura Long Short-Term Memory, LSTM**:

- **buňka** (*cell*)  $c_t$  – pomocná paměť
- 3 brány: **vstupní**, **výstupní** a **zapomínací** (*forget*) – regulace info do a z buňky



# Architektura Long Short-Term Memory (LSTM)

hlavní **výhoda LSTM** – schopnost nalézt **vzdálené závislosti**  
**nevýhody** – lineární postup, výpočet **nelze paralelizovat**

rekurence je **směřovaná** –  
zleva doprava  
⇒ jiný důraz při průběhu  
zprava doleva

**obousměrné** (*bidirectional*) **BiLSTM**  
spojení **dopředné LSTM**  
a **zpětné LSTM**  
výstupy se **spojí** (*concatenate*)  
do výsledných vektorů  
(2× dimenze)

# Architektura Long Short-Term Memory (LSTM)

hlavní **výhoda LSTM** – schopnost nalézt **vzdálené závislosti**

**nevýhody** – lineární postup, výpočet **nelze paralelizovat**

rekurence je **směřovaná** –

zleva doprava

⇒ jiný důraz při průběhu

**zprava doleva**

**obousměrné** (*bidirectional*) **BiLSTM**

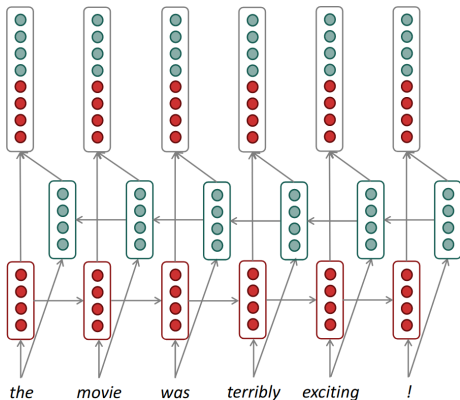
spojení **dopředné LSTM**

a **zpětné LSTM**

výstupy se **spojí** (*concatenate*)

do výsledných vektorů

(2× dimenze)

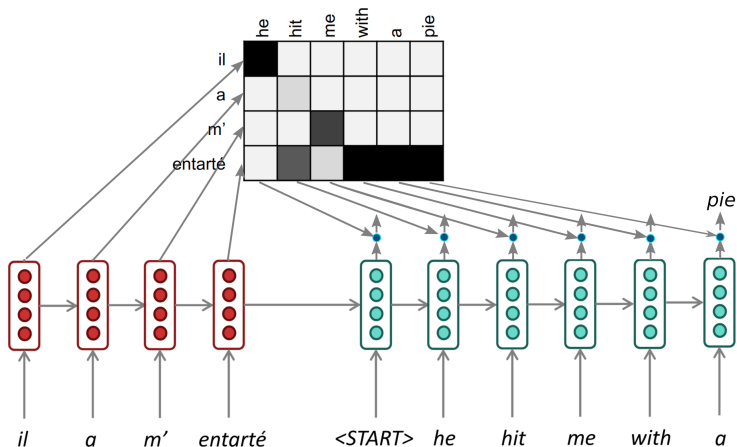


# Obsah

- 1 Od klasických k hlubokým neuronovým sítím
  - Klasické neuronové sítě a text
  - Hluboké učení
- 2 Neurální jazykové modely
  - S pevným kontextem
  - Rekurentní jazykový model
  - Praktické využití rekurentních sítí
- 3 Architektura Long Short-Term Memory (LSTM)
- 4 Architektura Transformer
  - Mechanismus attention
  - Architektura Transformer
  - Pokročilé jazykové modely

# Mechanismus attention

u **rekurentních** sítí – celá věta reprezentována jako **jeden vektor**  
 mechanismus **attention** (“pozornost”) – detailní provázání informací



# Architektura Transformer

“Attention is All You Need” (Vaswani et al, 2017)



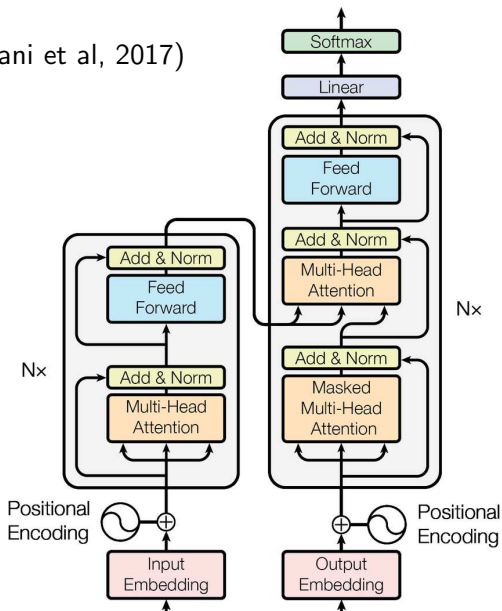


# Architektura Transformer

“Attention is All You Need” (Vaswani et al, 2017)

architektura **transformer**:

