Pokročilé metódy analýzy dát 5

úvod do hlbokého učenia

Peter Bednár

Pokročilé architektúry pre priestorové dáta

Obohacovanie dát

- Snažíme sa dosiahnuť čo najväčšiu robustnosť metódy
- V niektorých prípadoch nezáleží na tom, či je rozpoznávaný objekt na obrázku nejak transformovaný (posunutý, otočený, skosený, zrkadlený atď.)

- Manuálne vytvorenie trénovacích dát je prácne - z existujúcich trénovacích obrázkov si vygenerujeme nové klasifikované príklady:
  - Pridaním náhodného šumu
  - Priestorovým transformovaním

Afinné transformácie (1)

- Je možné ich reprezentovať maticou 3x3 – body (vektory) sa transformujú maticovým násobením
- Posunutie:

$$\begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

- Zrkadlenie (podľa osi Y)

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Skosenie

$$\begin{bmatrix} 1 & c_x & 0 \\ c_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Afinné transformácie (2)

- Otočenie:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Zmena mierky (zväčšenie/zmenšenie)

$$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Operácie je možné skladať násobením transformačných matíc

Prenos učenia

- *Transfer learning* – model, alebo jeho časť, najprv optimalizujeme na rozsiahlych dátach na jednej úlohe/ doméne a potom ich preučíme na cieľovú úlohu/doménu.

- Konvolučné vrstvy sa hierarchicky učia rozpoznávať stále zložitejšie a zložitejšie vizuálne vzory vo vstupnom obrázku
- Vyextrahované vzory môžu byť užitočné aj pre klasifikáciu nových objektov - konkrétne objekty rozpoznáva až dopredná sieť

Prenos učenia pri klasifikácii obrazu

1. Naučíme sieť na rozsiahlych dátach a triedach v jednej doméne
   – Napr. ImageNet – základná trénovacia množina má 1.2 mil. obrázkov klasifikovaných do 1000 tried)
2. Zachováme konvolučné vrstvy, doprednú vrstvu nahradíme pre nové triedy a celú sieť preučíme na cieľových dátach

- Pre ImageNet je možné stiahnuť už pred-učené modely rôznych architektúr
- Napr. VGG16 v Kerase:

```
keras.applications.vgg16.VGG16(include_top=False)
```

Rozšírené úlohy spracovania obrazu

Klasifikácia

trieda pre celý obrázok

Detekcia objektov

trieda + lokácia (obdĺžniková oblasť)

Sémantické segmentovanie

trieda pre každý pixel

Ďalšie rozšírenie: viac tried/objektov

Detekcia objektov

1. Rozdelíme obrázok na kandidátov - obdĺžnikové oblasti v ktorých sa pravdepodobne nachádzajú klasifikované objekty
2. Upravíme hranice oblastí a určíme triedu pre každého kandidáta

- Predikujeme dva výstupy:
  - Trieda – klasifikácia – krížová entropia
  - Hranice oblasti (4 číselné hodnoty: x, y, w, h) – regresia – kvadratická chybová funkcia
- Zložená chybová funkcia = $\alpha\, L_{cls} - (1-\alpha) L_2$
  - α – nastavenie algoritmu pre lineárnu kombináciu chýb (každá chyba má iný rozsah hodnôt)

Detekcia objektov – detekcia kandidátov

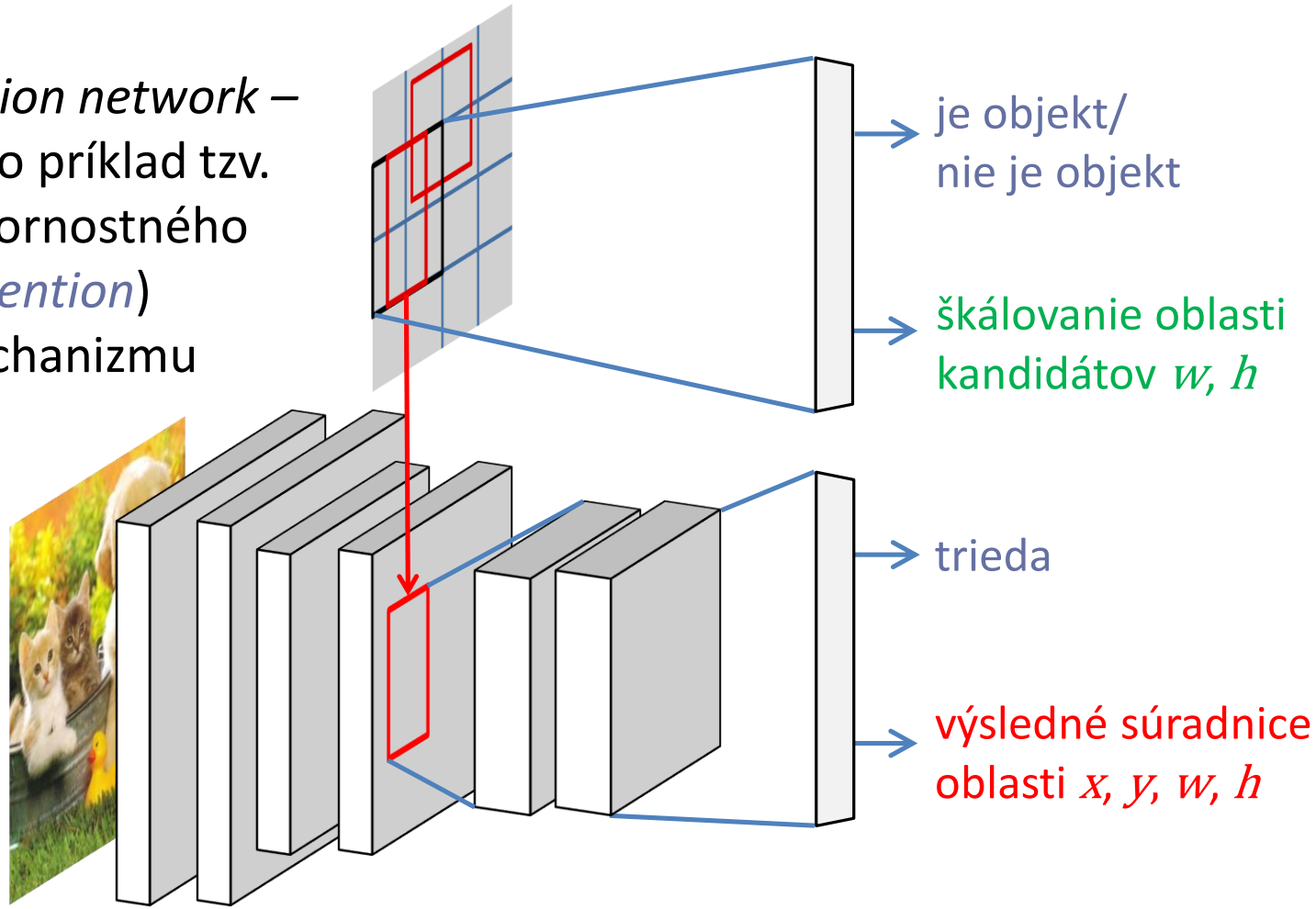
- Výstup konvolučnej siete – obrazové dáta (mapa príznakov - šírka x výška x počet filtrov na poslednej vrstve)
  - výstupná šírka a výška závisí od vstupných rozmerov obrázka a nastavení konvolúcie a poolingu
- Mapu príznakov rozdelíme na mriežku a umiestime na ňu stredy kandidátov
- Pre každý stred (ukotvenie)
  - Binárna klasifikácia "je objekt/nie je objekt"
  - Počiatočná výška a šírka oblasti w, h
- Dopredná sieť s dvoma výstupnými vrstvami, ktorá sa zdieľa pre všetky stredy

Detekcia objektov – klasifikácia kandidátov

- Na vstup doprednej siete premietneme príznaky z konvolučnej siete pre oblasť kandidáta
- Pre každého kandidáta
  - Klasifikácia do tried
  - Výsledná poloha a rozmery oblasti x, y, w, h
- Dopredná sieť s dvoma výstupnými vrstvami, ktorá sa zdieľa pre všetkých kandidátov

Detekcia objektov – R-FCN

Region network – ide o príklad tzv. pozornostného (*attention*) mechanizmu



Sémantické segmentovanie

- Výstup je mapa – obrazové dáta s rovnakým rozmerom ako vstupný obrázok x počet tried pre každý pixel
- Chybová funkcia – krížová entropia pre jeden pixel, suma pre všetky pixely

