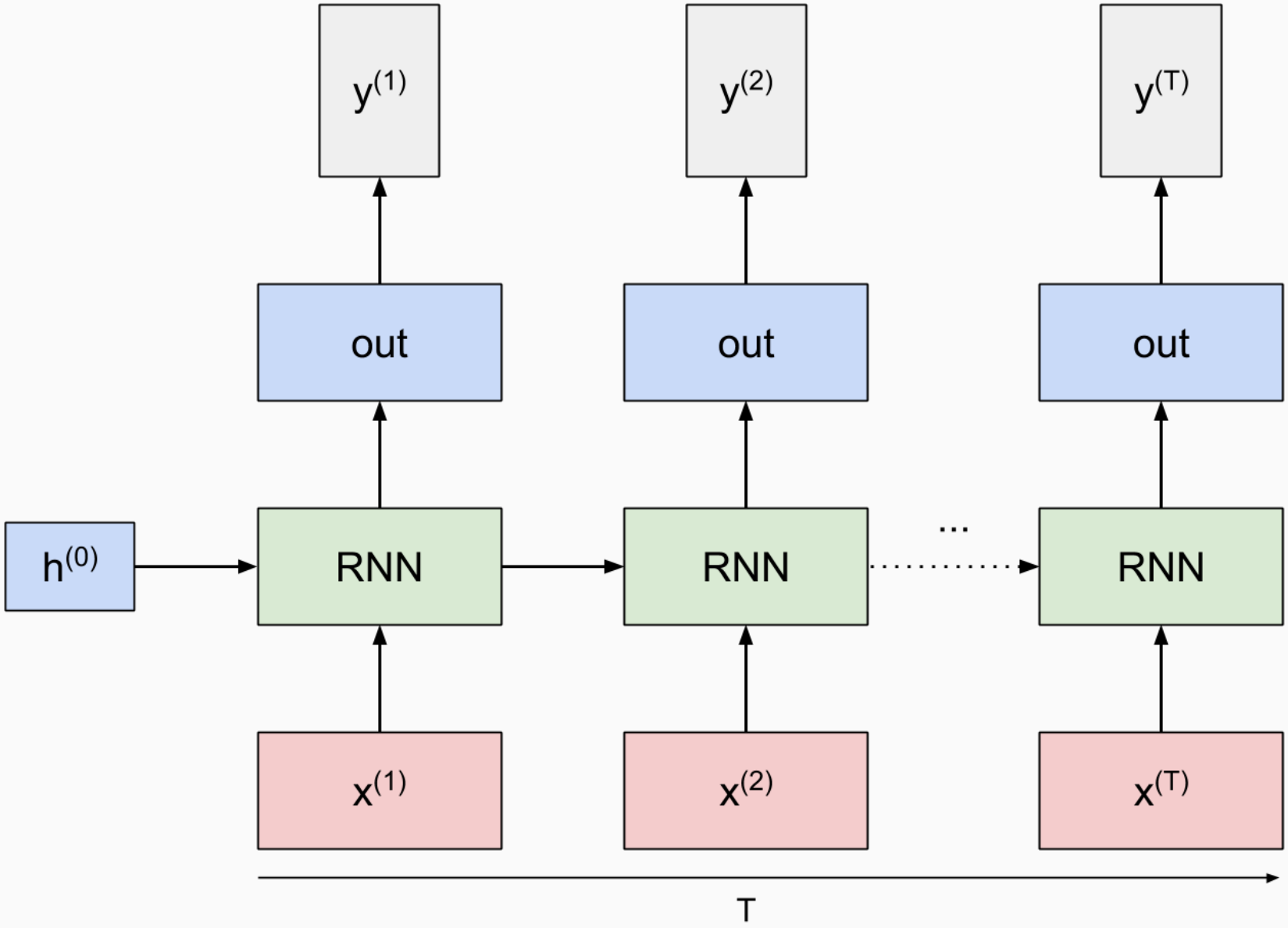
Advanced recurrent models

Martin Tutek, Petra Bevandić, Josip Šarić, Siniša Šegvić

2023.

Ponavaljanje

Obični povratni model (RNN označava povratnu ćeliju)



Osnovna povratna ćelija

Ažuriranje skrivenog stanja:

$$h^{(t)} = tanh(\underbrace{W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h}_{a^{(t)}})$$

Projiciranje izlaza:

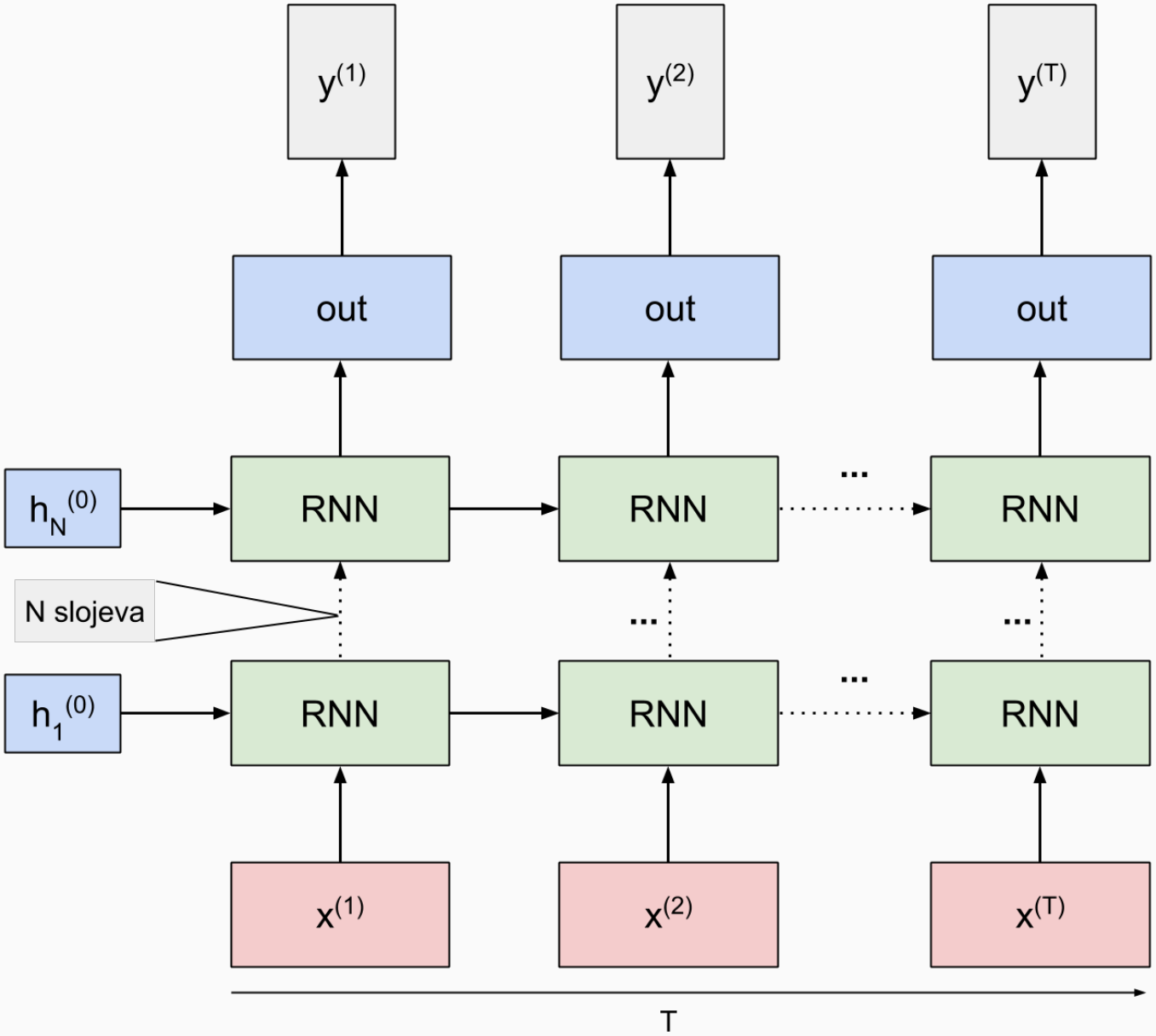
$$o^{(t)} = W_{hy}h^{(t)} + b_o \tag{1}$$

Obični povratni model obrađuje cijeli slijed *jednim* slojem:

- često nedovoljno za učenje složenih ovisnosti među elementima slijeda
- može se poboljšati dodavanjem latentnih slojeva između ulaza i predikcija

Duboki povratni modeli

Duboki (višerazinski) povratni modeli



Duboki povratni modeli

Možete li vidjeti problem na prethodnoj slici?

- $x^{(t)}$ i $h^{(t)}$ mogu imati različitu dimenzionalnost
- dimenzionalnost $W_{xh} \in \mathbb{R}^{h \times h}$ može se mijenjati preko slojeva

Sloj $n = 1$:

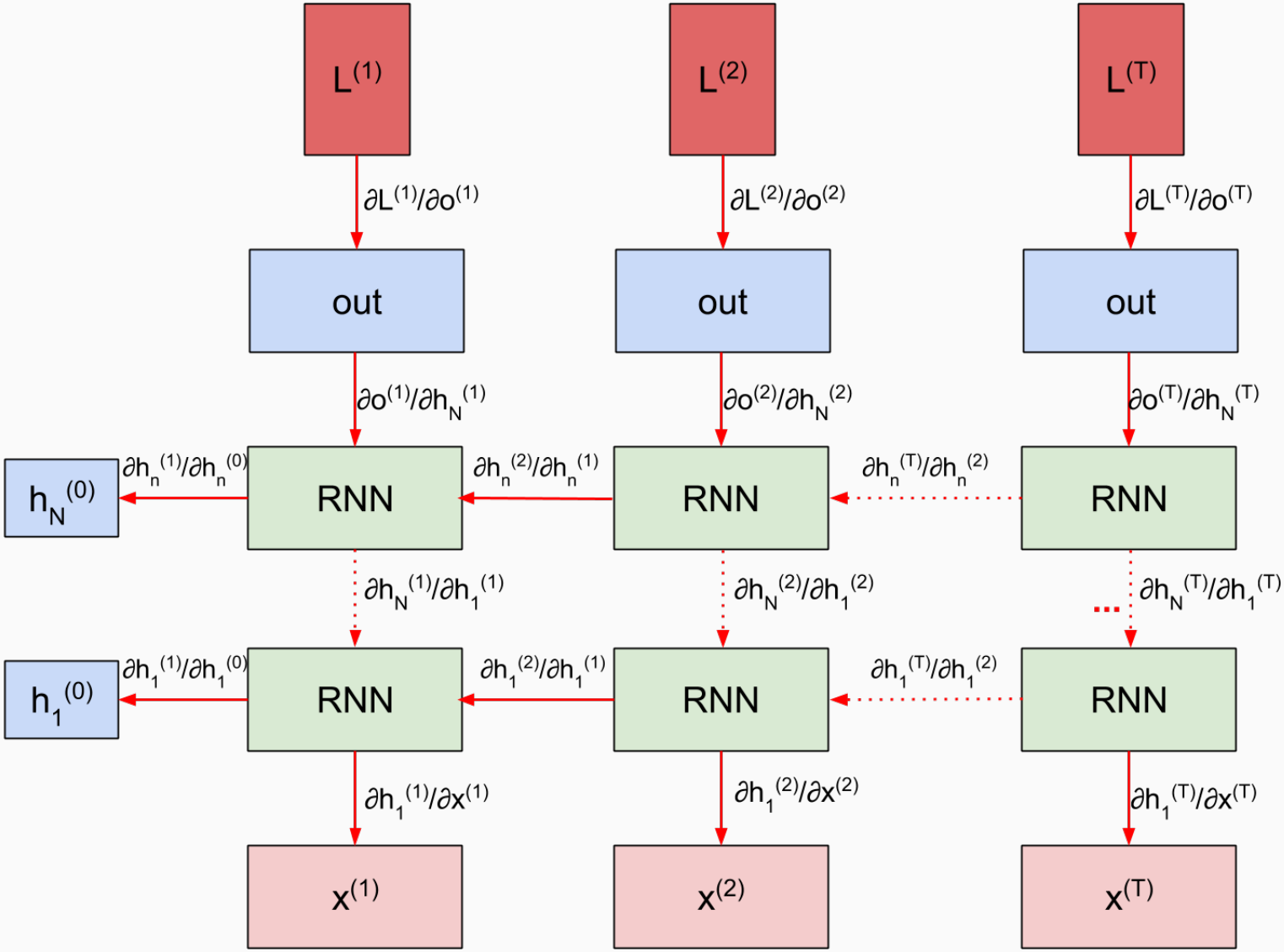
$$h_n^{(t)} = tanh(\underbrace{W_{nhh} h_n^{(t-1)} + W_{nxh} x^{(t)} + b_{nh}}_{a_n^{(t)}}) \tag{2}$$

Sloj $n > 1$:

$$h_n^{(t)} = tanh(\underbrace{W_{nhh} h_n^{(t-1)} + W_{nxh} h_{n-1}^{(t)} + b_{nh}}_{a_n^{(t)}}) \tag{3}$$

[!!] Ovu izmjenu ne morate raditi ručno: okviri nude povratne ćelije koje automatski prilagođavaju dimenzionalnost parametara.

Duboki povratni modeli: backprop



Duboki povratni modeli: sažetak

Povratni modeli duboki su kroz slojeve (vertikalno) i vrijeme (horizontalno):

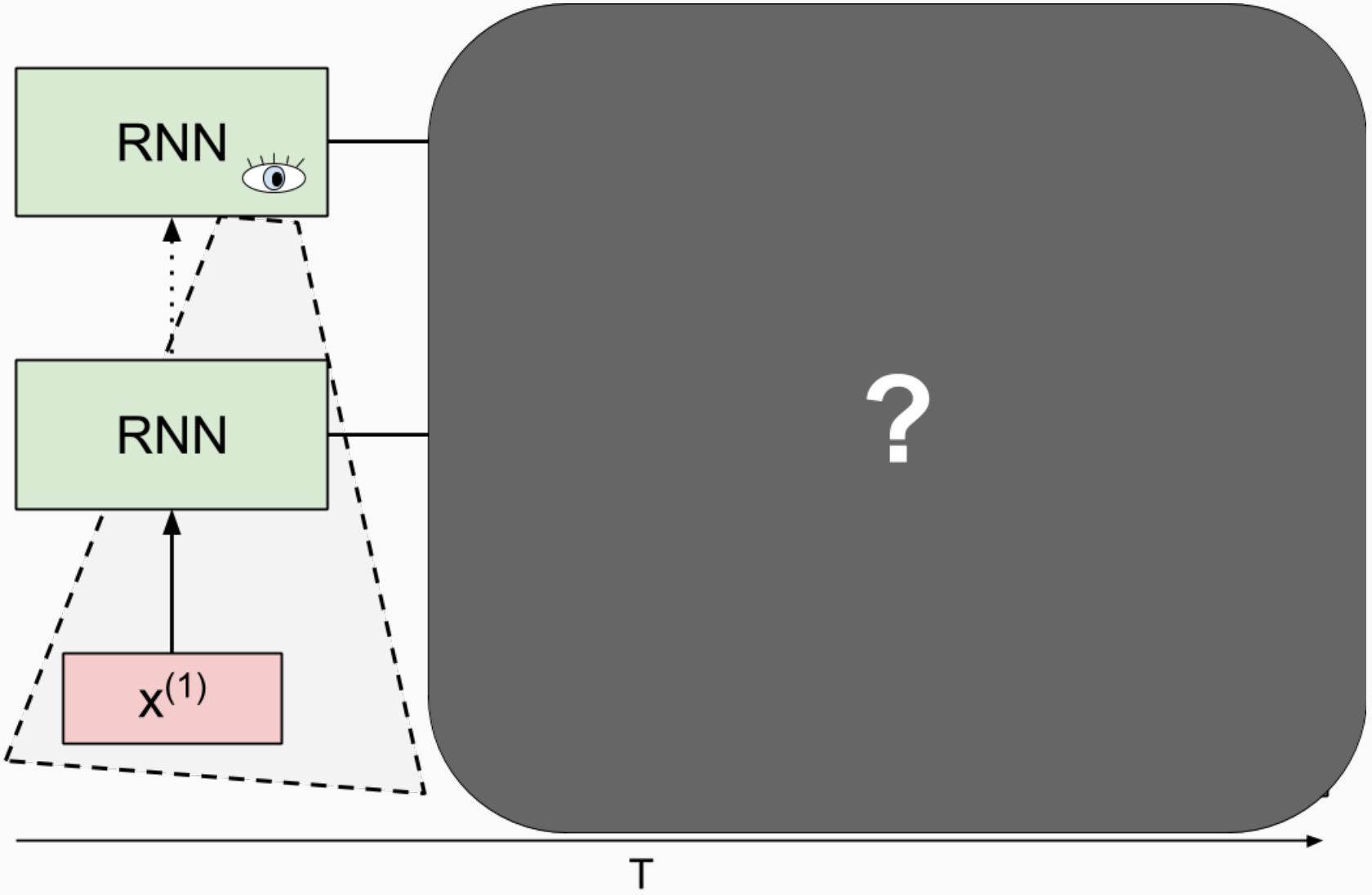
- praktične konfiguracije imaju 4 to 8 slojeva ovisno o količini podataka za učenje
- više od 8 povratnih slojeva ne dovodi do značajno bolje generalizacije (čak i za napredne ćelije)

Dimenzionalnost ulaza tipično je različita od dimenzionalnosti skrivenih slojeva:

- to najčešće ne komplicira programsku izvedbu
- slojeve konfiguriramo zadavanjem argumenata konstruktora odabrane povratne ćelije.

