nedelja informatike 3

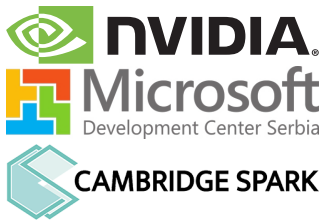
Neuralne mreže

Petar Veličković

Microsoft Development Center Serbia

Nedelja informatike 3

14. decembar 2016.



Roadmap

- Na ovom predavanju (i pratećoj Keras radionici) ćete se iz prve ruke upoznati sa *neuralnim mrežama*, trenutno jednim od najpopularnijih modela za (supervizirano) mašinsko učenje.
- Ovo smo zamislili kao predavanje iz šest delova...
  - Uvod u inteligentne sisteme;
  - Neuroni, neuralne mreže i višeslojni perceptroni;
  - Dizajn i treniranje neuralnih mreža;
  - Duboke neuralne mreže;
  - Konvolucijske neuralne mreže;
  - Regularizacija.
- ...praćeno radionicom na kojoj ćete moći *sami* da se oprobate u ovim tehnikama.
- Nenad i ja smo aktivni celog dana; slobodno razgovarajte sa nama o čemu god vas interesuje kod ovih tehnika!

NVIDIA Microsoft Development Center Serbia CAMBRIDGE SPARK

Motivacija: notMNIST

- Koja slova su na slici? (Kako ste to zaključili?)

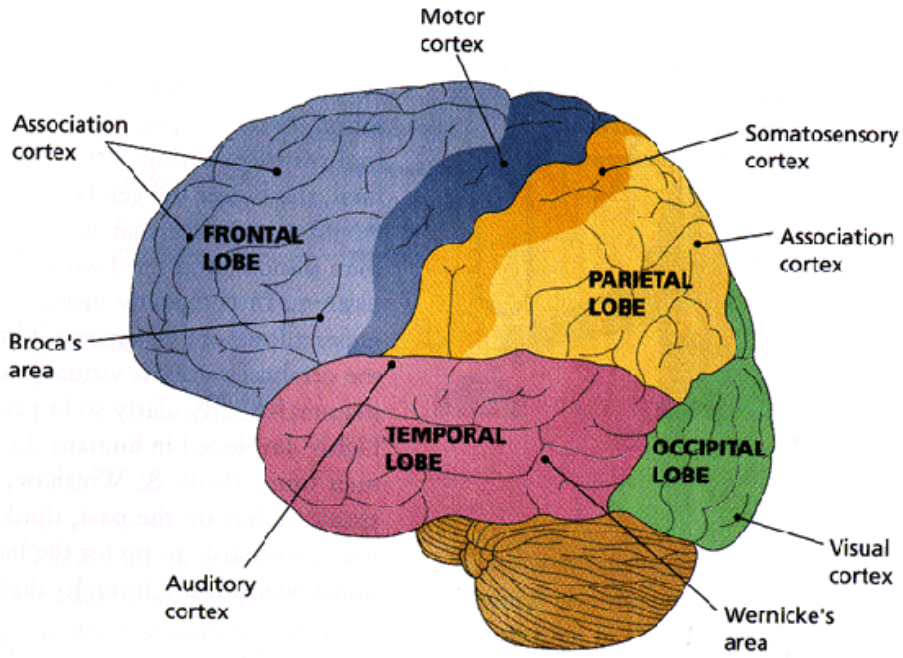
- Zamislite da vam je neko tražio program koji prepoznaje slova na glyph-ovima...kako biste pretvorili vaše rešenje u program?

Inteligentni sistemi

- Iako je vama (verovatno) prethodno pitanje bilo *jednostavno*, (verovatno) ne biste mogli da vaš proces razmišljanja jednostavno pretočite u niz instrukcija za program!
- Za razliku od *"glupog"* programa koji samo slepo izvršava preprogramirane instrukcije, vi ste tokom života bili izloženi velikom broju slova `A` i *"naučili"* kompleksna svojstva koja čine nešto slovom `A`!
- Želja da dizajniramo ovakve sisteme (koji su sposobni da *generalizuju* iz prethodnog iskustva) je upravo suština *mašinskog učenja*!
  - Koliko ovakvih sistema znate u prirodi?

Specijalizacija u mozgu

- Znamo da *različiti delovi* mozga rade *različite zadatke*:

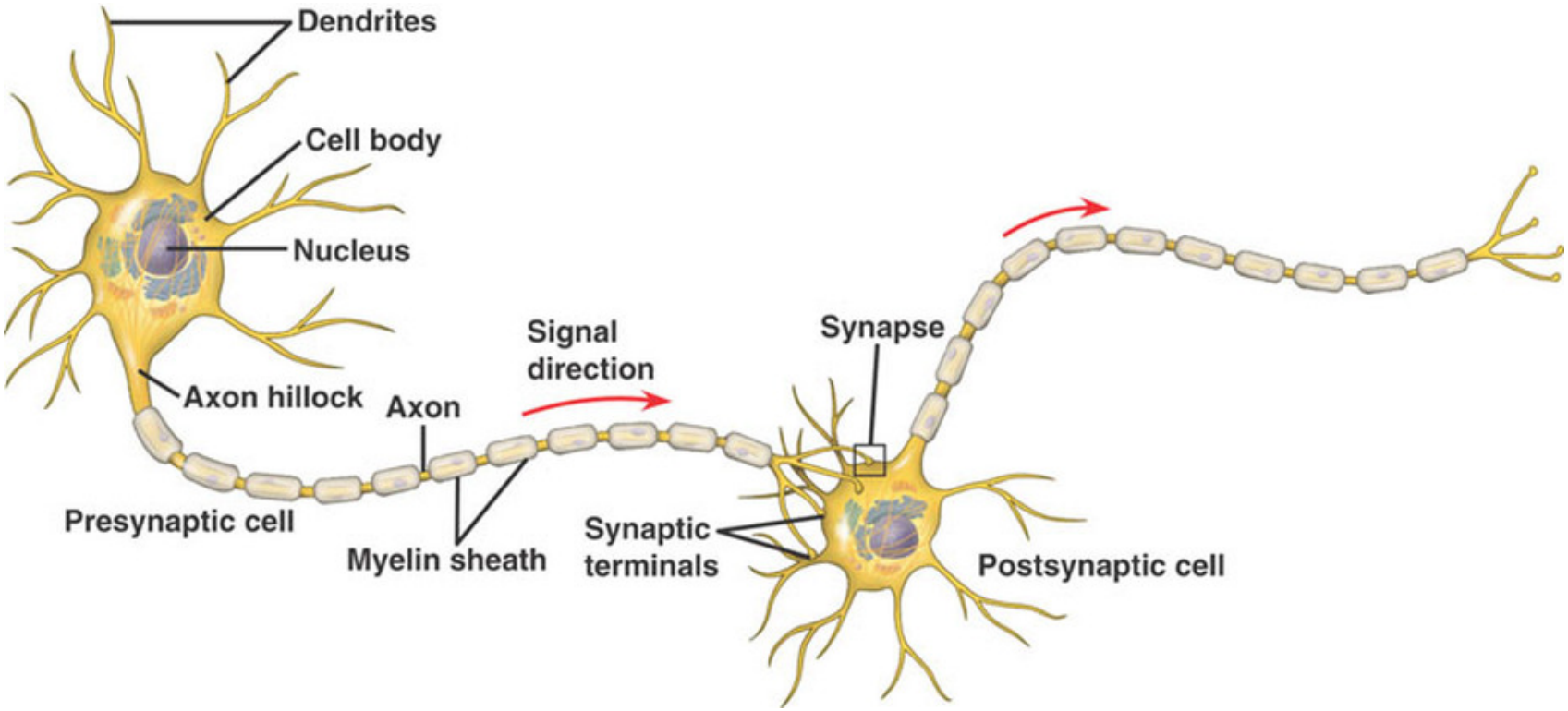


- Postoje sve veći dokazi da mozak:
  - Uči kad je *izložen podacima*;
  - *Nije* preprogramiran!

Mozak i podaci

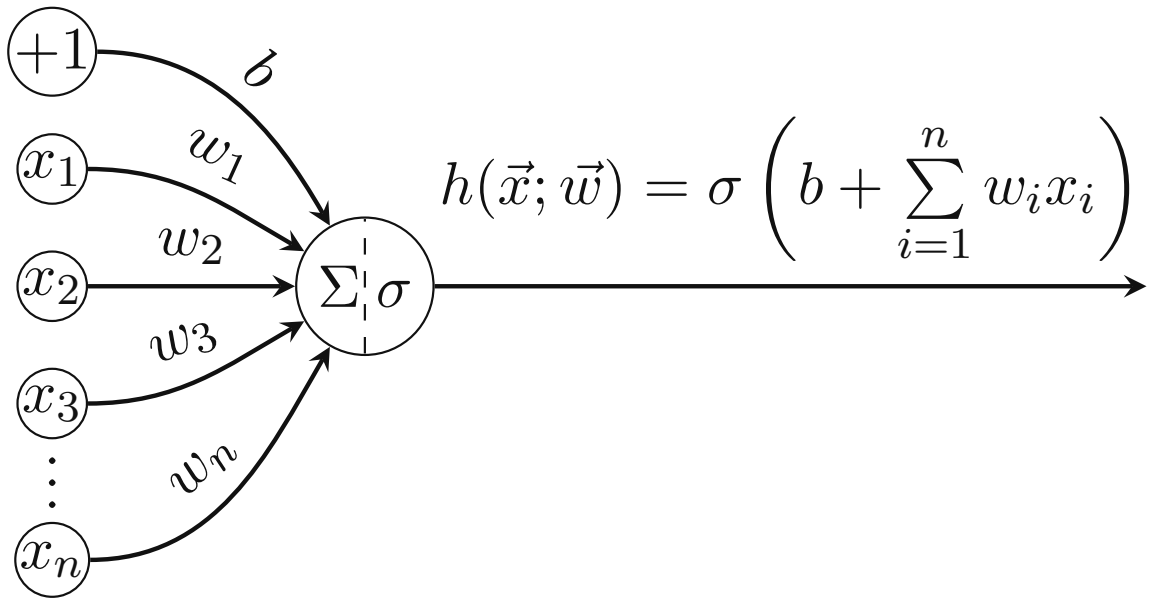
- Najveća znanja koja imamo o mozgu stižu od izučavanja *oštećenja mozga*:
  - Kod beba tvorova, ukoliko preusmerimo vizuelne ulaze u region koji normalno obrađuje zvuk, ovaj region će naučiti da procesira vizuelni ulaz!
  - Koliko (za sada) znamo, ovaj modifikovani region radi podjednako dobro kao i vizuelni korteks u zdravim tvorovima!
- Ukoliko nema velikih bioloških različitosti u obradi različitih tipova ulaza...
- $\Longrightarrow$ mozak verovatno koristi *jedinstven, generalan algoritam učenja* koji može da se prilagodi širokom spektru ulaza.

Jedan *stvaran* neuron!



Jedan *veštački* neuron!

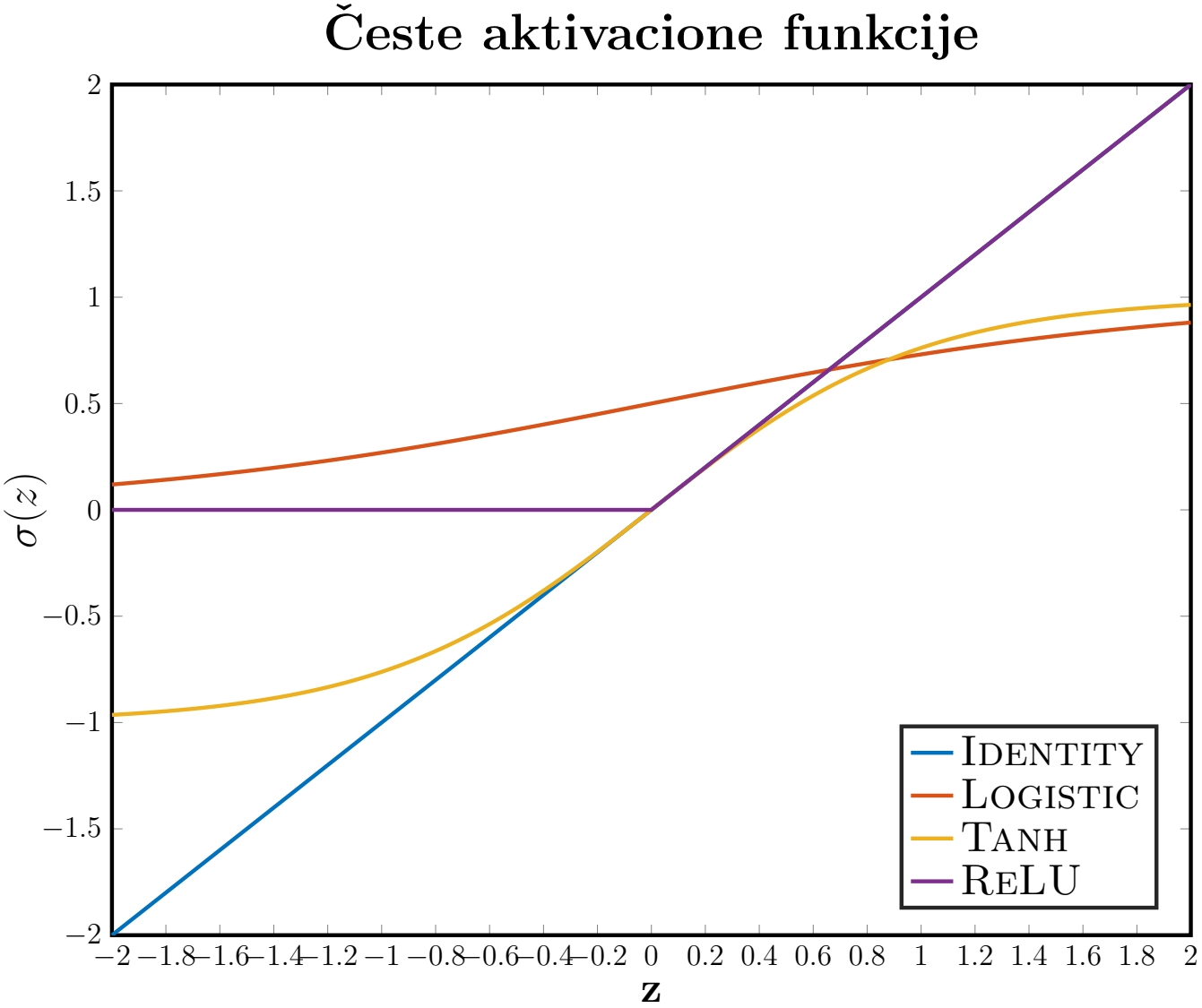
U ovom kontekstu često nazivan i *perceptronom* (...)



Česti izbori aktivacione funkcije σ:

- $\sigma(x) = x$ (*identitet*);
- $\sigma(x) = \max(0, x)$ (*ReLU*);
- $\sigma(x) = \frac{1}{1+\exp(-x)}$; $\sigma(x) = \tanh x$ (*sigmoidne funkcije*).

