Neuronové sítě 2 - Neuronové sítě a sekvenční data

18NES2 - 11.hodina, ZS 2024/25

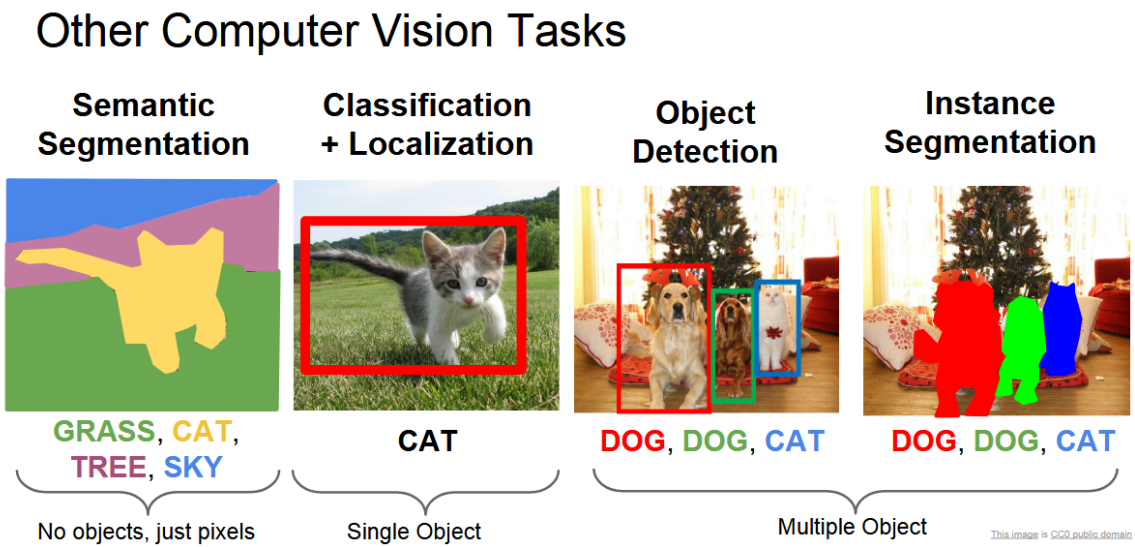
Zuzana Petříčková

11. prosince 2024

Neuronové sítě 2 - Neuronové sítě a sekvenční data

1. Co jsme dělali minule
2. Sekvenční data - časové řady
   - Příklad: predikce teploty
3. Rekurentní neuronové sítě
   - Jednoduchá rekurentní neuronová síť
   - LSTM (Long Short-Term Memory)
   - GRU (Gated Recurrent Unit)
4. Pokročilé aspekty RNN
   - RNN a zobecňování

Co jsme dělali minule: Aplikace konvolučních neuronových sítí



- 2D konvoluce: restaurování či stylyzace obrazu, generování obrazu, určení rysů obličeje, hodnocení podobnosti obrazů,...
- 3D konvoluce: analýza videa, object tracking, rozpoznávání akcí (např. sport)
- 1D konvoluce: časové řady, audio záznamy, obecně sekvenční data (např. přirozený jazyk - omezeně)

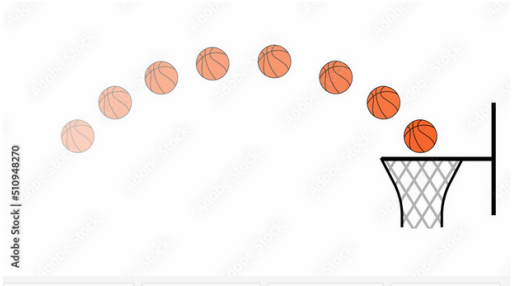
Dnes: Neuronové sítě a sekvenční data

- Sekvenční data a jejich reprezentace
- Sekvenční data a jejich zpracování pomocí MLP, CNN
  - Příklad: predikce časové řady
- Rekurentní neuronové sítě (RNN)
  - Základní model: jednoduchá (Vanilla /Elmann) RNN
  - LSTM, GRU
  - rekurentní dropout, vícevrstvé RNN, obousměrné RNN

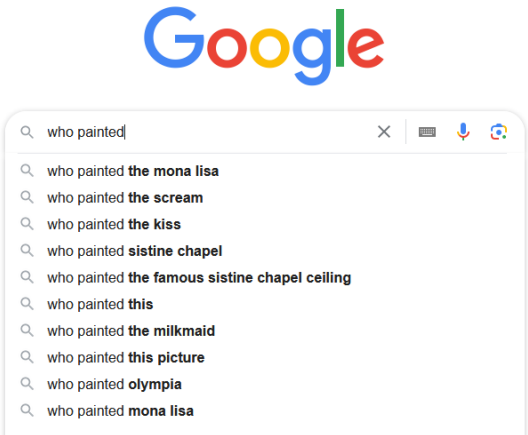
Sekvenční data

Příklady úloh:

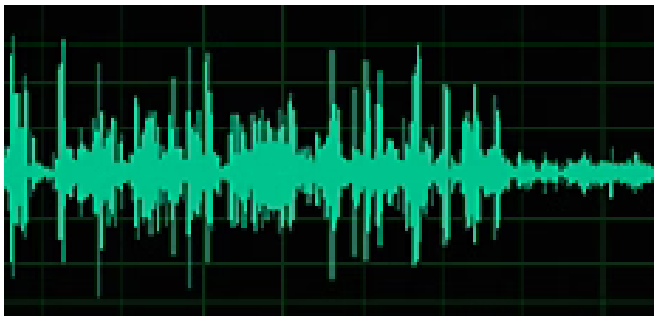
- Jaká bude poloha hozeného míče v dalším okamžiku?