- **pod-slovní** vektory (*subword embeddings*)
- vektory **pozice**
- **self-attention**
- více **hlav** (*multi-head attention*)
- **reziduální spojení**, **normalizace** a **škálování**

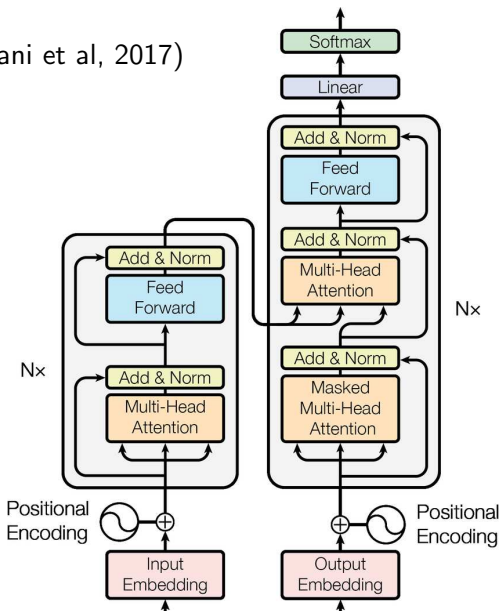


# Architektura Transformer

“Attention is All You Need” (Vaswani et al, 2017)

architektura **transformer**:

- **pod-slovní** vektory (*subword embeddings*)
- vektory **pozice**
- **self-attention**
- více **hlav** (*multi-head attention*)
- **reziduální spojení**, **normalizace** a **škálování**

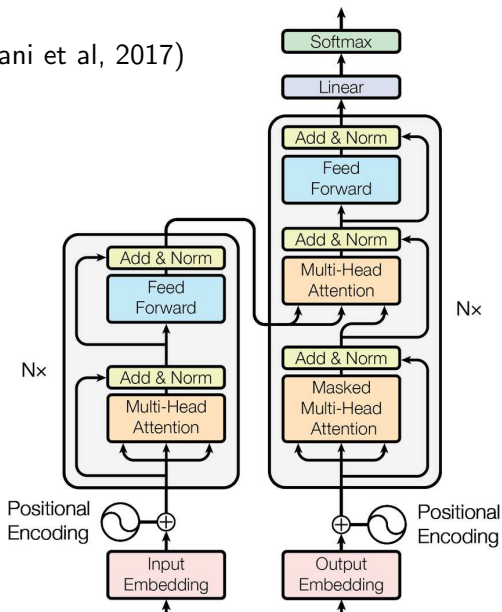


# Architektura Transformer

“Attention is All You Need” (Vaswani et al, 2017)

architektura **transformer**:

- **pod-slovní** vektory (*subword embeddings*)
- vektory **pozice**
- **self-attention**
- více **hlav** (*multi-head attention*)
- **reziduální spojení**, **normalizace** a **škálování**

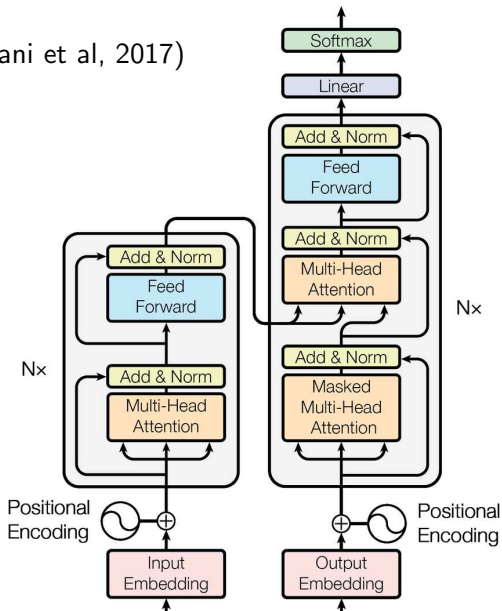


# Architektura Transformer

“Attention is All You Need” (Vaswani et al, 2017)

architektura **transformer**:

- **pod-slovní** vektory (*subword embeddings*)
- vektory **pozice**
- **self-attention**
- více **hlav** (*multi-head attention*)
- **reziduální spojení**,  
normalizace  
a škálování