Aktivacione funkcije

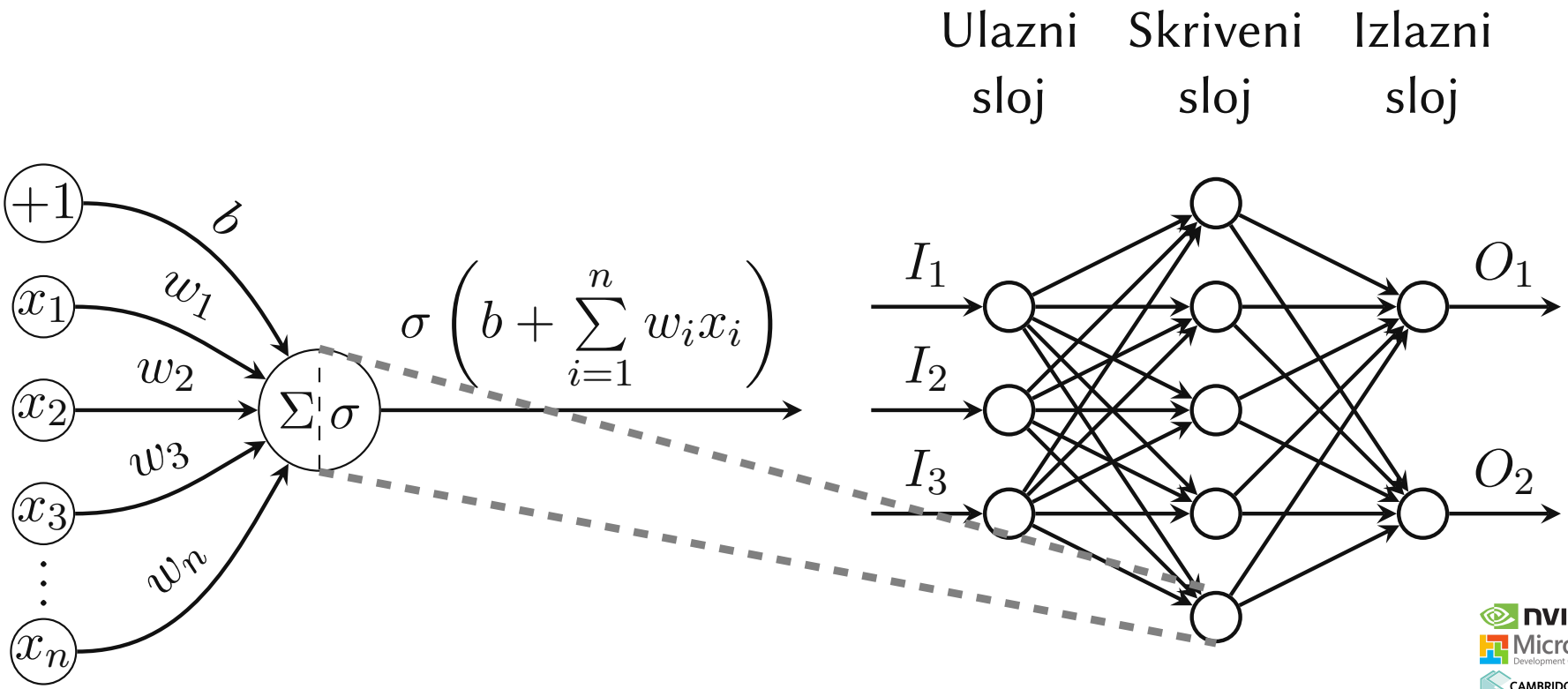


Neuralne mreže

- Lako je proširiti jedan neuron na *neuralnu mrežu—povezivanjem izlaza neurona sa ulazima drugih neurona* (uz specijalne *ulazne* i *izlazne* neurone za celu mrežu).
- Ovo možemo uraditi na dva načina:
  - **Aciklično** (*feedforward*): ne postoji *ciklus* u napravljenom grafu.
  - **Rekurentno** (*recurrent*): postoji ciklus u napravljenom grafu.
- Ovde ćemo se fokusirati *samo* na aciklične neuralne mreže:
  - Podesnije su za probleme kojima ćemo se baviti (∼*prepoznavanje slika*);
  - Lakše su za treniranje;
  - Nedavno se probijaju i u oblastima gde su istorijski dominirale rekurentne mreže (∼ *mašinsko prevođenje*)!

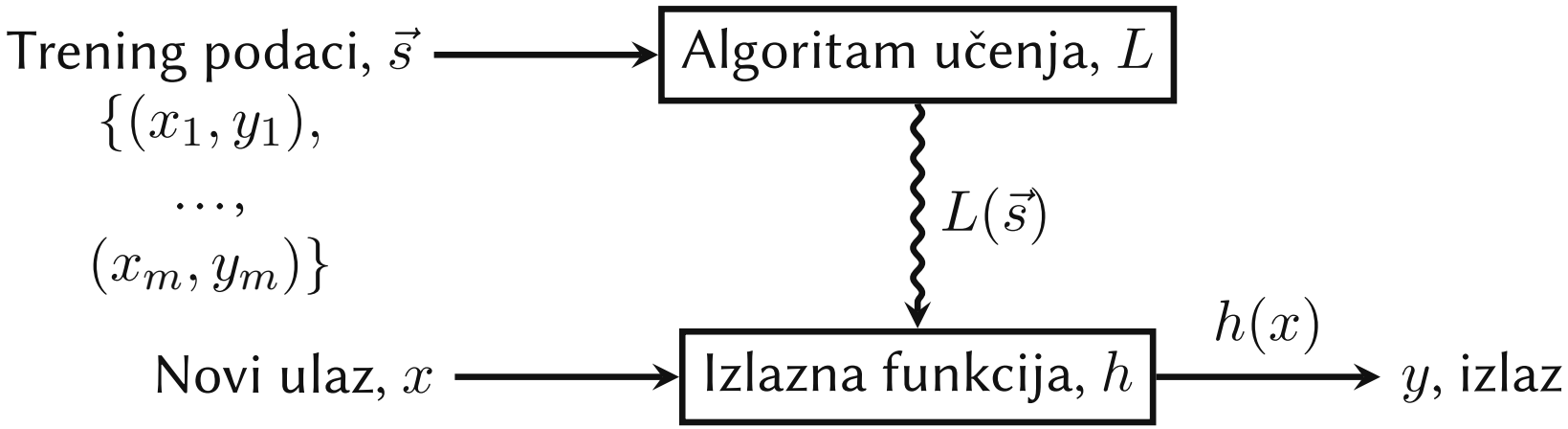
Višeslojni perceptroni

Aciklična arhitektura sa najvećim kapacitetom dozvoljava potpunu povezanost između susednih slojeva—ponekad nazivana i **višeslojnim perceptronom** (*multilayer perceptron*—MLP).



Supervizirano učenje

- Neuralne mreže su dizajnirane za probleme *superviziranog učenja*, gde je zadatak napraviti *algoritam učenja* koji će, na osnovu skupa ulaznih podataka sa poznatim izlazima (*trening primerima*), proizvesti funkciju sposobnu da dodeli izlaze do tada *neviđenim podacima* (*test primerima*).



Klasifikacija

- Najjednostavniji oblik superviziranog učenja je *klasifikacija*; za datih k klasa, odrediti kojoj klasi pripada ulazni podatak x.
  - Ovo je možda ujedno i *najvažniji* oblik superviziranog učenja—ako znamo dobro da klasifikujemo, možemo mnoge druge probleme (regresiju, segmentaciju, učenje sa pojačavanjem…) pretvoriti u ovakav.
- Da bismo modelirali ovaj problem sa neuralnim mrežama, napravićemo jedan izlazni neuron za svaku klasu, i odabrati onu klasu čiji neuron da *najveći* izlaz.
- **Problem:** nema jasnog *cilja* za algoritam treniranja: vrednosti mogu arbitrarno da rastu/opadaju!

Softmaks

- Da bismo postavili jasan *cilj* za algoritam treniranja, primenjujemo *softmaks* aktivaciju na izlaznim neuronima:

$$\sigma(\vec{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

  gde je z_i izlaz neurona vezanog za klasu i.

- Ova funkcija je *monotona* i sabija vrednosti u opseg $[0, 1]$, tako da se sumiraju u 1—efektivno, $\sigma(\vec{z})_i = \mathbb{P}(\textit{klasa}\ i)$.

- Tada lako možemo definisati i cilj, tj. *očekivani izlaz* mreže za neki (trening) objekat klase i. To je *"one-hot"* vektor veličine k, u kome je samo i-ti član 1, a svi ostali 0.

$$\vec{\hat{y}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$$

Funkcija gubitka

- Treniranje neuralnih mreža podrazumeva iterativno ažuriranje težina ($\vec{w}$ i $\vec{b}$ vrednosti) u mreži, tako da se minimizira neka *funkcija gubitka*, $\mathcal{L}(\vec{w})$, nad primerima iz trening skupa.
- Ukoliko ne znamo mnogo o traženom problemu, često je najbolje pribeći funkciji *kvadratnog odstupanja*. Za dati trening primer $(x, \hat{y})$:

$$\mathcal{L}(\vec{w}) = (\hat{y} - h(\vec{x}; \vec{w}))^2$$

- Međutim, ukoliko radimo klasifikaciju sa softmaks izlazima, prikladnije je koristiti *unakrsnu entropiju* (za detaljnije izvođenje konsultujte booklet!). Za dati trening primer (x, c), sa one-hot vektorom $\vec{\hat{y}}$ i izlazom mreže $\vec{y} = h(\vec{x}; \vec{w})$:

$$\mathcal{L}(\vec{y}, \vec{\hat{y}}) = -\sum_{i=1}^{k} \hat{y}_i \log y_i$$

Gradijentni spust

- Težine treniramo algoritmom gradijentnog spusta (*gradient descent*)—krenuvši od nasumičnih* težina, u svakom koraku ažuriramo težine u smeru najbržeg pada funkcije gubitka (tj. njenog negativnog gradijenta):

$$\vec{w} \leftarrow \vec{w} - \eta \sum_{p=1}^{m} \nabla \mathcal{L}_p(\vec{w})$$

  gde je $\nabla \mathcal{L}_p(\vec{w})$ gradijent funkcije gubitka za trening primer (x_p, y_p), a η "brzina učenja" (*learning rate*).

- Gradijent računamo *propagirajući* gubitke na izlaznim neuronima *unazad* ka ulaznim (*backpropagation*)—ovo je već implementirano za nas u Kerasu!

Gradijentni spust *(& friends)*

- Ako radimo sa *velikim* skupom podataka, sabiranje gradijenata za sve trening primere odjednom je:
  - *teško (numerički) kontrolisati*;
  - *previše korelacije između zasebnih iteracija (uvek "isti" primeri).*
- Ove probleme rešavamo *stohastičnim gradijentnim spustom*—u jednom koraku ažuriramo po gradijentu u *samo jednom* (nasumičnom) trening primeru $(x_{p'}, y_{p'})$:

$$\vec{w} \leftarrow \vec{w} - \eta \nabla \mathcal{L}_{p'}(\vec{w})$$

- U praksi, koristimo "zlatnu sredinu"—u jednom koraku ažuriramo po nasumično odabranoj grupi (*minibatch*) trening primera $\mathcal{B} = \{(x_1, y_1), \ldots, (x_{bs}, y_{bs})\}$.

$$\vec{w} \leftarrow \vec{w} - \eta \sum_{p=1}^{bs} \nabla \mathcal{L}_p(\vec{w})$$

Mistična η

- Korektan odabir vrednosti η za svaku iteraciju algoritma je **kritičan** za uspešne rezultate:
  - Preveliko η—beskonačno oscilovanje...
  - Premalo η—beskonačno iteracija do konvergencije...

- U praksi, imamo algoritme koji ovo mogu da rade za nas—na osnovu *istorijskih vrednosti gradijenata* automatski računaju šta bi trebalo da bude dobra vrednost η_t u svakom trenutku t.

- Preporučujemo *Adam* i *RMSProp* algoritme, koji obično daju sasvim dobre rezultate (i oba su implementirana u Keras-u).

Hiperparametri

- Gradijentni spust optimizuje samo težine i *bias*-e—a šta je sa:
  - brojem skrivenih slojeva?
  - brojem neurona u svakom skrivenom sloju?
  - aktivacijama ovih neurona?
  - brojem iteracija treninga?
  - veličinom jednog minibatch-a (bs)?
  - ...
- Ovi parametri *moraju biti određeni pre nego što trening zapravo počne*! Zato ih nazivamo **hiperparametrima**.
- Njihova optimizacija je *jako težak problem*—ne znamo za pametnije rešenje od odvajanja određenog podskupa trening podataka (*validacijski skup*) radi evaluacije različitih kombinacija hiperparametara.

Duboke neuralne mreže

- Zadržao bih se na jednom specifičnom hiperparametru: *broju skrivenih slojeva*, tj. *dubini mreže*.
- Šta mislite, koliko skrivenih slojeva je *dovoljno* da možemo naučiti bilo koju (neprekidnu i ograničenu) realnu funkciju?

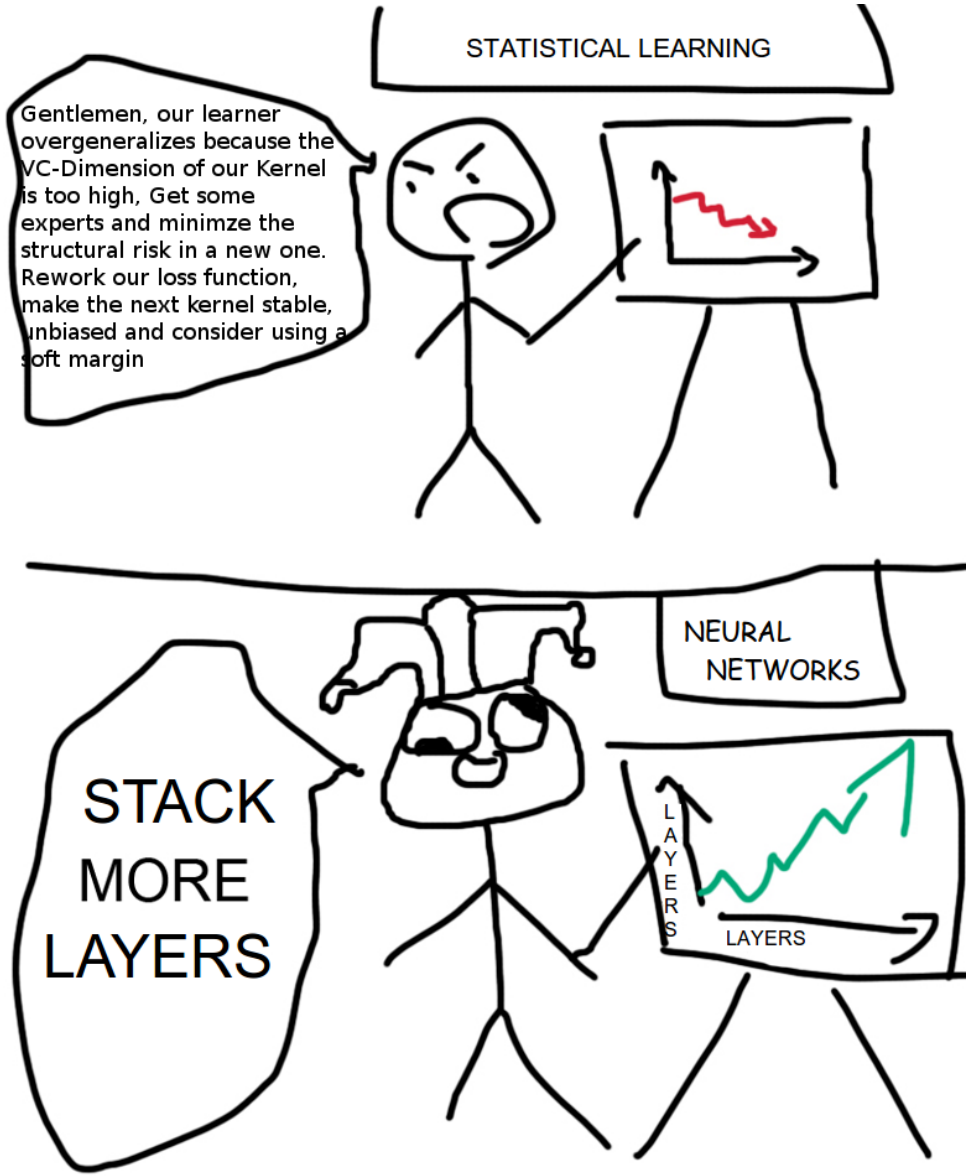
Duboke neuralne mreže

- Zadržao bih se na jednom specifičnom hiperparametru: *broju skrivenih slojeva*, tj. *dubini mreže*.
- Šta mislite, koliko skrivenih slojeva je *dovoljno* da možemo naučiti bilo koju (neprekidnu i ograničenu) realnu funkciju?
  - Jedan! (*Cybenkova teorema*, 1989.)
  - Za ljubitelje matematike među vama: `http://deeplearning.cs.cmu.edu/pdfs/Cybenko.pdf`

Duboke neuralne mreže

- Zadržao bih se na jednom specifičnom hiperparametru: *broju skrivenih slojeva*, tj. *dubini mreže*.
- Šta mislite, koliko skrivenih slojeva je *dovoljno* da možemo naučiti bilo koju (neprekidnu i ograničenu) realnu funkciju?
  - Jedan! (*Cybenkova teorema*, 1989.)
  - Za ljubitelje matematike među vama: `http://deeplearning.cs.cmu.edu/pdfs/Cybenko.pdf`
- Međutim, dokaz ove teoreme *nije konstruktivan*, tj. ne daje nam ni optimalnu širinu ovog sloja, a ni algoritam za treniranje.
- *We must go deeper...*
  - Svaka mreža sa > 1 skrivenim slojem se smatra *dubokom*!
  - Današnje *state-of-the-art* mreže imaju preko 150 slojeva.