Sémantické segmentovanie

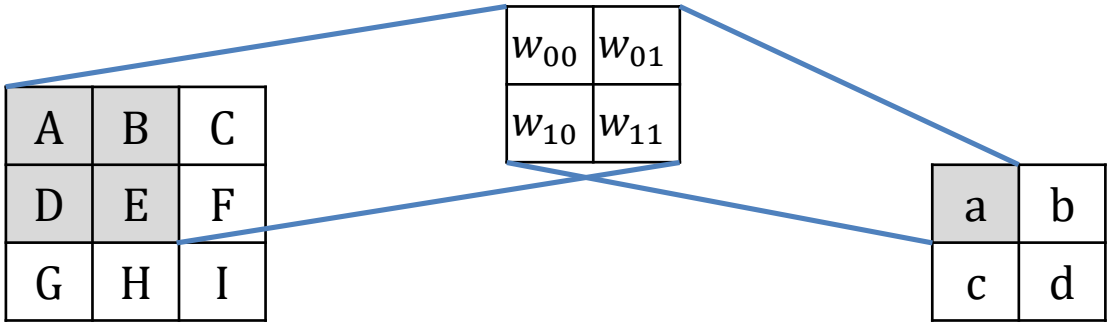
- Kodér
  - Konvolučná sieť – najčastejšie pred-učená pre klasifikáciu/detekciu objektov
  - Postupne znižuje rozmer, informácie sa zakódujú do väčšieho počtu kanálov
  - Napr. pre VGG16 z 224x224x3 na 7x7x512
- Dekodér
  - Dekonvolučná sieť
  - Postupne zväčšuje rozmer - aproximuje hodnoty vložených bodov
  - Počet kanálov poslednej vrstvy zodpovedá počtu tried

Dekonvolúcia – transponovaná konvolúcia

Príklad konvolúcie:
Vstup 3x3

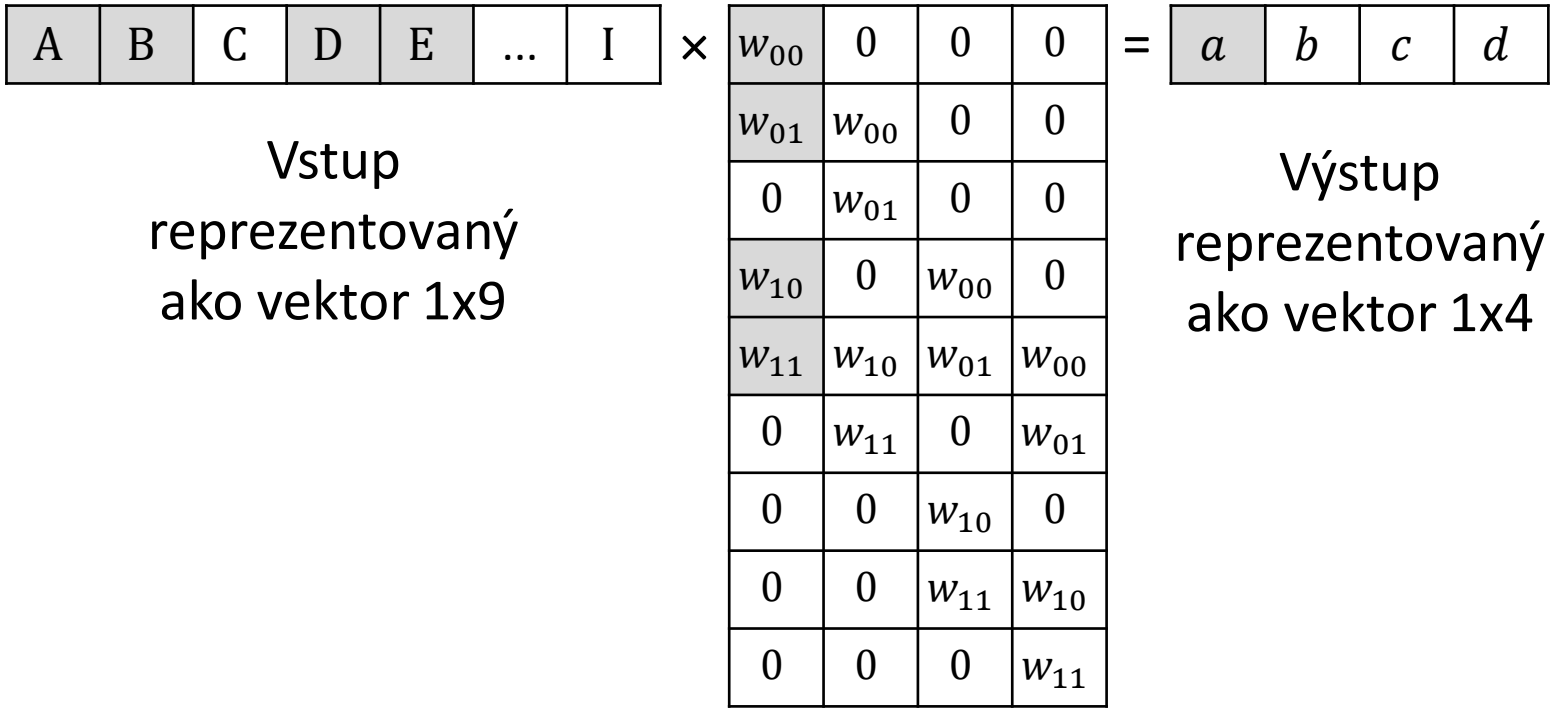
Filter 2x2,
posun 1,1,
okraj 0,0

Výstup 2x2



- Operáciu konvolúcie na celom vstupe zapíšeme pomocou transformačnej matice zostavenej z hodnôt filtra

Dekonvolúcia – transponovaná konvolúcia (1)

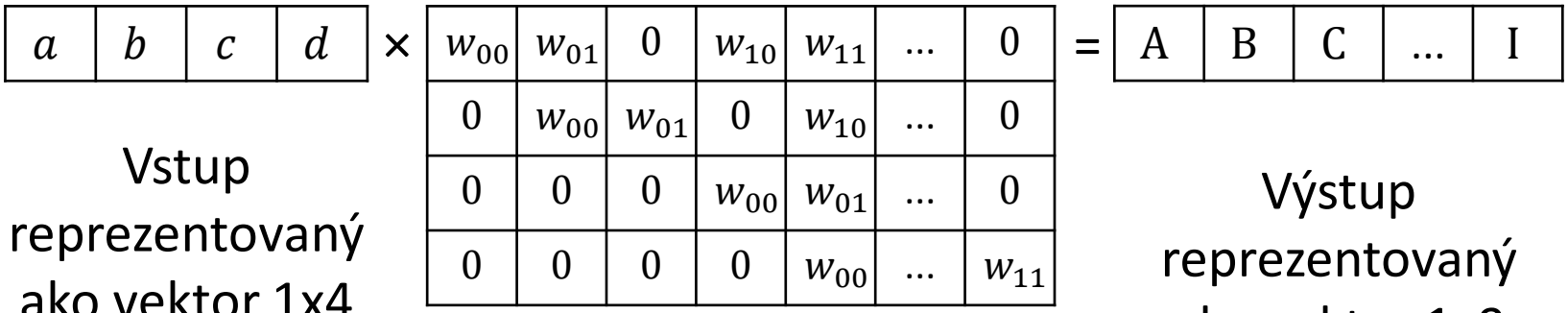


Vstup reprezentovaný ako vektor 1x9

Transformačná matica 9x4

Výstup reprezentovaný ako vektor 1x4

Dekonvolúcia – transponovaná konvolúcia (2)