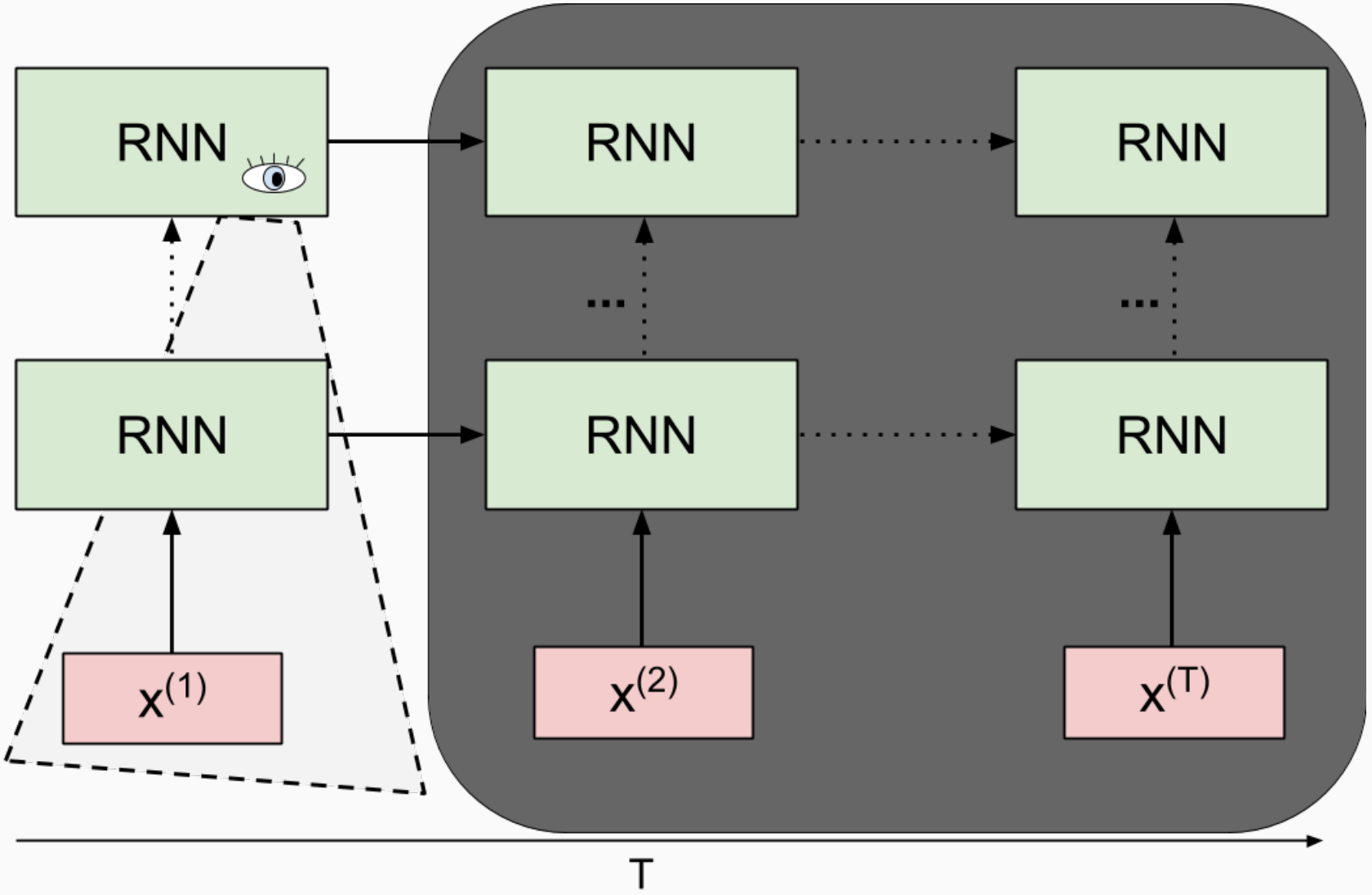
Problemi

Koliko je **receptivno polje** povratne ćelije?



Problemi

Koliko je **receptivno polje** povratne ćelije?



Problemi: receptivno polje

Povratna ćelija (u bilo kojem sloju) u trenutku t vidi samo $x^{(t)} \leq t$:

- predikcija u trenutku t određena je samo s do tada viđenim ulazima!
- ako zadatak ne pretpostavlja skrivanje *budućeg* konteksta, htjeli bismo omogućiti modelu da vidi cijeli slijed prije donošenja odluke.

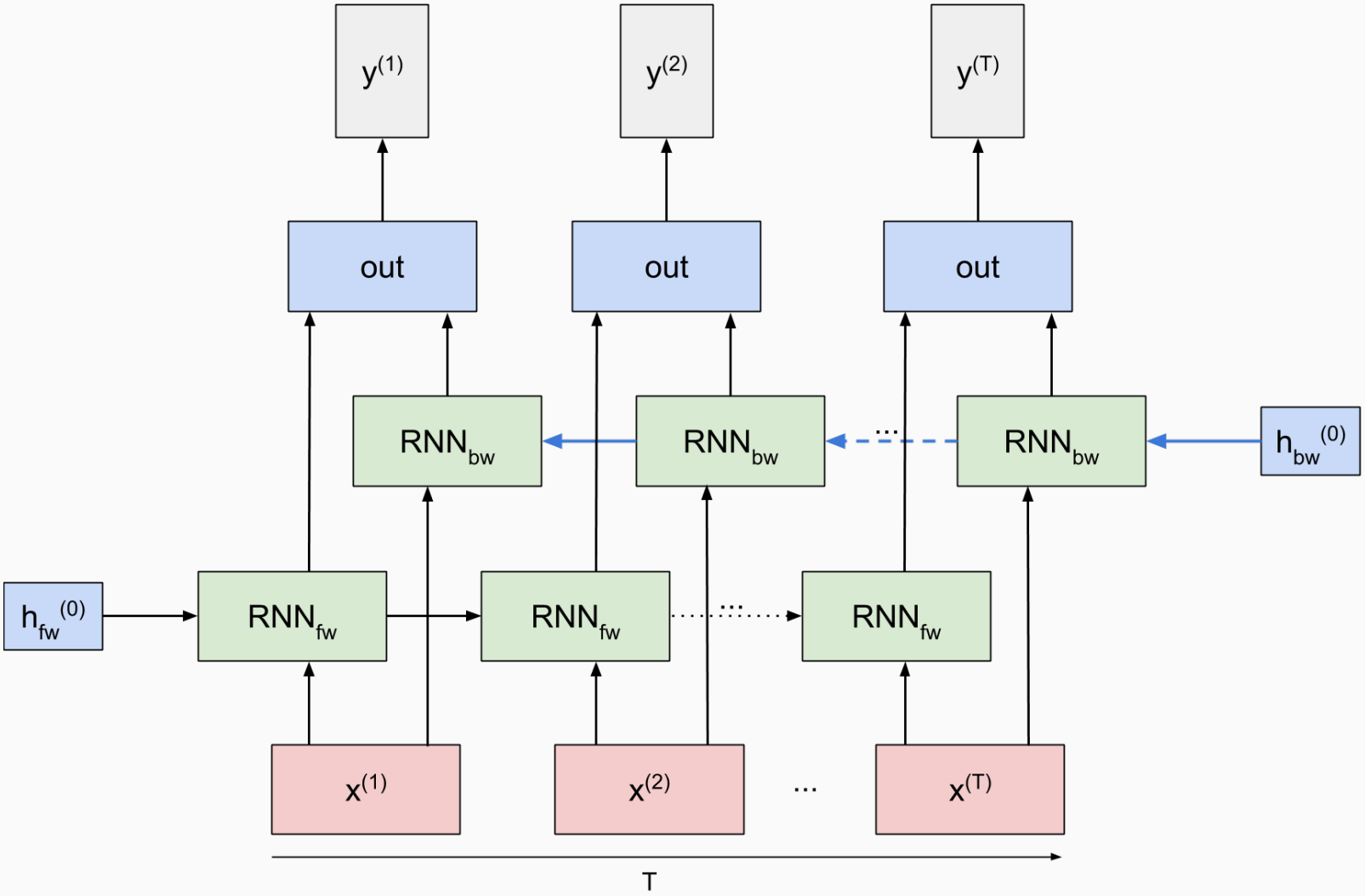
Ideja: ako ciljno stanje $h^{(t)}$ ćelije koja gleda s lijeva na desno vidi $x^{(t)} \leq t$, tada će ćelija koja gleda u suprotnom smjeru vidjeti sve preostale ulaze $x^{(t)} > t$

- zajedno, te dvije ćelije vide cijeli ulazni niz

Bidirekcionalni povratni modeli

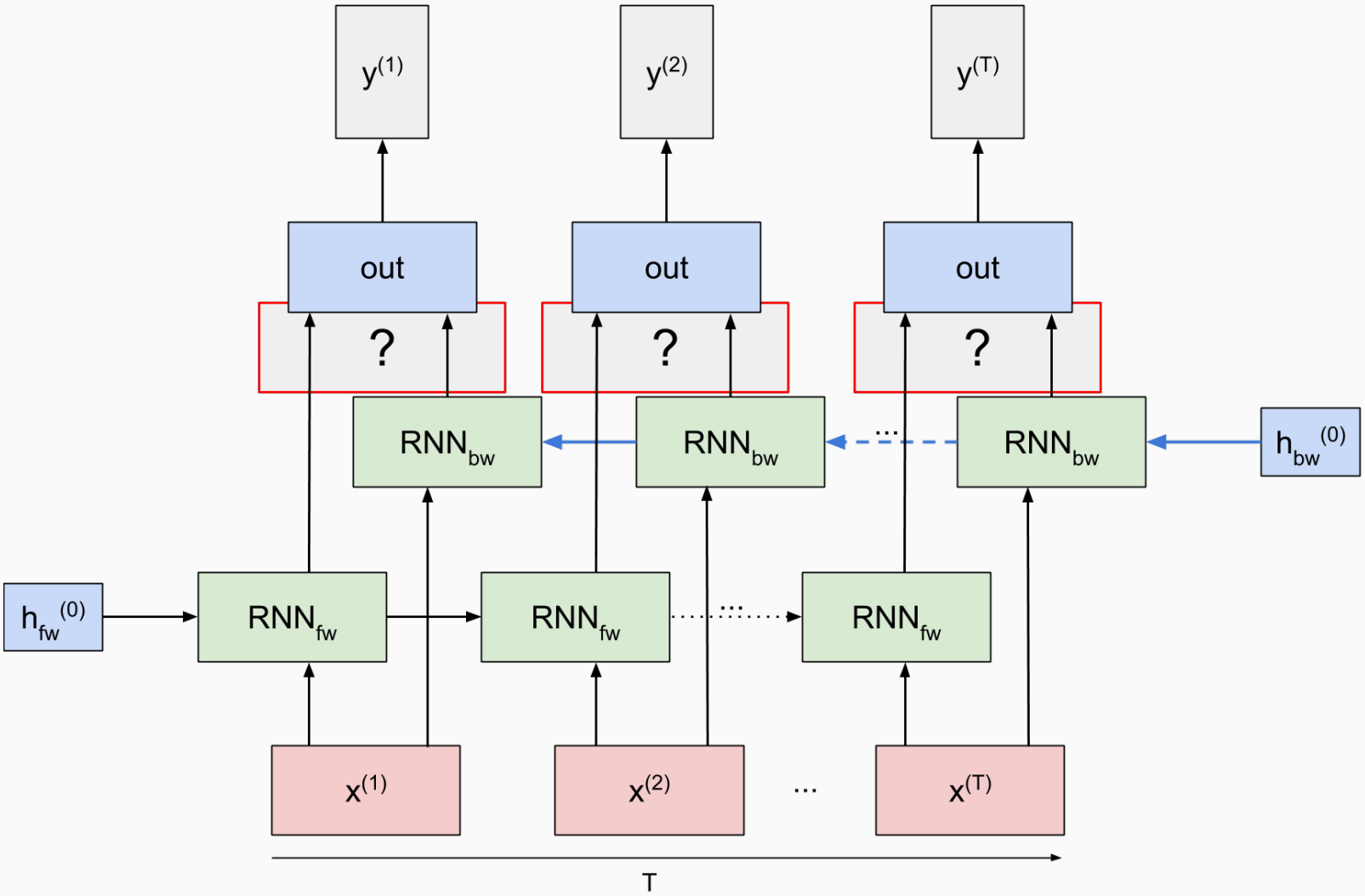
Dvosmjerni povratni modeli

Dodajemo **nezavisan** povratni model ($\overleftarrow{RNN}$) koji gleda u **suprotnom** smjeru s obzirom na originalni model ($\overrightarrow{RNN}$)



Dvosmjerni povratni modeli: agregiranje stanja

Kako agregirati izlaze povratnih ćelija prije predikcije?



Dvosmjerni povratni modeli: detalji

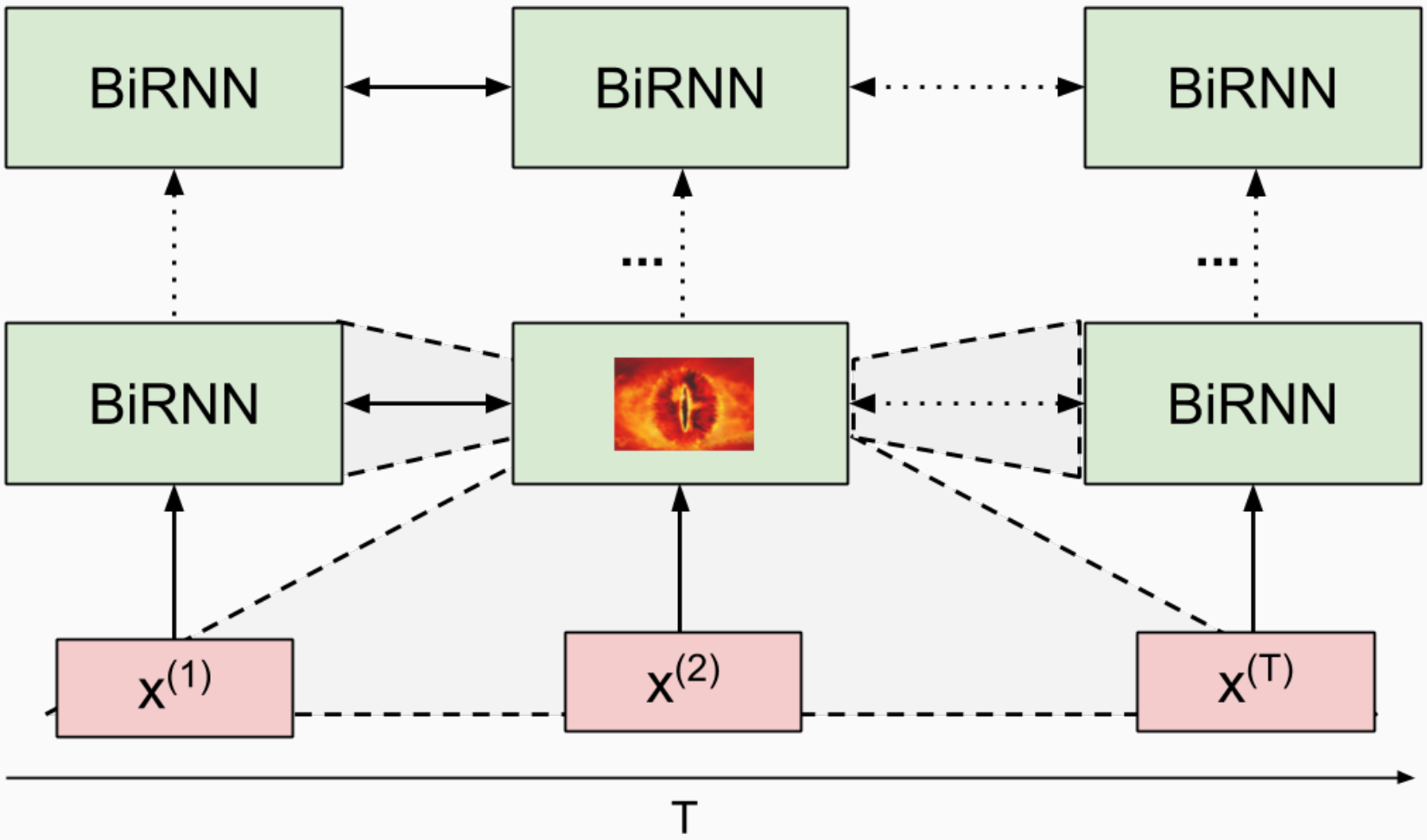
Dvosmjerni povratni model (BiRNN) sastoji se od dvija odvojena povratna modela koji funkcioniraju u suprotnim smjerovima:

- $\overrightarrow{RNN}$ čita s lijeva na desno
- $\overleftarrow{RNN}$ čita s desna na lijevo

Kako agregirati skrivena stanja?

1. konkateniranjem:
$$h^{(t)} = [\overrightarrow{h}^{(t)}, \overleftarrow{h}^{(t)}]$$
   - ovo udvostručuje dimenzionalnost sljedećeg sloja
   - podrazumijevani izbor u postojećim okvirima
2. usrednjavanjem
3. proizvoljnom (parametriziranom) funkcijom

Dvosmjerni povratni modeli: receptivno polje



Dvosmjerni povratni modeli: sažetak

Dvosmjerni modeli sastoje se od dva povratna modela koji napreduju u suprotnim smjerovima:

- konkateniranje stanja sljedećem sloju omogućava pregled svih ulaza

Konkatenacija povećava dimenzionalnost sljedećih slojeva:

- podrazumijevano ponašanje
- alternative: usrednjavanje, sažmanje + projekcija, ...

Važno je razmotriti **dozvoljava** li zadatak pristup cjelokupnom ulazu (prognoziranje vs gusta predikcija).

Učenje povratnih modela

Nestajući i eksplodirajući gradijenti

Gradijenti povratnih modela podložni numeričkoj nestabilnosti:

- uzrokovani dijeljenjem parametara u uzastopnim operacijama
- preciznije: uzastopno množenje s W_{hh}

Podsjetnik:

$$h^{(t)} = tanh(W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h)$$

Prvo razmatramo *skalarni* kontekst:

$$h^{(t)} = tanh(\underbrace{w_{hh}h^{(t-1)} + w_{xh}x^{(t)} + b_h}_{a^{(t)}})$$

- Vrijedi: $w_{hh}, w_{xh}, b_h, h, x \in \mathbb{R}$

Gradijenti u skalarnom slučaju

$$h^{(t)} = tanh(\underbrace{w_{hh}h^{(t-1)} + w_{xh}x^{(t)} + b_h}_{a^{(t)}})$$

Razmatramo gradijent između susjednih stanja:

$$\begin{aligned}
\frac{\partial h^{(t)}}{\partial h^{(t-1)}} &= \frac{\partial h^{(t)}}{\partial a^{(t)}} \frac{\partial a^{(t)}}{\partial h^{(t-1)}} \\
&= \frac{\partial th(a^{(t)})}{\partial a^{(t)}} w_{hh} && \searrow\ tanh = th \\
&= (1 - th^2(a^{(t)}))w_{hh} && \searrow\ \frac{dth(x)}{dx} = 1 - th^2(x)
\end{aligned}$$

Derivacija hiperbolnog tangensa ograničena na jedinični interval:

$$tanh(x) \in (-1, 1)$$

$$\frac{\partial tanh(x)}{x} = (1 - tanh^2(x)) \in (0, 1)$$

Gradijenti u skalarnom slučaju (2)

$$\frac{\partial h^{(t)}}{\partial h^{(t-1)}} = (1 - th^2(a^{(t)}))w_{hh}$$

Primijenimo supstituciju:

$$\gamma_t = \partial tanh(x)/\partial x\big|_{a^{(t)}} < 1$$

Slično: $\gamma_{\sigma t} = \partial \sigma(x)/\partial x\big|_{a^{(t)}} < 1/4$

$$\left.\begin{aligned} \frac{\partial h^{(t)}}{\partial h^{(t-1)}} &= \gamma_t w_{hh} \\ \frac{\partial h^{(T)}}{\partial h^{(t_0)}} &= \prod_{t_0}^{T} \gamma_t w_{hh} \end{aligned}\right\} t \to T$$

$$\left.\frac{\partial h^{(T)}}{\partial h^{(t_0)}} = (\overline{\gamma} w_{hh})^{T-t_0}\right\} \overline{\gamma}, w_{hh} \text{ ne ovisi o } t$$

Gradijenti u skalarnom slučaju (3)

$$\frac{\partial h^{(T)}}{\partial h^{(t_0)}} = (\overline{\gamma} w_{hh})^{T-t_0}$$

Kod dugih slijedova imamo: $T - t_0 \gg 0$

- numerička stabilnost gradijenta ovisi o $\overline{\gamma} w_{hh}$:

$$(\overline{\gamma} w_{hh})^{T-t_0} \to \begin{cases} \infty & \text{if } \overline{\gamma} w_{hh} > 1 \quad \text{(eksplodira)} \\ 0 & \text{if } \overline{\gamma} w_{hh} < 1 \quad \text{(nestaje)} \\ 1 & \text{if } \overline{\gamma} w_{hh} \approx 1 \quad \text{(stabilan)} \end{cases}$$

Ako pretpostavimo $\overline{\gamma} = 1$, tada gore navedeni uvjet primijenjujemo na **parametar** w_{hh}.