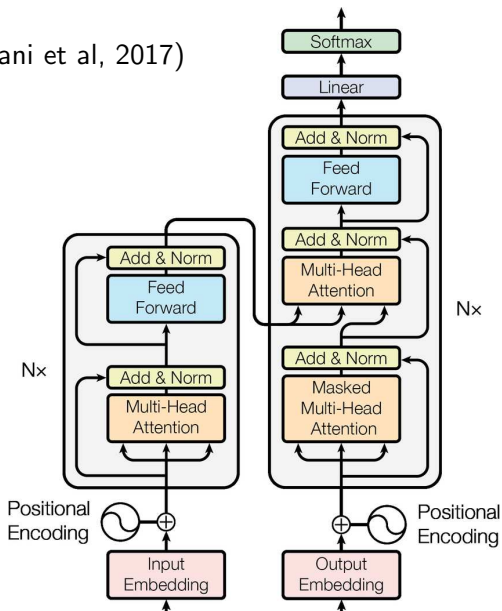


# Architektura Transformer

“Attention is All You Need” (Vaswani et al, 2017)

architektura **transformer**:

- **pod-slovní** vektory (*subword embeddings*)
- vektory **pozice**
- **self-attention**
- více **hlav** (*multi-head attention*)
- **reziduální** spojení,  
**normalizace**  
a **škálování**



# Architektura Transformer

## výhody:

- základ **pokročilých jazykových modelů s předtrénováním**
- aktuálně **nejlepší výsledky** téměř ve všech NLP úlohách
- trénování je dobře **paralelizovatelné**

## nevýhody:

- kvadratický výpočet (plné) self-attention vs. lineární růst u rekurentních modelů  
návrhy – random attention, window attention, ...
- lineární reprezentace pozice  
návrhy – relativní pozice, syntaktická pozice, ...

# Architektura Transformer

## výhody:

- základ pokročilých jazykových modelů s předtrénováním
- aktuálně nejlepší výsledky téměř ve všech NLP úlohách
- trénování je dobře paralelizovatelné

## nevýhody:

- kvadratický výpočet (plné) self-attention vs. lineární růst u rekurentních modelů  
návrhy – random attention, window attention, ...
- lineární reprezentace pozice  
návrhy – relativní pozice, syntaktická pozice, ...

# Architektura Transformer

## výhody:

- základ pokročilých jazykových modelů s předtrénováním
- aktuálně nejlepší výsledky téměř ve všech NLP úlohách
- trénování je dobře paralelizovatelné

## nevýhody:

- kvadratický výpočet (plné) self-attention vs. lineární růst u rekurentních modelů  
návrhy – random attention, window attention, ...
- lineární reprezentace pozice  
návrhy – relativní pozice, syntaktická pozice, ...



# Architektura Transformer

## výhody:

- základ pokročilých jazykových modelů s předtrénováním
- aktuálně nejlepší výsledky téměř ve všech NLP úlohách
- trénování je dobře paralelizovatelné

## nevýhody:

- kvadratický výpočet (plné) self-attention vs. lineární růst u rekurentních modelů  
návrhy – random attention, window attention, ...
- lineární reprezentace pozice  
návrhy – relativní pozice, syntaktická pozice, ...

# Architektura Transformer

## výhody:

- základ pokročilých jazykových modelů s předtrénováním
- aktuálně nejlepší výsledky téměř ve všech NLP úlohách
- trénování je dobře paralelizovatelné

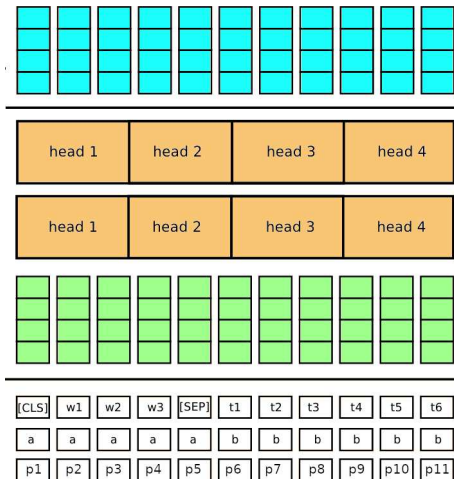
## nevýhody:

- kvadratický výpočet (plné) self-attention vs. lineární růst u rekurentních modelů  
návrhy – random attention, window attention, ...
- lineární reprezentace pozice  
návrhy – relativní pozice, syntaktická pozice, ...