Duboko mašinsko učenje



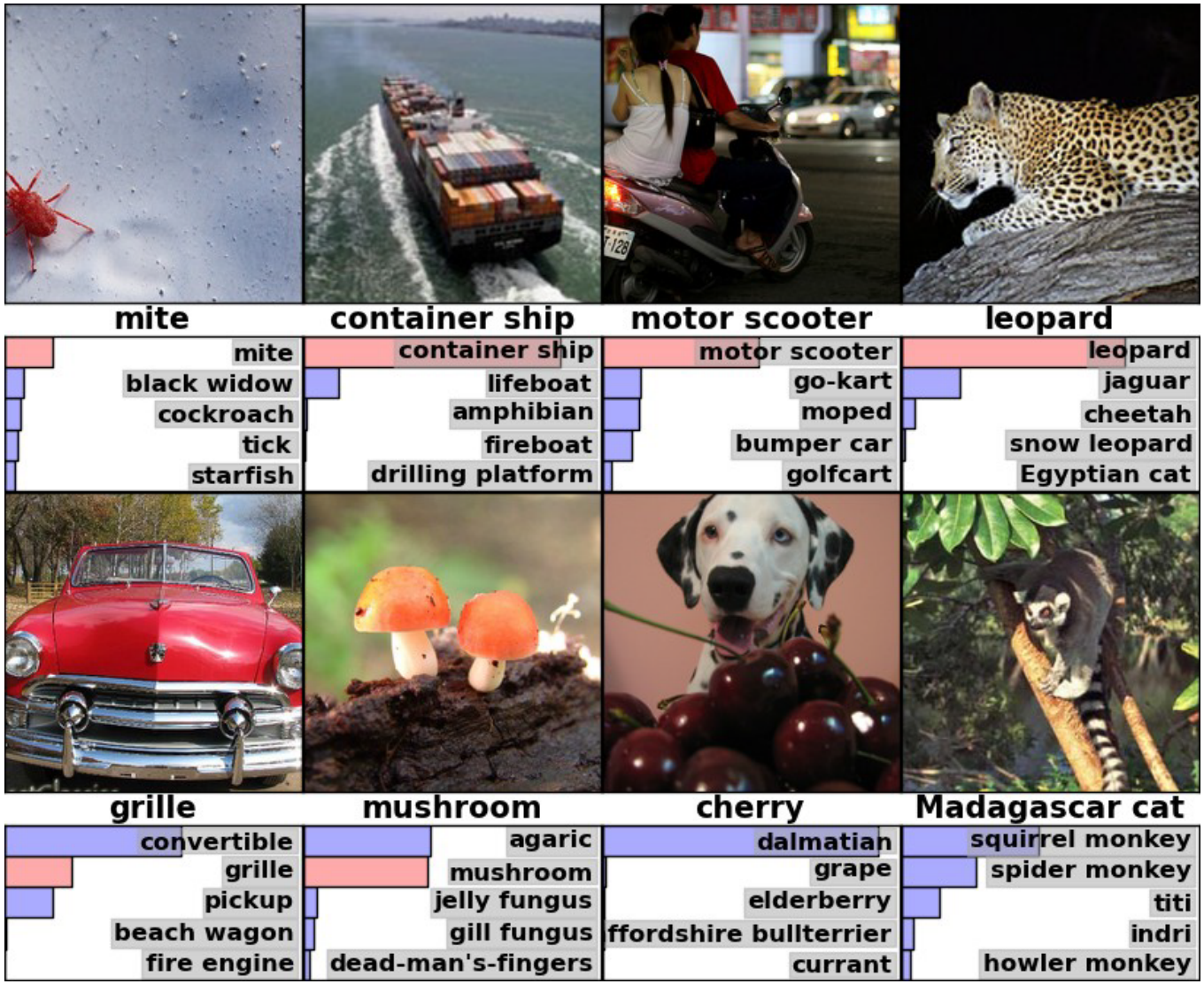
Feature engineering

- Istorijski, problemi mašinskog učenja su rešavani tako što se najpre definiše skup *karakterističnih svojstava*, koji se onda izvlače iz sirovih podataka i daju kao ulaz za *"plitke"* modele.
  - Mnogi naučnici su *gradili čitave doktorate* fokusirajući se samo na jednom ovakvom problemu!
  - Generalizacija—izuzetno mala (neretko nula)!
- Kod dubokog učenja, mreža *sama uči* najbolja svojstva za dati problem iz *sirovih podataka*!
  - Po prvi put spojila istraživače iz, do tada, praktično nespojivih oblasti, npr. *obrade prirodnog jezika* i *računarske vizije*.
  - $\Longrightarrow$ osoba sposobna za rad sa dubokim neuralnim mrežama može direktno da primeni svoje znanje za kreiranje najjačih modela u praktično svakom domenu*!

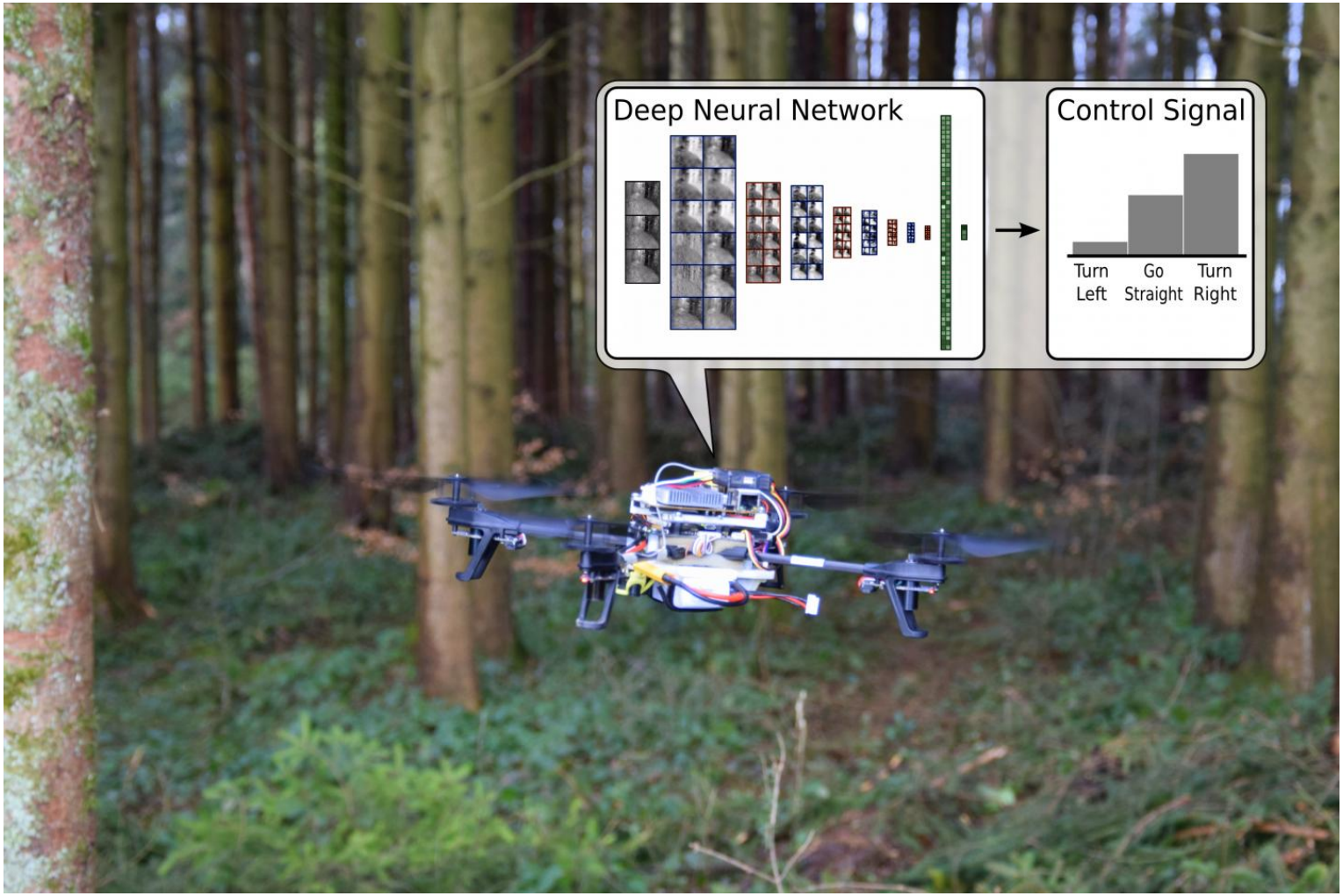
Kviz: šta je na slikama?