Nastavljamo s analizom u kontekstu vektorskog skrivenog stanja.

Gradijenti u vektorskom slučaju: spektralna norma

Razmatramo svojstva spektralne norme kvadratne matrice A:

- norma produkta manja je ili jednaka produktu normi (vrijedi za sve matrične norme):

$$\|AB\| \leq \|A\| \, \|B\|$$

- spektralna norma odgovara najvećoj singularnoj vrijednosti
  - ili, ekvivalentno, korijenu najveće svojstvene vrijednosti $A^\top A$
- spektralna norma inducirana je L2-normom:

$$\|Ax\| \leq \|A\| \, \|x\|$$

Gradijenti u vektorskom slučaju: jedan korak

Ažuriranje skrivenog stanja (podsjetnik):

$$h^{(t)} = tanh(\underbrace{W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h}_{a^{(t)}})$$

Gledamo gradijent između dva uzastopna stanja:

$$\frac{\partial h^{(t)}}{\partial h^{(t-1)}} = \frac{\partial h^{(t)}}{\partial a^{(t)}} W_{hh}$$

Postavljamo gornju ogradu gradijenta:

$$\left\| \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \right\| \leq \left\| \frac{\partial h^{(t)}}{\partial a^{(t)}} \right\| \|W_{hh}\| \leq \gamma_{\max} \lambda_1$$

- λ_1 ... najveća singularna vrijednost W_{hh}
- $\gamma_{\max} = \max(\frac{\partial tanh(a^{(t)})}{\partial a^{(t)}})$ ... gornja ograda gradijenta aktivacije

Gradijenti u vektorskom slučaju: svi koraci

Razmatamo prethodnu jednadžbu kroz vrijeme:

$$\frac{\partial h^{(T)}}{\partial h^{(t_0)}} \leq (\gamma_{\max} \lambda_1)^{T-t_0}$$

Za dugačke slijedove ($T - t_0 \gg 0$) numerička stabilnost radijenta ovisi o $\gamma_{\max} \lambda_1$:

$$(\gamma_{\max} \lambda_1)^{T-t_0} \to \begin{cases} \infty & \text{if } \gamma_{\max} \lambda_1 > 1 \text{ (eksplodira)} \\ 0 & \text{if } \gamma_{\max} \lambda_1 < 1 \text{ (nestaje)} \\ 1 & \text{if } \gamma_{\max} \lambda_1 \approx 1 \text{ (stabilan)} \end{cases}$$

- Matrica W_{hh} mora zadovoljiti **stroge uvjete** ako želimo stabilnu optimizaciju
- za detaljniju analizu preporučamo pogledati [pascanu13icml]:
  Razvan Pascanu, Tomás Mikolov, Yoshua Bengio: On the difficulty of training recurrent neural networks. ICML 2013.

Učenje dugih veza

Povratni modeli konzistentno podbacuju na dugim slijedovima:

- problem nastaje zbog numeričke nestabilnosti gradijenata uslijed uzastopnog množenja s W_{hh}:

  [bengio94tnn] Y Bengio, PY Simard, P Frasconi: Learning long-term dependencies with gradient descent is difficult, IEEE TNN 1994.

Simptome možemo ublažiti:

- osiguravanjem umjerenih singularnih vrijednosti povratne veze

  M Arjovsky, A Shah, Y Bengio: Unitary Evolution Recurrent Neural Networks. ICML 2016
- izostavljanjem matričnog množenja iz povratne veze.

Rješenje: *razdvojiti* zadatke povratne veze

- W_{hh} i W_{xh} sprežu filtriranje inforacija, pamćenje ulaza i projekciju novih elemenata u skriveno stanje
- skriveno stanje h spreže projekciju izlaza i pamćenje informacija za buduće izlaze.

Povratna ćelija s dugoročnom memorijom (LSTM: Long short-term memory)

LSTM: notacija

Sada skriveno stanje $h^{(t)}$ koristimo **samo za računanje izlaza**.

Uvodimo **stanje ćelije** $c^{(t)}$ koji **samo pamti viđenu informaciju**.

Uvodimo **doprinos stanju ćelije** $\hat{c}^{(t)}$ s obzirom na ulaz u trenutku t.

Uvodimo logičke vektore $f^{(t)}$ i $i^{(t)}$:

- $f^{(t)}$ nazivamo propusnicom (vratima) *zaboravljanja*
- $i^{(t)}$ nazivamo propusnicom (vratima) *ulaza*

Izbacujemo matrično množenje iz povratne jednadžbe stanja ćelije:

$$c^{(t)} = \quad c^{(t-1)} + \quad \hat{c}^{(t)}$$

$$\frac{\partial c^{(t)}}{\partial c^{(t-1)}} = \mathbb{I}$$

[hochreiter96nips] LSTM can solve hard long time lag problems

LSTM: jednadžbe

$$c^{(t)} = \quad c^{(t-1)} + \quad \hat{c}^{(t)}$$

LSTM ćelija **zaboravlja** dio informacije iz prethodnog stanja:

$$f^{(t)} = \sigma(W_{fhh} h^{(t-1)} + W_{fxh} x^{(t)} + b_{fh}) = \sigma(a_f^{(t)})$$

- svaka vrata imaju svoj vlastiti skup parametara W_{hh}, W_{xh}, b_h.

LSTM ćelija propušta **samo podskup** ulaza:

$$i^{(t)} = \sigma(W_{ihh} h^{(t-1)} + W_{ixh} x^{(t)} + b_{ih}) = \sigma(a_i^{(t)})$$

- povratni put stanja ćelije upotpunjavamo vratima f i i:

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$

LSTM: intuicija

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$

Hadamardov produkt ($\odot$): tenzorsko množenje po elementima

$$a \odot b = \begin{pmatrix} a_0 b_0 \\ \dots \\ a_i b_i \end{pmatrix}$$

Svrha vrata: filtriranje informacije ($\sigma : \mathbb{R} \to (0, 1)$).

Sigmoidna funkcija može se probabilistički interpretirati kao **dio informacije koji želimo zadržati**.

Ograničavanje $f^{(t)}$ i $i^{(t)}$ na jedinični interval $(0, 1)$ *eliminira* eksplodirajuće gradijente

- u teoriji, kodomena sigmoide je otvorena ($\sigma(x) < 1 \quad \forall x$)
- u praksi imamo podljev izraza exp(-x) zbog konačne preciznosti

LSTM: detalji

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$

Izraz za računanje doprinosa stanju ćelije $\hat{c}$:

$$\hat{c}^{(t)} = tanh(W_{chh} h^{(t-1)} + W_{cxh} x^{(t)} + b_{ch}) = tanh(a_c^{(t)})$$

- određujemo ga kao afinu transformaciju **skrivenog stanja** i ulaza.

Naša notacija malo je **različita** nego u knjizi

- u knjizi: $s^{(t)} := c^{(t)}; g^{(t)} := i^{(t)}; q^{(t)} := o^{(t)}$
- umjesto estetske supstitucije $\hat{c}^{(t)}$, knjiga ima razmotani izraz:

$$s^{(t)} = f^{(t)} s^{(t-1)} + g^{(t)} \left(tanh(W h^{(t-1)} + U x^{(t)} + b_s) \right)$$

LSTM: skriveno stanje

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$

Skriveno stanje računamo kao funkciju stanja ćelije:

$$h^{(t)} = o^{(t)} \odot tanh(c^{(t)})$$

Logički vektor $o^{(t)}$ nazivamo *izlaznim vratima*:

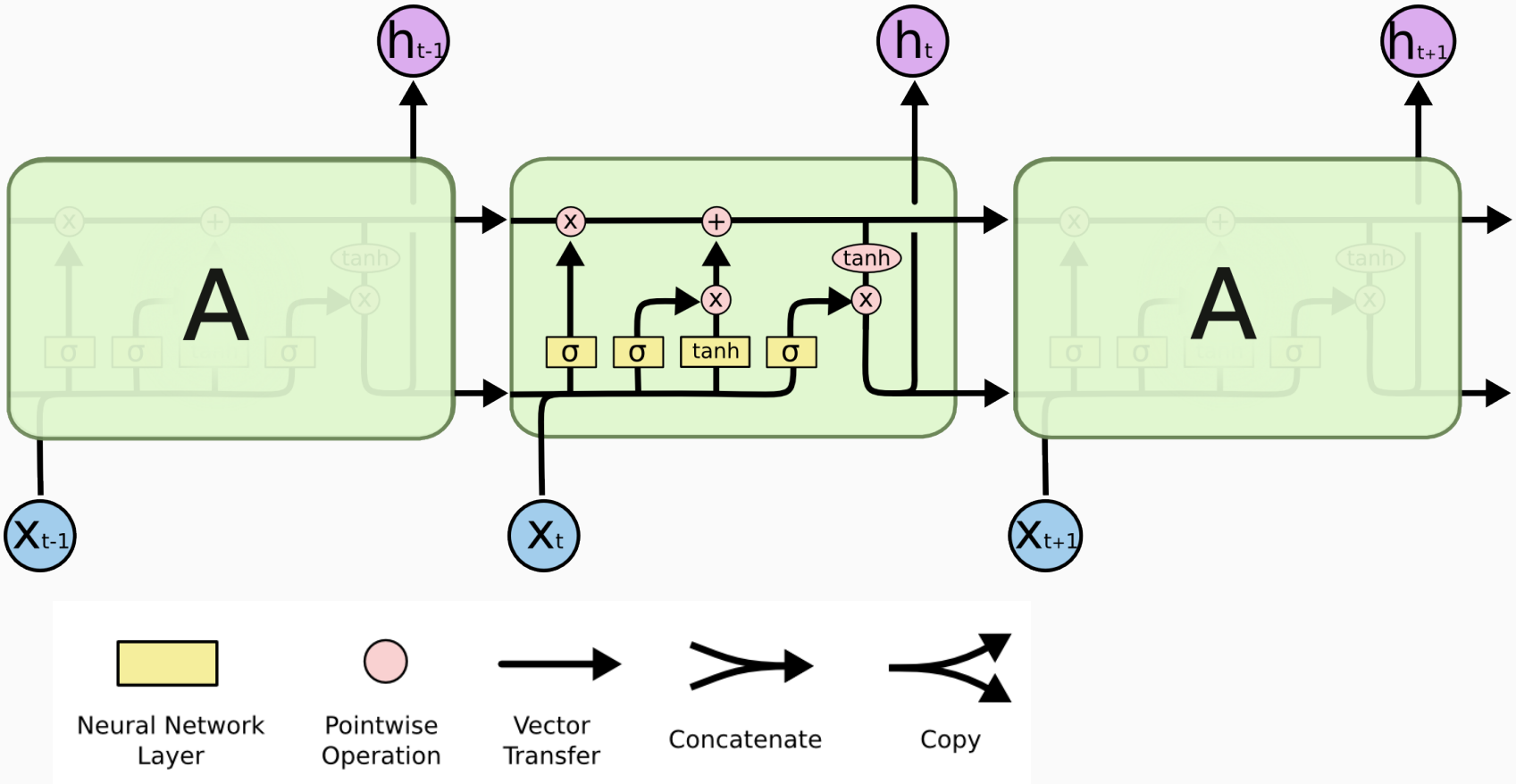
$$o^{(t)} = \sigma(W_{ohh} h^{(t-1)} + W_{oxh} x^{(t)} + b_{oh}) = \sigma(a_o^{(t)})$$

Sažetak:

- Razdvojili smo stanje ćelije ("memoriju") $c^{(t)}$ od skrivenog stanja $h^{(t)}$ iz kojeg se računa izlaz
- arhitektura ne dopušta lako mijenjanje stanja ćelije
- imamo **4**$\times$ više parametara

LSTM: vizualizacija

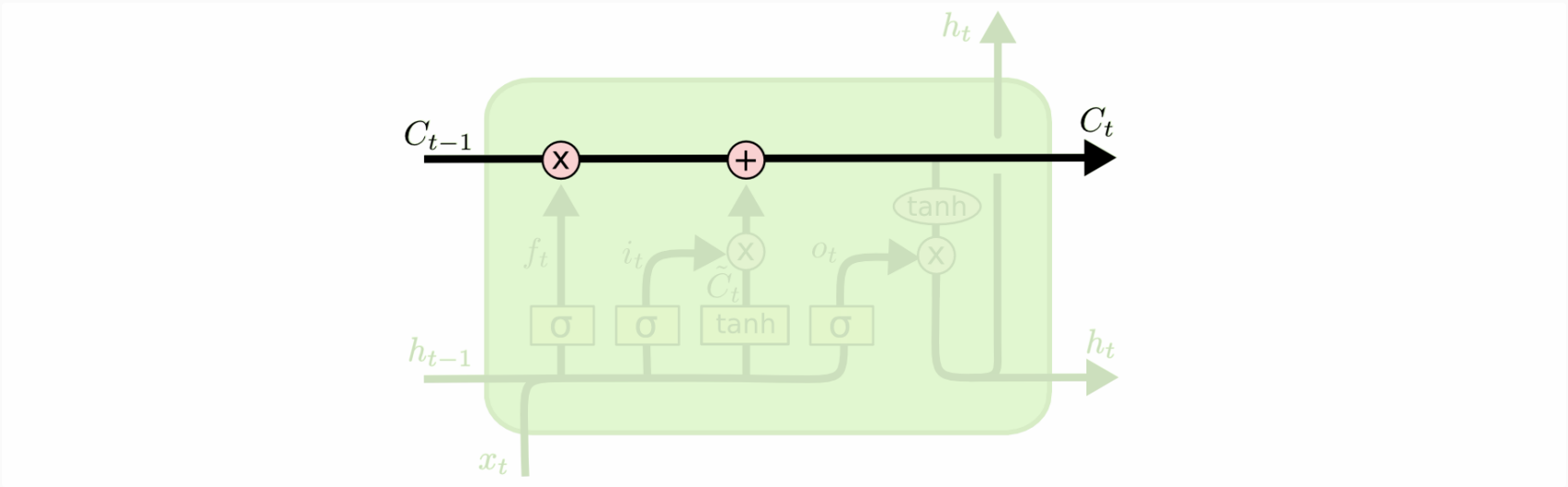
Prikazujemo nekoliko slika s [bloga] Cristophera Olaha



Pitanje: koji je redoslijed vrata na skici?

LSTM: vizualizacija stanja ćelije

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$

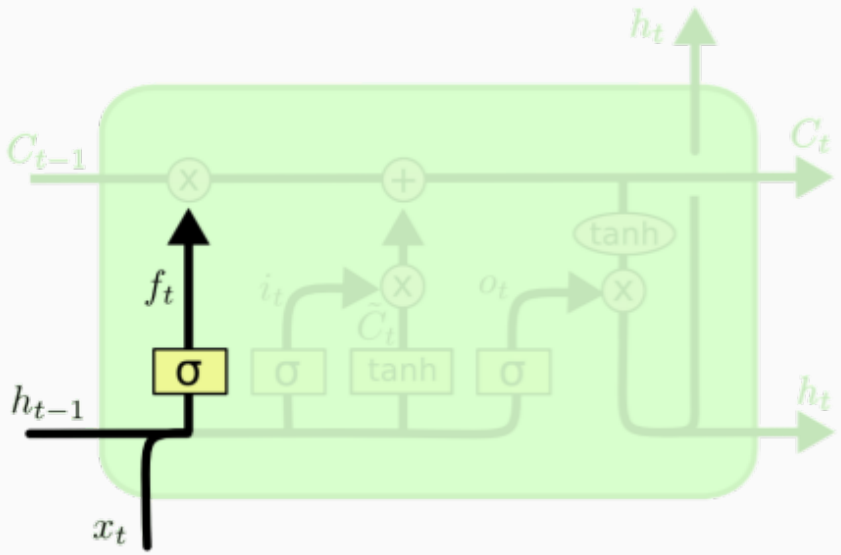


Stanje ćelije mijenjamo množenjem po elementima i zbrajanjem - informacijski tok je jednostavan

- Stanje ćelije je **teško** izmijeniti: svaku promjenu moraju *podržati* dvoja vrata.

