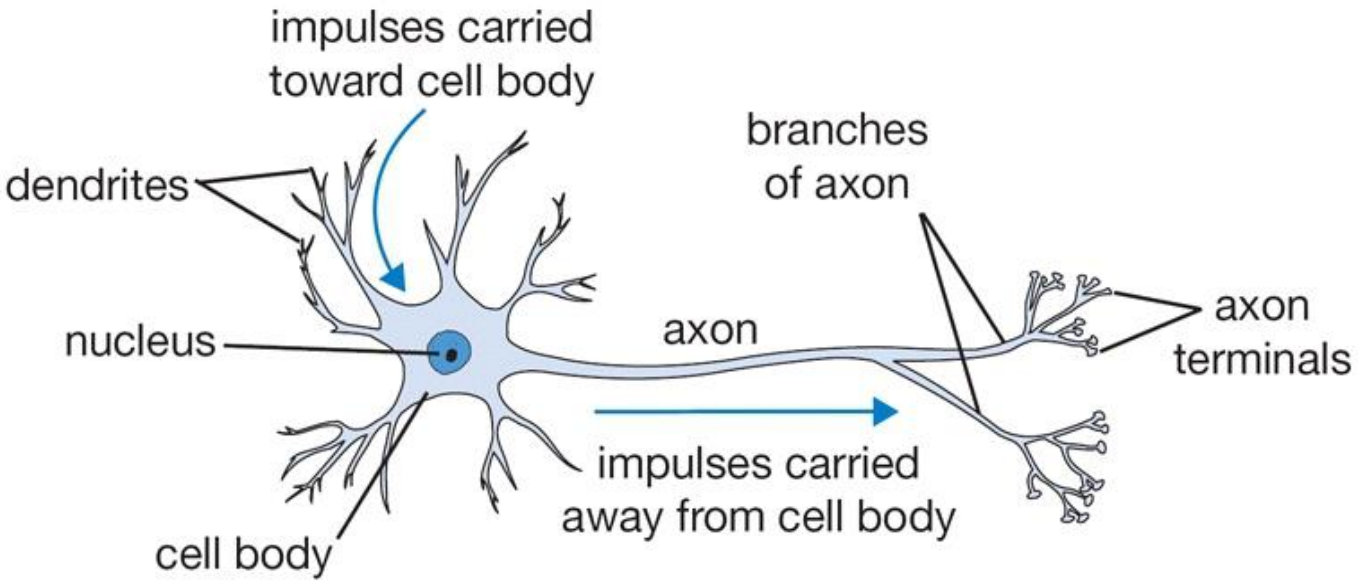
Konvolucione neuronske mreže

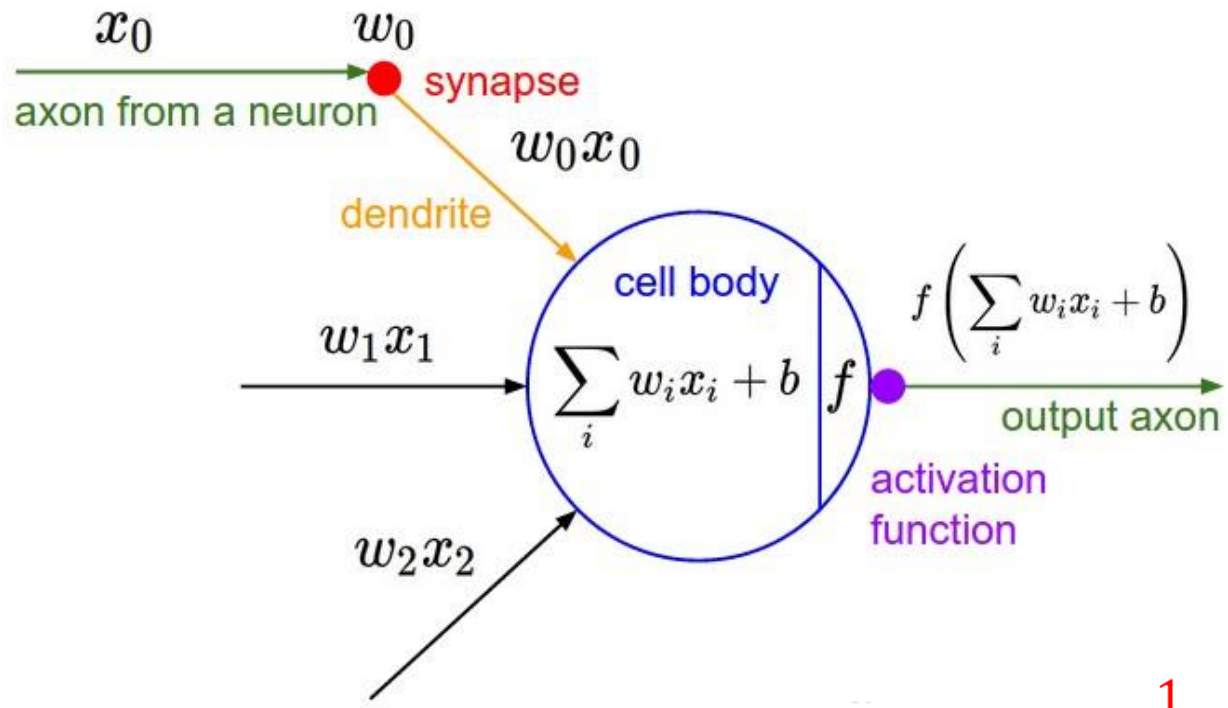
(*Convolutional Neural Networks*, CNN)