Klasifikacija saobraćajnih znakova

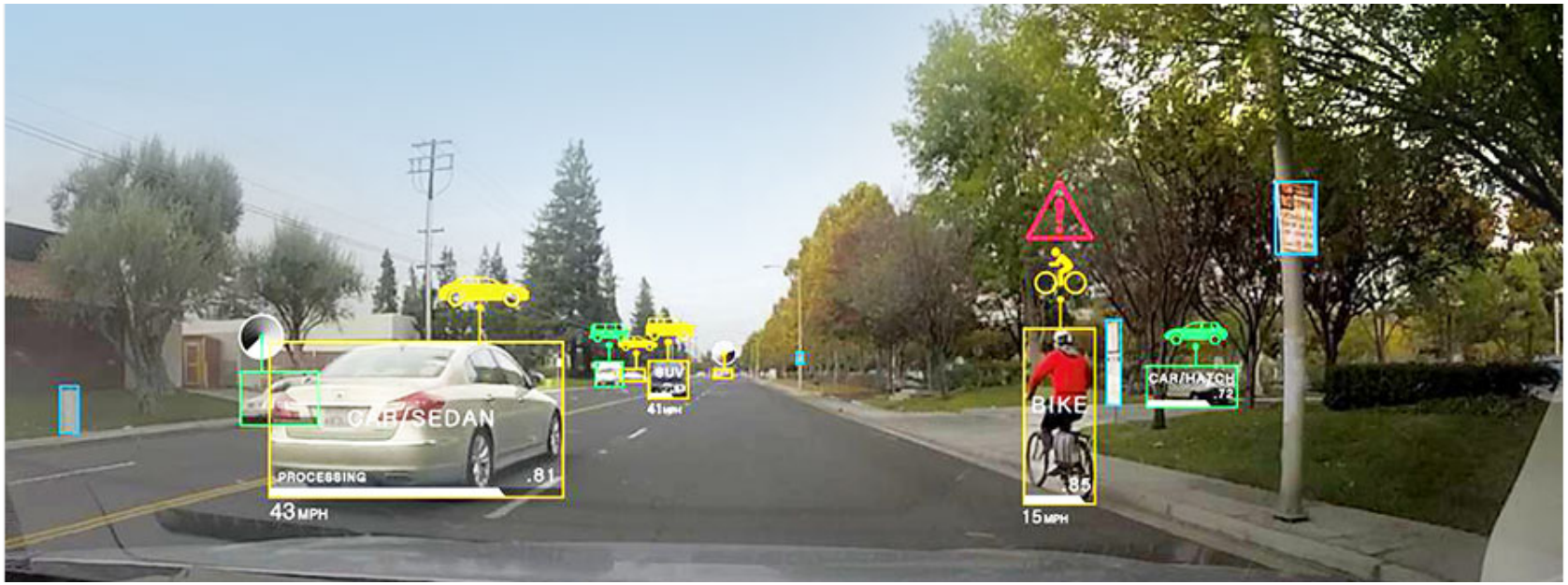
ImageNet klasifikacija



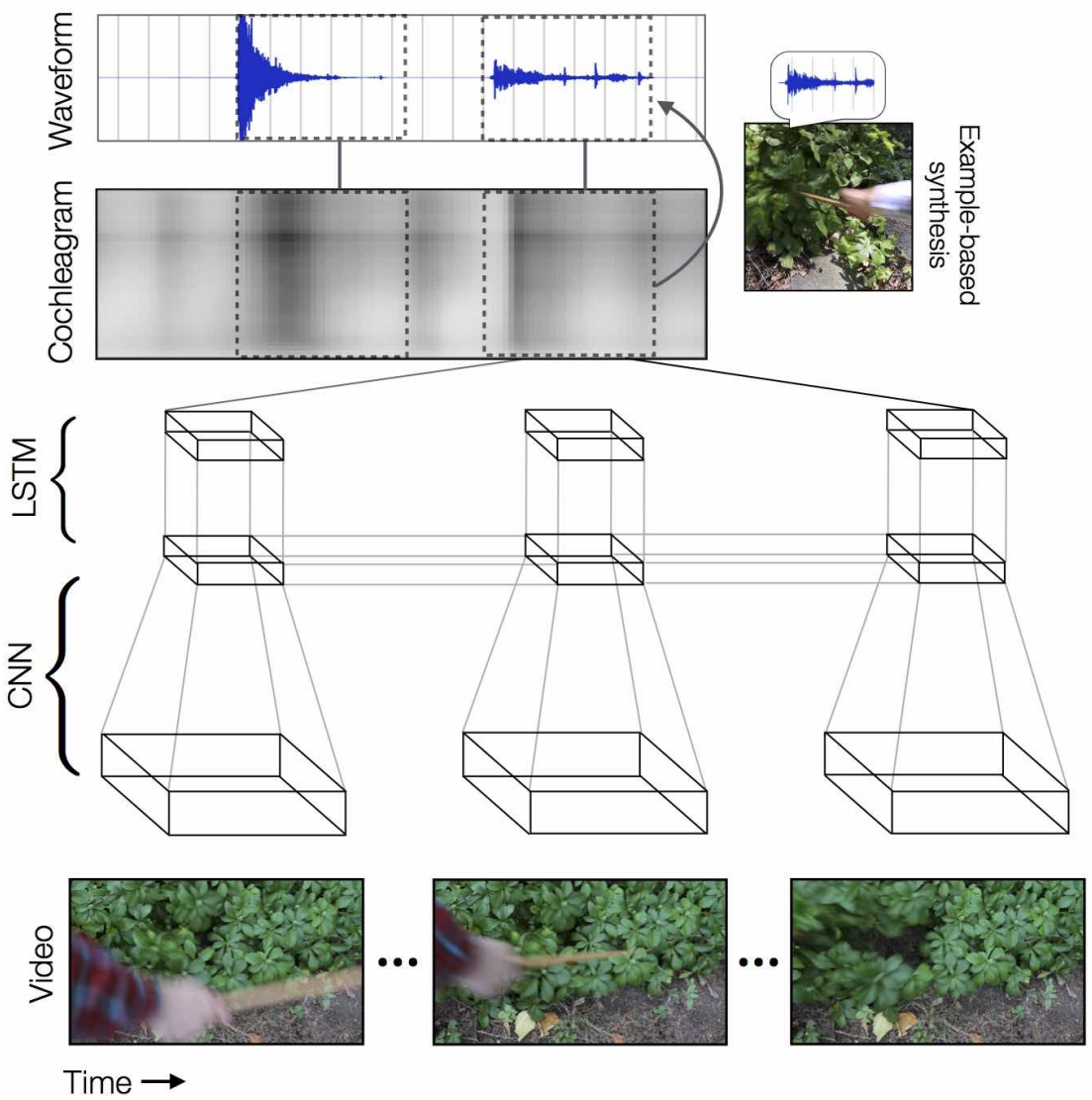
“Jednostavna” navigacija



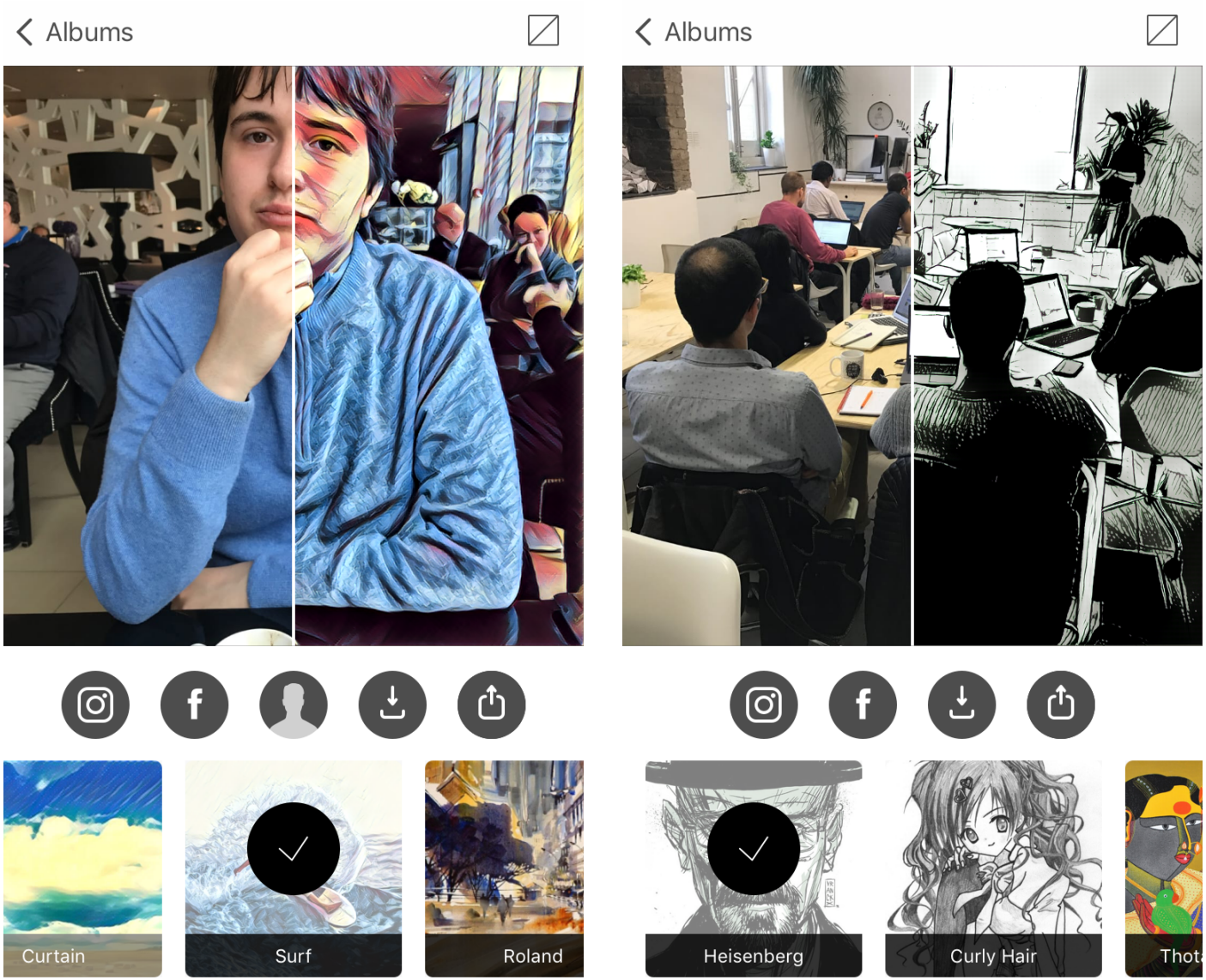
Samovozeći automobili



"Greatest hits"

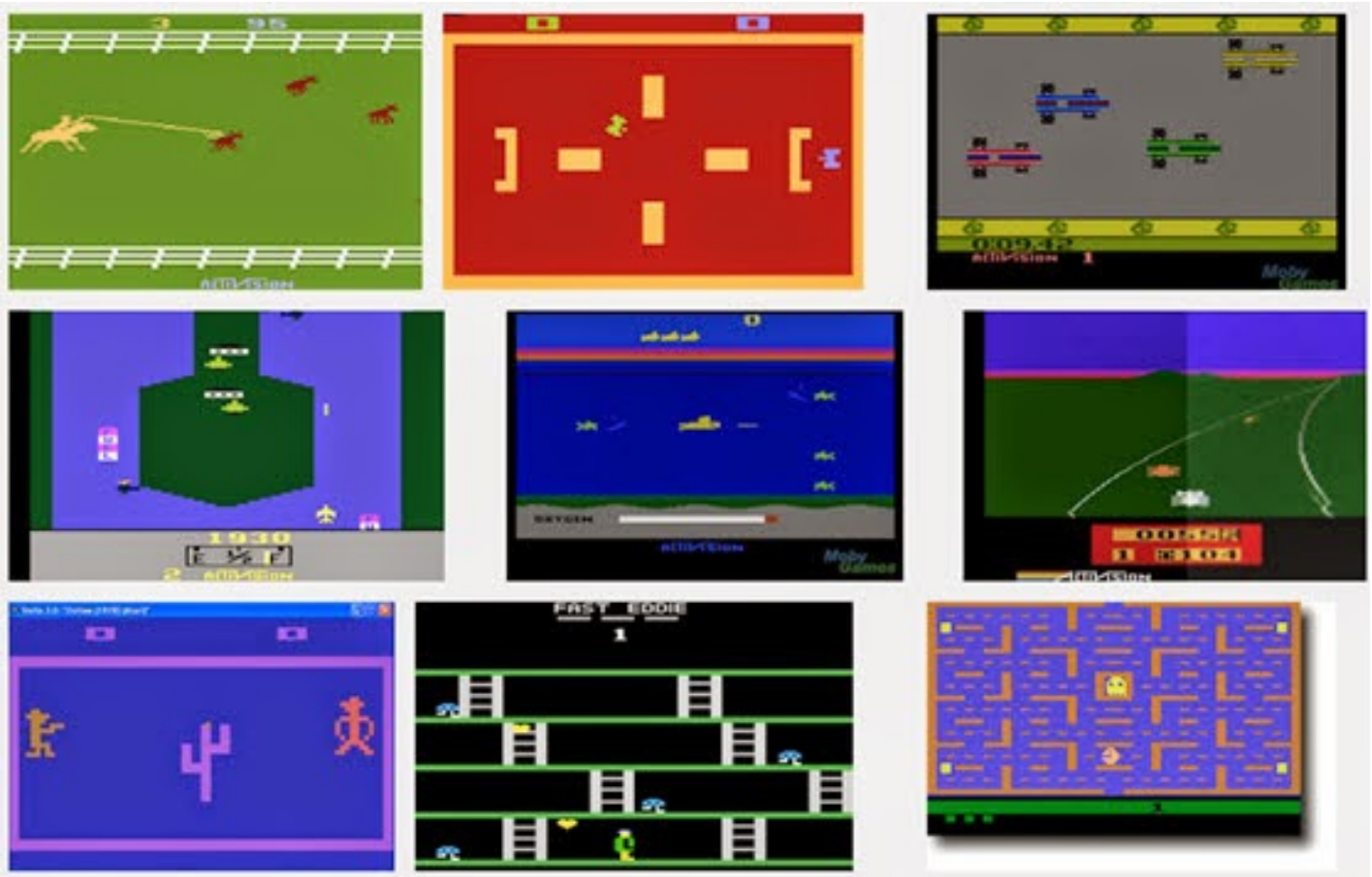


Neuralni transfer stila (*Prisma*)

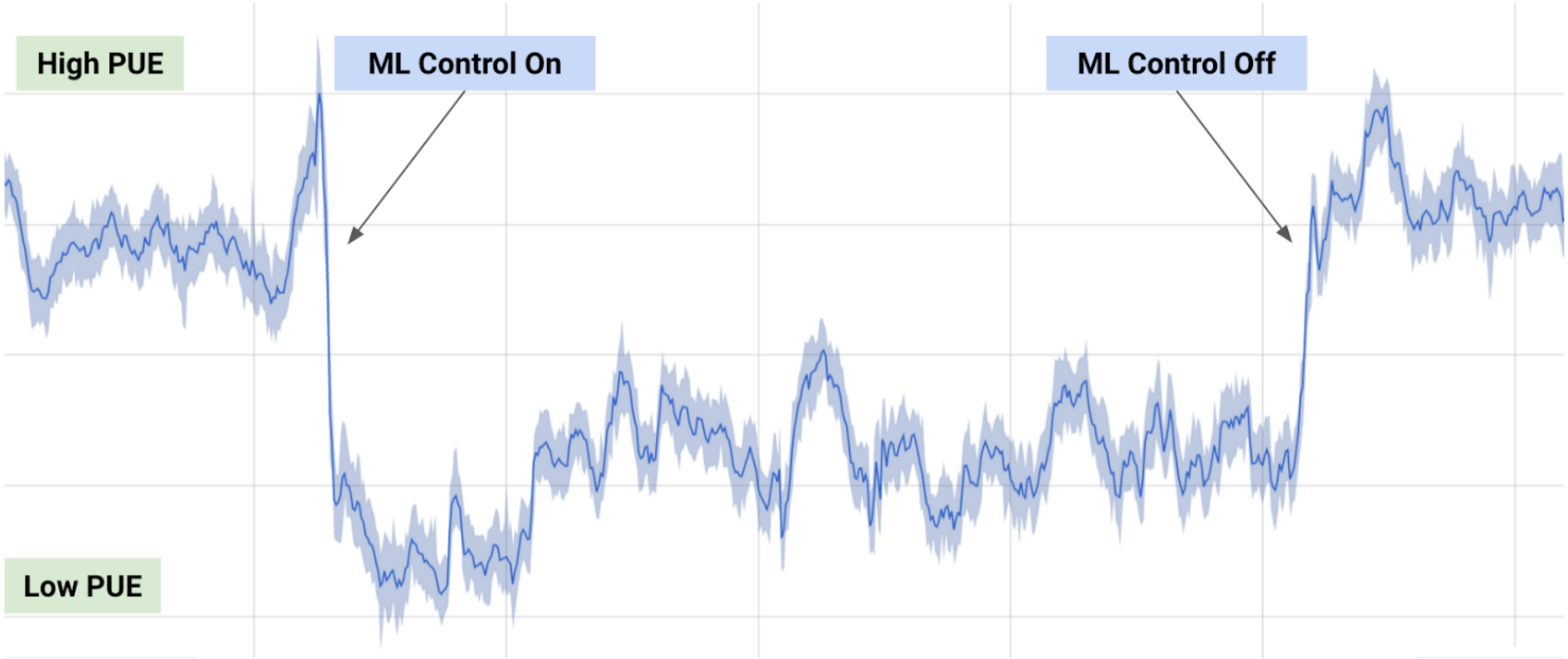


Opisivanje filma

Igranje Atarija...



. . . i hlađenje



. . . u praktično *svakom* domenu*

- Zašto duboko mašinsko učenje nije eksplodiralo ranije? Za neuralne mreže smo znali decenijama (i bile su izuzetno popularne u 90-tim. . .)

. . . u praktično *svakom* domenu*

- Zašto duboko mašinsko učenje nije eksplodiralo ranije? Za neuralne mreže smo znali decenijama (i bile su izuzetno popularne u 90-tim. . .)
  - *zahtevni gejmeri* ($\sim$ razvoj GPUova!)

NVIDIA. Microsoft Development Center Serbia CAMBRIDGE SPARK

. . . u praktično *svakom* domenu*

- Zašto duboko mašinsko učenje nije eksplodiralo ranije? Za neuralne mreže smo znali decenijama (i bile su izuzetno popularne u 90-tim. . .)
  - *zahtevni gejmeri* (∼ razvoj GPUova!)
  - *velike firme* (∼ mnogo resursa!)

NVIDIA
Microsoft Development Center Serbia
CAMBRIDGE SPARK

. . . u praktično *svakom* domenu*

- Zašto duboko mašinsko učenje nije eksplodiralo ranije? Za neuralne mreže smo znali decenijama (i bile su izuzetno popularne u 90-tim. . .)
  - *zahtevni gejmeri* (∼ razvoj GPUova!)
  - *velike firme* (∼ mnogo resursa!)
  - *big data* (∼ za veliki broj parametara!)

NVIDIA
Microsoft Development Center Serbia
CAMBRIDGE SPARK

. . . u praktično *svakom* domenu*

- Zašto duboko mašinsko učenje nije eksplodiralo ranije? Za neuralne mreže smo znali decenijama (i bile su izuzetno popularne u 90-tim. . .)
  - *zahtevni gejmeri* (∼ razvoj GPUova!)
  - *velike firme* (∼ mnogo resursa!)
  - *big data* (∼ za veliki broj parametara!)

- Duboko učenje je, zato, jako efektivno ako ste velika firma sa mnogo GPU servera i možete da zaposlite 10000 ljudi samo da vam naprave veći trening skup za dati problem. . .

- Notorna oblast gde ovo često **nije** moguće—*zdravstvo*! *(big data in the wrong dimension. . .)*

NVIDIA
Microsoft Development Center Serbia
CAMBRIDGE SPARK

Motivacija: notMNIST

$28 \times 28 \times 1$ slike, 10 klasa (A–J)

$85.5\% \longrightarrow ???$

Rad sa slikama

- Iako su višeslojni perceptroni teoretski najjače neuralne mreže, uglavnom su neuspešni na visoko-dimenzionalnim podacima (npr. *slikama*).
  - Ako tretiramo svaki piksel kao nezavisan ulaz...

Rad sa slikama

- Iako su višeslojni perceptroni teoretski najjače neuralne mreže, uglavnom su neuspešni na visoko-dimenzionalnim podacima (npr. *slikama*).
  - Ako tretiramo svaki piksel kao nezavisan ulaz...
  - ...dobijamo $h \times w \times d$ novih parametara po neuronu u prvom skrivenom sloju...