LSTM: vizualizacija vrata zaboravljanja

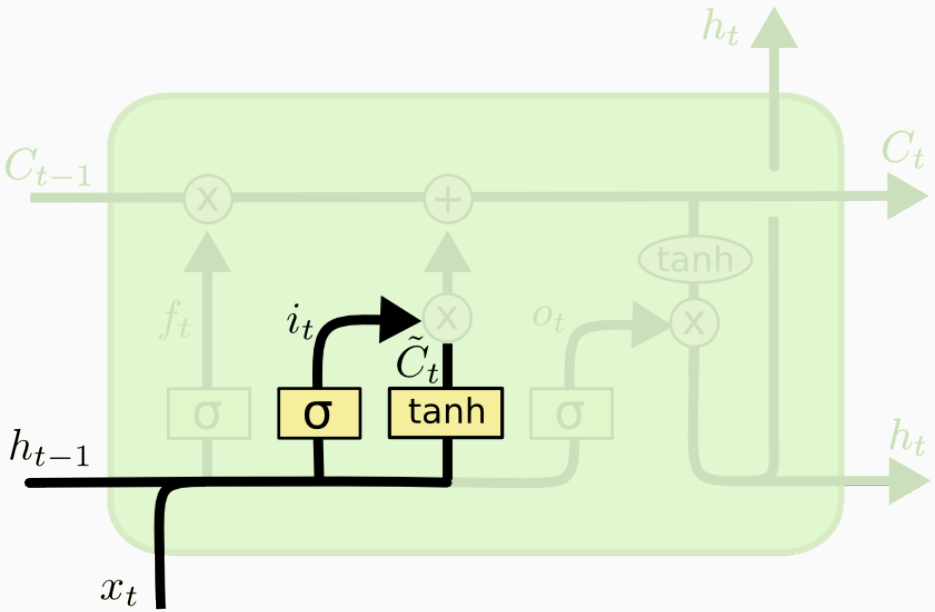
$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$



$$f^{(t)} = \sigma(W_{fhh} h^{(t-1)} + W_{fxh} x^{(t)} + b_{fh})$$

LSTM: vizualizacija ulaznih vrata

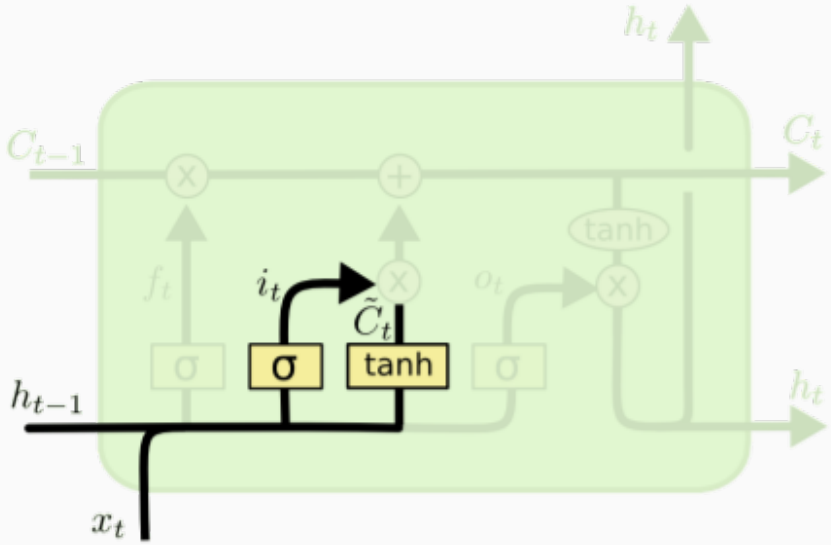
$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + \boxed{i^{(t)} \odot \hat{c}^{(t)}}$$



$$i^{(t)} = \sigma(W_{ihh} h^{(t-1)} + W_{ixh} x^{(t)} + b_{ih})$$

$$\hat{c}^{(t)} = tanh(W_{chh} h^{(t-1)} + W_{cxh} x^{(t)} + b_{ch})$$

LSTM: vizualizacija ulaznih vrata (2)



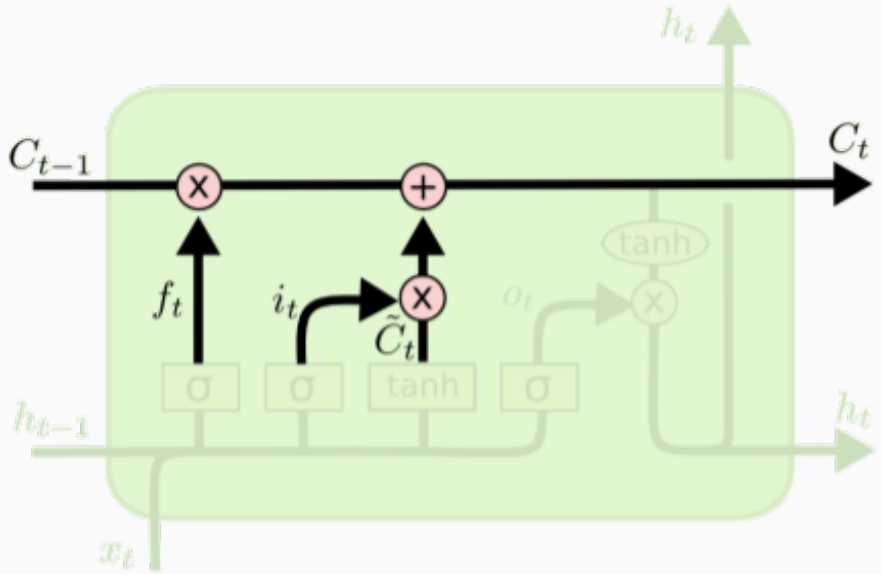
$$\hat{c}^{(t)} = tanh(W_{chh}h^{(t-1)} + W_{cxh}x^{(t)} + b_{ch})$$

Prema literaturi, doprinos stanju $\hat{c}^{(t)}$ može biti aktiviran sigmoidom ili hiperbolnim tangensom

- PyTorch i Tensorflow koriste *tanh*

LSTM: vizualizacija ažuriranja stanja

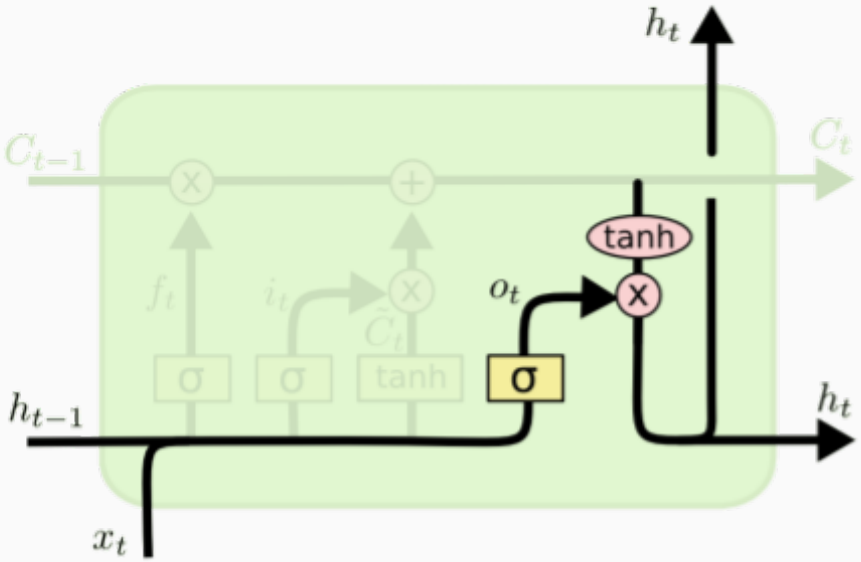
$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$



LSTM: vizualizacija izlaznih vrata

$$h^{(t)} = o^{(t)} \odot tanh(c^{(t)})$$

$$o^{(t)} = \sigma(W_{ohh} h^{(t-1)} + W_{oxh} x^{(t)} + b_{oh})$$



LSTM: sažetak

Obične povratne modele nije lako naučiti

- često susrećemo eksplodirajuće i nestajuće gradijente zbog uzastopnog množenja s W_{hh}
- skriveno stanje mora pamtiti informacije i voziti izlaz
- zbog toga se ovi modeli loše ponašaju na dugim slijedovima.

Zbog toga smo uveli ćeliju s dugoročnim pamćenjem (LSTM)

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$

$$h^{(t)} = o^{(t)} \odot tanh(c^{(t)})$$

- izostavili smo matrično množenje iz unaprijednog i povratnog puta
- uveli smo troja vrata za filtriranje informacija
- raspregnuta **odgovornost** olakšava pritisak na stanje ćelije.

LSTM: backprop

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i \odot \hat{c}^t$$

Razmatramo gradijent za povratnu vezu:

$$\frac{\partial c^{(t)}}{\partial c^{(t-1)}} = f^{(t)} = \sigma(a_f^{(t)}) \in (0,1)$$

Pravilo ulančavanja daje:

$$\frac{\partial c^{(T)}}{\partial c^{(t_0)}} = \prod_{t=t_0}^{T} f^{(t)} \leq 1$$

Čini se da ova jednadžba ne dozvoljava **eksplodirajuće** gradijente!

- *Je li uistinu tako?*
- LSTM ćelija ima *dvojno* skriveno stanje ($c^{(t)}$, $h^{(t)}$)

LSTM: backprop (2)

$$h^{(t)} = o^{(t)} \odot tanh(c^{(t)})$$

Pogledajmo izlazna vrata:

$$o^{(t)} = \sigma(a_o^{(t)}) = \sigma(W_{ohh} h^{(t-1)} + W_{oxh} x^{(t)} + b_{oh})$$

Primjećujemo sličan oblik kao u običnoj povratnoj ćeliji:

$$h^{(t)} = \sigma(W_{ohh} h^{(t-1)} + W_{oxh} x^{(t)} + b_{oh}) \odot tanh(c^{(t)})$$

$$\frac{\partial h^{(t)}}{\partial h^{(t-1)}} = \frac{\partial h^{(t)}}{\partial a_o^{(t)}} \frac{\partial a_o^{(t)}}{\partial h^{(t-1)}} = \frac{\partial h^{(t)}}{\partial a_o^{(t)}} W_{ohh} = \dots$$

Zbog toga, eksplodirajući gradijent je **ipak moguć** pri unatražnom prolazu kroz $h^{(t)}$

- međutim, to se rijetko događa u praksi

LSTM varijante: ćelija sa špijunkom

Uključiti stanje ćelije $c^{(t-1)}$ u jednadžbu za računanje vrata:

$$f^{(t)} = \sigma(\underbrace{W_{fch}c^{(t-1)} + W_{fhh}h^{(t-1)} + W_{fxh}x^{(t)} + b_{fh}}_{a_f^{*(t)}})$$

Ova ideja može se primijeniti na sva vrata.

- **Prednost:** dodatna informacija može **pomoći** [gers00ijcnn]
- **Nedostatak:** veći broj parametara
- **Nedostatak:** još jedan put kroz koji može dovesti do eksplodirajućeg gradijenta.

LSTM varijante: spojena vrata (eng. fused gates)

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \hat{c}^{(t)}$$

Ideja: ako smo neku informaciju zaboravili, trebamo je zamijeniti

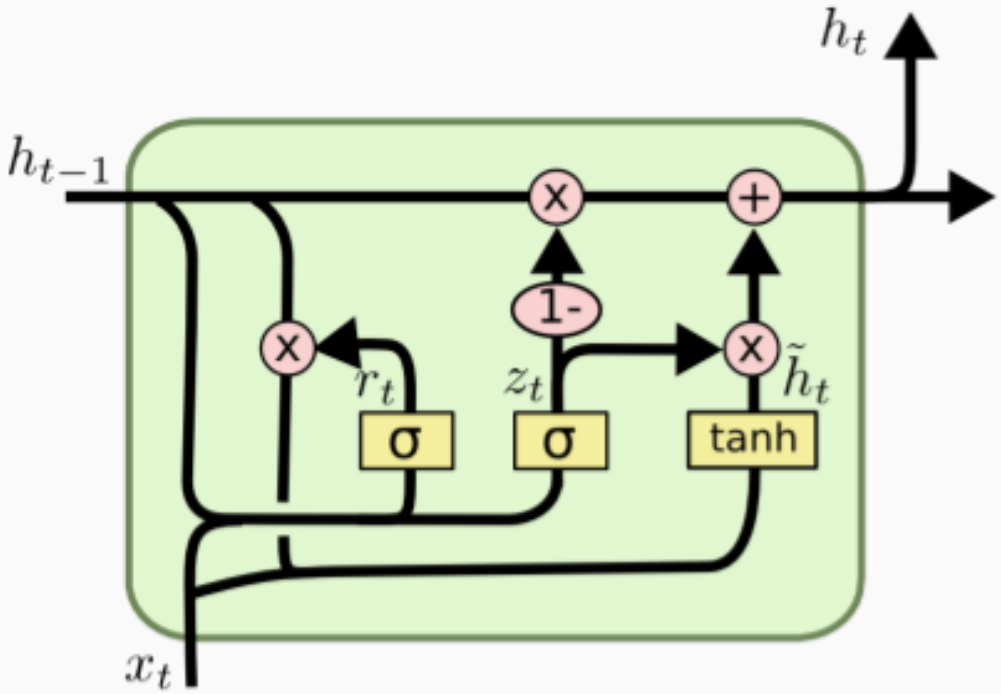
$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + (1 - f^{(t)}) \odot \hat{c}^{(t)}$$

- **Prednost:** 25% manje parametara
- **Nedostatak:** radi lošije od standardnog LSTM-a (više parametara pomaže)
- **Nedostatak:** smanjuje izražajnost modela (nemogućnost akumulacije)

LSTM varijante: propusna povratna ćelija

Propusna ćelija (eng. gated recurrent unit): pojednostavnjeni LSTM

- izvrsno radi iako ne izdvaja stanje ćelije iz latentnog stanja
- ovo sugerira da su naše intucije u vezi LSTM-ova nepotpune.



LSTM varijante: propusna povratna ćelija (2)

GRU ćelije imaju dvoja vrata: $r^{(t)}$ and $u^{(t)}$:

- $u^{(t)}$ nazivamo vratima *ažuriranja* (eng. update gate)

$$u^{(t)} = \sigma\left(W_{uhh}h^{(t-1)} + W_{uxh}x^{(t)} + b_{uh}\right) \tag{4}$$

- $r^{(t)}$ nazivamo vratima *resetiranja* (eng. reset gate)

$$r^{(t)} = \sigma\left(W_{rhh}h^{(t-1)} + W_{rxh}x^{(t)} + b_{rh}\right) \tag{5}$$

Važan međurezultat: privremeno stanje $\hat{h}^{(t)}$

$$\hat{h}^{(t)} = \sigma\left(W_{hh}\left(r^{(t)} \odot h^{(t-1)}\right) + W_{xh}x^{(t)} + b_h\right) \tag{6}$$

Povratno stanje $h^{(t)}$ ponovo ima višestruke odgovornosti:

$$h^{(t)} = u^{(t)}h^{(t-1)} + (1 - u^{(t)})\hat{h}^{(t)} \tag{7}$$

LSTM varijante: sažetak

Eksplodirajući i nestajući gradijenti mogu se pojaviti i u LSTM-ovima

- međutim, oni se javljaju znatno rjeđe u praksi

Uspjeh LSTM-ova doveo je do razvoja više varijanti.

1. **LSTM sa špijunkom**:
   - stanje ćelije koristi se u jednadžbi ažuriranja
   - ima smisla jer LSTM-ovi ionako ne uspijevaju u potpunosti izbjeći numeričke probleme s gradijentom
2. **LSTM sa spojenim vratima**
   - spajanje vrata zaboravljanja s vratima ulaza dovodi do veće učinkovitosti i manjeg broja parametara
3. **Propusna povratna ćelija (GRU)**
   - spojena vrata i izmjenjena semantika stanja
   - slična generalizacijska moć kao i LSTM-ovi unatoč spregnutom stanju

Analiza: slijed-u-slijed

Strojno prevođenje

Želimo naučiti generirati prijevod danog ulaznog slijeda:

- model cilja izlazne slijedove **unknown length**.
- prediktiramo kategoričku distribuciju preko izlaznog rječnika u svakom izlaznom elementu.

Skupovi podataka: WMT, IWSLT (povremeno se ažuriraju):

- `https://www.statmt.org/wmt15/translation-task.html`
- `https://sites.google.com/site/iwsltevaluation2015/mt-track`

Primjer ulaza i izlaza za jedan primjer WMT-14 en-de skupa:

```
Parliament Does Not Support Amendment Freeing Tymoshenko
Keine befreiende Novelle für Tymoshenko durch das Parlament
```

Strojno prevođenje: pretpostavke

Ciljne varijable:

- slijed riječi ciljanog jezika: $\{keine, befreiende, Novelle, \ldots\}$
- ciljne varijable pretvaramo u indekse: $\{0, \ldots, V_{out}\}$
- biramo veličinu ciljnog vokabulara.