- Dokonči větu:



- Kdo to mluví?



Sekvenční data

Časové řady

- různá granularita: denní ceny akcií, hodinová spotřeba elektřiny, týdenní tržby v obchodě,...
- různý charakter dat: návštěvnost webových stránek, transakce na kreditní kartě, seismická aktivita, vývoj počasí,...

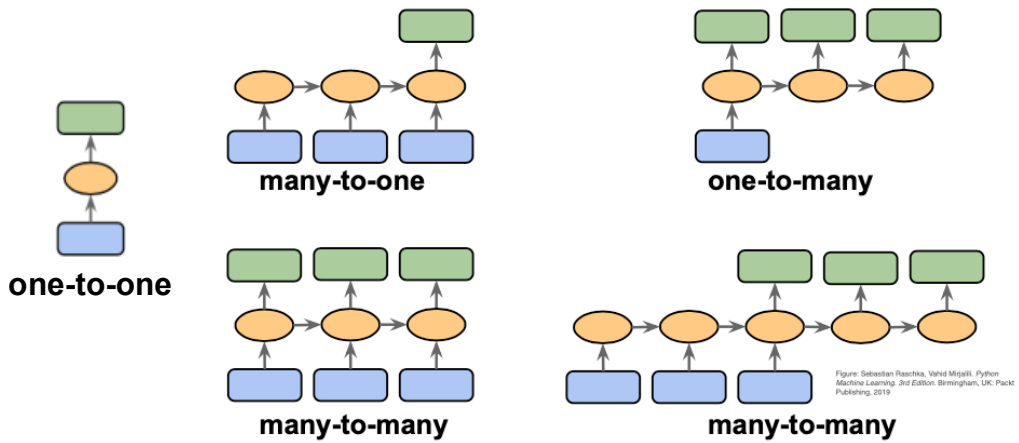
Sekvenční data nejsou jen časové řady:

- **audiodata:** rozpoznávání řeči, identifikace mluvčího, rozpoznání emocí, akustická lokalizace, analýza hudby,...
- **text:** analýza sentimentu, strojový překlad, predikce následujících slov ve větě,...
- **video:** object tracking, predikce trajektorií, generování popisků videí,...
- **biologická data:** analýza DNA sekvencí, monitorování srdečního tepu,...

Typy úloh nad sekvenčními daty

Typy úloh nad sekvenčními daty

- **one-to-one** - např. obyčejná klasifikace
- **many-to-one** - např. klasifikace sentimentu, rozpoznání akce na videu
- **one-to-many** - např. vygenerovat slovní popis k obrázku, sentence tagging
- **many-to-many** - strojový překlad, object tracking



zdroj: S. Raschka: Introduction to Recurrent Neural Networks, https://sebastianraschka.com/blog/2021/dl-course.html

Příklad: Predikce časové řady

Časová řada

- sekvenční data, pro které je typická periodičnost (denní, roční,....)

Typické úlohy

- **predikce** - předpovídání následující hodnoty (many-to-one) / následujících hodnot (many-to-many)
- **klasifikace** - např. rozpoznání, zda je návštěvník stránky bot nebo člověk, zhodnocení rizika budoucího infarktu pacienta z EKG snímku,...
- **detekce událostí** - např. seismická aktivita, rozpoznání klíčových slov v audio-proudu („OK, Google")
- **detekce anomálií** - např. neobvyklá aktivita na síti
  řeší se obvykle technikami učení bez učitele (shlukování, autoencodery)

Příklad: Predikce časové řady

Příklad: Jena Climate Dataset

- meteorologická data naměřená v Ústavu Maxe Plancka v Jeně v letech 2009-2016 (8 let)
- 15 příznaků (čas, teplota, atmosferický tlak, vlhkost, rychlost a směr větru,...)
- měřeno každých 10 minut, celkem kolem 400000 vzorků

Úloha

- predikce teploty ,,zítra v tuto dobu" na základě dat z posledních 5 dnů
- pro predikci použijeme hodnoty 14 příznaků (čas vynecháme)
- bereme v úvahu jen data měřená jednou za hodinu $\rightarrow$ 24 * 5 $=$ 120 hodnot
- vstupní vzor je tvaru 120×14, výstup je jedno číslo (teplota za 24 hodin)

Příklad: Predikce časové řady

Načtení a příprava dat

- **Ukázka:** načtení CSV souboru a vytvoření datasety pro časovou řadu v Kerasu
- **Pozor**: vzory ve validační a testovací množině musí v čase následovat za vzory v trénovací množině
  - typická chyba: náhodné promíchání vzorů před rozdělením dat na trénovací, validační a testovací
  - vzory v trénovací množině již můžeme seřadit náhodně

Základní referenční úroveň

- **Primitivní predikce:** ,,Teplota za 24 hodin bude stejná jako je teď"
- $MAE_{test} = 2.62$, tj. model se v průměru mýlí o 2.62 °C (to není mnoho)

Příklad: Predikce časové řady

Řešení pomocí MLP

- průměrná chyba zhruba dosahuje základní referenční úrovně
- Proč není lepší?
  - hledání jehly v kupce sena