Rad sa slikama

- Iako su višeslojni perceptroni teoretski najjače neuralne mreže, uglavnom su neuspešni na visoko-dimenzionalnim podacima (npr. *slikama*).
  - Ako tretiramo svaki piksel kao nezavisan ulaz...
  - ...dobijamo $h \times w \times d$ novih parametara po neuronu u prvom skrivenom sloju...
  - ...što brzo eksplodira sa veličinom slika—zahtevajući daleko više podataka da bi se ti parametri pravilno naučili!

Rad sa slikama

- Iako su višeslojni perceptroni teoretski najjače neuralne mreže, uglavnom su neuspešni na visoko-dimenzionalnim podacima (npr. *slikama*).
  - Ako tretiramo svaki piksel kao nezavisan ulaz...
  - ...dobijamo $h \times w \times d$ novih parametara po neuronu u prvom skrivenom sloju...
  - ...što brzo eksplodira sa veličinom slika—zahtevajući daleko više podataka da bi se ti parametri pravilno naučili!
- **Ključna ideja:** smanjiti sliku dok ne postane dovoljno mala da možemo primeniti višeslojni perceptron nad njom!
  - Idealno bismo želeli da prvo izvučemo neka interesantna svojstva slike...
- $\Longrightarrow$ iskoristićemo *prostornu strukturu* slike!

Konvolucija

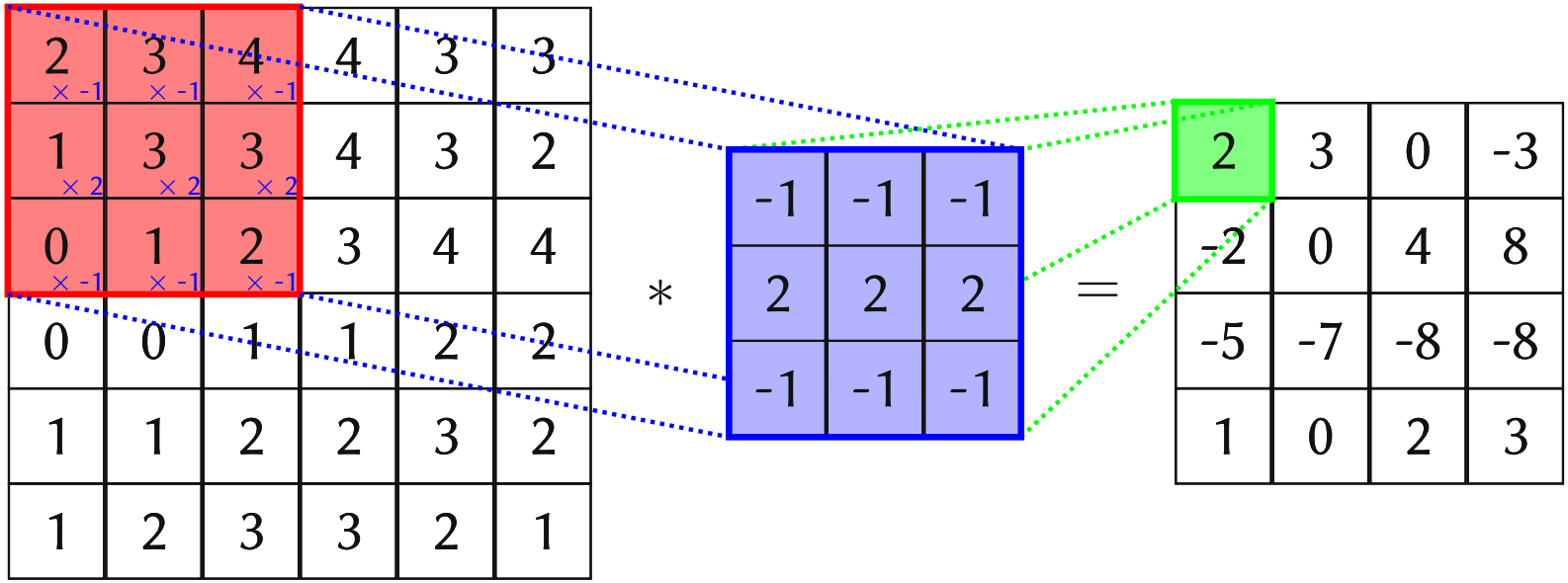
Konvolucijski operator

- Definišimo malu (npr. 3×3) matricu (**kernel**, $\mathbf{K}$).
- Postavimo je na sve moguće načine preko **ulazne slike**, $\mathbf{I}$.
- Zabeležimo *sume element-wise proizvoda* u novoj slici.

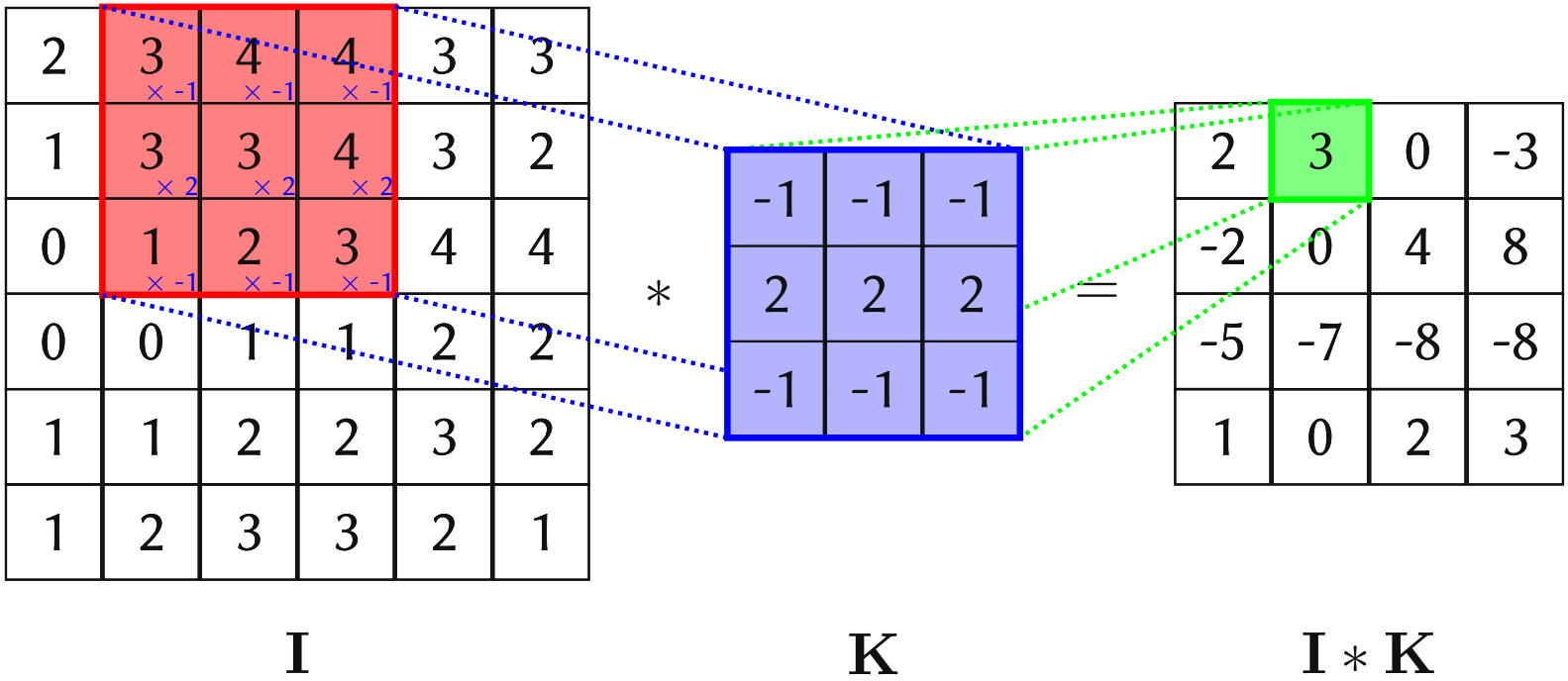
$$(\mathbf{I} * \mathbf{K})_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} \mathbf{K}_{ij} \cdot \mathbf{I}_{x+i-1,y+j-1}$$

- Ova operacija iskorišćava *strukturu* u slici—susedni pikseli utiču jedan na drugog više nego pikseli na suprotnim ćoškovima!