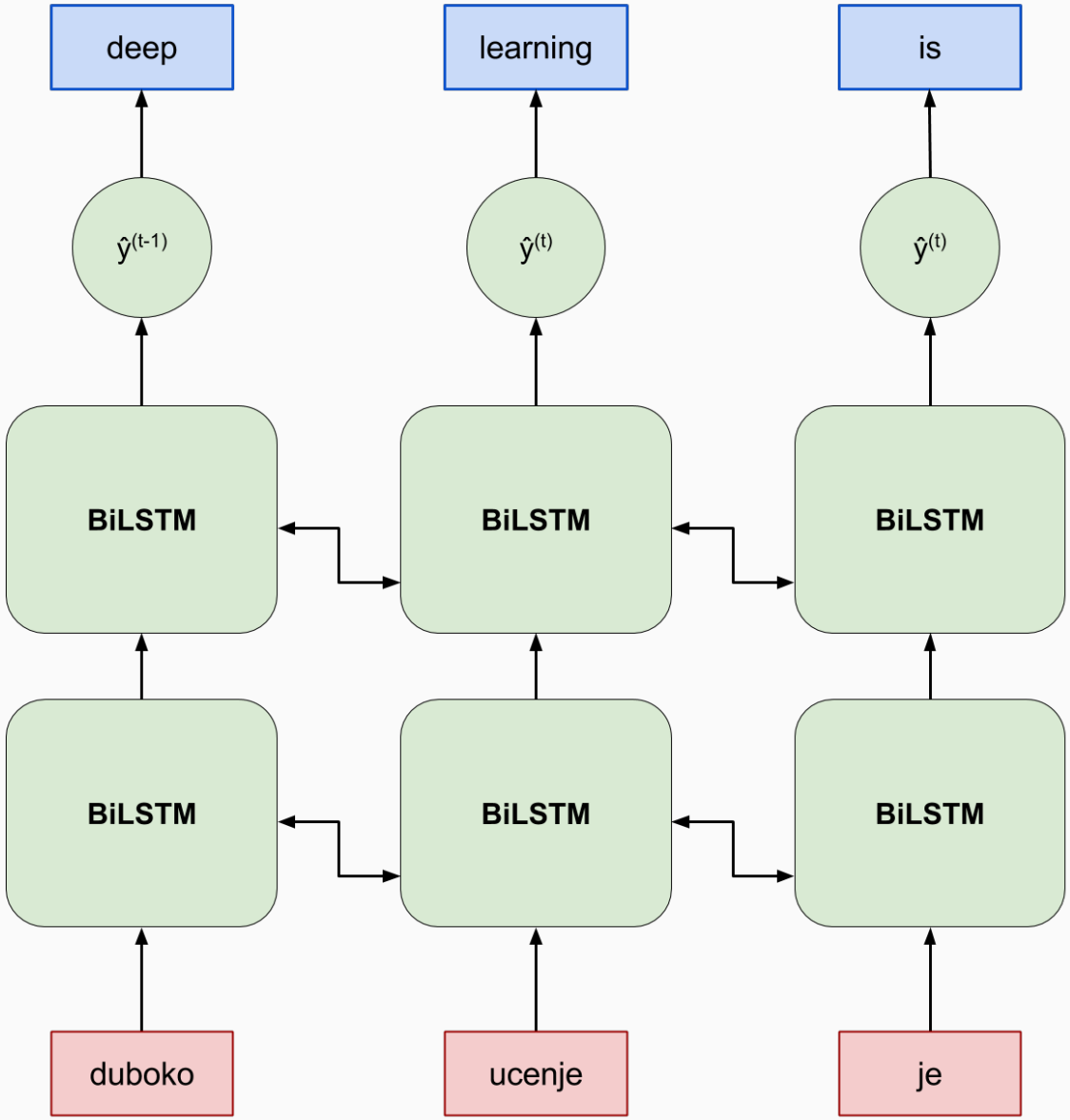
Ulazne varijable:

- biramo veličinu ulaznog vokabulara
- ulazne riječi pretvaramo u indekse koji odgovaraju indeksima u matrici ugrađivanja

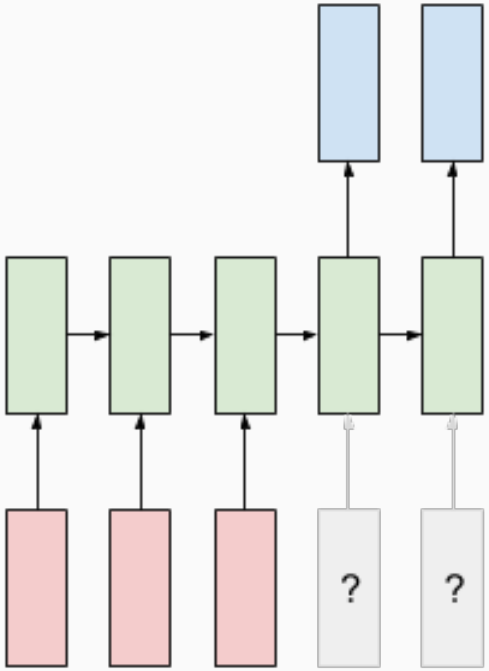
Ovaj pristup je sličan gustoj predikciji (npr. predikciji vrste riječi), ali:

1. možemo započeti predikciju tek nakon što smo vidjeli cijeli ulaz
2. ne znamo broj izlaznih simbola
3. nije jasno o kojim ulazima ovisi tekući izlaz

Strojno prevođenje: naivno rješenje koje ne funkcionira



Slijed u slijed: pretpostavke



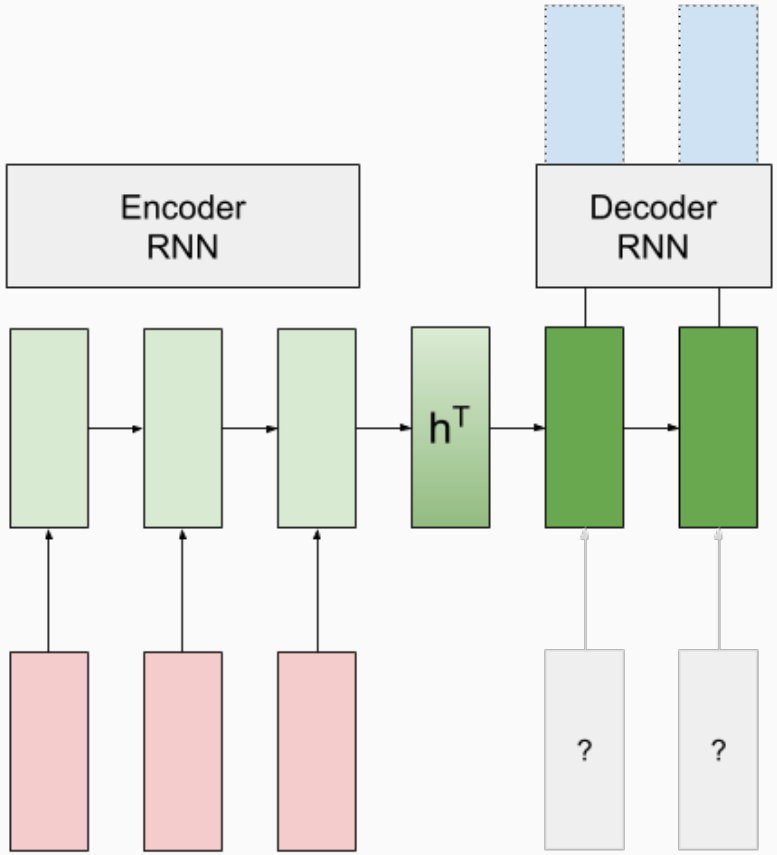
Započinjemo formalizacijom problema slijed-u-slijed (gore).

Nakon toga, prikazat ćemo neka konkretna rješenja.

Slijed u slijed: formalizacija

Predviđamo rješenje s dva modula:

1. **koder** ("čitač"): čita ulazni slijed i gradi skrivenu reprezentaciju izrečenog
2. **dekoder** ("pisač") generira prijevod na temelju skrivene reprezentacije



Slijed u slijed: formalizacija (2)

Posljednje stanje kodera određuje prvo stanje dekodera:

$$h_{dec}^{(0)} = f(h_{enc}^{(T)})$$

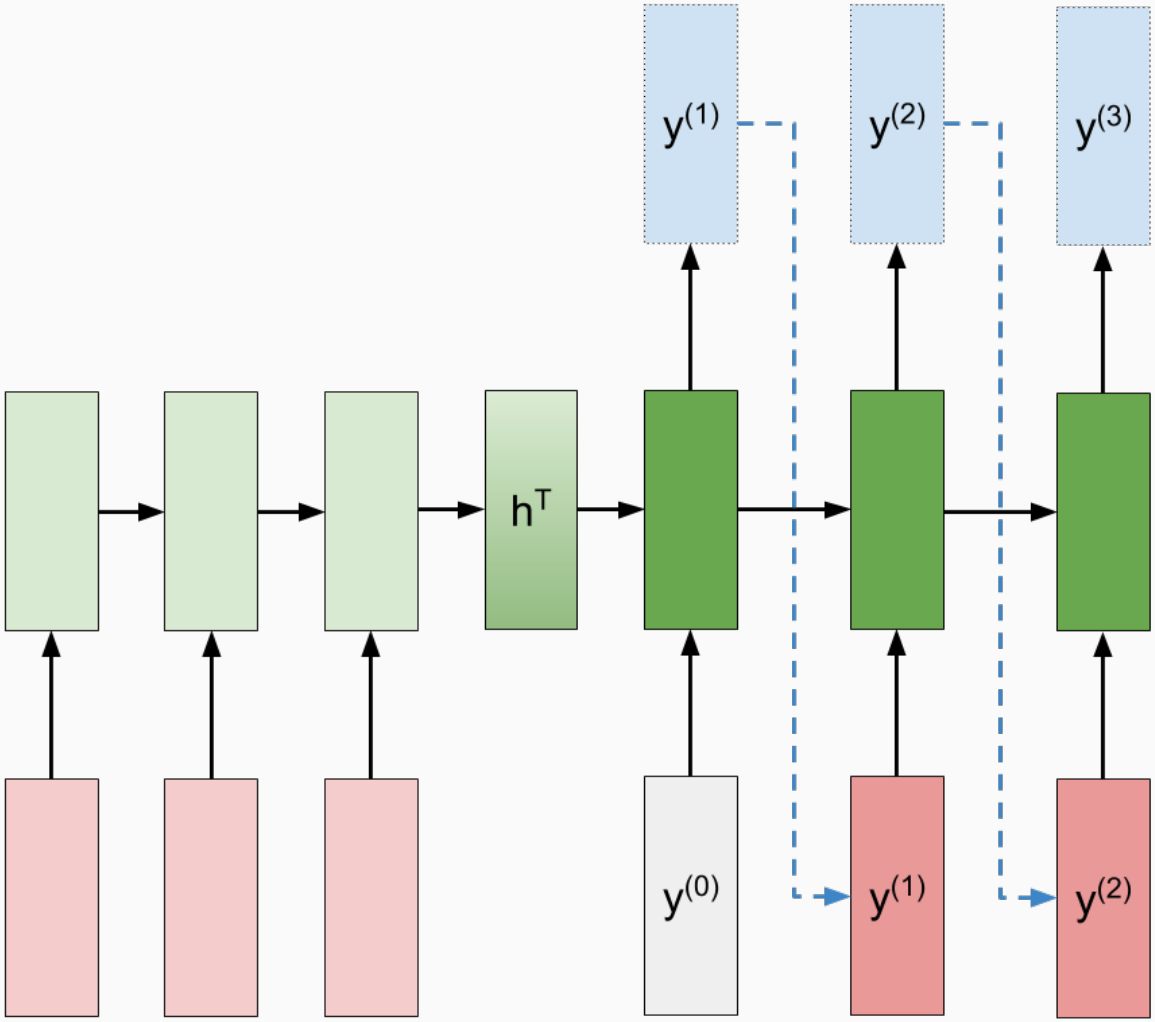
- u praksi često imamo: $h_{dec}^{(0)} = h_{enc}^{(T)}$
- f može biti bilo kakva parametrizirana glatka funkcija
- pitanje: kada bi to bilo **potrebno**?

Koder **ne prima** gubitak izravno

- gradijenti prvo moraju proći kroz cijeli **dekoder**

Problem: što ulazi u povratne ćelije dekodera?

Slijed u slijed: ulazi dekodera



Povratne ćelije dekodera primaju izlaze iz prethodnog koraka obrade.

Slijed u slijed: generiranje izlaza

Ulaz dekodera u trenutcima $t > 0$ sadrži najizgledniji izlaz iz prethodnog koraka:

- što se zbiva u koraku $t = 0$?
- na ulaz dekodera u $t = 0$ dovodimo *poseban simbol* koji označava početak slijeda (<sos>)

Kako znamo da je izlaz upotpunjen?

- model označava kraj prijevoda predkcijom *posebnog simbola* koji označava kraj slijeda (<eos>)
- simbol <eos> dodajemo na **kraj** svakog ciljnog niza
- generiranje prijevoda zaustavljamo kada dobijemo simbol <eos> ili kada duljina slijeda premaši maksimalnu duljinu

Slijed u slijed: generiranje izlaza (2)

Prevođenje slijeda nije lako naučiti, a posebno u ranim fazama optimizacije.

Zato često koristimo **forsiranje učitelja** (engl. teacher forcing) gdje ulaze dekodera postavljamo stohastički na:

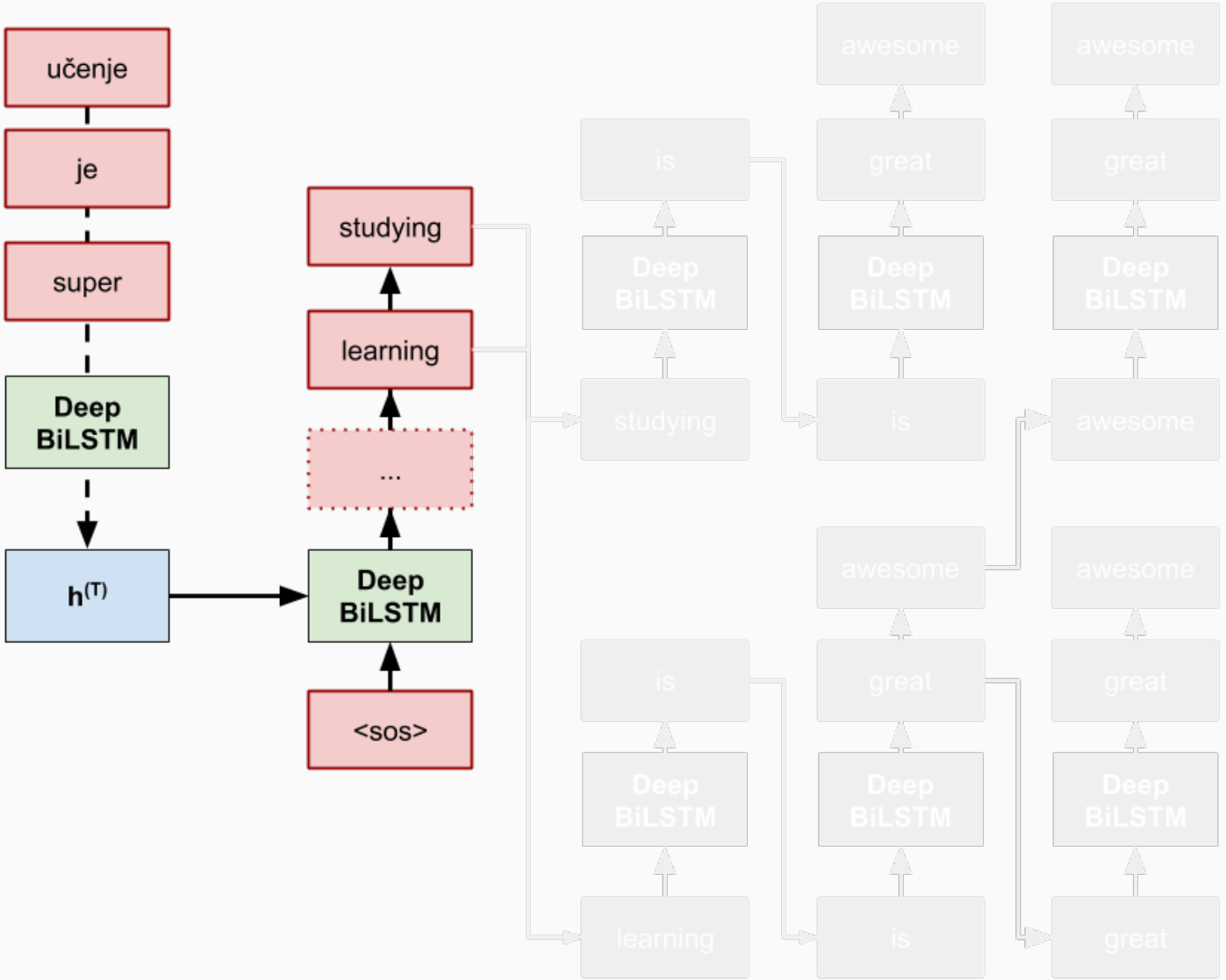
- **točne simbole** prethodnog koraka u p primjeraka za učenje
- **predikcije** prethodnog koraka u $1 - p$ primjera za učenje
- $p \in [0, 1]$ je hiper-parametar koji započinje s $p = 1$ i može se smanjiti kad učenje uznapreduje.

Slijed u slijed: zaključivanje

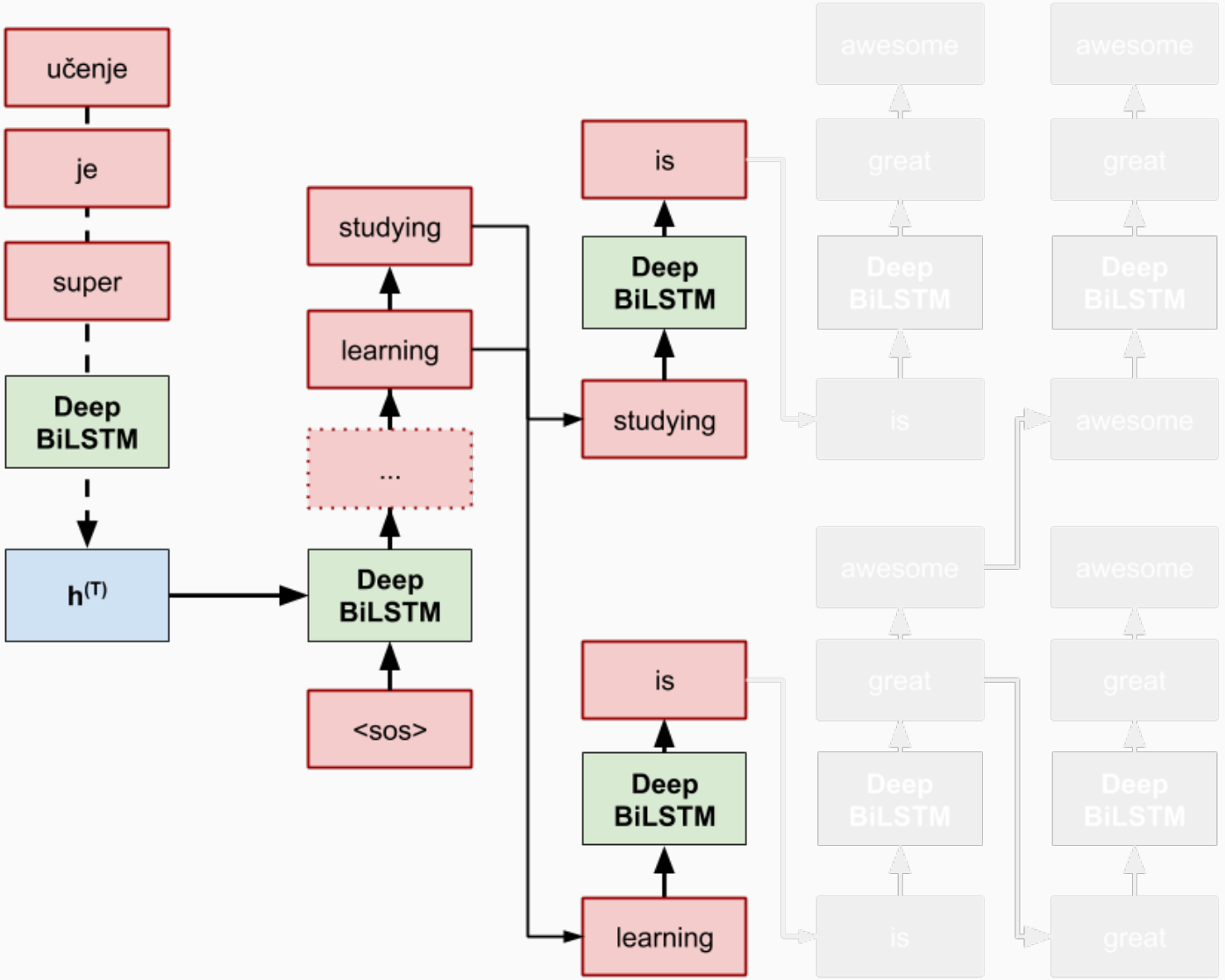
Pristupi za generiranje izlaza:

1. odabrati najvjerojatniju riječ u svakom koraku:
   - pohlepni pristup, može biti suboptimalan
   - moguće je da najbolji prijevod ne sadrži najvjerojatniju riječ u svakom koraku
   - nije dobro za uzorkovanje jer proizvodi **determinističke sljedove**
2. uzorkovanje s vjerojatnosnim težinama (roulette wheel selection):
   - uvodi slučajnost u postupak prevođenja i ohrabruje **raznolikost** izlaza
   - nije jasno je li nam prihvatljivo da model može izabrati lošu riječ s vjerojatnošću koja je **veća od nule** .
3. fokusirano uzorkovanje pretraživanjem zrakom:
   - razmatramo k najboljih prijevoda u svakom koraku
   - hiperparametar k označava širinu zrake

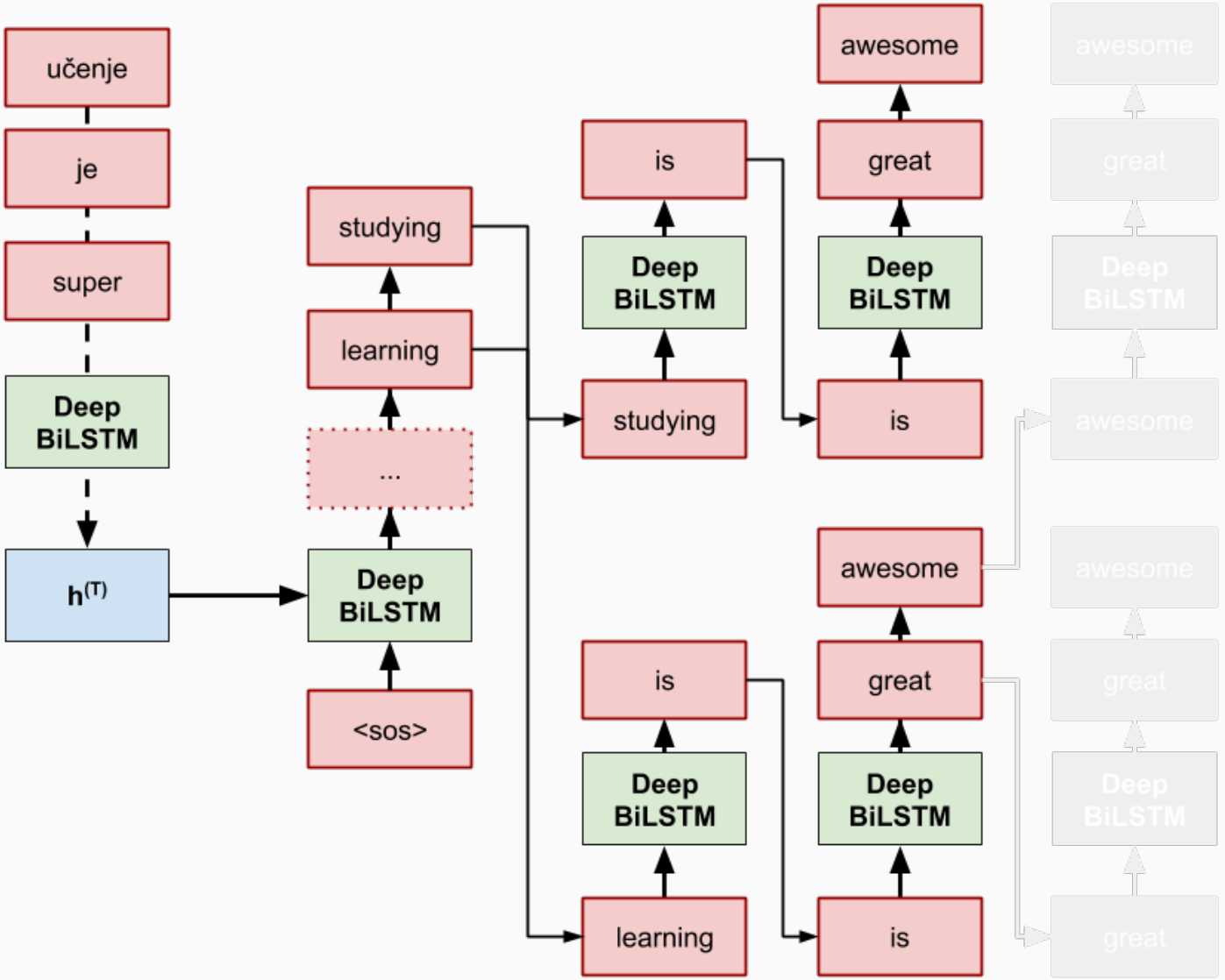
Slijed u slijed: pretraživanje zrakom



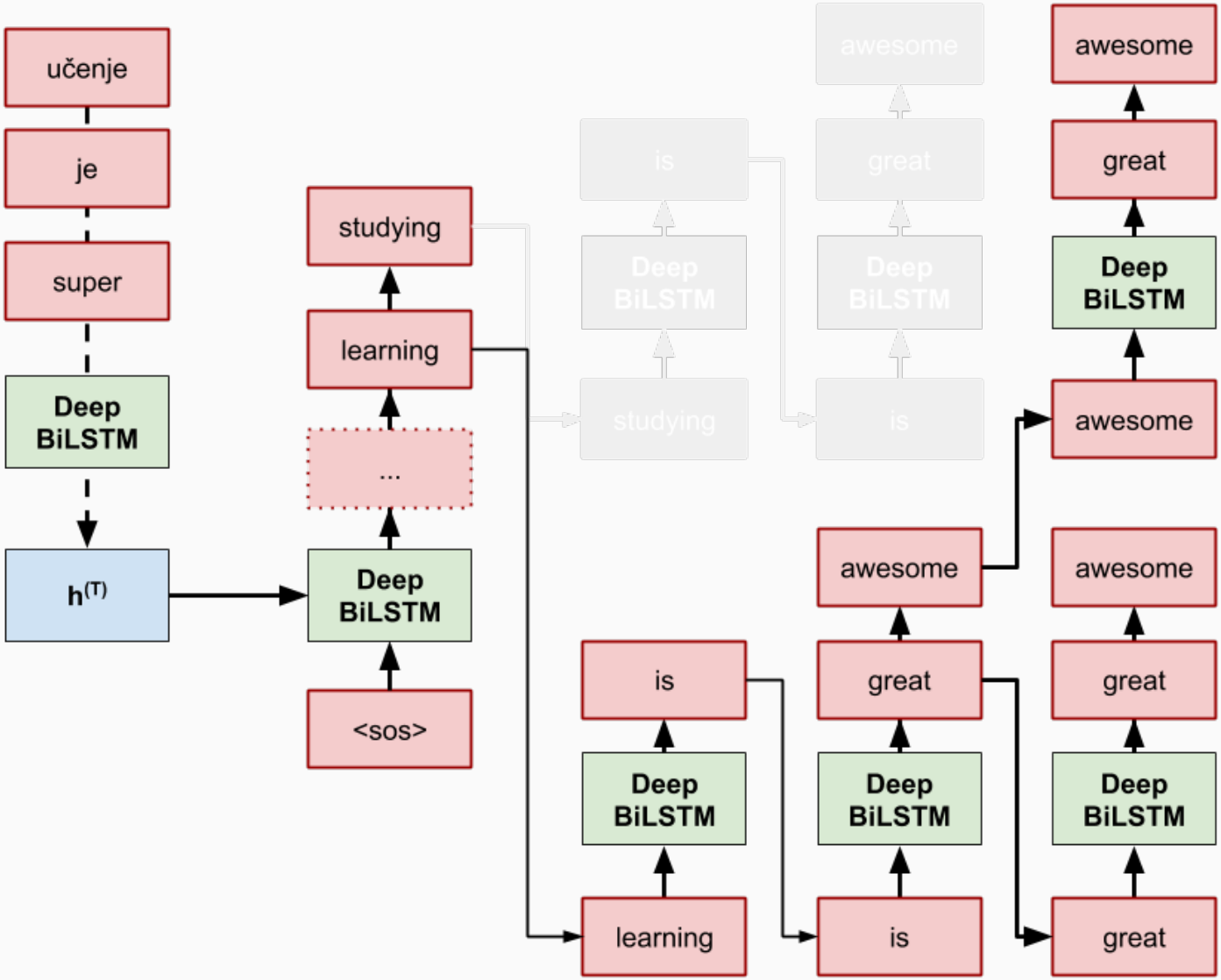
Slijed u slijed: pretraživanje zrakom (2)



Slijed u slijed: pretraživanje zrakom (3)



Slijed u slijed: pretraživanje zrakom (4)



Slijed u slijed: sažetak

Složenost prevođenja proizlazi iz varijabilne duljine ciljnog slijeda:

- umjesto "jednostavne" klasifikacije iz konteksta, model mora naučiti prediktirati cijele slijedove.

Ovom problemu pristupamo dekompozicijom na **i)** čitanje ulaznog slijeda i **ii)** generiranje izlaznog slijeda:

- koder i dekoder imaju odvojene parametre.

Slijed u slijed: sažetak (2)

Rana faza učenja je posebno problematična:

- može se olakšati **forsiranjem učitelja** ("učenjem prema šalabahteru") u nekom udjelu ulaznih primjera

Generiranje sljedova je teško:

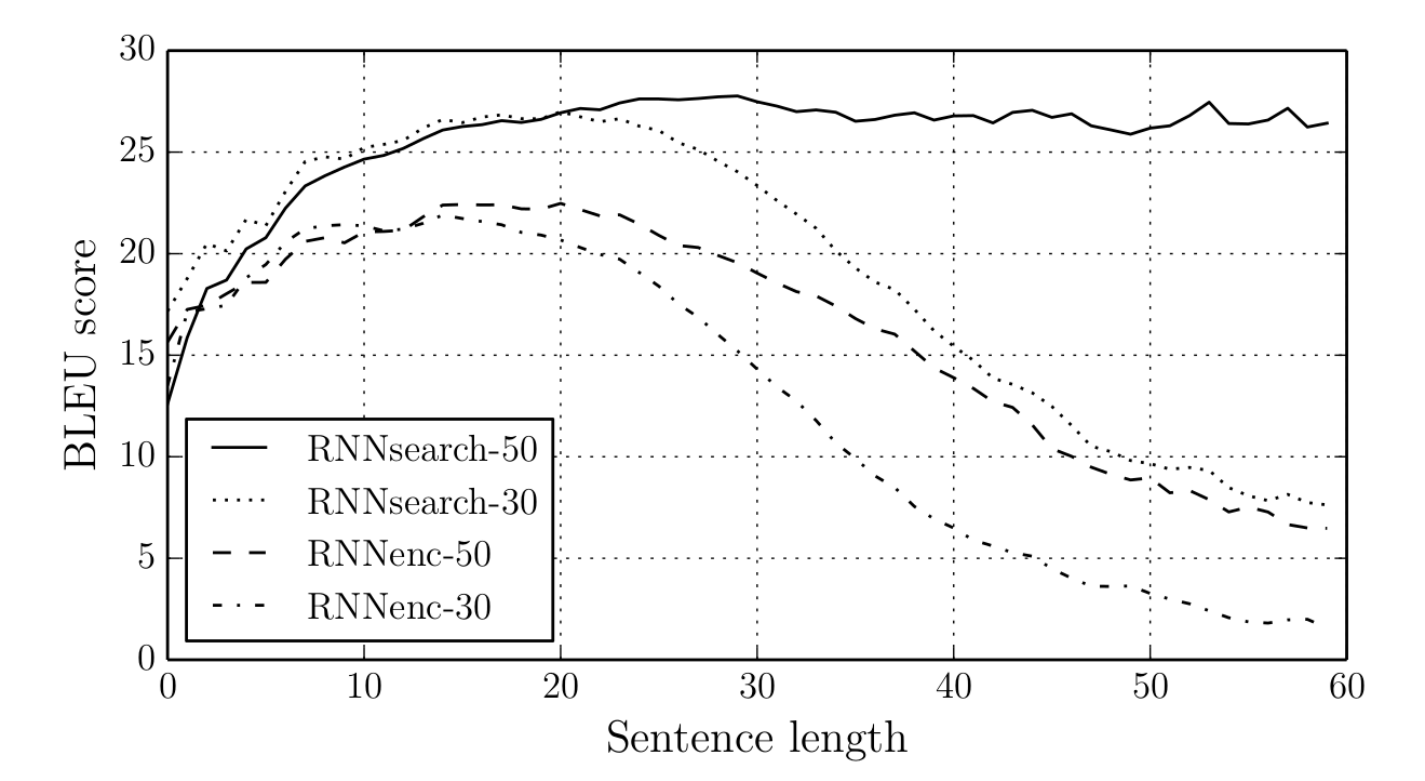
- želimo maksimizirati vjerojatnost **slijeda** umjesto vjerojatnost pojedinačnih dijelova
- tom problemu možemo pristupiti **pretraživanjem zrakom** koje prati k najvjerojatnijih slijedova u svakom koraku procesa generiranja prijevoda.

Pažnja (ili pozornost)

Pažnja

Uspješnost strojnog prevođenja za rečenice različite duljine:

- RNNsearch modeli [bahdanau14iclr] koriste pažnju.



Čak i najbolje povratne ćelije znatno lošije prevode duge rečenice:

- to sugerira da povratni modeli imaju slabo pamćenje.

Pažnja: ideja

Motivacija (https://distill.pub/2016/augmented-rnns/):

- *"When I'm translating a sentence, I pay special attention to the word I'm presently translating. When I'm transcribing an audio recording, I listen carefully to the segment I'm actively writing down. And if you ask me to describe the room I'm sitting in, I'll glance around at the objects I'm describing as I do so."*

Naše skrivene reprezentacije **nisu** savršene (ograničena veličina).

Ako ćelija ne može zapamtiti sve, može li barem zaključiti gdje se tražena informacija može naći?

- označimo tekuću reprezentaciju kao **upit**
- označimo prethodne reprezentacije kao **ključeve** (memoriju)
- pronađimo *sličnost* između upita i ključeva
- aggregirajmo prethodne reprezentacije **otežanim sažimanjem** gdje težine odgovaraju sličnosti.

Pažnja: osnovna formulacija

Funkcija attn vraća skalarnu sličnost između dva vektora:

$$a^{(t_{\text{dec}},t_{\text{enc}})} = \text{attn}(q^{(t_{\text{dec}})}, k^{(t_{\text{enc}})}), \qquad a \in \mathbb{R}, q \in \mathbb{R}^{d_q}, k \in \mathbb{R}^{d_k} \ .$$

- pri tome su $q^{(t_{\text{dec}})} = h_{dec}^{(t_{\text{dec}})}$ i $k^{(t_{\text{enc}})} = h_{enc}^{(t_{\text{enc}})}$ skrivena stanja modela

Treba nam sličnost između upita i *svih* ključeva:

$$a = \text{attn}(q, K), \qquad a \in \mathbb{R}^T, K = [k^{(1)}, \ldots, k^{(T)}] \ .$$

Mjeru sličnosti normaliziramo na vjerojatnosnu distribuciju:

$$\alpha = \text{softmax}(a) \ .$$

Izlaz pažnje je linearna kombinacija skrivenih stanja kodera:

$$out_{\text{attn}} = \sum_t^T \alpha_t k^{(t)} \ .$$

Pažnja: osnovna formulacija (2)

Izlaz pažnje je linearna kombinacija skrivenih stanja kodera:

- rezultat konkateniramo sa stanjem dekodera neposredno prije generiranja izlaza (**pažnja se ne koristi u povratnim ćelijama**)

$$h_{dec}^{*(t)} = [h_{dec}^{(t)}; out_{\text{attn}}] \ .$$

Kako formulirati funkciju sličnosti attn?