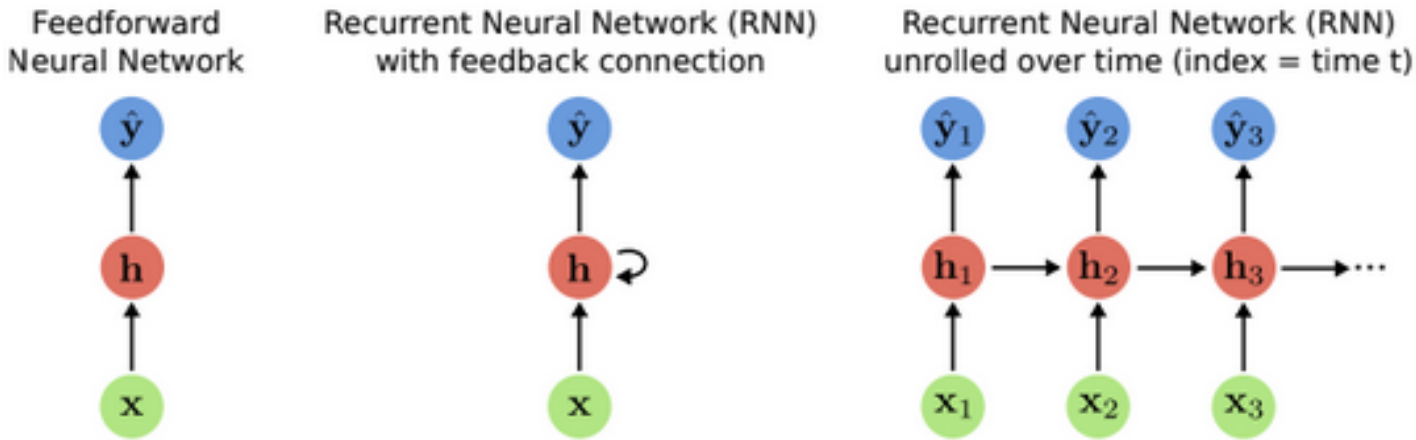
Řešení pomocí CNN

- 1D konvoluce
- dokonce horší než základní referenční úroveň
- Proč se nedaří?
  - data nejsou invariantní vůči posunutí
  - záleží na pořadí dat (nedávná minulost je důležitější než vzdálená)

Řešení pomocí jednoduchého LSTM modelu (rekurentní model neuronové sítě)

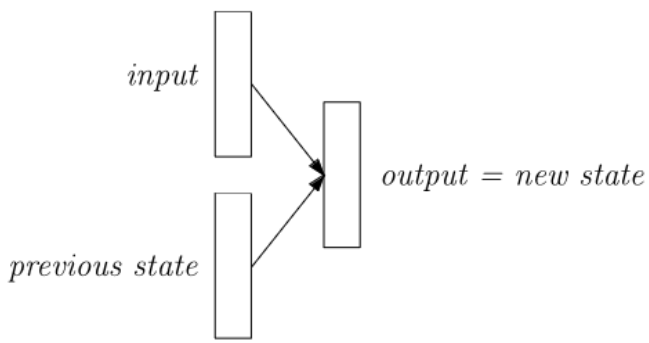
- konečně překonává základní referenční úroveň

Rekurentní neuronová síť (RNN)

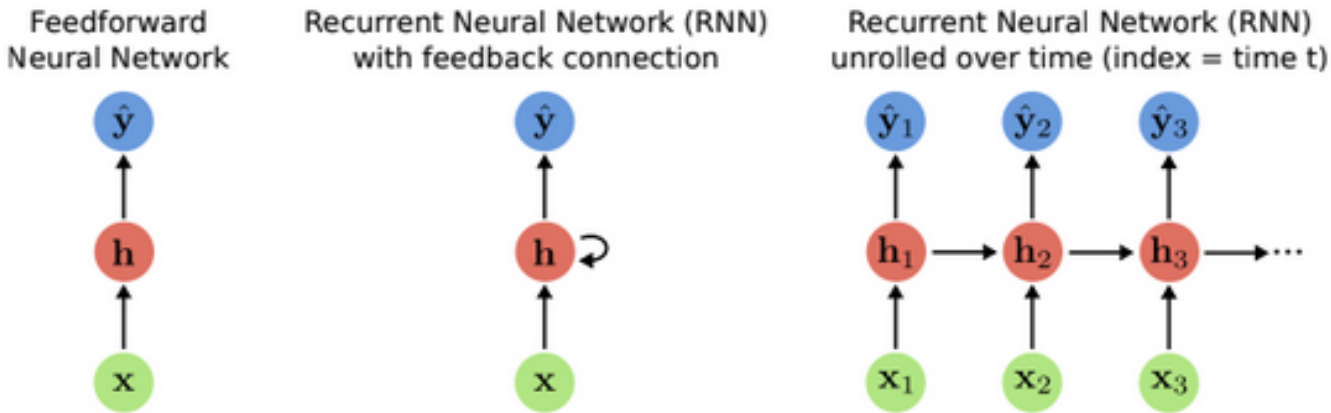


Myšlenka:

- neurony si udržují svůj vnitřní stav (mají vnitřní paměť)
  - neuronům v RNN se někdy říká „buňky" (*memory cells*)
- výstup neuronu závisí nejen na vstupních datech, ale i na jeho stavu (minulém výstupu)