Biološka motivacija

- Osnovna procesna jedinica ljudskog mozga je neuron
  - 10^{11} neurona
  - $10^{14} - 10^{15}$ sinapsi (svaki neuron je povezan sa oko $10^3 - 10^4$ drugih neurona)
  - Do neurona preko dendrita dolaze impulsi
    - Impulsi povećavaju ili smanjuju verovatnoću da će neuron proizvesti impuls na aksonu



Matematički model veštačkog neurona



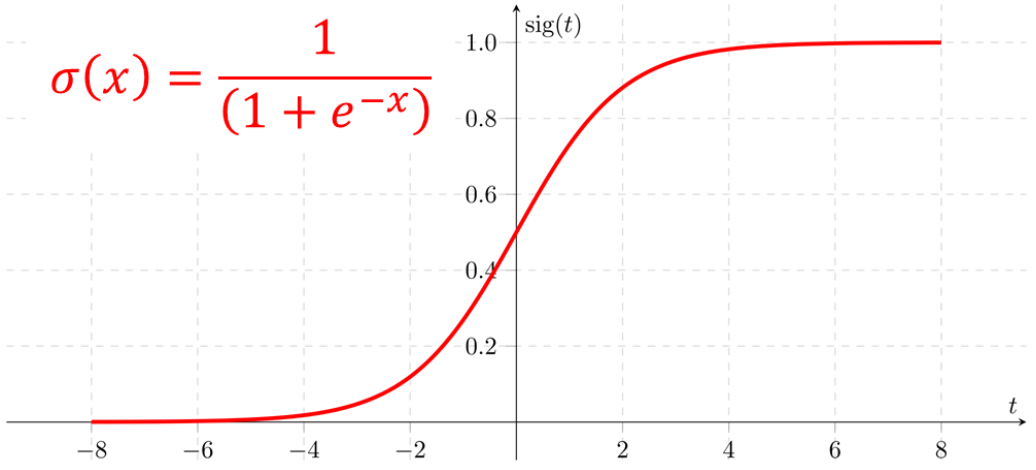
f – aktivaciona funkcija

- Određuje nivo pobuđenosti neurona za date ulaze
- Postoji više tipova aktivacionih funkcija