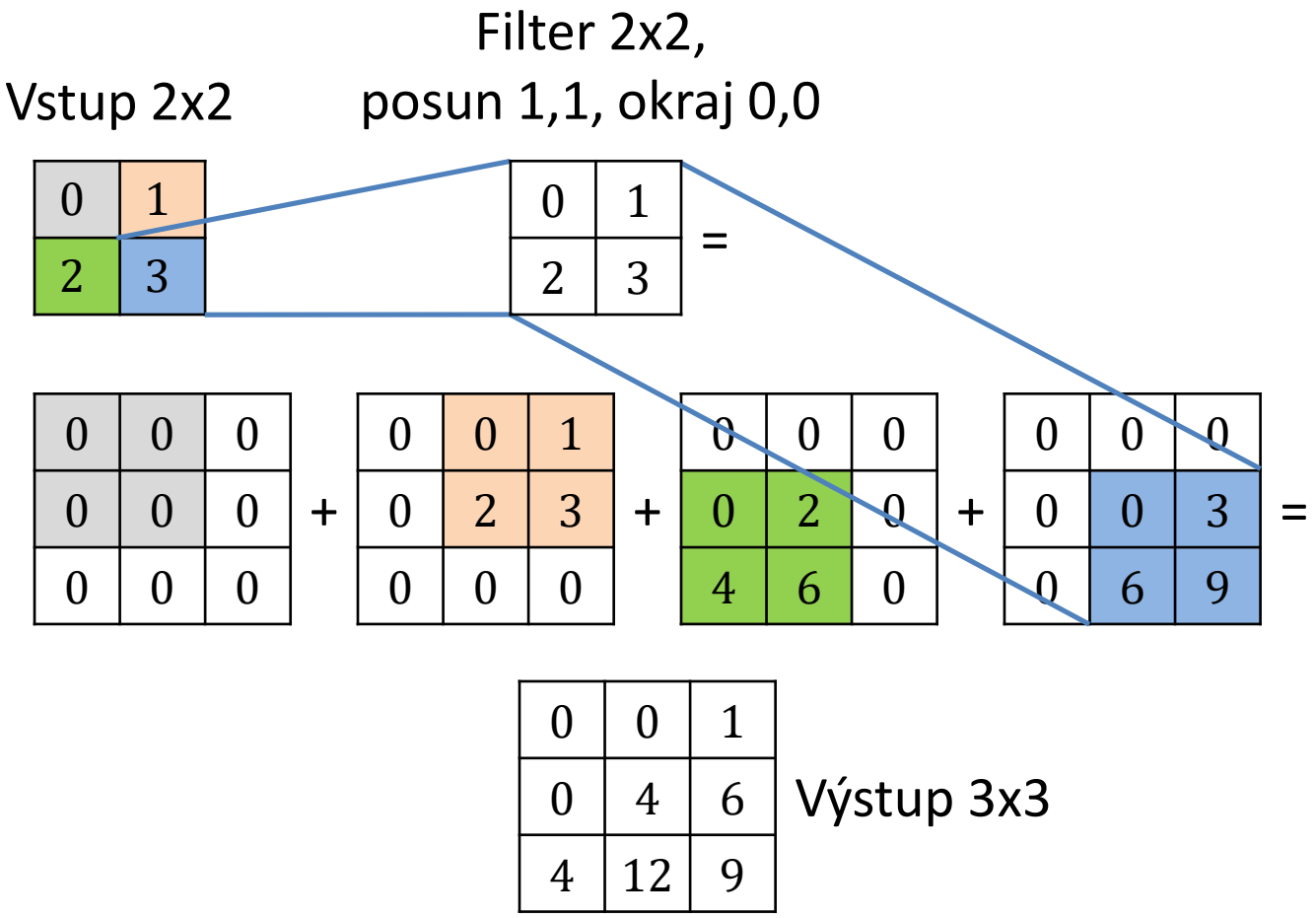


Vstup reprezentovaný ako vektor 1x4

Transponovaná transformačná matica 4x9

Výstup reprezentovaný ako vektor 1x9

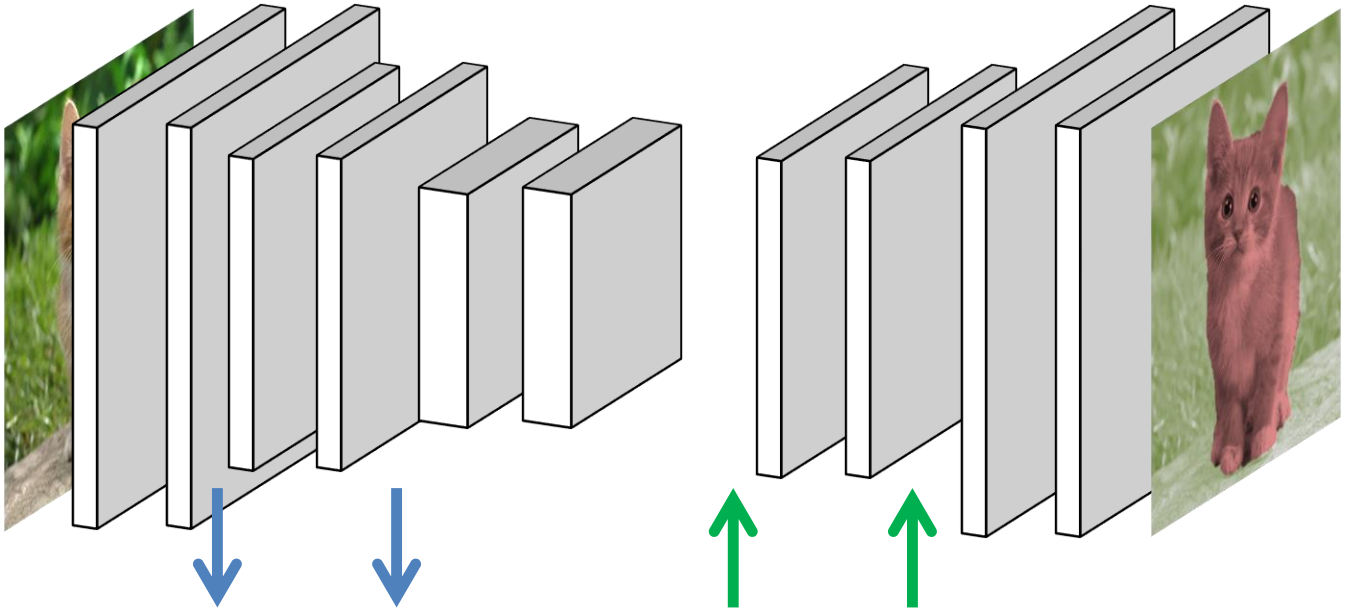
Dekonvolúcia – transponovaná konvolúcia (3)