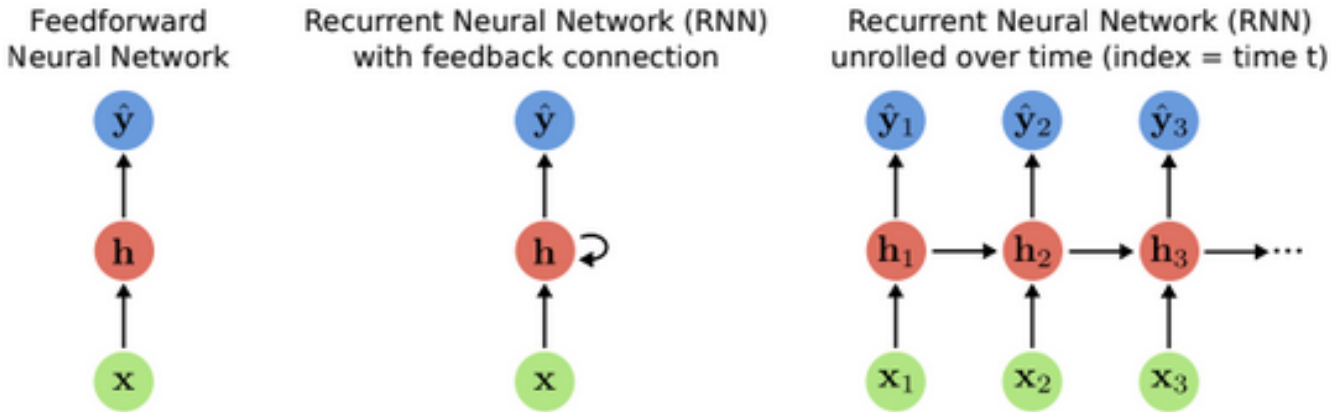


Jednoduchá rekurentní neuronová síť (vanilla/Elman RNN)



- neuron si udržuje svůj vnitřní stav: $h_t = f_h(h_{t-1}, x_t)$
- výstup neuronu: $y_t = f_y(h_t)$
- modelu předložíme celou posloupnost dat
  - libovolné délky
  - model si můžeme ,,rozbalit" (obrázek vpravo)
  - f_y, f_h se (pro jednu vstupní posloupnost) v čase nemění
- pro novou posloupnost se stav resetuje

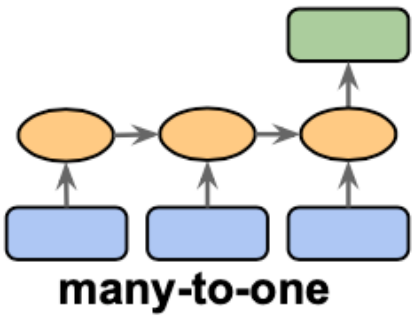
Jednoduchá rekurentní síť (vanilla/Elman RNN)

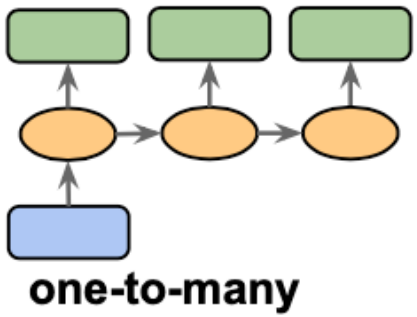


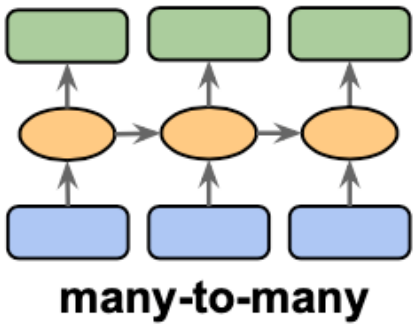
- vnitřní stav: $h_t = f_h(h_{t-1}, x_t)$
  - obvyklá přenosová funkce f_h je tanh
  - vnitřní stav: $h_t = tanh(w_h h_{t-1} + w_x x_t + b_h)$
- výstup neuronu $y_t = f_y(h_t) = w_y h_t + b_y$
- modelu předložíme celou posloupnost dat
  - libovolné délky
  - model si můžeme ,,rozbalit" (obrázek vpravo)
  - váhy a biasy w, b se (pro jednu vstupní posloupnost) v čase nemění

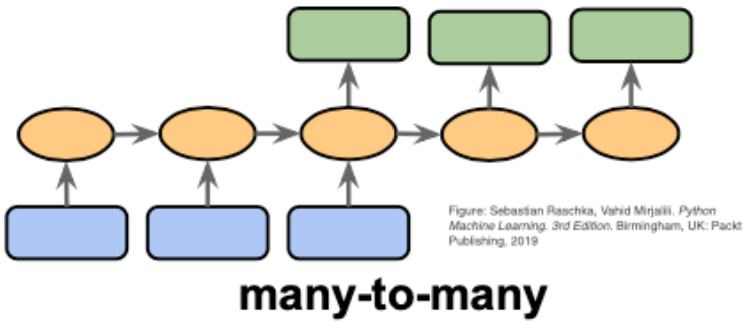
Jednoduchá rekurentní síť (vanilla/Elman RNN)

- ukázky rozbalení neuronu pro jednotlivé typy úloh:



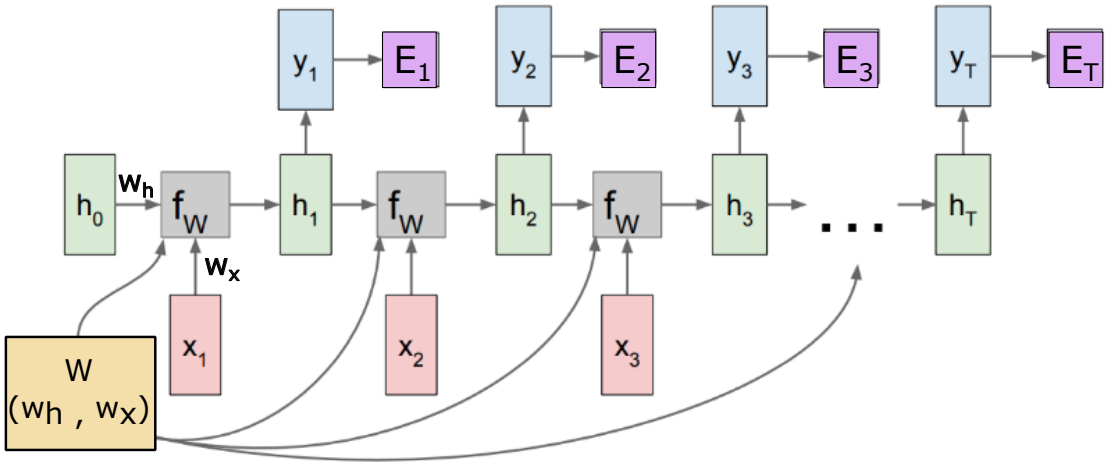






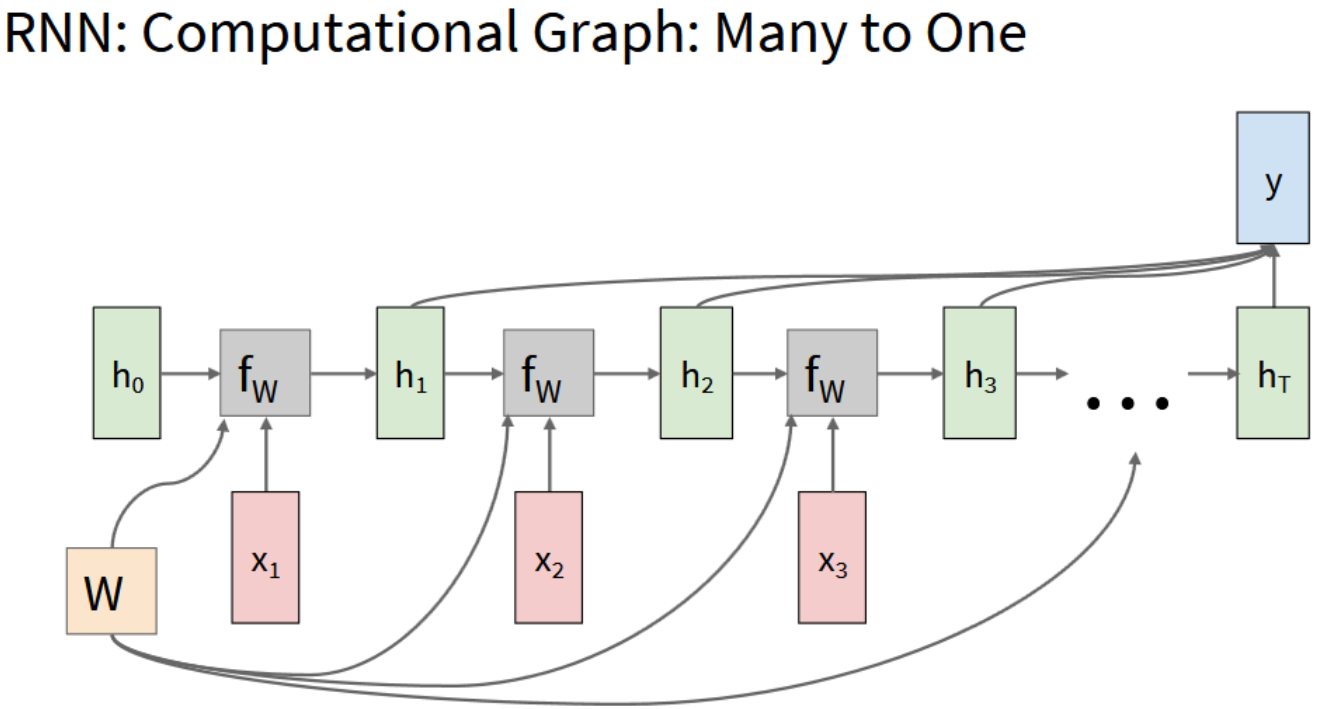
Jednoduchá rekurentní síť (vanilla/Elman RNN)

Algoritmus učení: backpropagation through time (BPTT)



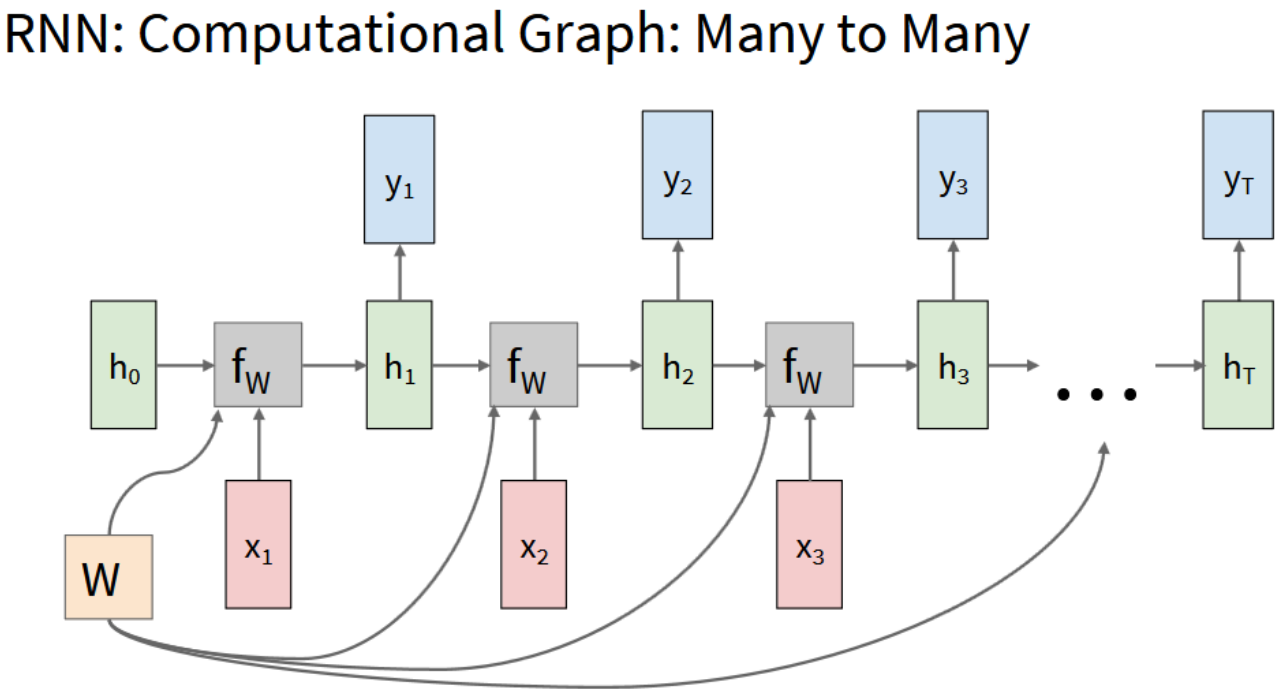
- ukázka pro architekturu many-to-many
- neuron si můžeme představit rozbalený do MLP
- zpětná propagace pro celou vstupní posloupnost ,,najednou"