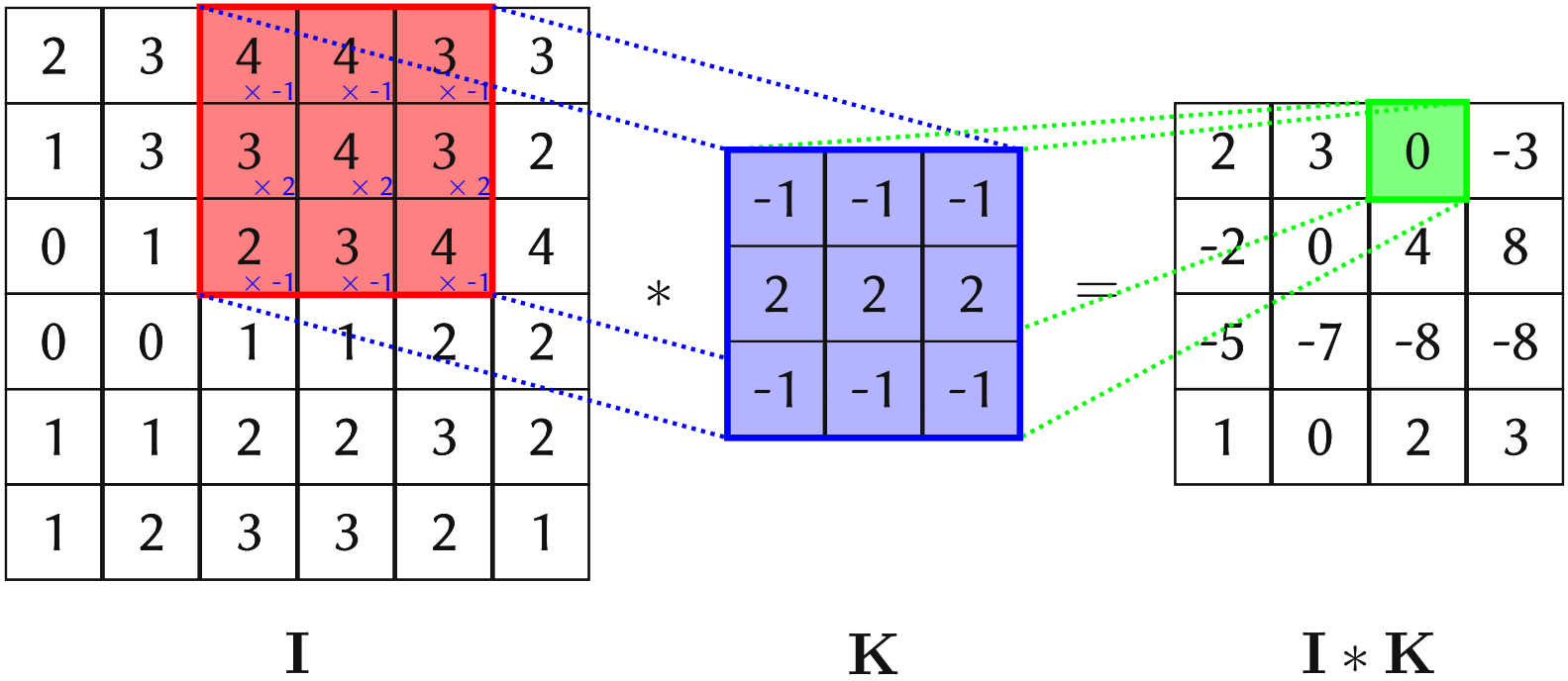
Primer konvolucije



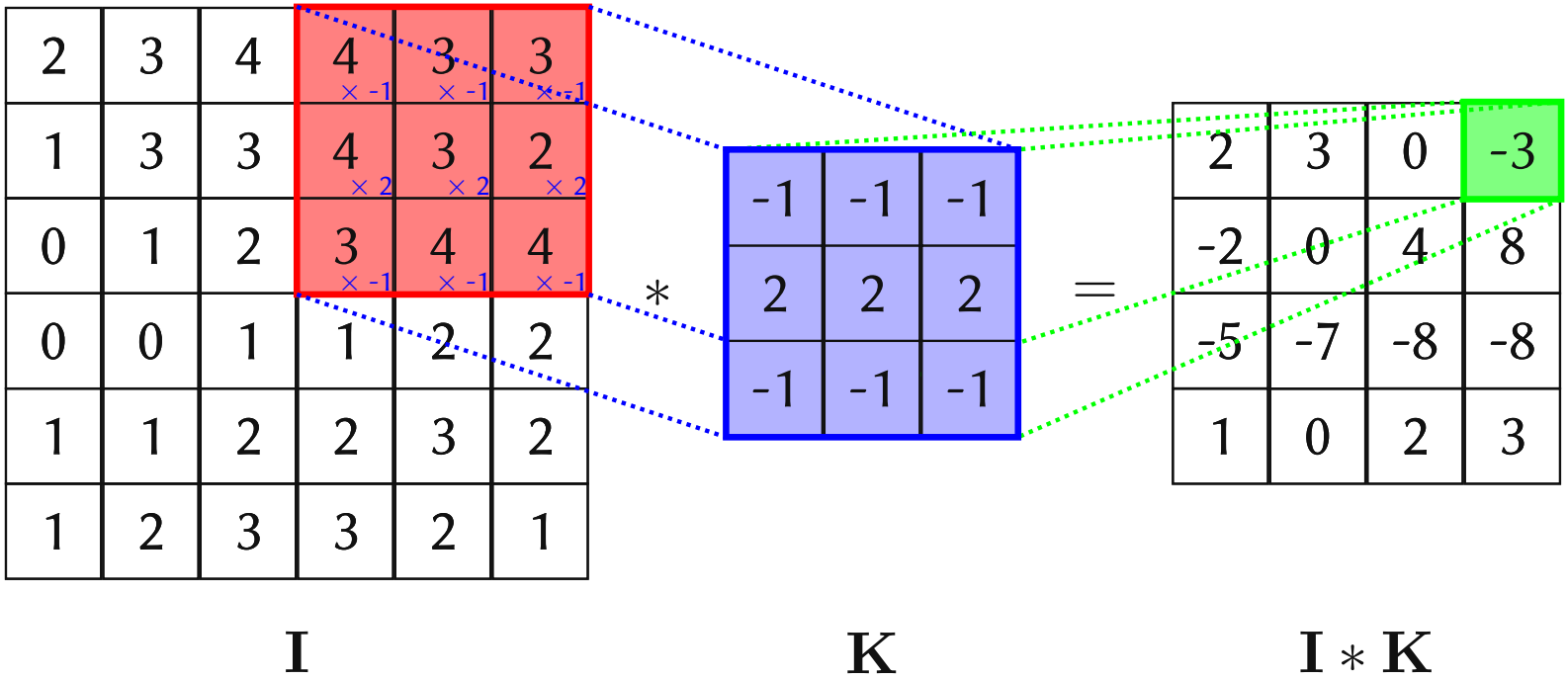
Primer konvolucije



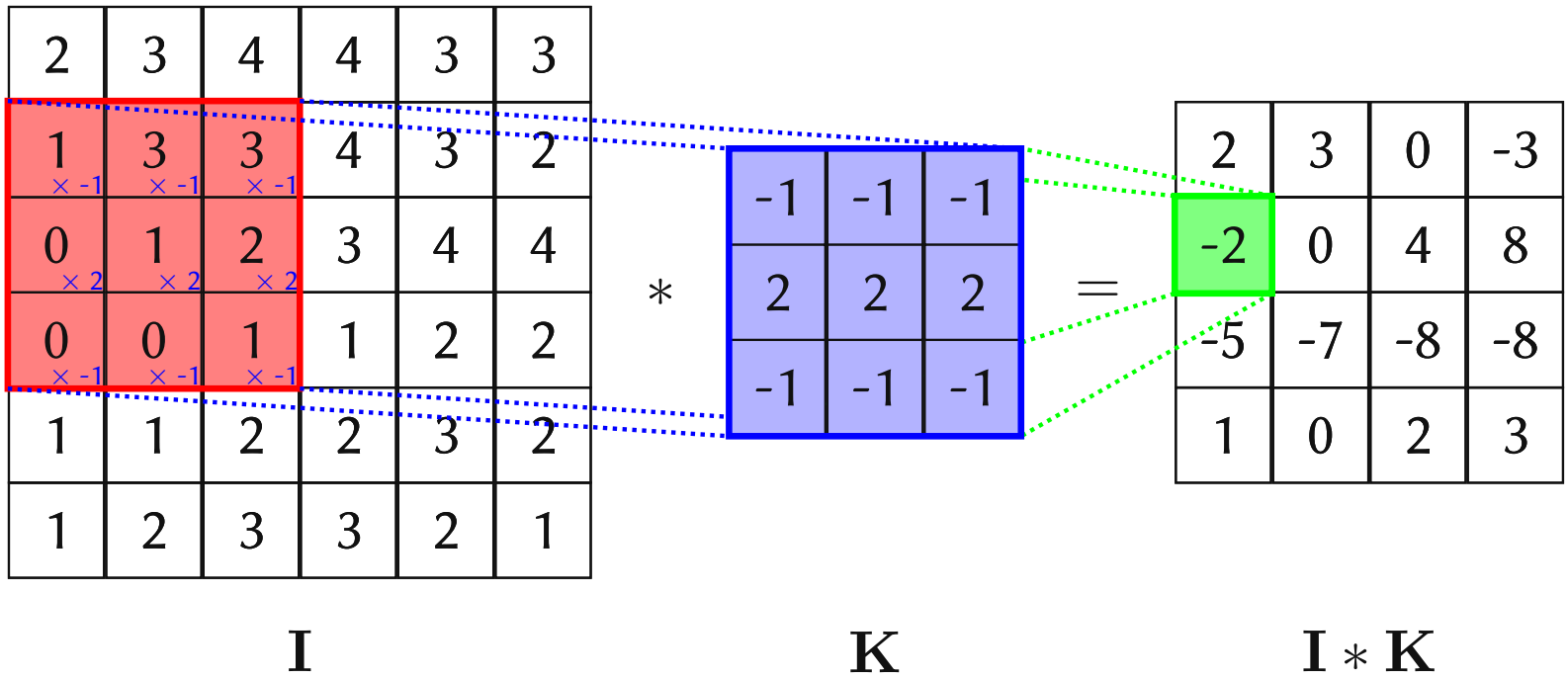
Primer konvolucije



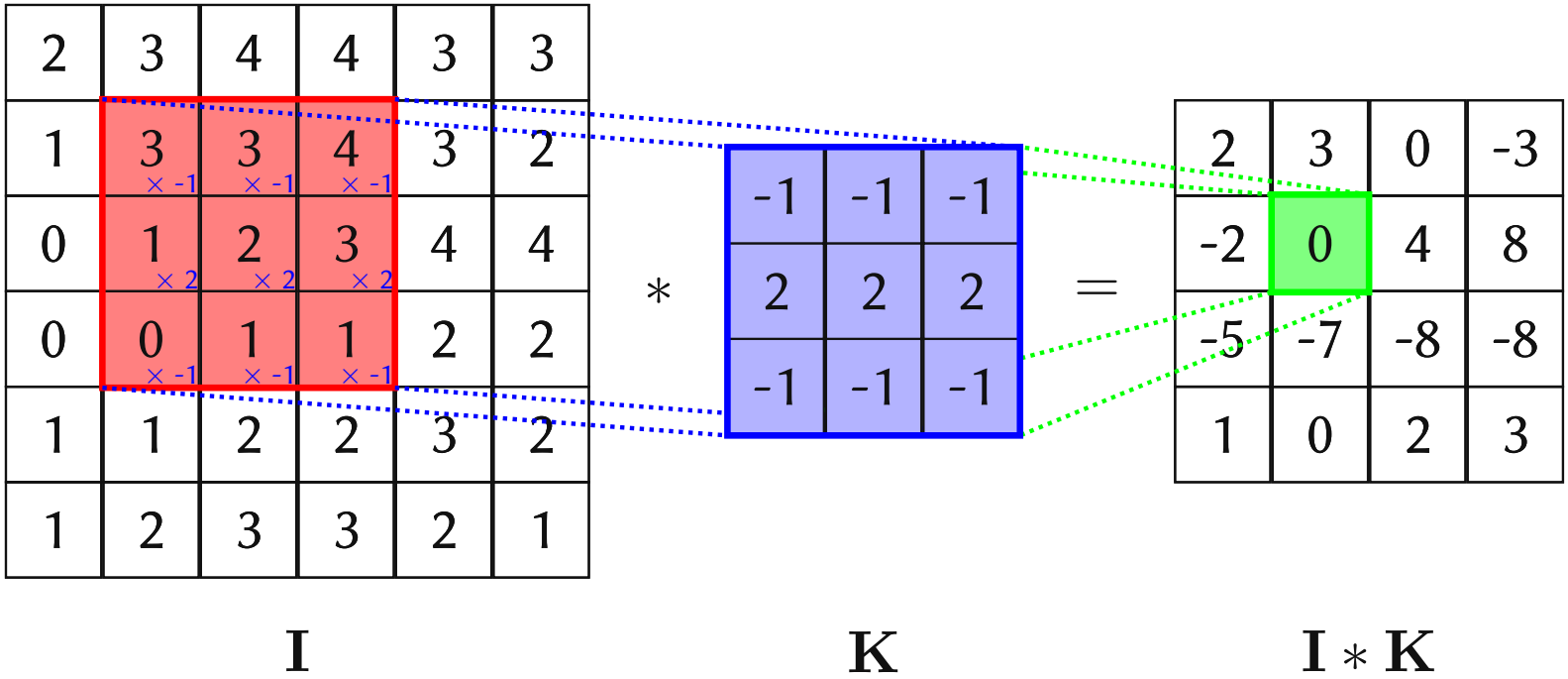
Primer konvolucije



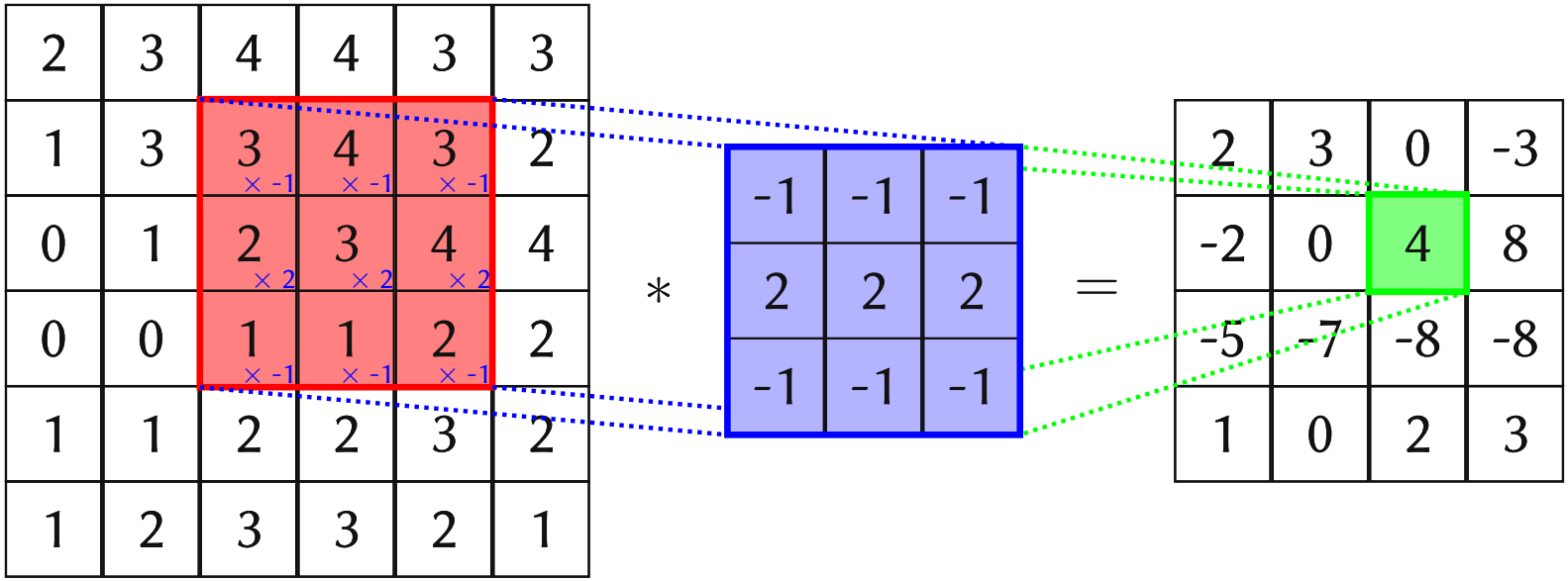
Primer konvolucije



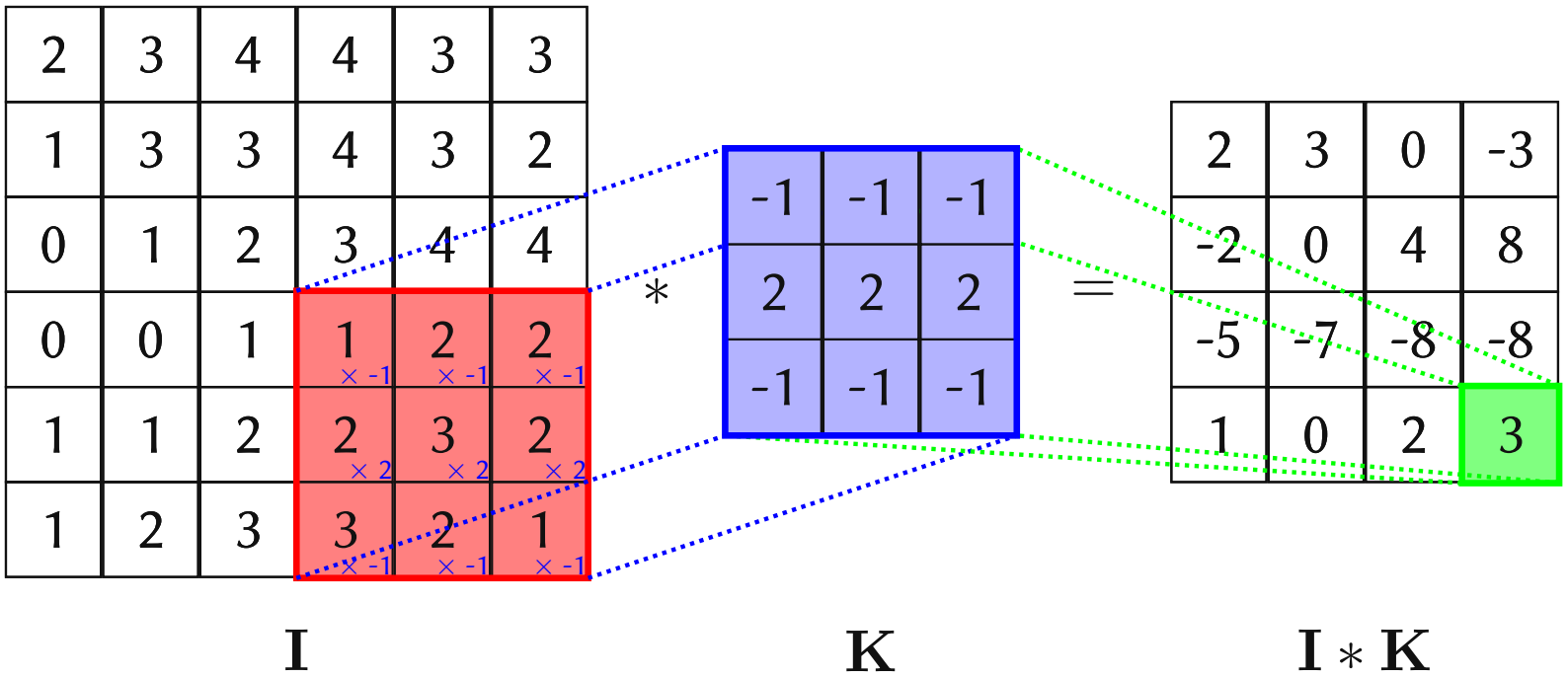
Primer konvolucije



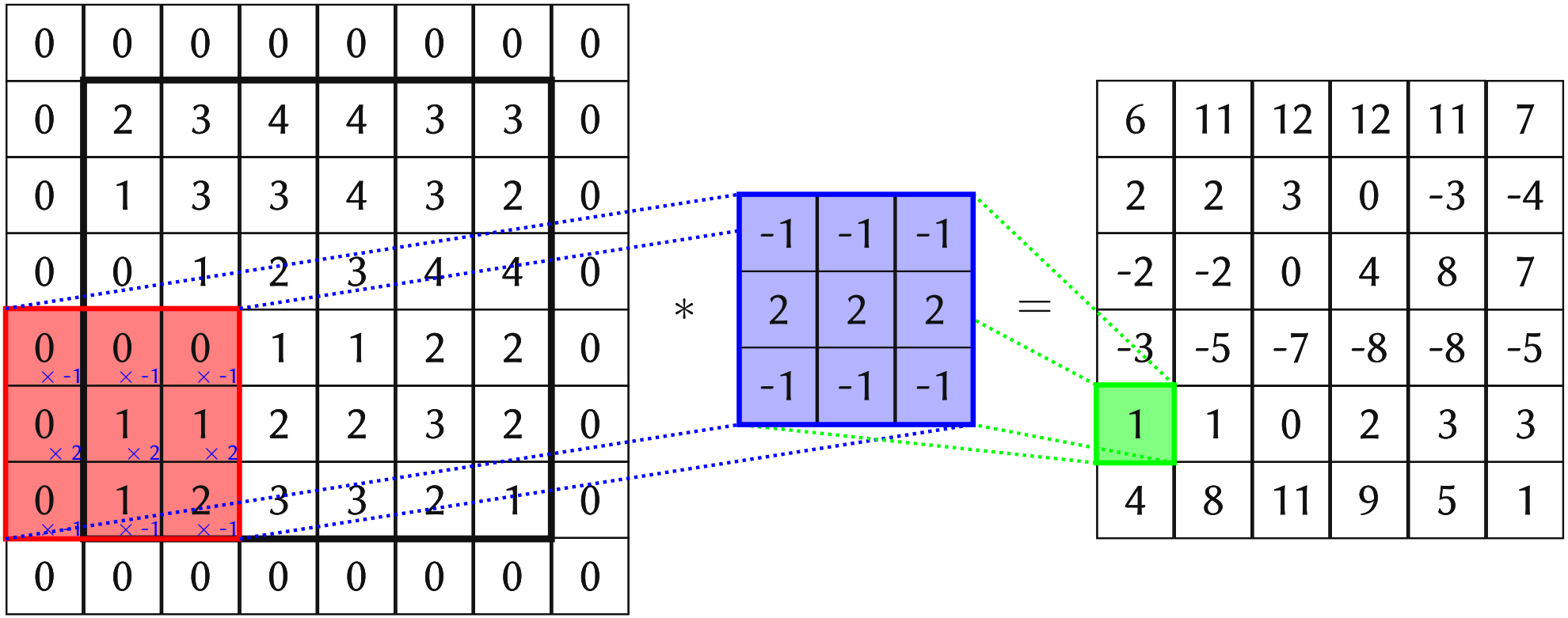
Primer konvolucije



Primer konvolucije

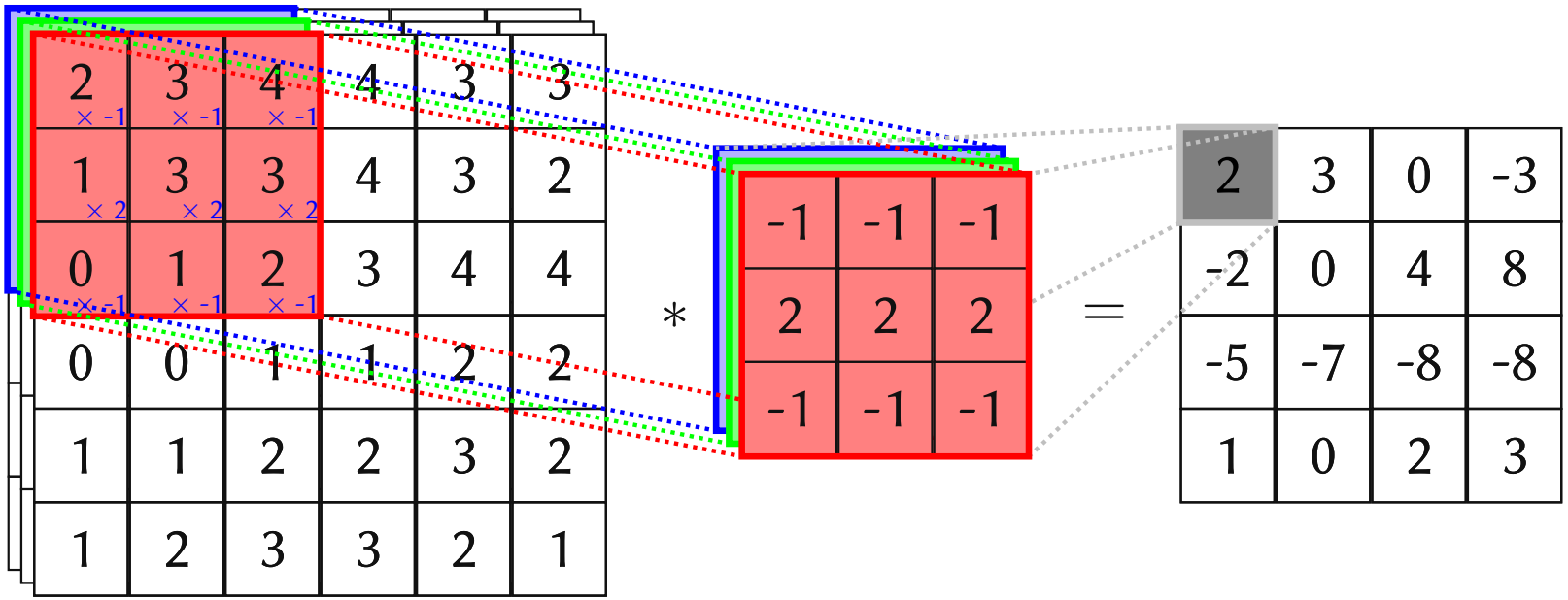


Primer konvolucije (sa *padding*-om)



$\mathbf{I}$ (*padded*) $\mathbf{K}$ $\mathbf{I} * \mathbf{K}$

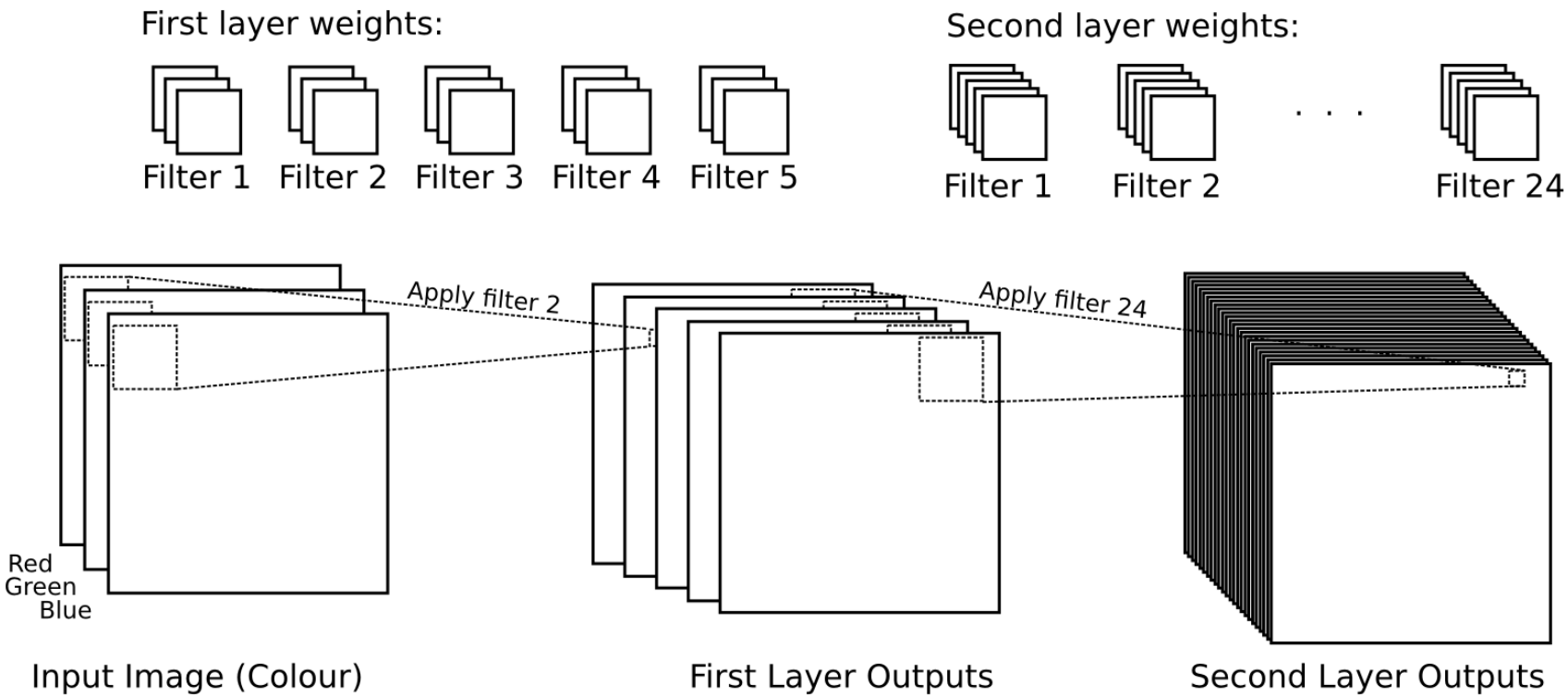
Konvolucija sa *bojom/kanalima*



Konvolucijski sloj

- Konvolucijski sloj je definisan preko d kernela veličine $h \times w$, koje treba primeniti na sliku iz prethodnog sloja.