Sémantické segmentovanie

Kodér - konvolučná sieť

Dekodér - dekonvolučná sieť



Zmenšenie rozmeru
pooling/
konvolúcia

Zväčšenie rozmeru
transponovaná konvolúcia

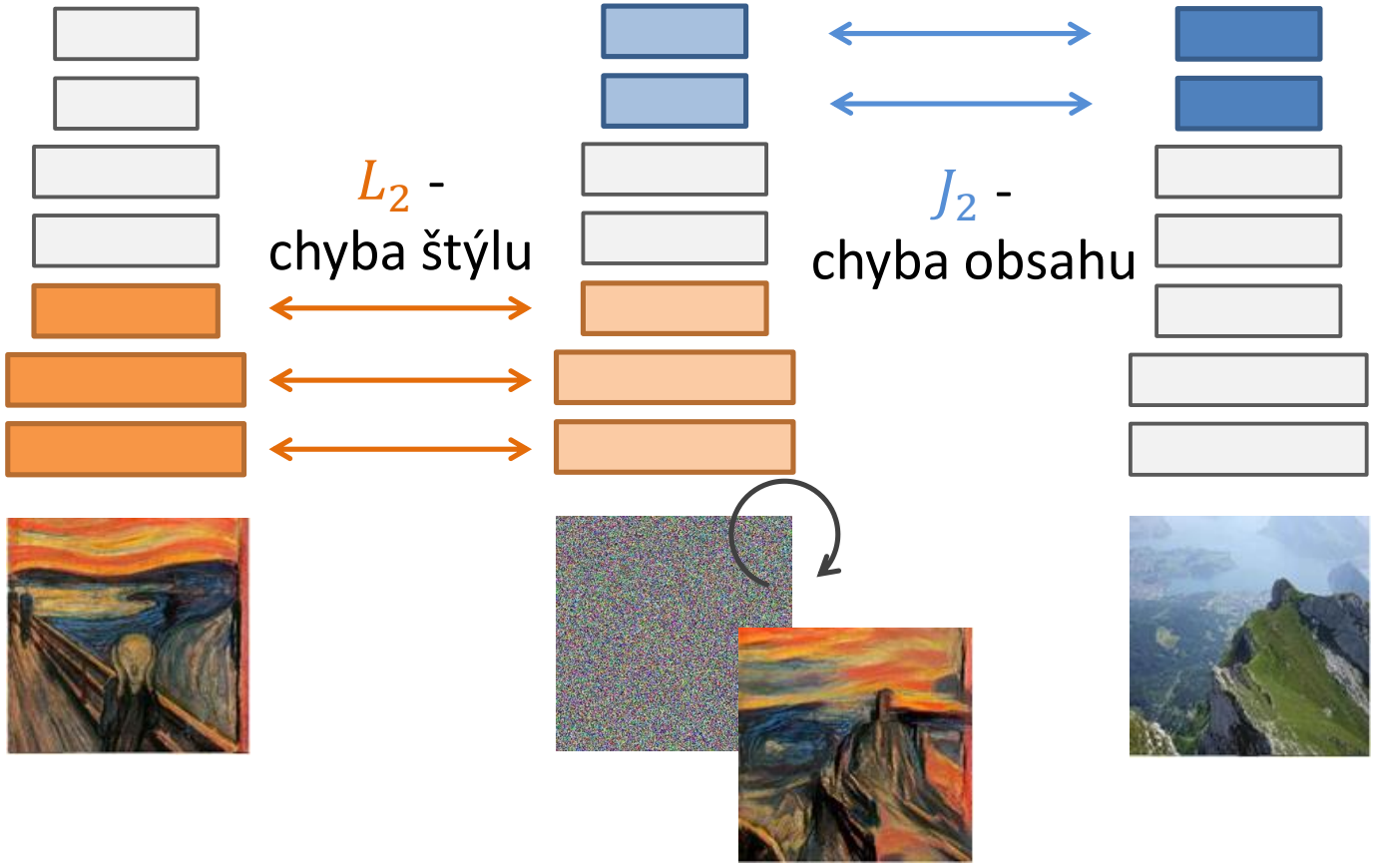
Zmena štýlu

- Chceme preniesť grafický štýl z jedného obrázka na iný

- Hierarchia príznakov konvolučnej siete
  - Príznaky štýlu (základné vzory ako napr. hrany, farebné prechody) – nižšie vrstvy
  - Príznaky obsahu (zložitejšie vzory) – vyššie vrstvy

Zmena štýlu

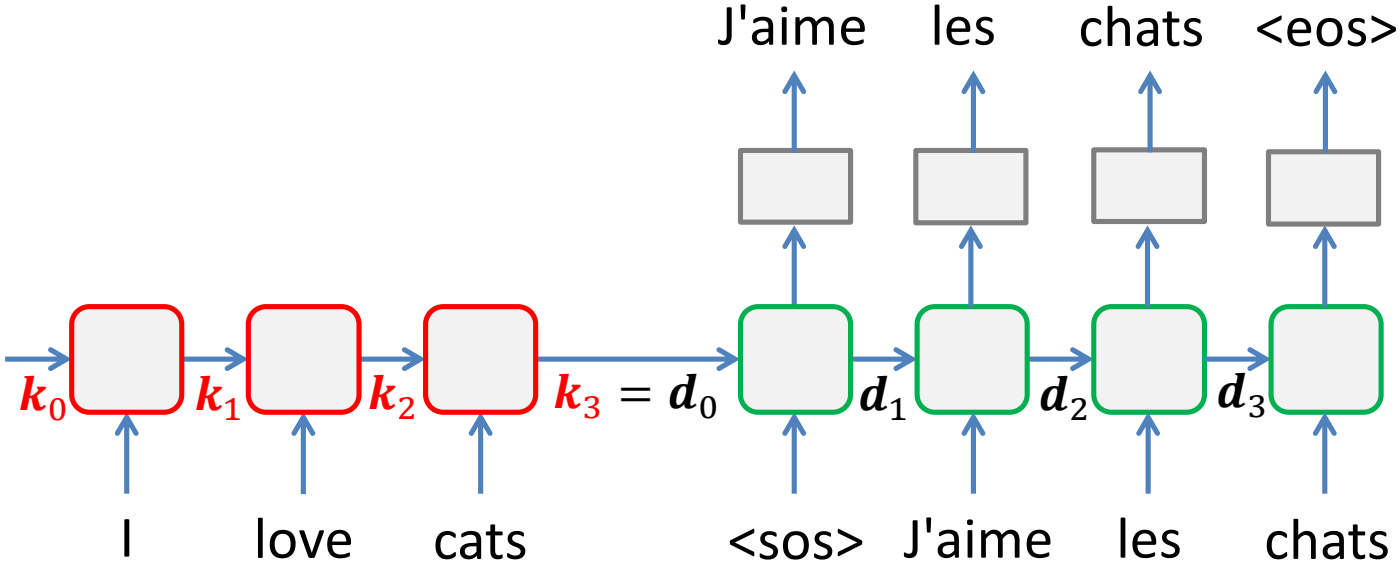
- Optimalizujeme obrázok – nie parametre konvolučnej siete



Pokročilé architektúry pre sekvenčné dáta

Pozornostné modely pre sekvencie

- Problém: pri mapovaní typu sekvencia-sekvencia (napr. pri automatickom preklade) musia byť všetky informácie zo vstupnej sekvencie zakódované do jedného vektora (posledného stavu), z ktorého musí dekodér vygenerovať celú výstupnú sekvenciu