Jednoduchá rekurentní síť (vanilla/Elman RNN)



zdroj: S. Raschka: Introduction to Recurrent Neural Networks,
https://sebastianraschka.com/blog/2021/dl-course.html

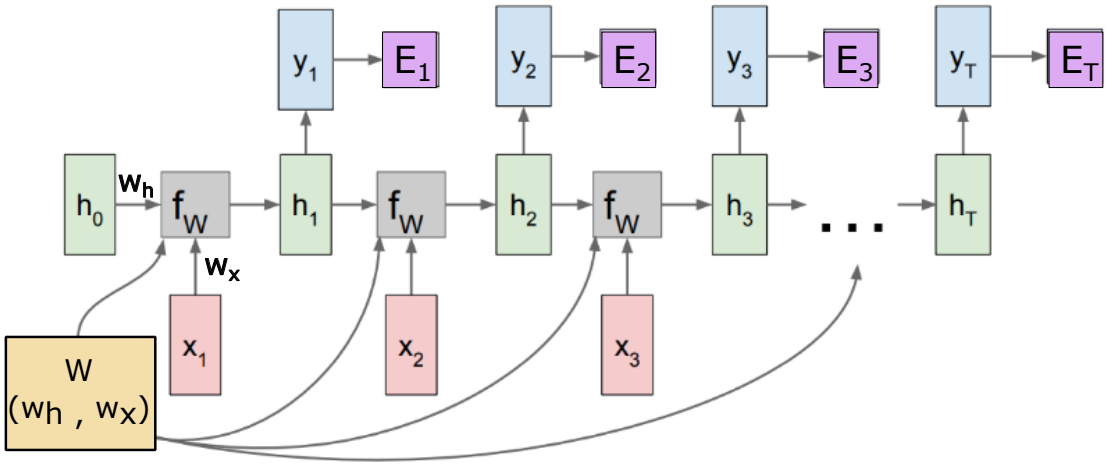
Jednoduchá rekurentní síť (vanilla/Elman RNN)



zdroj: S. Raschka: Introduction to Recurrent Neural Networks,
https://sebastianraschka.com/blog/2021/dl-course.html

Jednoduchá rekurentní síť (vanilla/Elman RNN)

Algoritmus učení: backpropagation through time (BPTT)



- ukázka pro architekturu many-to-many
- neuron si můžeme představit rozbalený do MLP
- zpětná propagace pro celou vstupní posloupnost ,,najednou"

Jednoduchá rekurentní síť (vanilla/Elman RNN)

Problémy algoritmu BPTT

- při zpětné propagaci se navzájem násobí velké množství čísel (derivace stavu vůči předchozímu, vůči vahám)
- **problém mizejících gradientů:**
  - pokud jsou derivace přenosové funkce nebo váhy blízké 0 $\rightarrow$ násobíme hodně malých čísel $\rightarrow$ moc malý krok učení
  - Výsledek: model má problém zachovat dlouhodobé závislosti, zapomíná.
- **problém explodujících gradientů**
  - pokud jsou derivace přenosové funkce nebo váhy moc velké $\rightarrow$ násobíme hodně velkých čísel $\rightarrow$ moc velký krok učení
  - Výsledek: model má nestabilní učení (často vede k NaN hodnotám)

$\rightarrow$ model má problém se učit

Řešení problému mizejících a explodujících gradientů

Explodující gradienty: Gradient clipping

- pokud je průměrný gradient větší než daná mez, normalizujeme ho (vydělíme průměrem)