# BERT, ALBERT, RoBERTa

## Bidirectional Encoder Representations from Transformers (BERT)

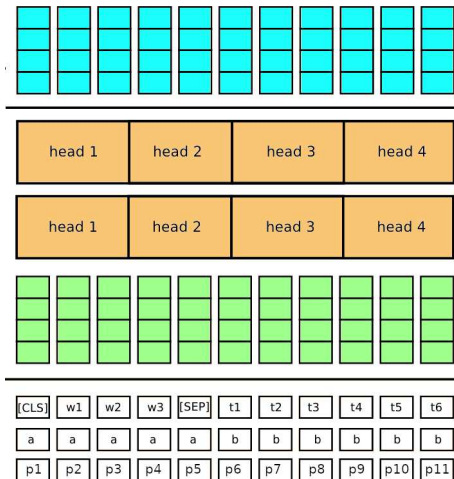
- jen **enkodér**
- symboly **[CLS]**, **[SEP]** a **[MASK]**
- vektory **segmentů**
- **maskovaný vstup**
- **úloha predikce následující věty**  
(*Next Sentence Prediction*)  
u ALBERT **predikce pořadí vět**  
(*Sentence Order Prediction*)



## BERT, ALBERT, RoBERTa

## Bidirectional Encoder Representations from Transformers (BERT)

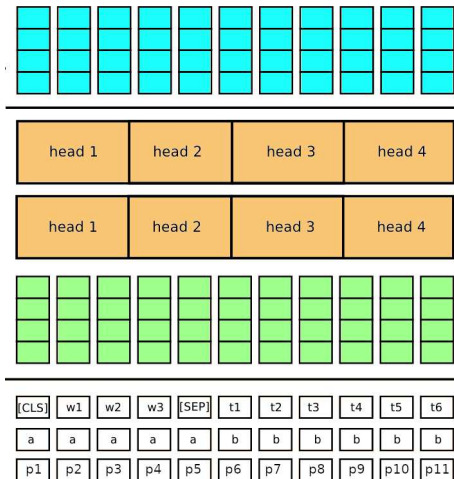
- jen **enkodér**
- symboly **[CLS]**, **[SEP]** a **[MASK]**
- vektory **segmentů**
- **maskovaný vstup**
- **úloha predikce následující věty**  
(*Next Sentence Prediction*)  
u ALBERT **predikce pořadí vět**  
(*Sentence Order Prediction*)



# BERT, ALBERT, RoBERTa

## Bidirectional Encoder Representations from Transformers (BERT)

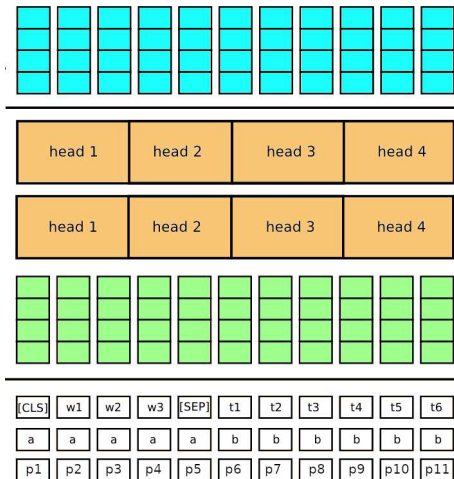
- jen **enkodér**
- symboly **[CLS]**, **[SEP]** a **[MASK]**
- vektory **segmentů**
- **maskovaný vstup**
- **úloha predikce následující věty**  
(*Next Sentence Prediction*)  
u ALBERT **predikce pořadí vět**  
(*Sentence Order Prediction*)



## BERT, ALBERT, RoBERTa

## Bidirectional Encoder Representations from Transformers (BERT)

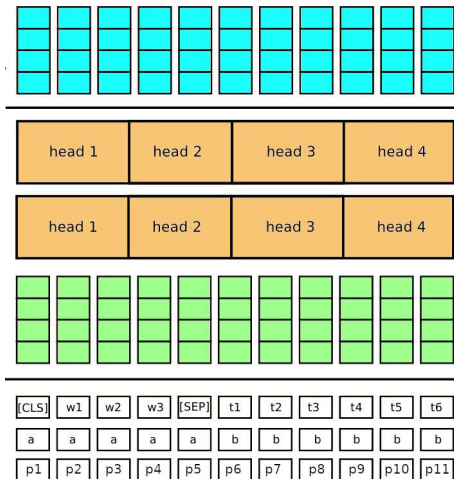
- jen **enkodér**
- symboly **[CLS]**, **[SEP]** a **[MASK]**
- vektory **segmentů**
- **maskovaný** vstup
- úloha **predikce následující věty**  
(*Next Sentence Prediction*)  
u ALBERT **predikce pořadí vět**  
(*Sentence Order Prediction*)