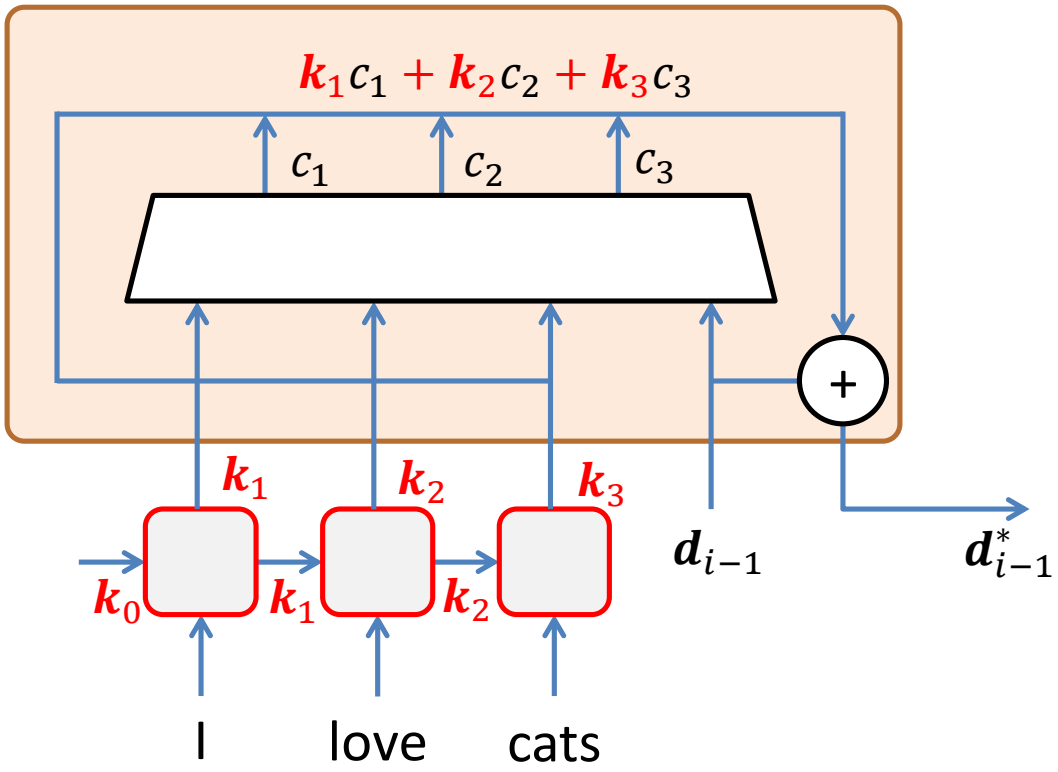


Pozornostné modely pre sekvencie

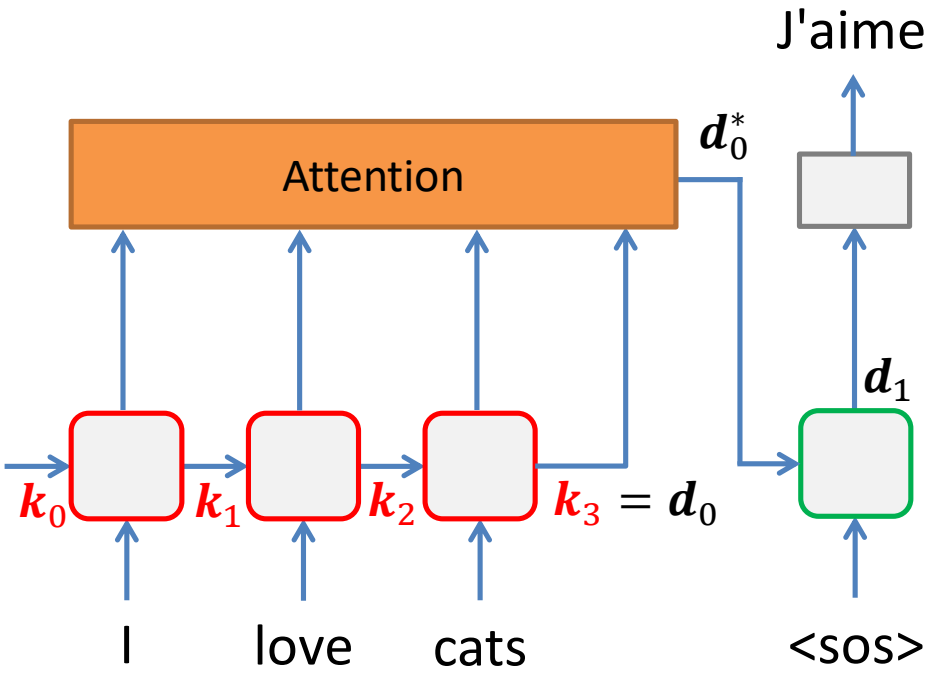
- Pozornostný model

- Dopredná sieť
- V každom kroku pri dekódovaní má na vstupe:
  - Všetky stavy kodéra
  - Predchádzajúci stav dekodéra
- Pre každý vstup vypočíta, s akou váhou je stav kodéra dôležitý pre generovanie aktuálneho výstupu
- Stav dekodéra sa modifikuje lineárnou kombináciou všetkých stavov kodéra