Problém mizejících gradientů

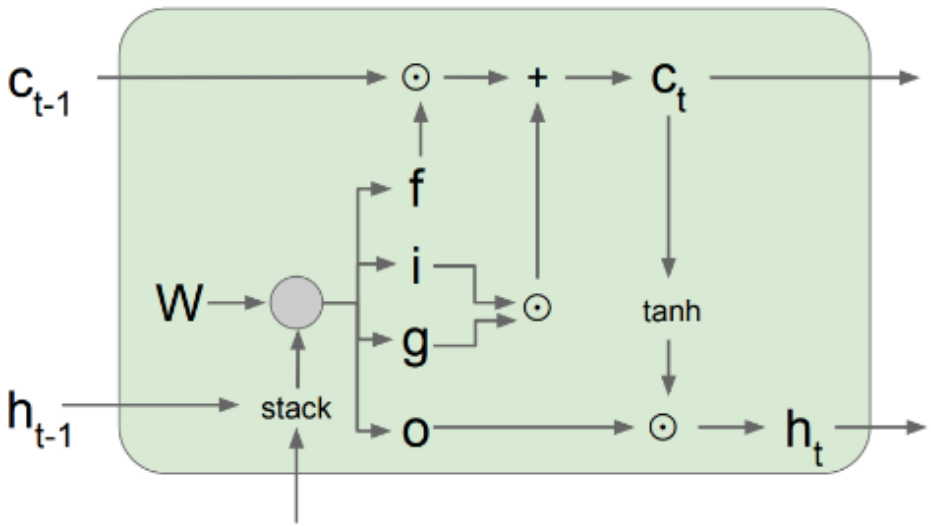
- model zapomíná, má špatnou dlouhodobou paměť
- ukázka obtížného příkladu: ,,The dogs in the neighborhood are ..." (barking)
- **řešení**:
  - echo state networks
  - LSTM (Long Short-Term Memory, 1997)
  - GRU (Gated recurrent unit, 2014)

LSTM (Long Short-Term Memory)

Myšlenka:

- násobení nahradíme sčítáním
- krátkodobý stav buňky: h_t
- dlouhodobý stav buňky: c_t uchovává dlouhodobou informaci
- používá brány (gates): vstupní (i), zapomínací (f), výstupní (o), brána aktuálního stavu (g)

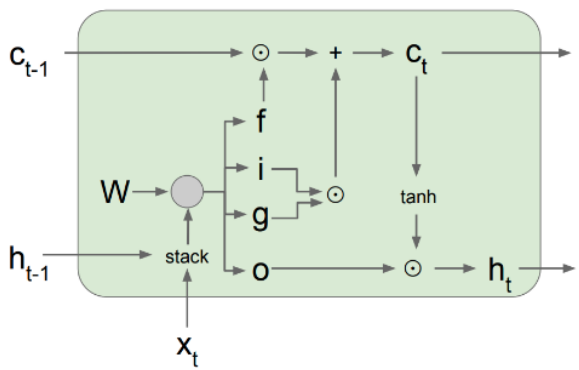
Buňka (memory cell):



LSTM (Long Short-Term Memory)

Možná interpretace:

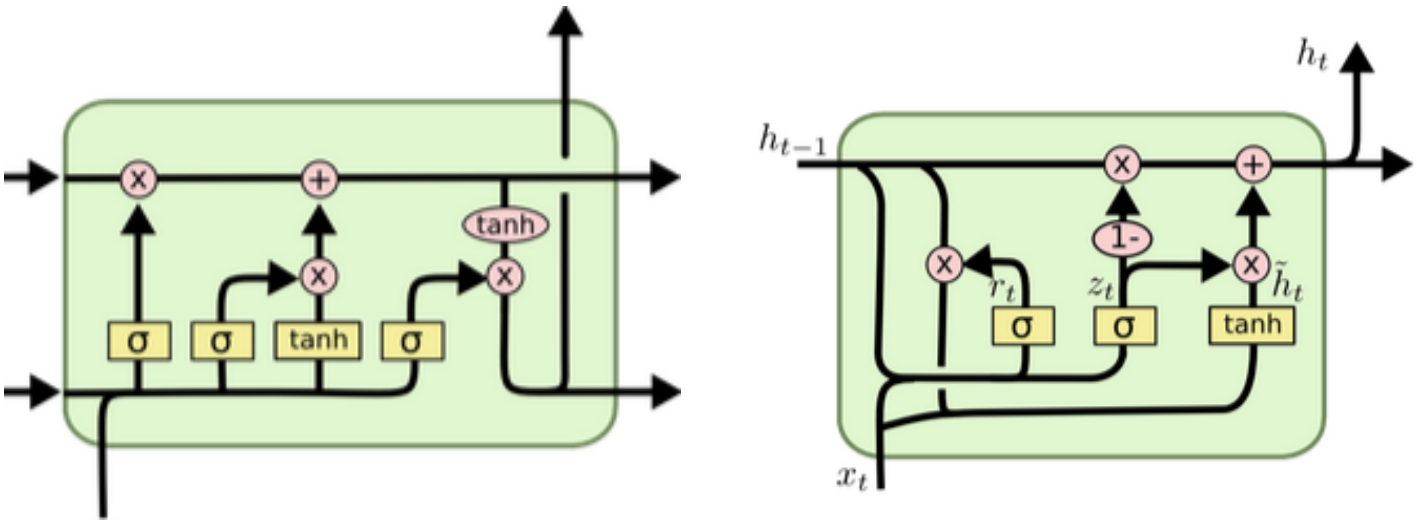
- **vstupní brána: zda informaci zapsat do paměti**
  $i_t = \sigma(w_h^{(i)} h_{t-1} + w_x^{(i)} x_t + b_h^{(i)})$
- **brána aktuálního stavu: kolik informace zapsat**
  $g_t = tanh(w_h^{(g)} h_{t-1} + w_x^{(g)} x_t + b_h^{(g)})$
- **zapomínací brána: zda zapomenout**
  $f_t = \sigma(w_h^{(f)} h_{t-1} + w_x^{(f)} x_t + b_h^{(f)})$
- **výstupní brána: kolik informace dát na výstup**
  $o_t = \sigma(w_h^{(o)} h_{t-1} + w_x^{(o)} x_t + b_h^{(o)})$



Gated Recurrent Unit (GRU)

- jednodušší buňky oproti LSTM, méně parametrů, efektivnější výpočet
- vstupní a výstupní brána kombinovány do jedné:
  - **aktualizační brána:** kolik informace zapsat.
  - **resetovací brána:** kolik stavu zapomenout.
- dlouhodobý a krátkodobý stav v jednom

LSTM (vlevo) a **GRU** (vpravo)



[zdroj]

Gated Recurrent Unit (GRU)

Výhody GRU:

- Jednodušší struktura a rychlejší trénování ve srovnání s LSTM.
- V některých úlohách dosahuje podobné přesnosti jako LSTM.
- Lépe funguje na menších datasetech.