- Konvolucijom jednog takvog kernela (preko svih kanala ulazne slike) dobijamo *jedan kanal* izlazne slike. Na kraju možemo primeniti aktivacijsku funkciju na svaki piksel (uglavnom ReLU).
- *Kviz*: Ukoliko je ulazna slika imala d' kanala, *koliko parametara ima jedan konvolucijski sloj*?

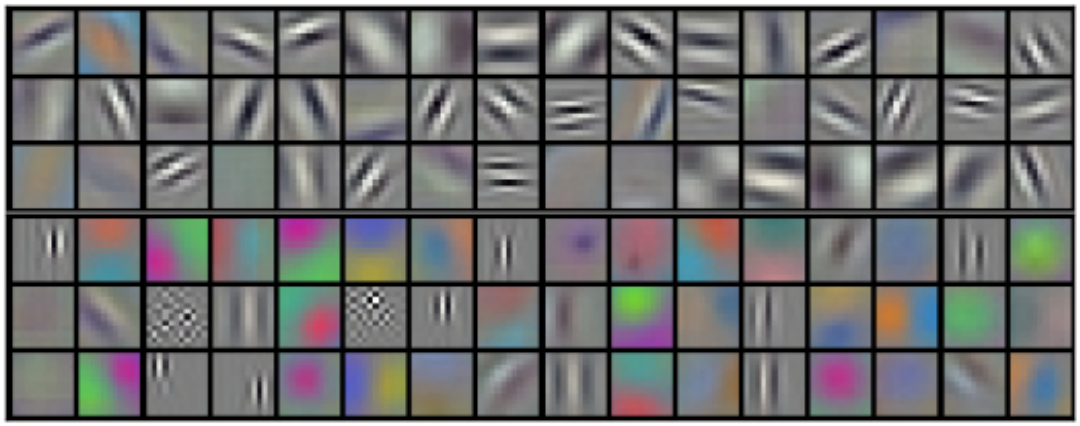
Slaganje konvolucijskih slojeva



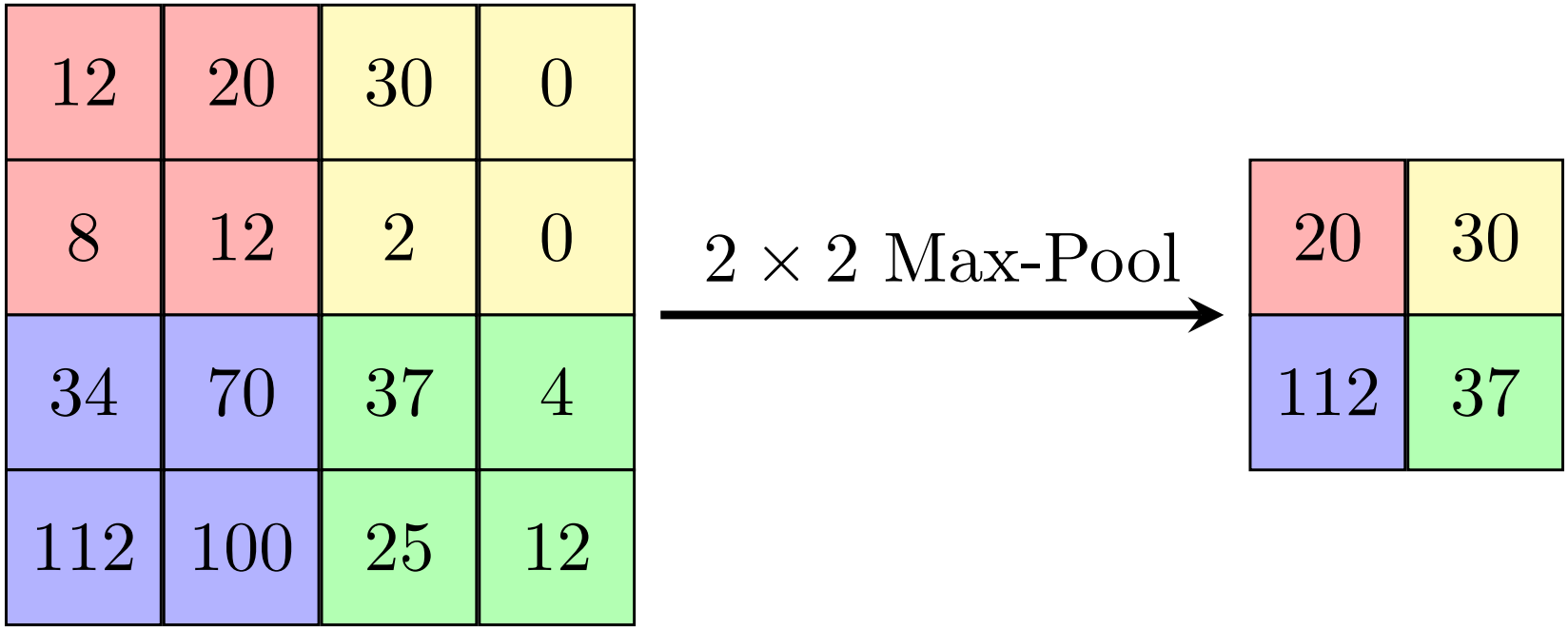
Treniranje konvolucijskog sloja

- Počnimo sa **nasumičnim** kernelima, i pustimo mrežu da *sama* nauči optimalne kernele!
  - **N.B.** sve što radimo je množenje ulaza sa težinama i sabiranje
    $\Longrightarrow$ možemo trenirati *na isti način* kao i ranije!

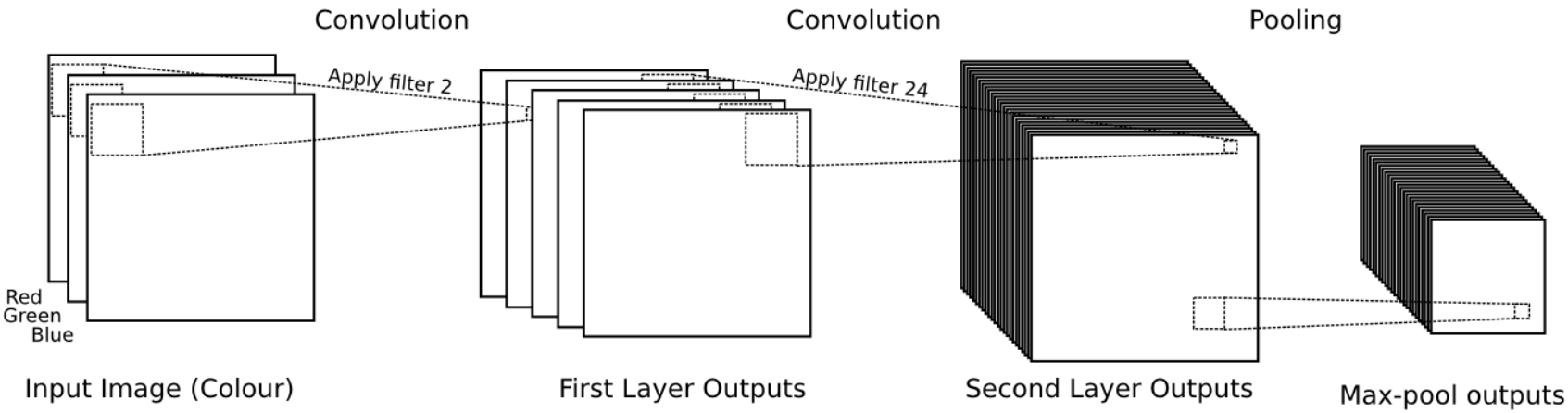
- Gledajući naučene kernele, u gotovo svim slučajevima prvi sloj nauči da bude *detektor ivica*...