1. Diferencijabilni modul, npr. Bahdanauova funkcija s parametrima W_1 (matrica) i w_2 (vektor):

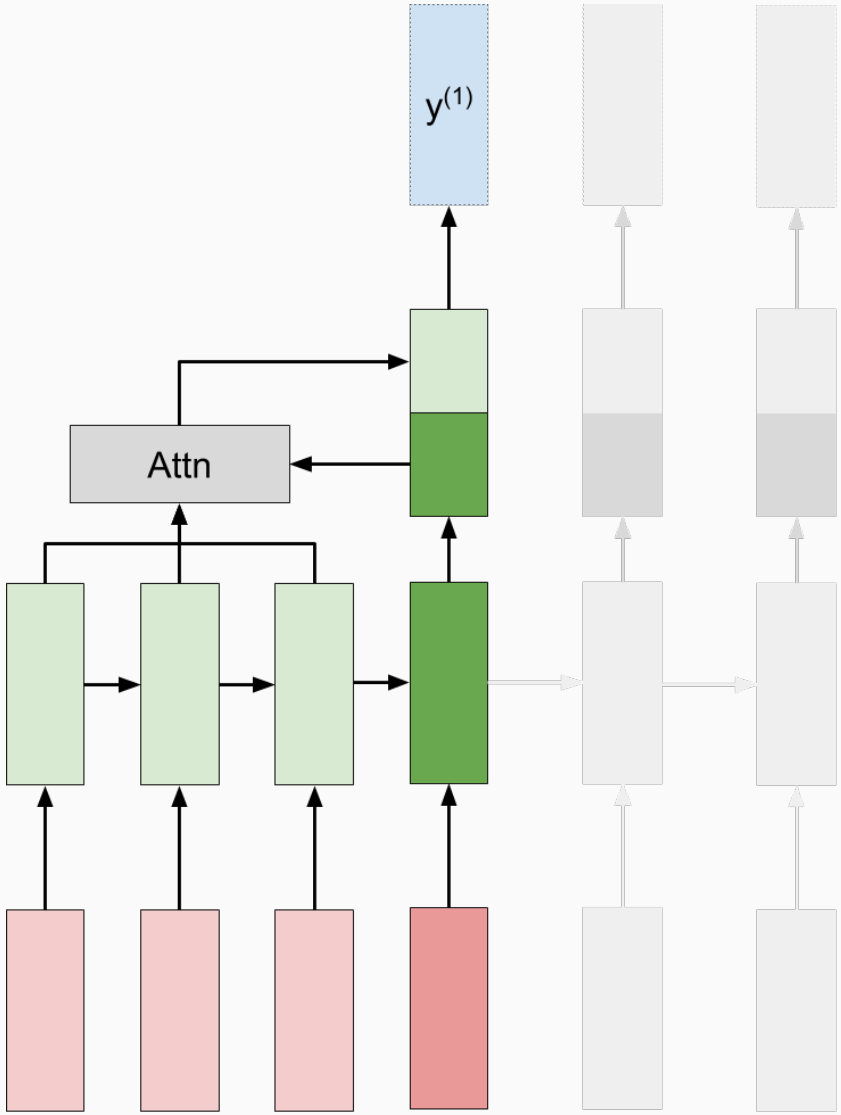
$$a^{(t)} = w_2^\top \cdot \tanh(W_1 \cdot [q^{(t)}; k^{(t)}]) \ .$$

2. Skalarni produkt (uvjet: $\dim(q) = \dim(k)$):

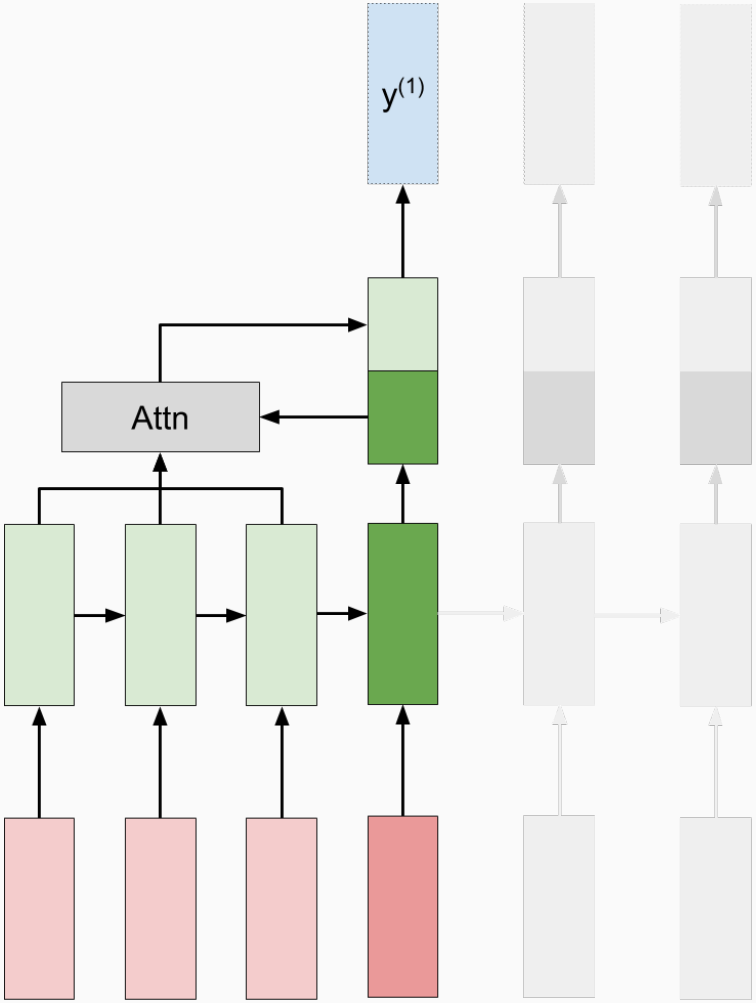
$$a^{(t)} = \frac{q^{(t)\top} \cdot k^{(t)}}{\sqrt{\dim(k)}} \ .$$

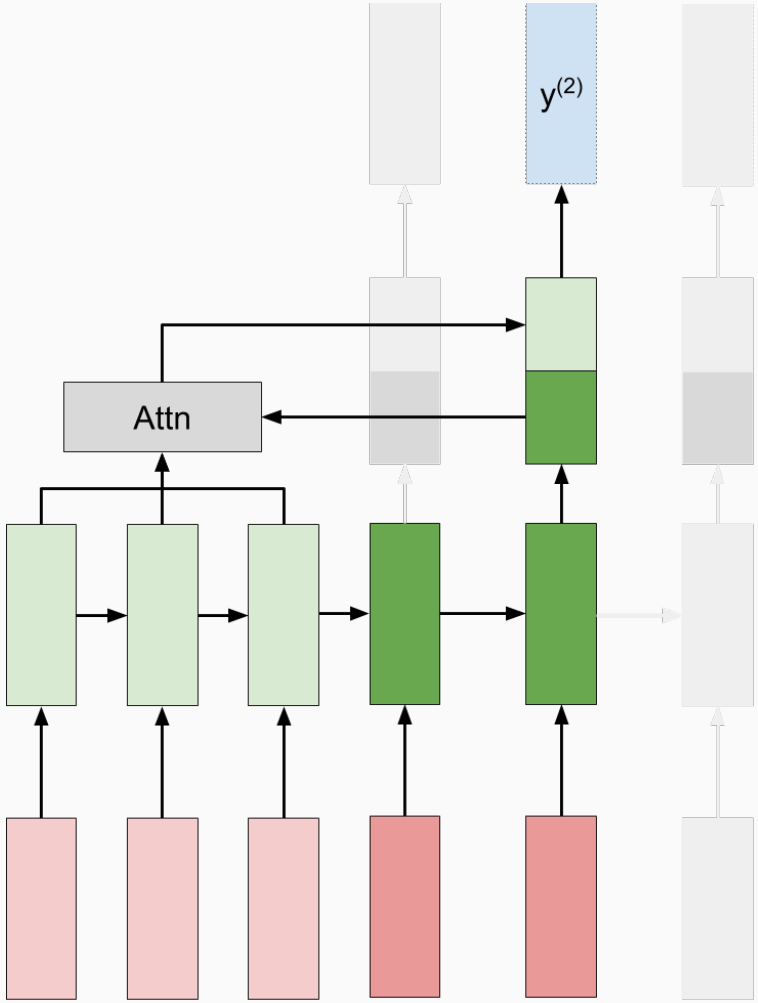
   - zašto skaliramo s dimenzionalnošću k?

Pažnja: vizualizacija

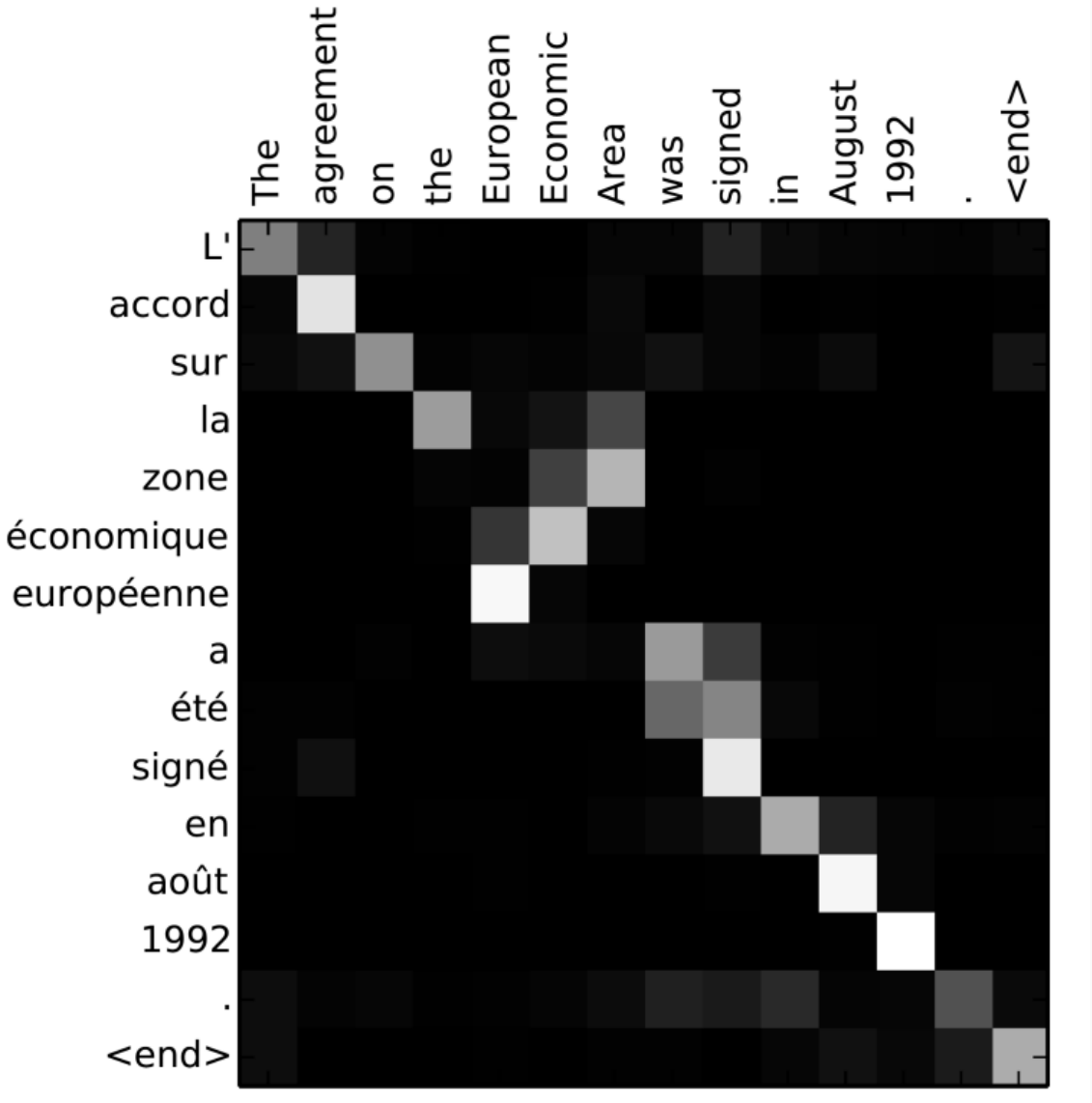


Pažnja: vizualizacija (2)





Pažnja: vizualizacija sličnosti



Sličnost između skrivenih stanja kodera i dekodera za slučaj prijevoda iz francuskog u engleski jezik.

Pažnja: proširena formulacija

Uvodimo razliku između **ključeva** i **vrijednosti**:

$$k^{(t)} = f_k(h_{enc}^{(t)}), \qquad v^{(t)} = f_v(h_{enc}^{(t)}) \ .$$

Funkcije f_k i f_v transformiraju skrivena stanja u **ključeve** i **vrijednosti**.

U praksi, f_k i f_v su projekcije:

$$k^{(t)} = W_k h_{enc}^{(t)}, \qquad v^{(t)} = W_v h_{enc}^{(t)} \ .$$

Proširena pažnja:

$$\alpha = \text{softmax}(\text{attn}(q, K)),$$

$$out_{\text{attn}} = \sum_t^T \alpha_t \, v^{(t)} \ .$$

Ova formulacija može biti korisna i izvan prevođenja iz slijeda u slijed

Pažnja u klasifikaciji slijedova

Ako upit dolazi iz iste reprezentacije kao i ključevi, onda se pažnja $\text{attn}(k_i, K)$ može približiti jednojediničnom vektoru e_i:

- može se izbjeći **naučenim upitima**
- izgleda da računalni vid ne pati od ovog problema

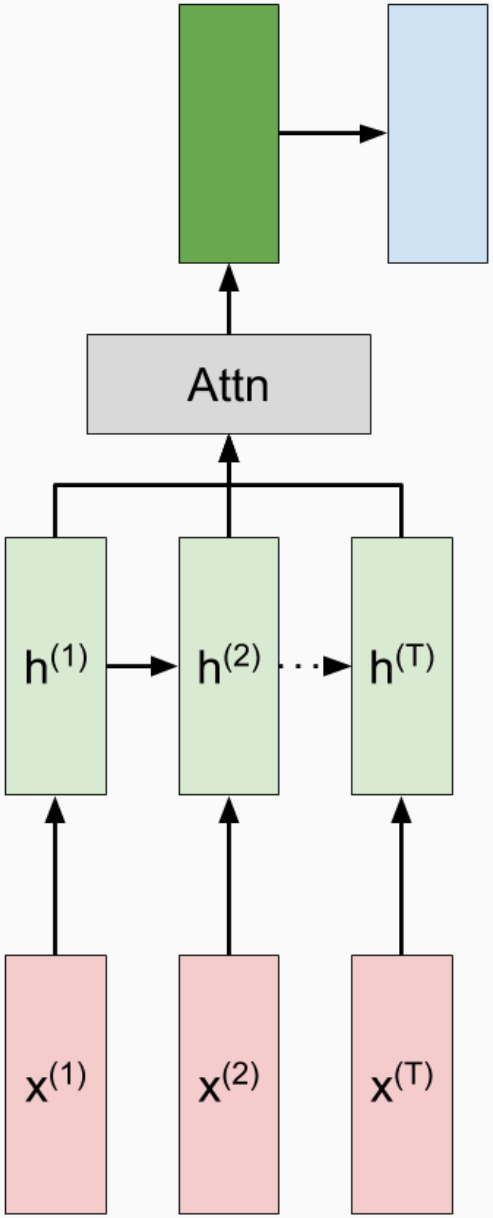
Pažnja s naučenim upitima w:

$$\hat{\alpha} = \text{softmax}(attn(w, K)),$$

$$out_{attn} = \sum_{t}^{T} \hat{\alpha}_t v^{(t)} .$$

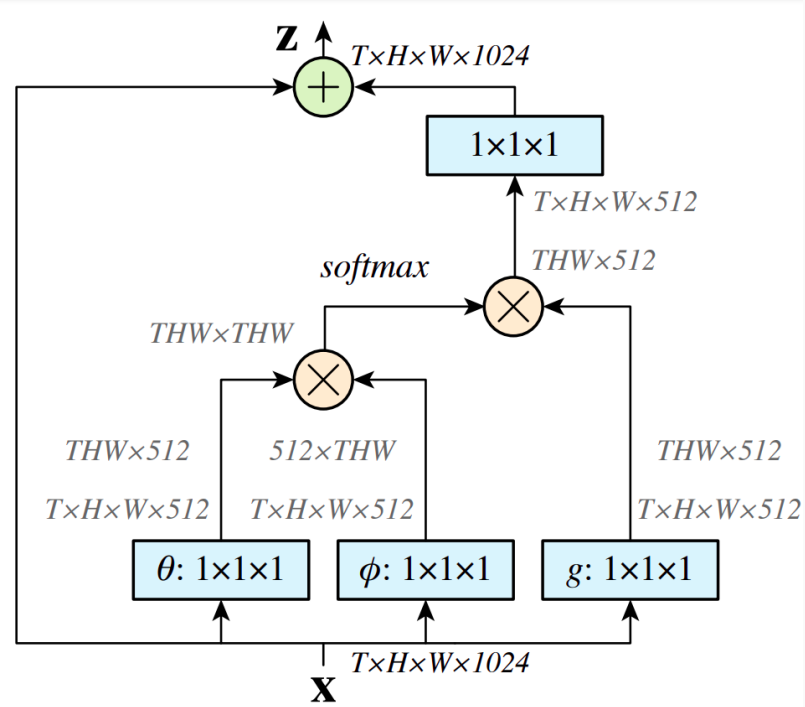
Intuitivno, naučeni upiti odgovaraju apstraktnim konceptima kao što su *formalni tekst*, *slang*, *nogomet*, *bliski istok*, itd.

Pažnja u klasifikaciji slijeda: vizualizacija



Pažnja u računalnom vidu (klasifikacija videa)

Neki algoritmi računalnog vida modeliraju **daleke** međuovisnosti proširenom pažnjom bez naučenih upita



[wang18cvpr]

Ulaz: apstraktna reprezentacija X

- tenzor 4. reda T×H×W×1024
- gledamo ga kao THW×1024.
- H - visina, W - širina, vrijeme

Izlaz: reprezentacija Z s poboljšanim dalekim vezama

Ulaz X projiciramo na upite (θ), ključeve (ϕ) i vrijednosti (g).

Svaka značajka $x_i \in R^{1024}$ istovremeno je i upit i ključ.

Matrica sličnosti A (THW $\times$ THW) uspoređuje upite s ključevima.

Pažnja u računalnom vidu (detalji)

Matricu A dobivamo matričnim množenjem:

- lako je umetnuti i drukčije formulacije sličnosti.
- tu bi dobro došlo i normaliziranje varijance ($\sqrt{1024}$)

$$\begin{aligned} A &= (W_\theta X^\top)^\top \cdot (W_\phi X^\top), \\ &= (X W_\theta^\top) \cdot (W_\phi X^\top) \ . \end{aligned}$$

Matricu težina α dobivamo aktiviranjem redaka softmaksom.

- α_{ij} odražavan sličnost upita $W_\theta x_i$ i vrijednosti $W_g x_j$

$$\alpha = softmax(A, axis = 1) \ .$$

` Izlazi $Z = \{z_i\}$ su linearne kombinacije vrijednosti $V = g(x_i)$:

- naravno, težine odgovaraju elementima matrice α

$$z_i = \sum_j \alpha_{ij} \cdot g(x_j) \ .$$

Pažnja: sažetak

Najbolji povratni modeli i dalje se muče s dugačkim rečenicama

Zato uvodimo **pažnju** za modeliranje dalekih međuovisnosti

- pažnja je linearna kombinacija skrivenih stanja
- težine modeliraju **sličnost** s obzirom na odgovarajući upit
  - važnost ulaznih informacija ovisi o trenutnom izlazu
- u povratnim modelima iz slijeda u slijed, **upit** je tekuće skriveno stanje dekodera, dok su ključevi skrivena stanja **kodera**
- proširenja su primjenjiva na klasifikaciju slijedova i gustu predikciju.

Pažnja: sažetak (2)

Kako formulirati sličnost:

- Bahdanauova pažnja: differencijabilni modul djeluje na konkatenaciju upita i ključa
- skalarni produkt: *izravna* usporedba (projiciranog) upita s (projiciranim) ključem

Pažnja se koristi kao dodatak svim novijim povratnim modelima.

Pažnja je kritična komponenta suvremenih algoritama strojnog učenja.

Pažnja je … sve što trebamo?

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions

Modeli zasnovani na pažnji

Komentirani članak s primjerima izvornog koda:
http://nlp.seas.harvard.edu/2018/04/03/attention.html

Primjene, primjeri, stanje tehnike

gail weiss
@gail_w

Follow

Roses are red
RNNs process sequences
But if you look closely
You'll find they have weaknesses

New work with @yoavgo and @yahave on the practical power of different RNN architectures available now on arxiv, and soon at @acl2018!

Nedostatci povratnih pristupa

Povratni model sa sigmoidalnom aktivacijom i beskonačnom numeričkom preciznošću može simulirati Turingov stroj.