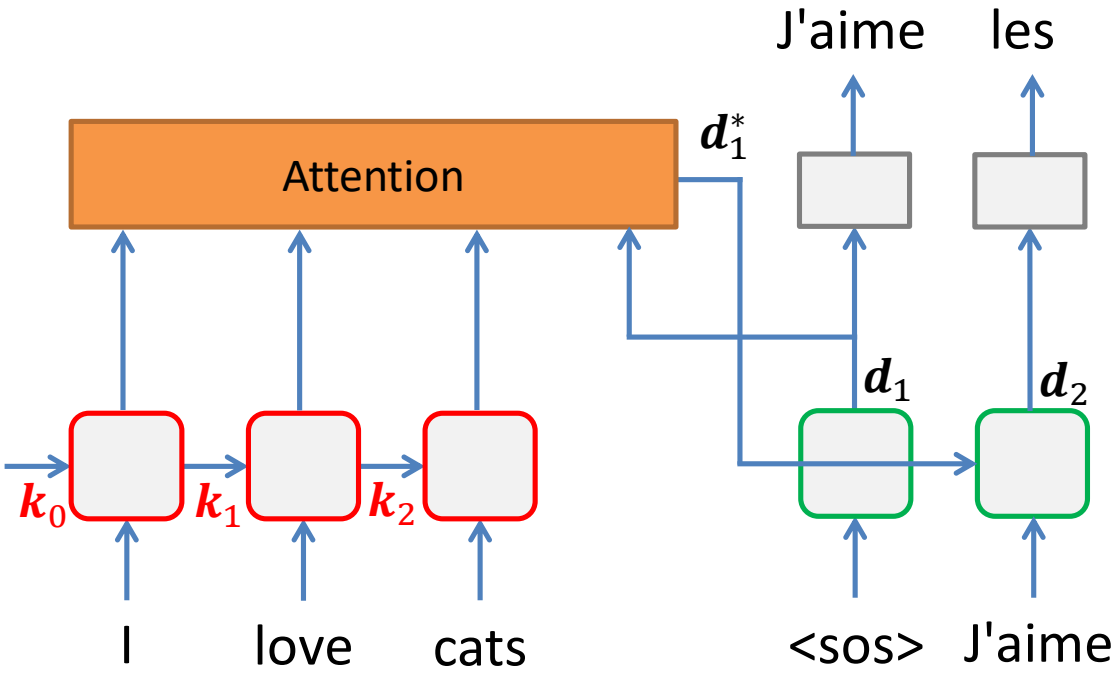
Pozornostné modely pre sekvencie



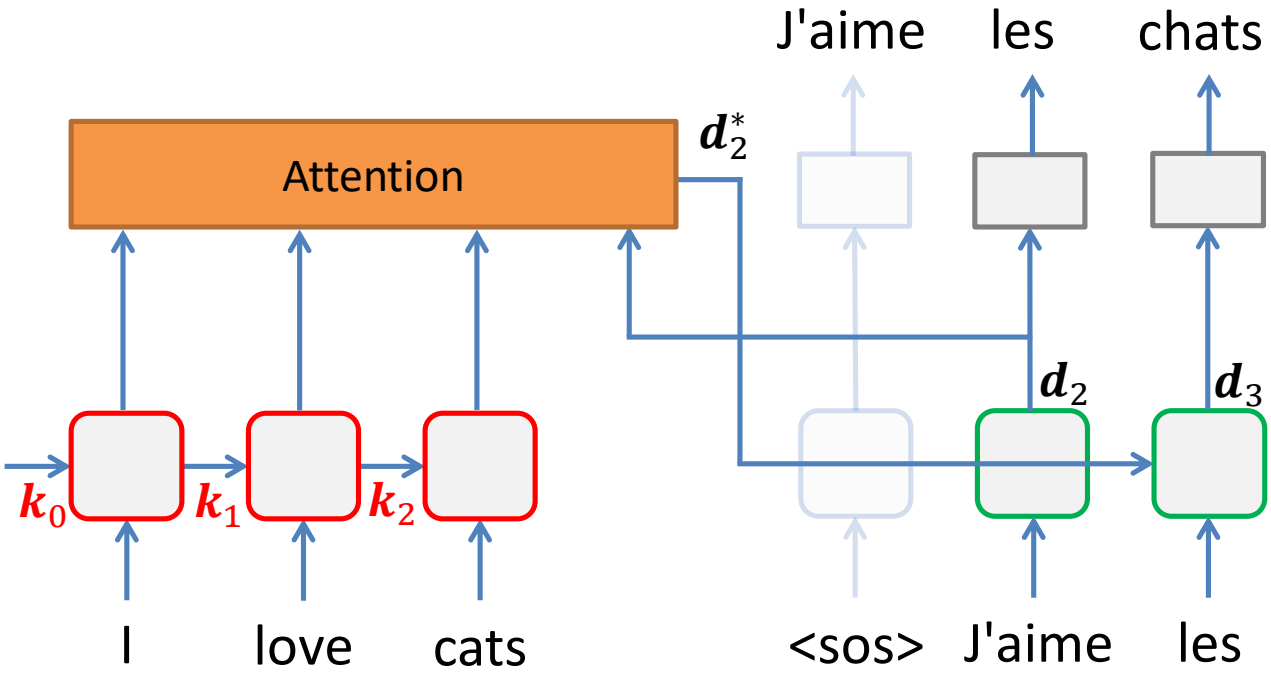
Pozornostné modely pre sekvencie



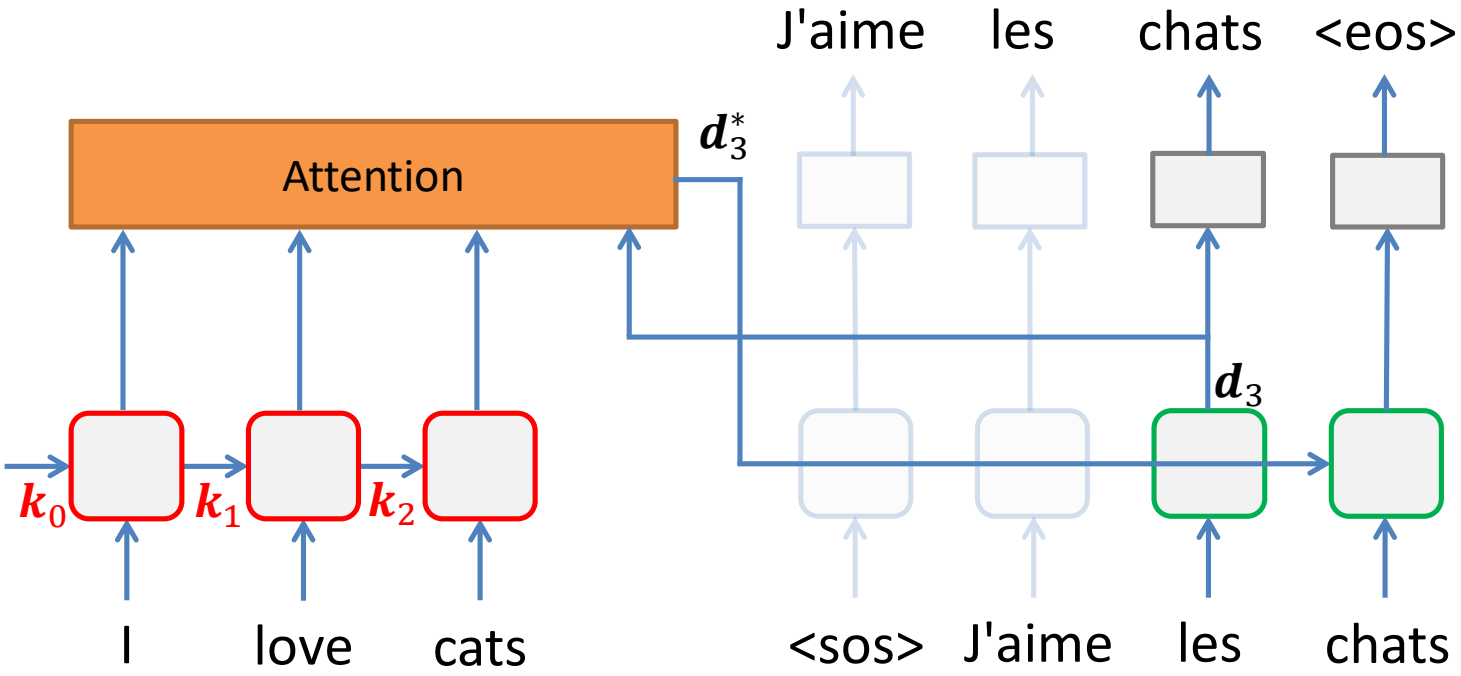
Pozornostné modely pre sekvencie



Pozornostné modely pre sekvencie



Pozornostné modely pre sekvencie



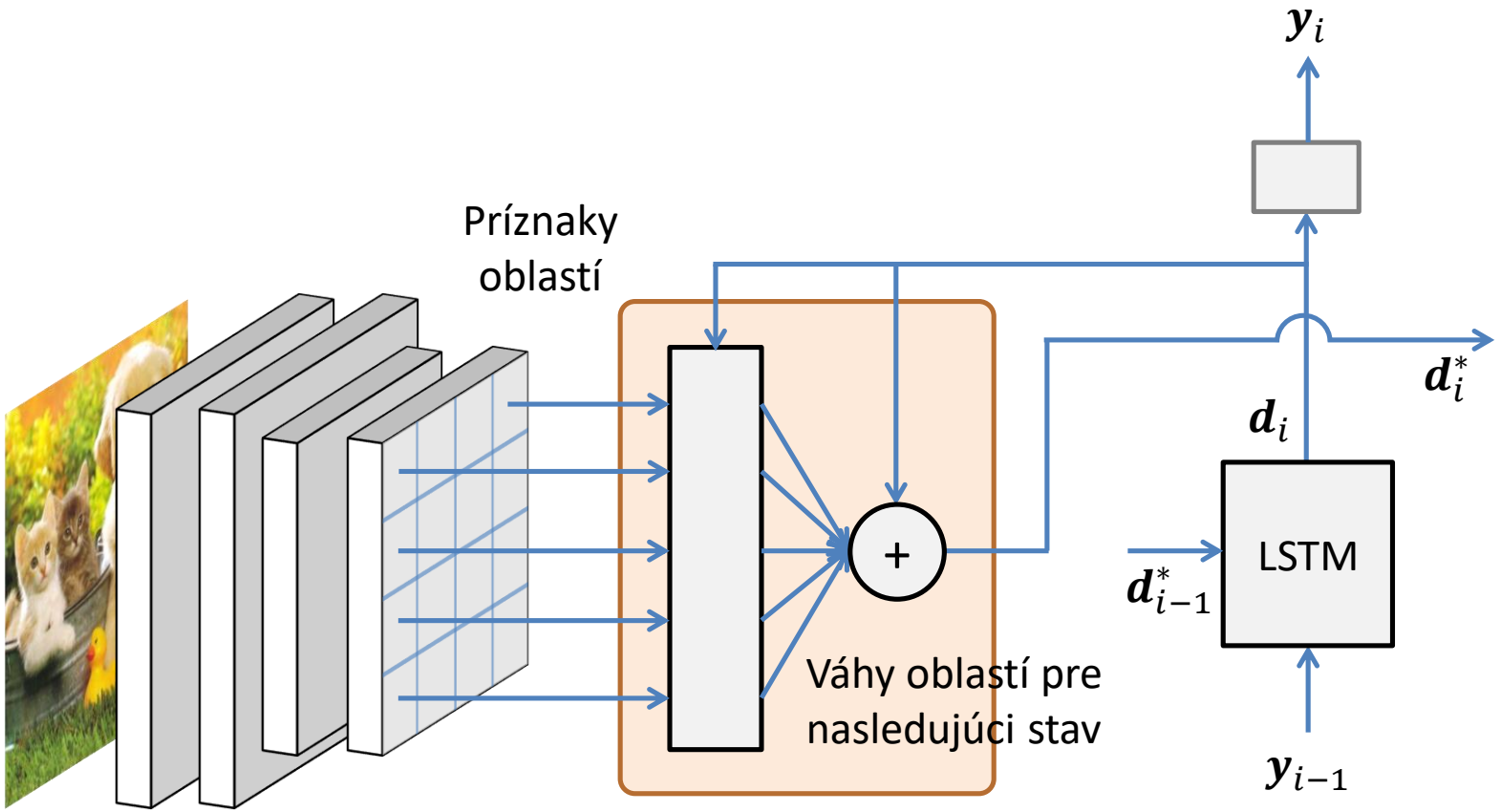
Titulkovanie obrázkov

- Úloha je vygenerovať pre zadaný obrázok titulok v prirodzenom jazyku, napr.:

"dve mačatá vo vedre vedľa psa a žltej kačičky."

- Mapovanie obrázka (priestorové dáta) na vetu (sekvenciu)
  - Kodér zakóduje obrázok – konvolučná sieť
  - Dekodér vygeneruje sekvenciu - LSTM

Konvolučné siete + vizuálna pozornosť + LSTM



GAN – Generative Adversarial Network

GAN (1)

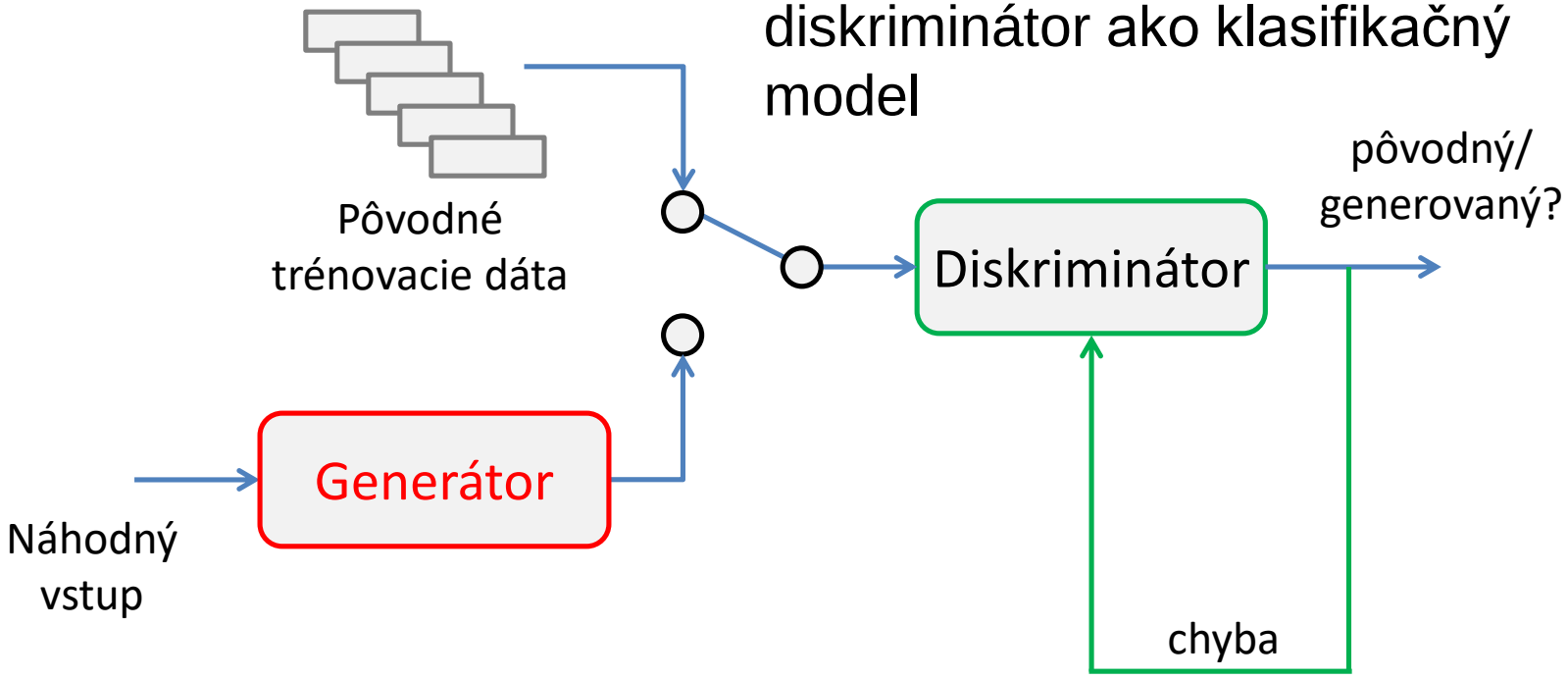
- Architektúra zložená z dvoch modelov
  - Generátor – generuje nové príklady čo najpodobnejšie skutočným príkladom z danej domény
  - Diskriminátor – klasifikačný model, ktorého úlohou je rozlíšiť či ide o skutočný, alebo vygenerovaný príklad
- Vznikla v kontexte semikontrolovaného učenia – diskriminátor je pred-trénovaný model pre klasifikáciu
  - ak model dokáže lepšie rozlíšiť „falošné“ príklady, zlepší sa aj jeho klasifikácia do tried