Nevýhody:

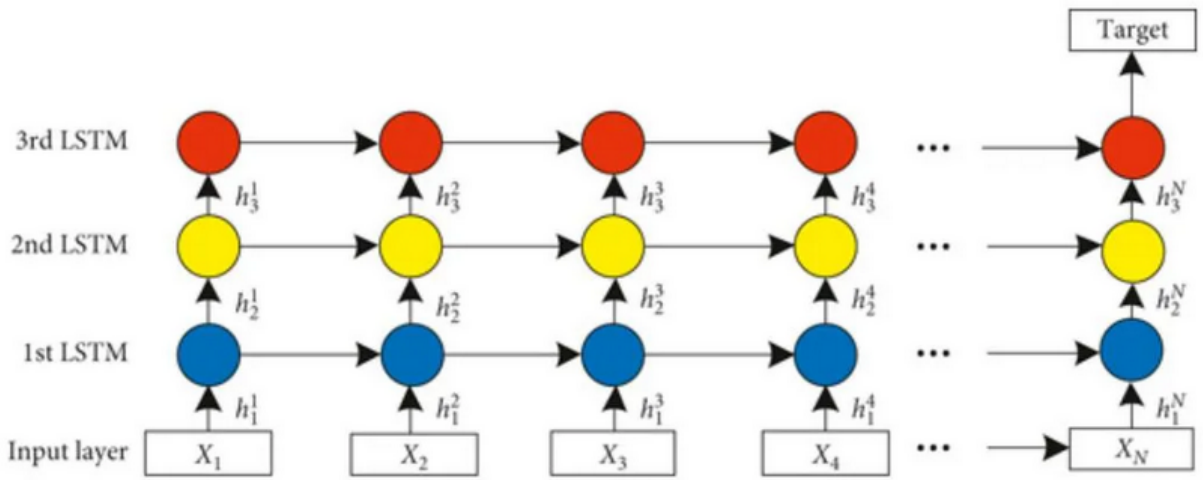
- Menší flexibilita než LSTM při modelování velmi složitých časových závislostí.
- Větší citlivost k volbě parametrů

Rekurentní neuronové sítě a zobecňování

- RNN jsou náchylné k přeučení, zejména u dlouhých sekvencí
- běžný dropout před rekurentní vrstvou nefunguje - narušuje učení dlouhodobých závislostí
- používá se speciální **rekurentní dropout**
  - náhodně vypíná rekurentní vazby (spojení mezi časovými kroky)
  - stejná maska v každém časovém kroku (kvůli konzistenci)
- **layer normalization** (namísto batch normalization) pro hlubší modely RNN

Vícevrstvé rekurentní neuronové sítě

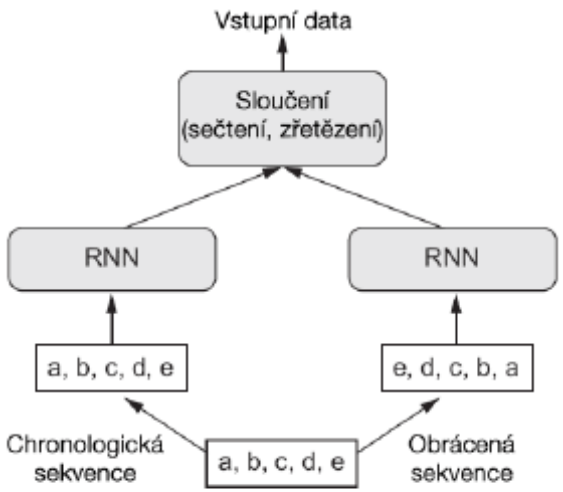
- zvětšení kapacity modelu
- umožňují modelovat složitější (a dlouhodobější) závislosti
- větší riziko přeučení $\rightarrow$ kombinujeme je s dropoutem nebo s normalizací



https://python.plainenglish.io/stacked-rnns-in-nlp-936e6eecf37a

Obousměrné rekurentní neuronové sítě (bidirectional RNN)

- na vstupní sekvenci se dívá v chronologickém i opačném pořadí
- vhodný pro úlohy zpracování přirozeného jazyka
- jedná se vlastně o ensemble model



Obrázek 10.14:
Jak pracuje vrstva obousměrné RNN

Zdroj obrázku: F. Chollet: Deep learning v jazyku Python, obr 10.14

Rekurentní neuronová síť (RNN) - shrnutí

- RNN umožňují zpracovávat vstupní vzory s různou délkou
- variabilní architektura (one-to-many, many-to-one, many-to-many,...)
- vhodné pro **sekvenční data**, kde existují závislosti mezi prvky dané posloupností (např. časové řady, text)
- jednoduché RNN se obtížně učí $\rightarrow$ v praxi se používají LSTM a GRU, gradient clipping
- RNN jsou citlivé na přeučení a volbu parametrů $\rightarrow$ regularizace
- architektura je ,,hluboká", ale v jiném - ,,časovém" rozměru (ale existují i vícevrstvé rekurentní sítě)
- existuje celá řada dalších architektur RNN
- **Slabiny**: problematická GPU akcelerace a paralelizace z důvodu sekvenčních výpočtů
- z RNN vycházejí i modernější **transformery**