## BERT, ALBERT, RoBERTa

## Bidirectional Encoder Representations from Transformers (BERT)

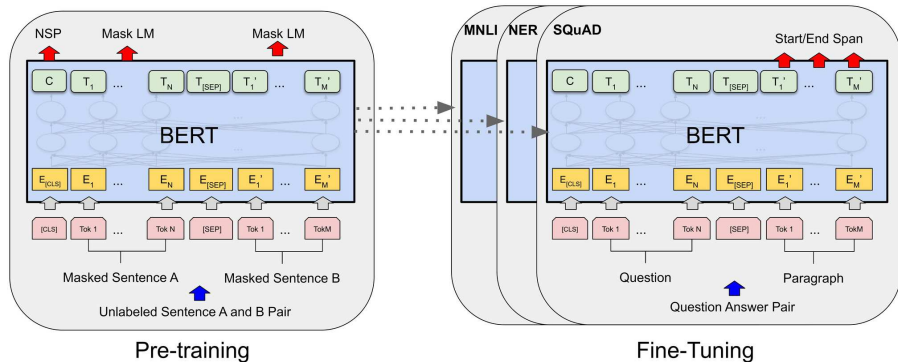
- jen **enkodér**
- symboly **[CLS]**, **[SEP]** a **[MASK]**
- vektory **segmentů**
- **maskovaný** vstup
- úloha **predikce následující věty**  
(*Next Sentence Prediction*)  
u ALBERT **predikce pořadí vět**  
(*Sentence Order Prediction*)



# BERT – předtrénování a vyladění

pro většinu úloh se BERT trénuje ve **dvou fázích**:

- **předtrénování** (*pre-training*) – na **obecných** velkých **textech**
- **vyladění** (*fine-tuning*) – dotrénování klasifikace pro **konkrétní úlohu**





# GPT, T5, ELMo, ERNIE, ELECTRA, ...

množství **variant** architektury **transformer** podle

- doplnění **vstupních vektorů**
- techniky **předtrénování**
- **obsahu a velikosti textů** pro trénování

většinou platí: **větší model** → **lepší výsledky**



Slido

[platform.openai.com/examples](https://platform.openai.com/examples)

# GPT, T5, ELMo, ERNIE, ELECTRA, ...

množství **variant** architektury **transformer** podle

- doplnění **vstupních vektorů**
- techniky **předtrénování**
- **obsahu** a **velikosti textů** pro trénování

většinou platí: **větší model** → **lepší výsledky**



Slido

[platform.openai.com/examples](https://platform.openai.com/examples)

# GPT, T5, ELMo, ERNIE, ELECTRA, ...

množství **variant** architektury **transformer** podle

- doplnění **vstupních vektorů**
- techniky **předtrénování**
- **obsahu** a **velikosti textů** pro trénování

většinou platí: **větší model** → **lepší výsledky**



Slido

[platform.openai.com/examples](https://platform.openai.com/examples)

# GPT, T5, ELMo, ERNIE, ELECTRA, ...

množství **variant** architektury **transformer** podle

- doplnění **vstupních vektorů**
- techniky **předtrénování**
- **obsahu** a **velikosti textů** pro trénování

většinou platí: **větší model** → **lepší výsledky**



Slido

[platform.openai.com/examples](https://platform.openai.com/examples)

# GPT, T5, ELMo, ERNIE, ELECTRA, ...

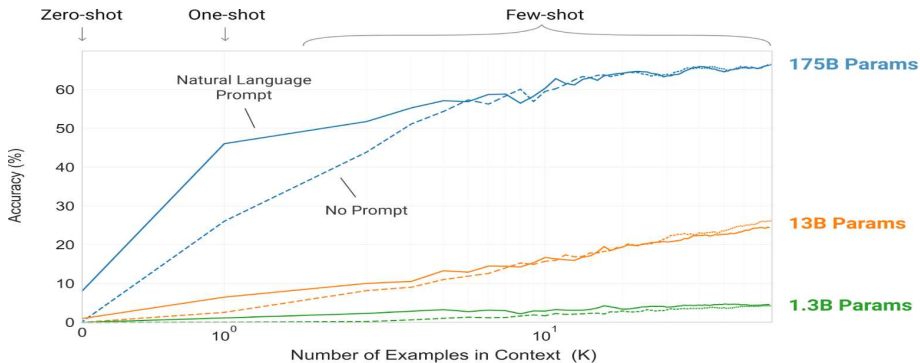
množství **variant** architektury **transformer** podle

- doplnění **vstupních vektorů**
- techniky **předtrénování**
- **obsahu** a **velikosti textů** pro trénování

většinou platí: **větší model** → **lepší výsledky**



Slido



[platform.openai.com/examples](https://platform.openai.com/examples)