GAN (2)

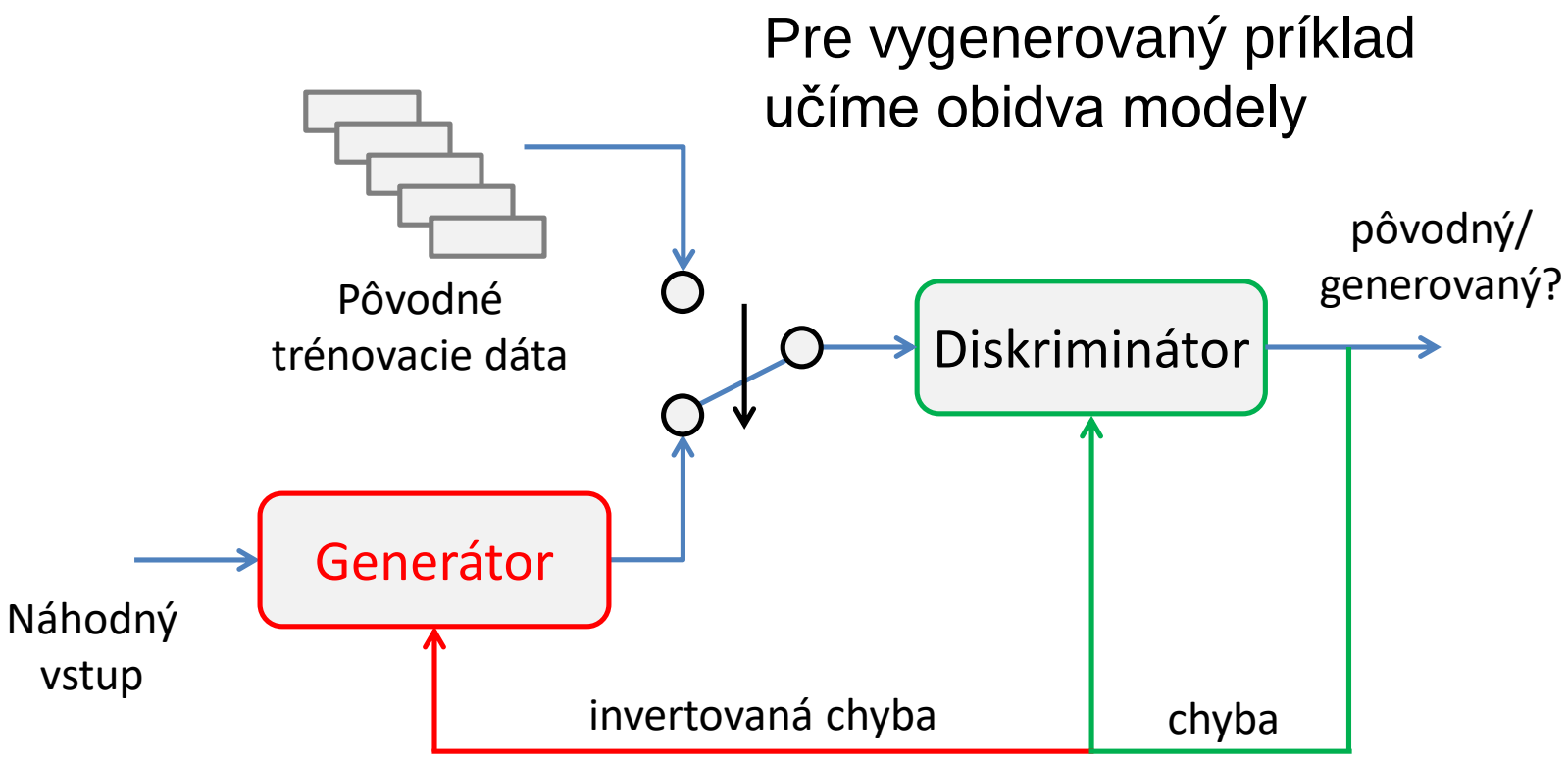
- Pri učení sa optimalizujú parametre oboch modelov:
  - Diskriminátor minimalizuje chybu klasifikácie
  - Generátor maximalizuje podobnosť medzi vygenerovanými a skutočnými príkladmi – pre diskriminátor ich potom bude ťažšie rozoznať
- Učenie prebieha ako hra dvoch hráčov s 0 súčtom až kým sa nedosiahne rovnováha

GAN – Architektúra (1)