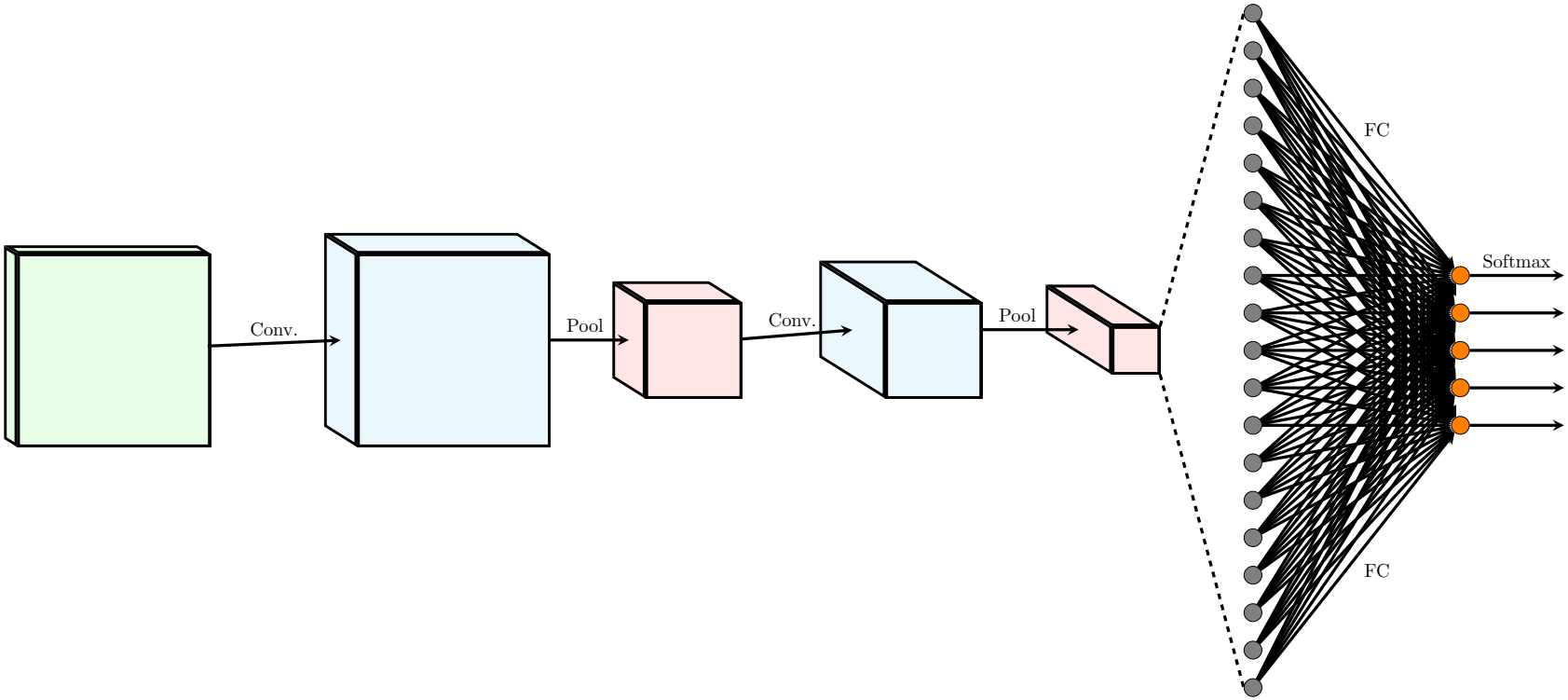
Pooling sloj



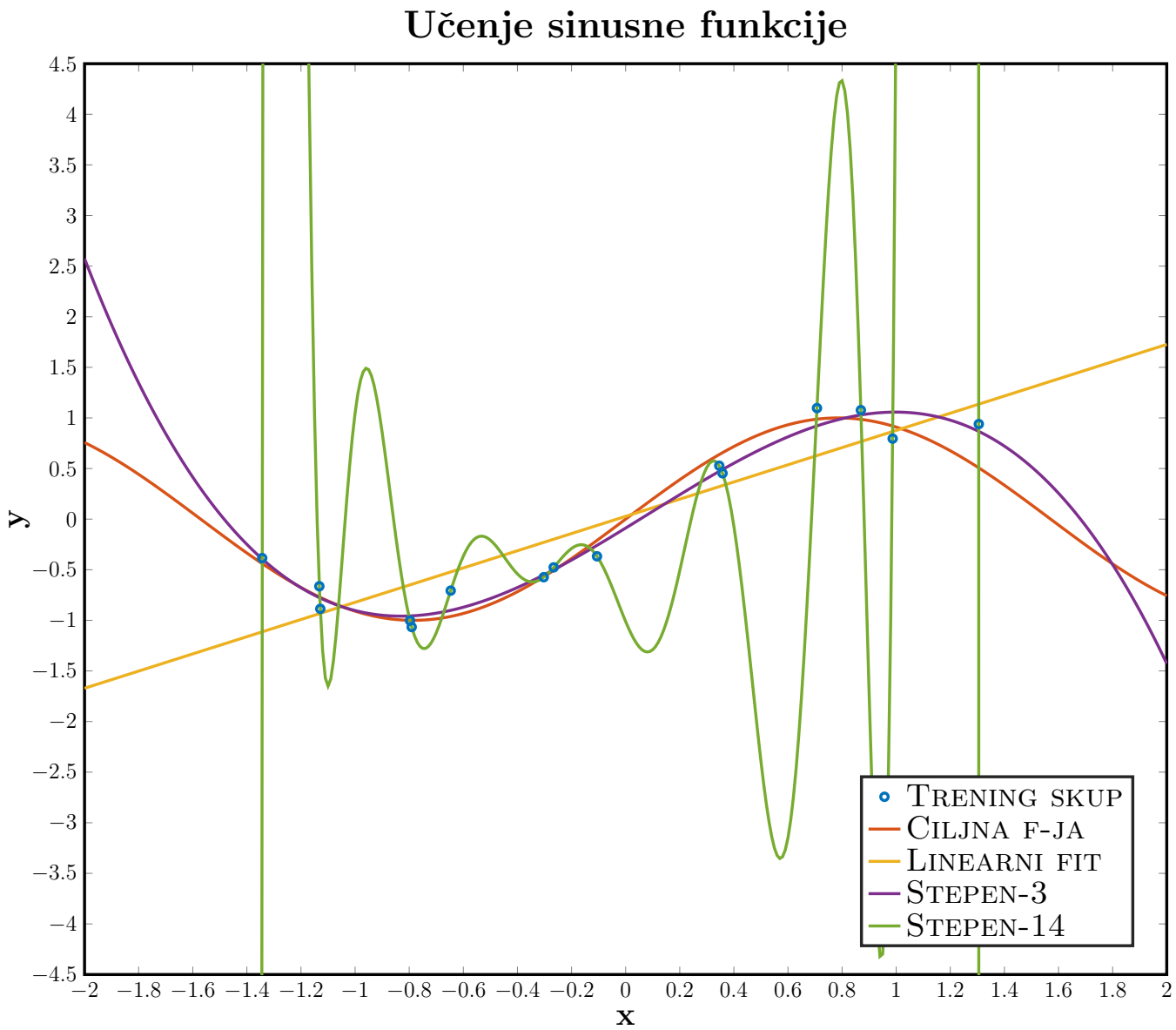
Dodavanje *max-pool* sloja



Sastavimo sve zajedno!



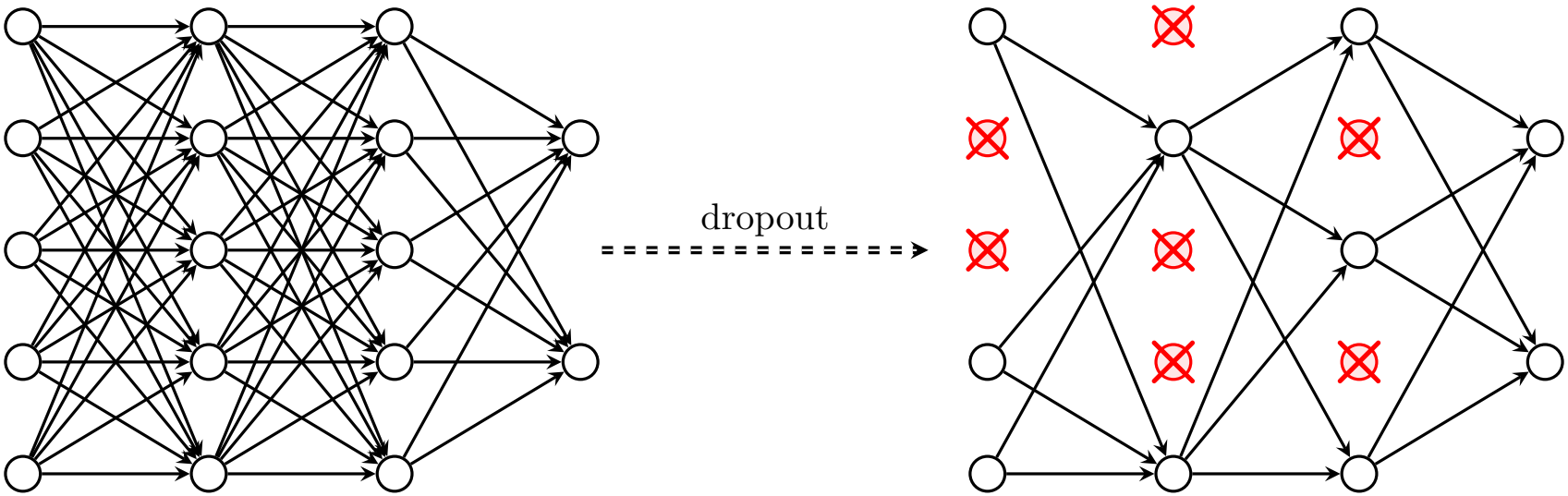
Prekomerno fitovanje/*Overfitting*



Zašto sada?

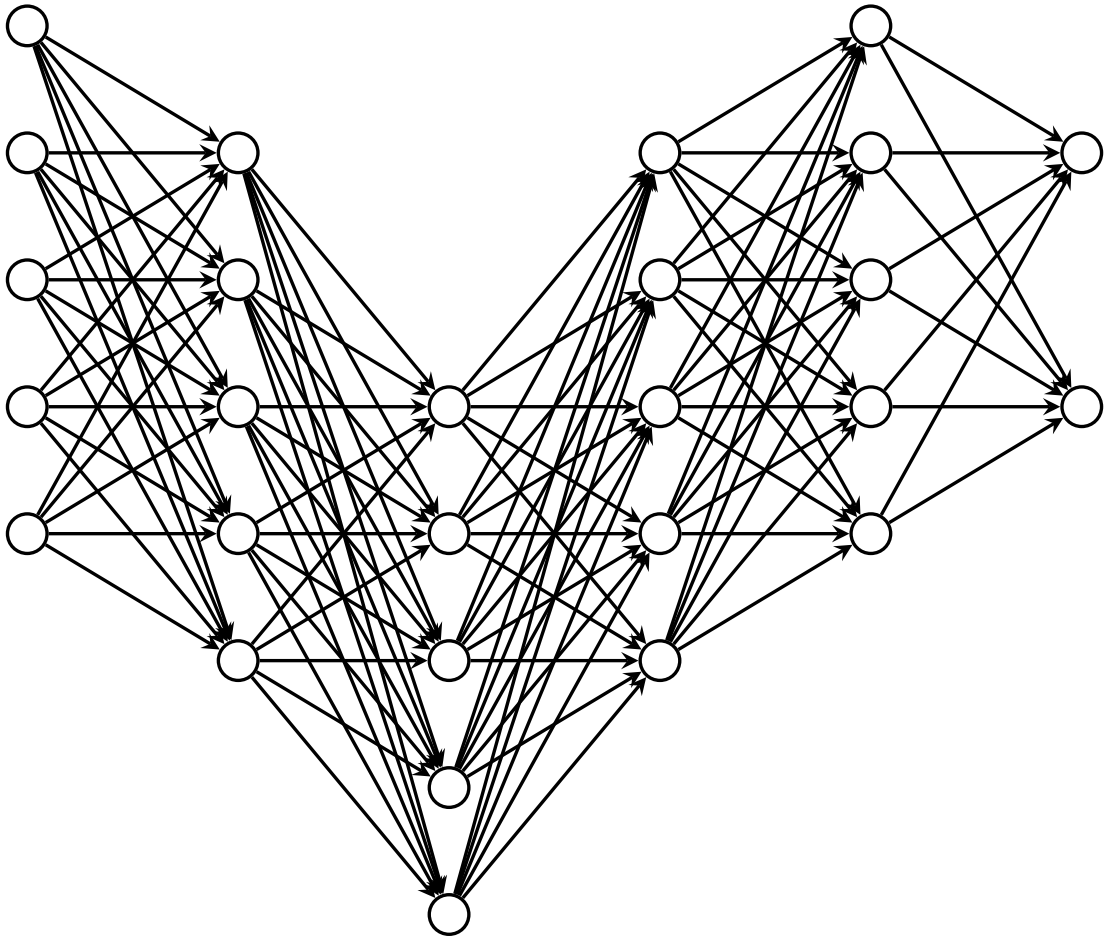
- Kod konvolucijskih neuralnih mreža, i većine problema prepoznavanja nad slikama, *overfitting* postaje izuzetno veliki problem!
- Metode sa kojima činimo da se naš model "lepše" ponaša u nedostatku dovoljno velikog trening skupa su metode **regularizacije**.
- Ovde ćemo pokriti dve regularizacione metode *"crne magije"* koje uglavnom rade ekstremno dobro u praksi...

Dropout



- Nasumično "ubiti" svaki neuron u nekom sloju sa verovatnoćom p, samo tokom treninga...?!

Batch normalisation



- “Unutrašnje pomeranje kovarijanse”…?!

Batch normalisation

- Rešenje: **renormalizacija** izlaza trenutnog sloja kroz trenutni *minibatch*, $\mathcal{B} = \{x_1, \ldots, x_m\}$ (ali dozvoliti mreži da se "vrati nazad" ako je tako bolje)!

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \varepsilon}} \qquad y_i = \gamma\hat{x}_i + \beta$$

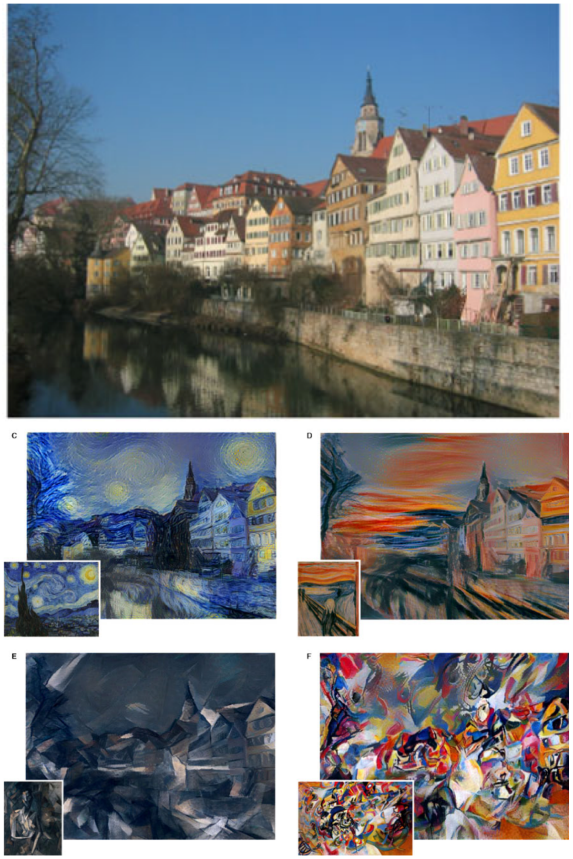
gde se γ i β mogu *trenirati*!

- Sada se *gotovo stalno* koristi za treniranje dubljih konvolucijskih mreža.
  - Rad objavljen u februaru 2015., $\sim$ 850 puta citiran do sada!

Jedan trik za kraj: *augmentacija podataka*

Kraj predavanja!

Pitanja? Mnogo interesantnih stvari o kojima možemo diskutovati!



NVIDIA
Microsoft Development Center Serbia
CAMBRIDGE SPARK