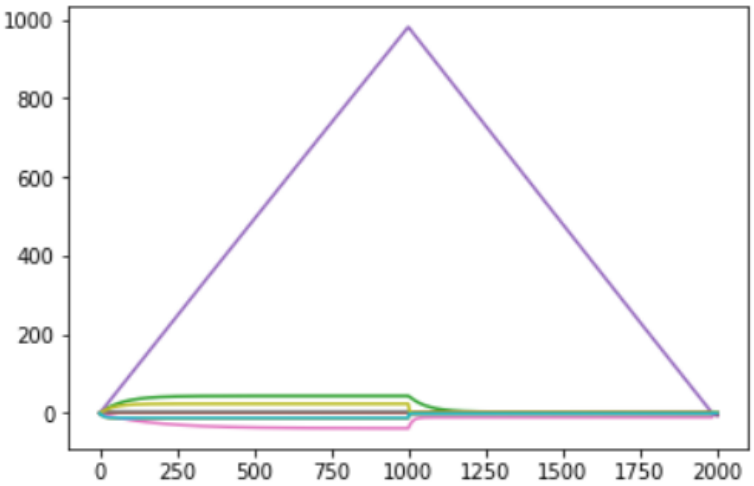
Rezultat je proširen i na ReLU aktivaciju.

Međutim, ovo vrijedi samo pod stanovitim okolnostima:

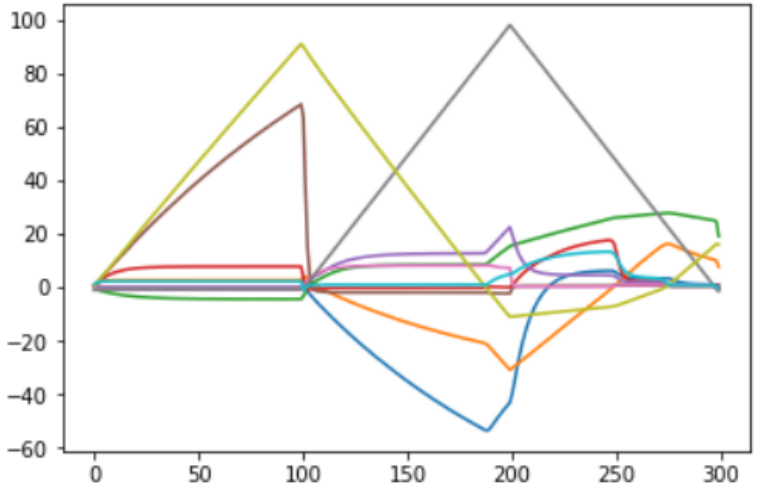
1. cijeli slijed je uveden u povratni model te imamo beskonačno vrijeme zaključivanja
2. **beskonačna** numerička preciznost.

Eksperiment brojanja: povratni model prepoznaje slijedove $a^n b^n$ ili $a^n b^n c^n$

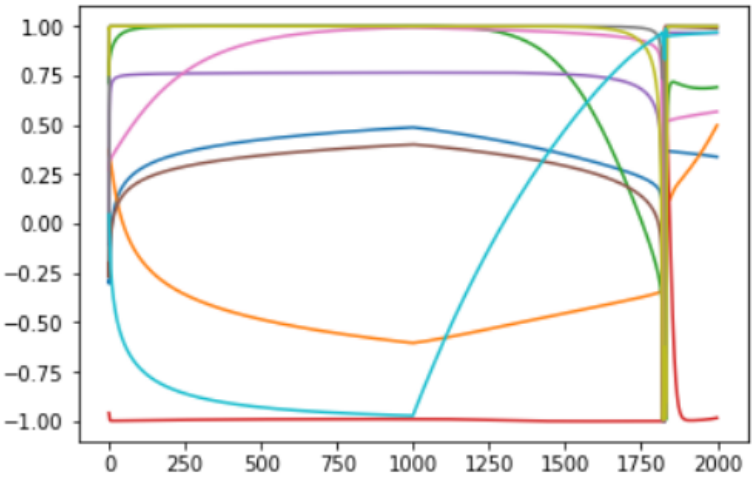
Članak: https://arxiv.org/pdf/1805.04908.pdf



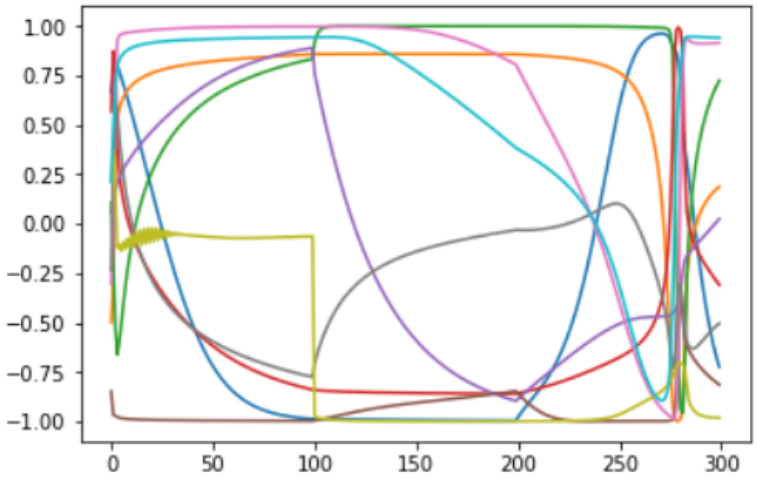
(a) $a^n b^n$-LSTM on $a^{1000} b^{1000}$



(b) $a^n b^n c^n$-LSTM on $a^{100} b^{100} c^{100}$



(c) $a^n b^n$-GRU on $a^{1000} b^{1000}$

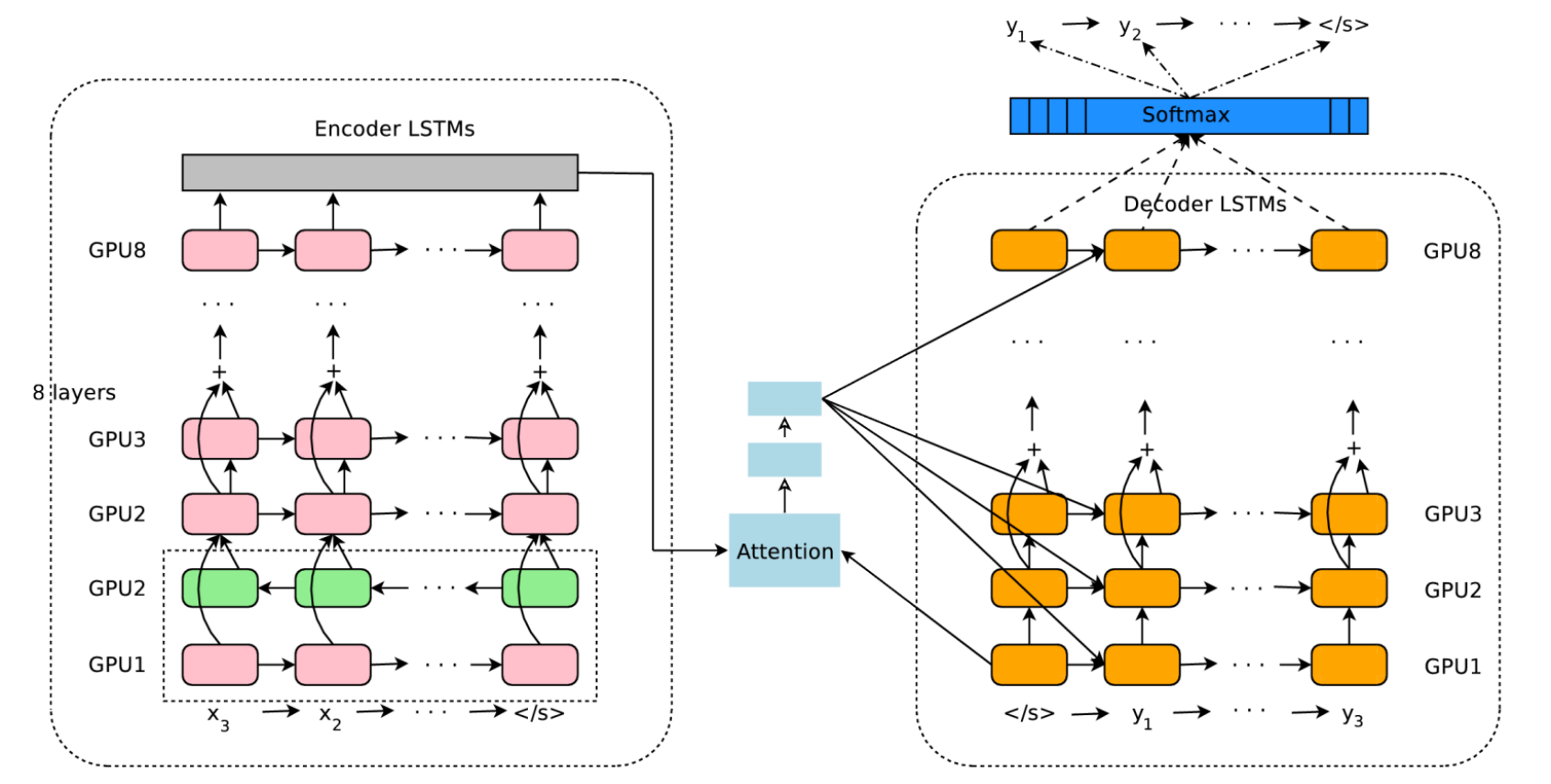


(d) $a^n b^n c^n$-GRU on $a^{100} b^{100} c^{100}$

Strojno prevođenje

Google translate

https://research.googleblog.com/2016/09/a-neural-network-for-machine.html



Google translate network arhitecture

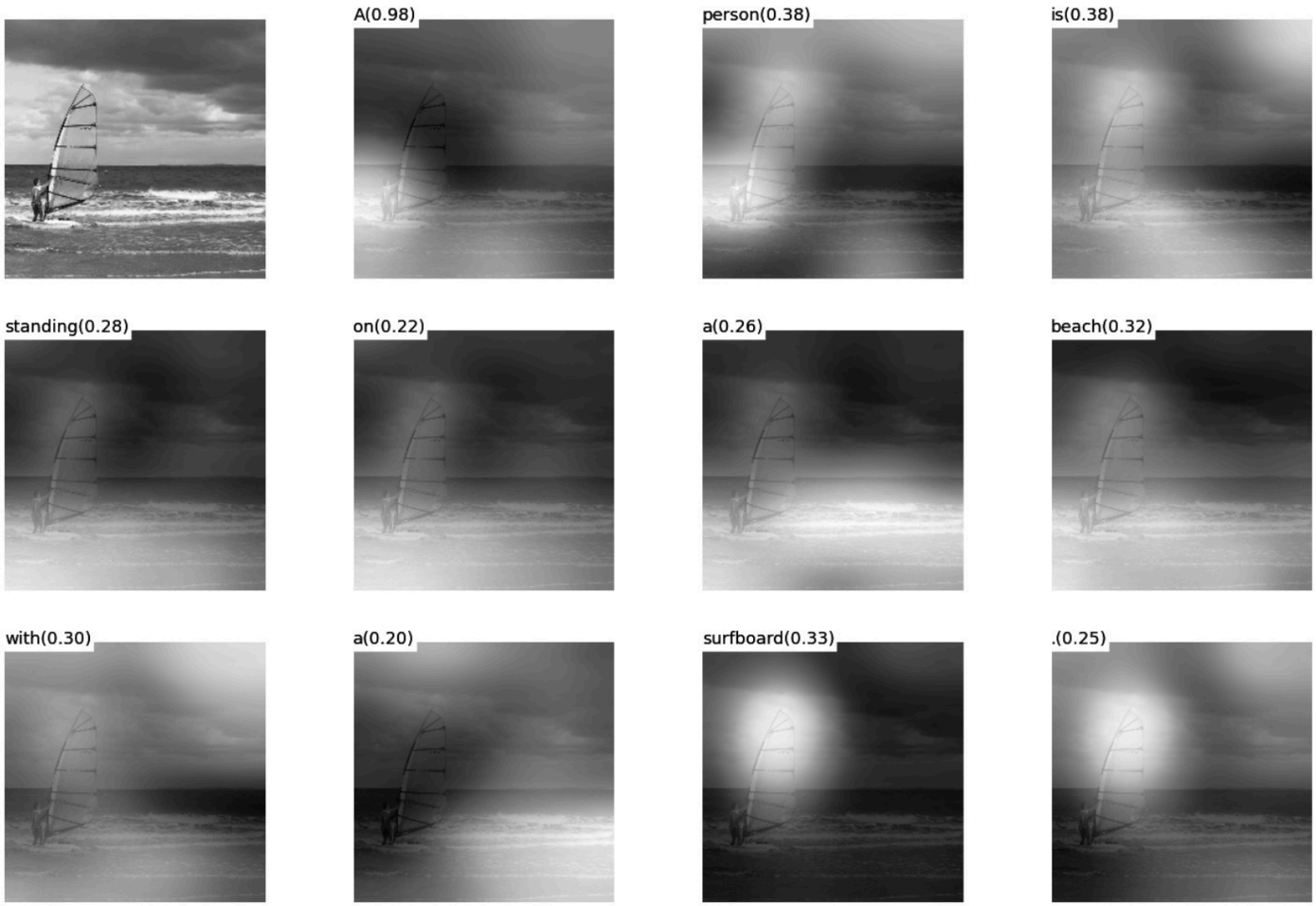
Generiranje govora

https://google.github.io/tacotron/

Generiranje zvuka iz teksta

https://deepmind.com/blog/wavenet-generative-model-raw-audio/

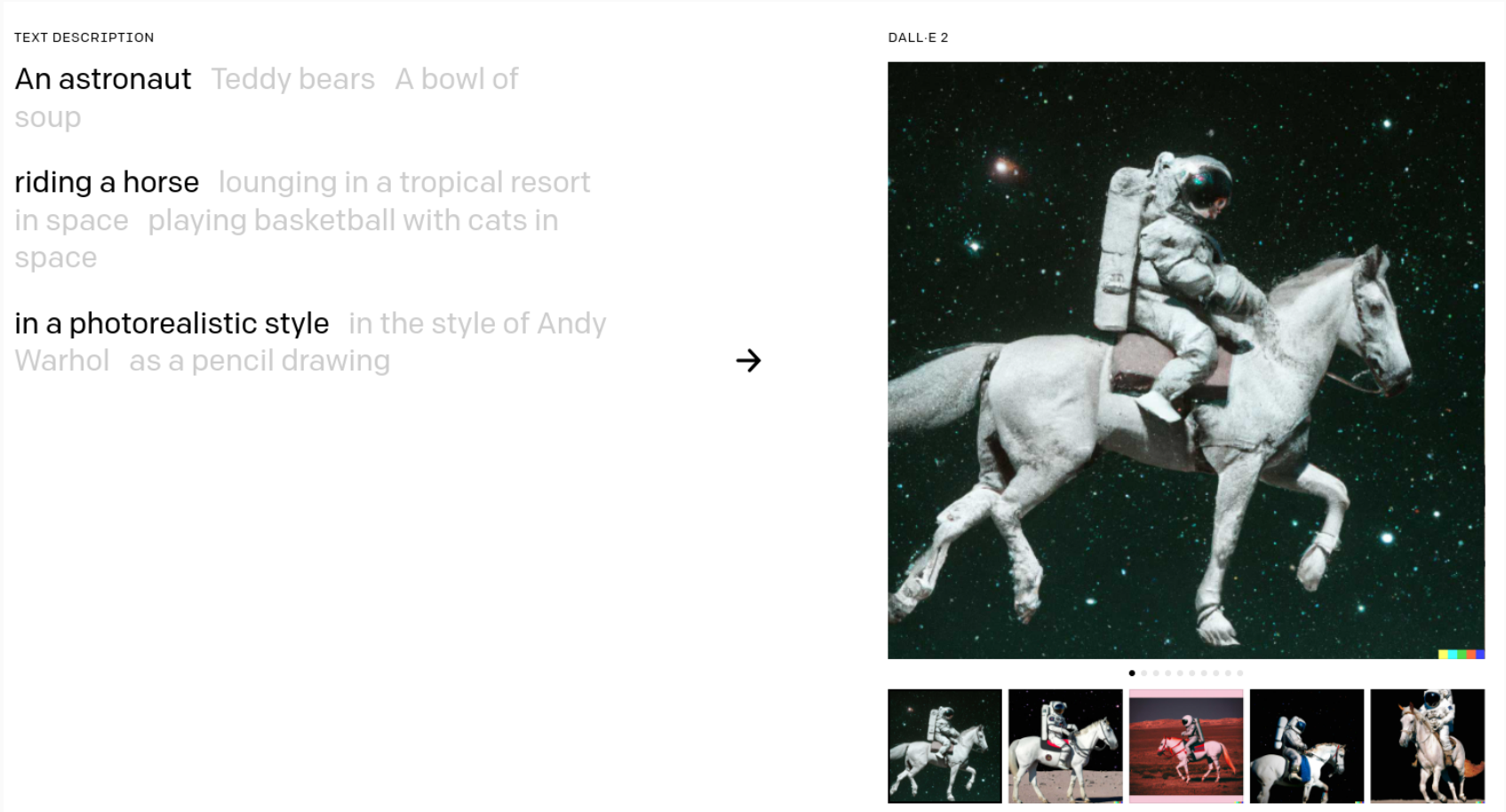
Opisivanje slika



(b) A person is standing on a beach with a surfboard.

Opisivanje slika povratnim modelima s pažnjom [xu2015show]

Uvjetno generiranje slika



DALLE-2

https://openai.com/dall-e-2/

Generiranje teksta

A robot wrote this entire article. Are you scared yet, human?

GPT-3

We asked GPT-3, OpenAI's powerful new language generator, to write an essay for us from scratch. The assignment? To convince us robots come in peace

- For more about GPT-3 and how this essay was written and edited, please read our editor's note below

https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3;

https://arxiv.org/abs/2005.14165

Rješavanje matematičkih problema

Question

Ali is a dean of a private school where he teaches one class. John is also a dean of a public school. John has two classes in his school. Each class has 1/8 the capacity of Ali's class which has the capacity of 120 students. What is the combined capacity of both schools?

Answer

Method: **175B Verification**

```
Ali's class has a capacity of 120 students.
Each of John's classes has a capacity of 120/8 = 15 students.
The total capacity of John's two classes is 15 students * 2 classes =
30 students.
The combined capacity of the two schools is 120 students + 30 students =
150 students.
```

Verifiers:

`https://openai.com/blog/grade-school-math/`

Generiranje koda prema instrukcijama

Demo:

`https://www.youtube.com/watch?v=SGUCcjHTmGY`

Blog post:

`https://openai.com/blog/openai-codex/`

Pitanja?

Knjiga

- relevantna poglavlja: 10.1, 10.2, 10.3, 10.4, 10.5, 10.7, 10.10, 10.11