Pre pôvodný príklad učíme iba diskriminátor ako klasifikačný model

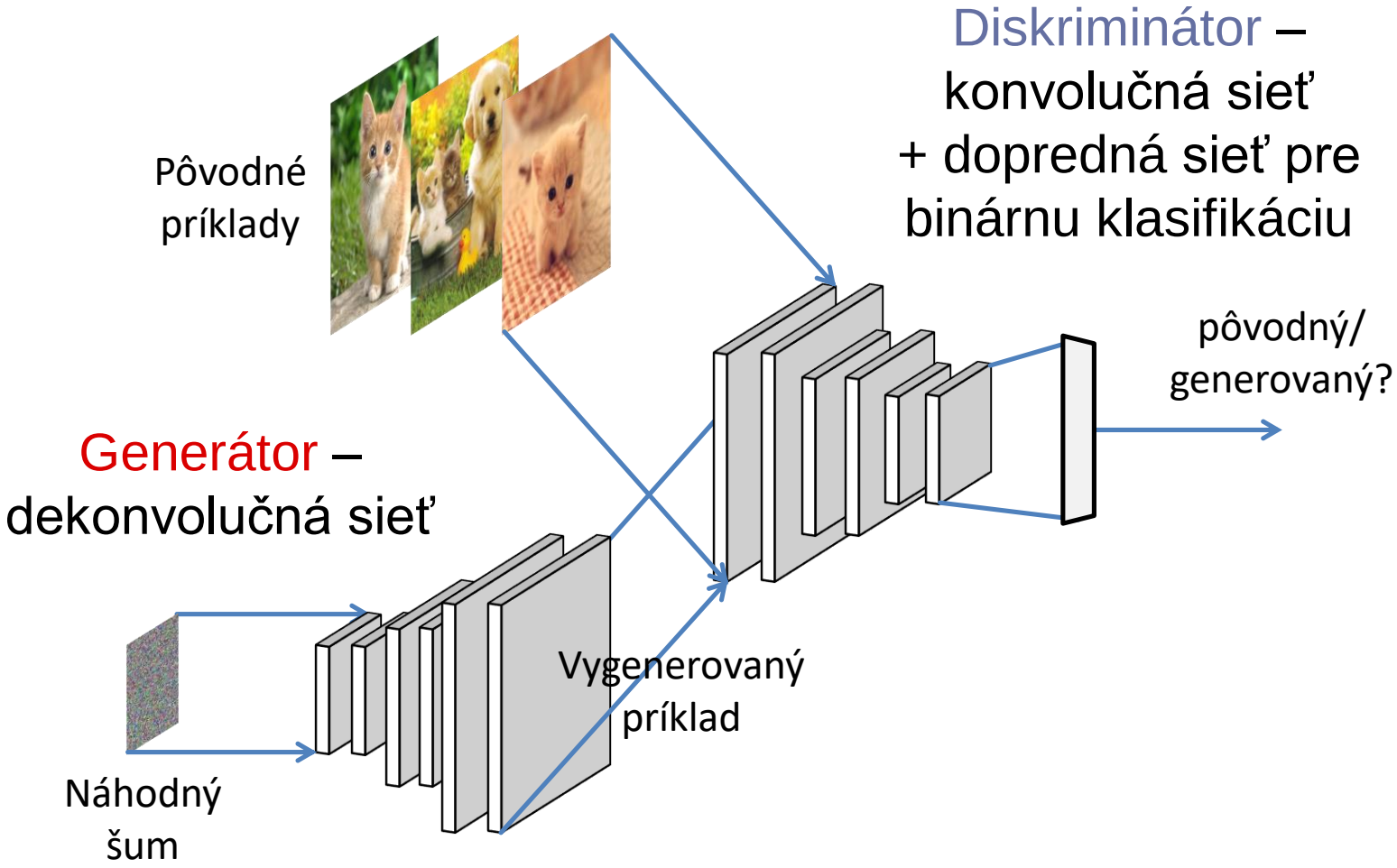


GAN – Architektúra (2)



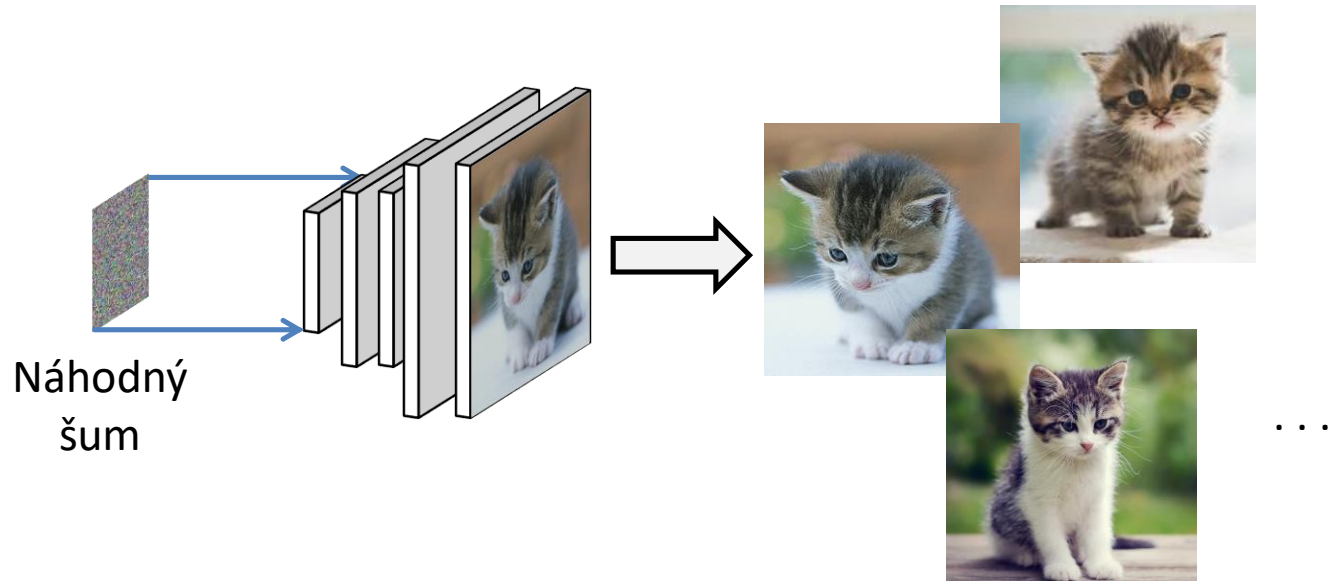
Pre generátor je chyba invertovaná – tzn. príklad bol chybný ak ho diskriminátor klasifikoval správne

GAN + obrazové dáta



GAN ako generátor dát

- Po naučení môžeme použiť samotný generátor na generovanie nových príkladov, ktoré budú na nerozoznanie od pôvodných dát



GAN ako generátor dát

2014

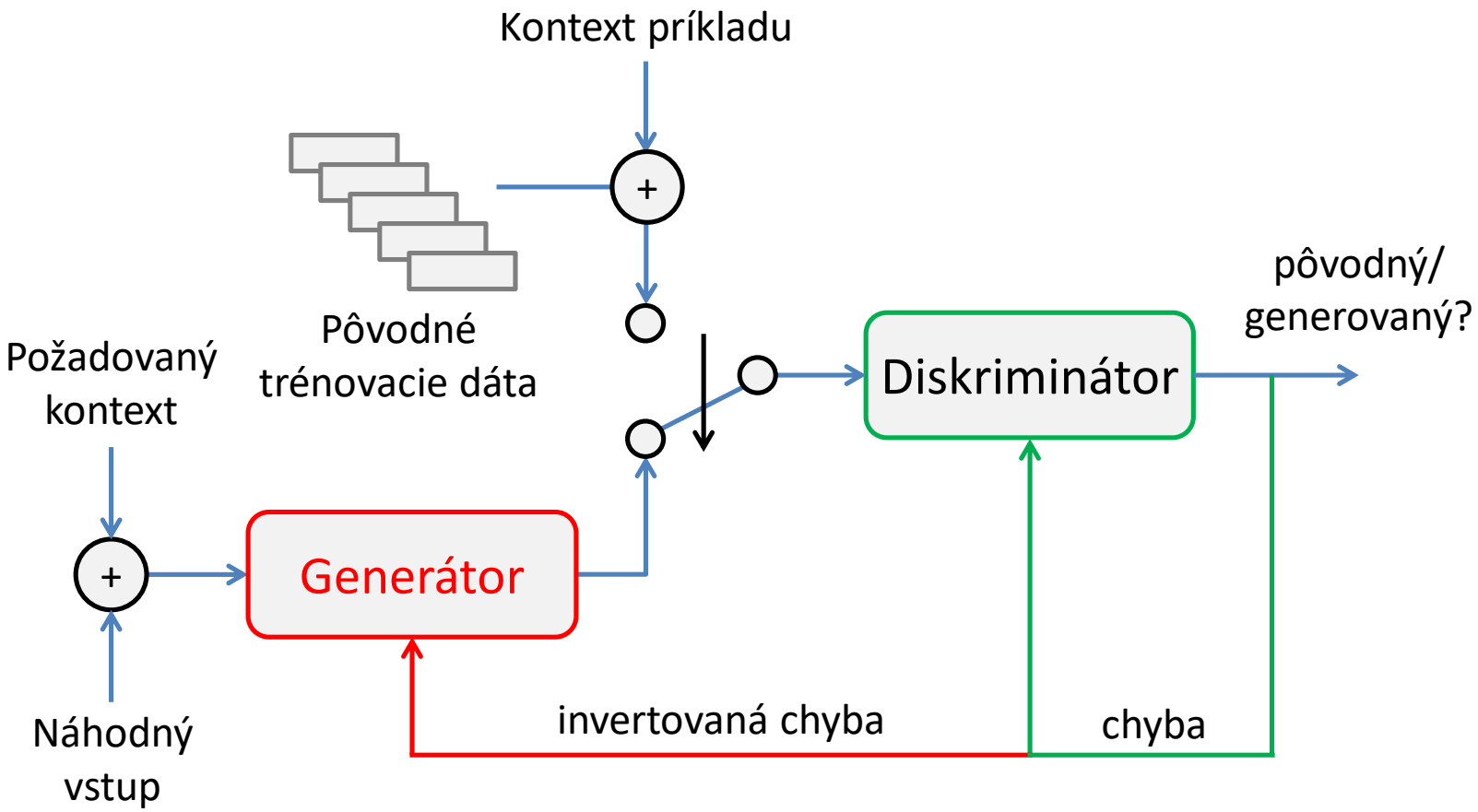
2015

2016

2017

Zdroj: https://arxiv.org/abs/1802.07228

GAN s ohraničením



GAN + obrazové dáta s ohraničením