Primer aktivacione funkcije: sigmoidna funkcija

transformiše $\sum_i w_i x_i + b$ iz opsega $(-\infty, +\infty)$ u opseg $[0,1]$

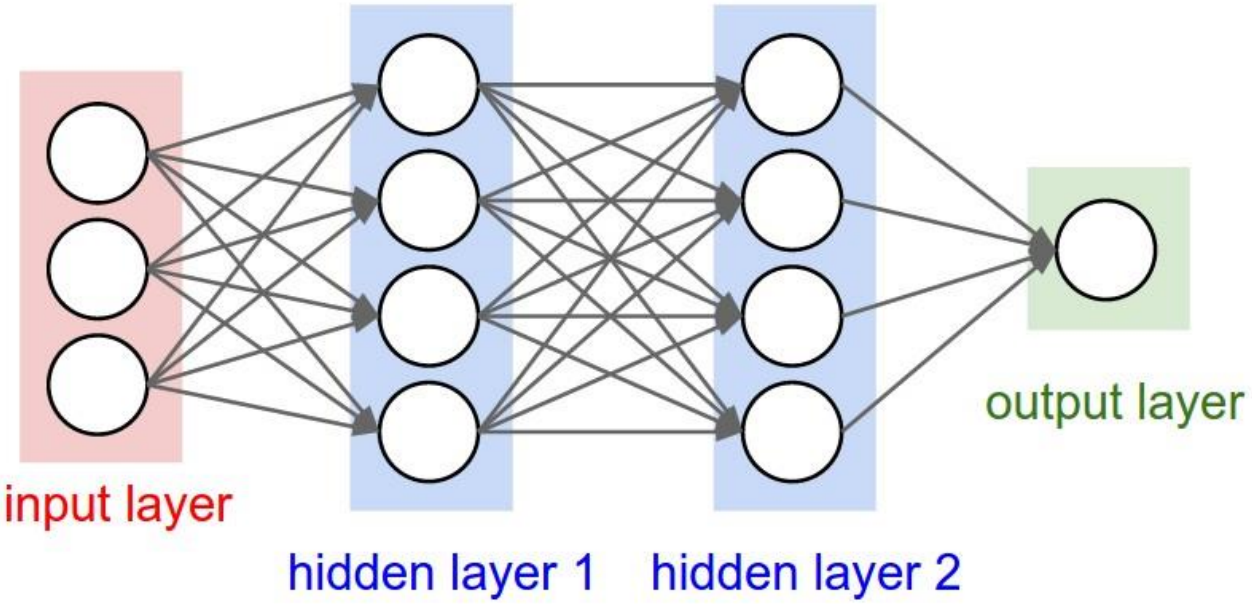
$$\sigma(x) = \frac{1}{(1 + e^{-x})}$$



Analogija prirodnog i veštačkog neurona

- Budite veoma pažljivi u pravljenju analogije između veštačkog i biološkog neurona!
- Biološki neuroni
  - Postoji više različitih tipova
  - Dendriti mogu da primenjuju kompleksne nelinearne funkcije
  - Sinapse nisu određene jedinstvenom težinom već kompleksnim nelinearnim dinamičnim sistemom
  - Aktivacionu funkciju obično „interpretiramo“ kao frekvenciju okidanja
    - Intuicija je da neuroni prenose signale povezanim neuronima u diskretnim impulsima – ako je frekvencija velika, signal je snažan
    - Od aktivacionih funkcija, najpribližnija aproksimacija je ReLU
    - Ali, posmatranje isključivo frekvencije impulsa verovatno nije dovoljno. Kod biološkog neurona impulsi nisu u ravnomernim vremenskim razmacima i ovo je verovatno od značaja

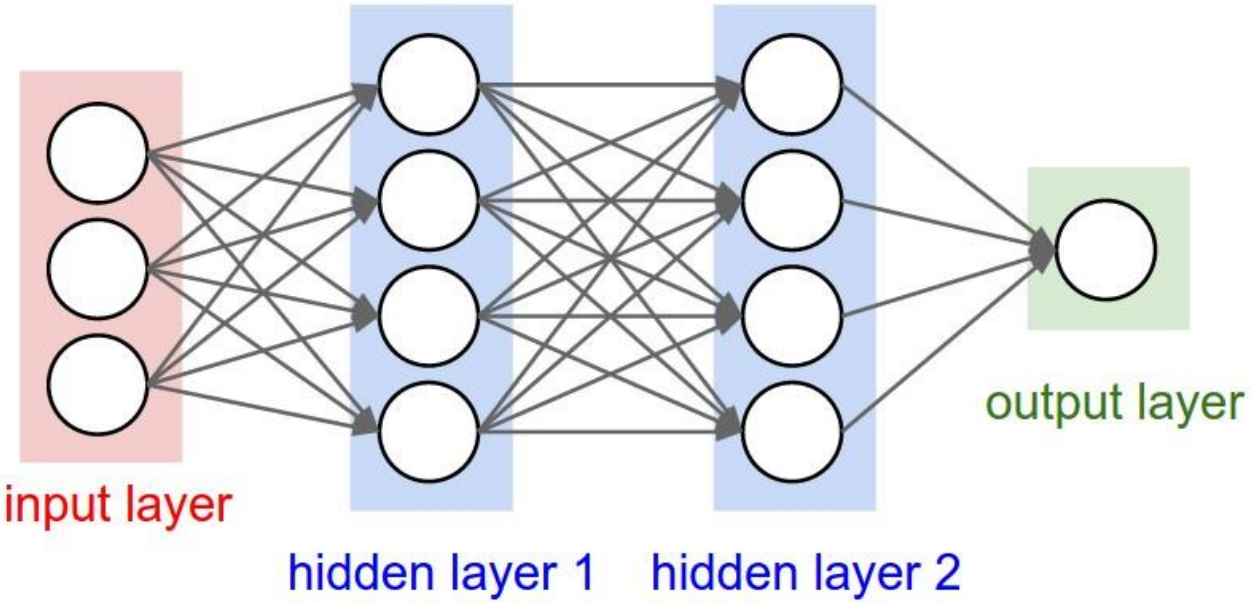
Organizacija po slojevima



- U neuronskoj mreži neuroni su grupisani u slojeve
  - Prvi sloj je ulazni i razlikuje se od ostalih: prosleđuje vrednosti samog uzorka
  - Poslednji sloj je izlazni i njegov izlaz predstavlja rezultat
  - Slojevi između ulaznog i izlaznog nazivaju se skriveni slojevi

- Na slici je dat primer neuronske mreže sa **tri** sloja
  - dva skrivena i jedan izlazni (ne brojimo ulazni sloj)

Feedforward arhitektura



- Prikazana je struktura sa propagacijom unapred (*feedforward*)
  - Izlazi neurona su povezani *isključivo* na neurone sledećeg sloja

- Slojevi su međusobno potpuno povezani (*fully connected layer*)
  - Neuroni jednog sloja su povezani sa svim neuronima sledećeg sloja

Model

- Arhitektura
  - Broj skrivenih slojeva i broj neurona u skrivenim slojevima spadaju u osnovne hiperparametre ovog modela

- Svaki neuron ima jedan dodatni ulaz koji se naziva *bias* (b)
  - *bias*, zajedno sa težinama ostalih veza, predstavlja parametre modela koji se uče

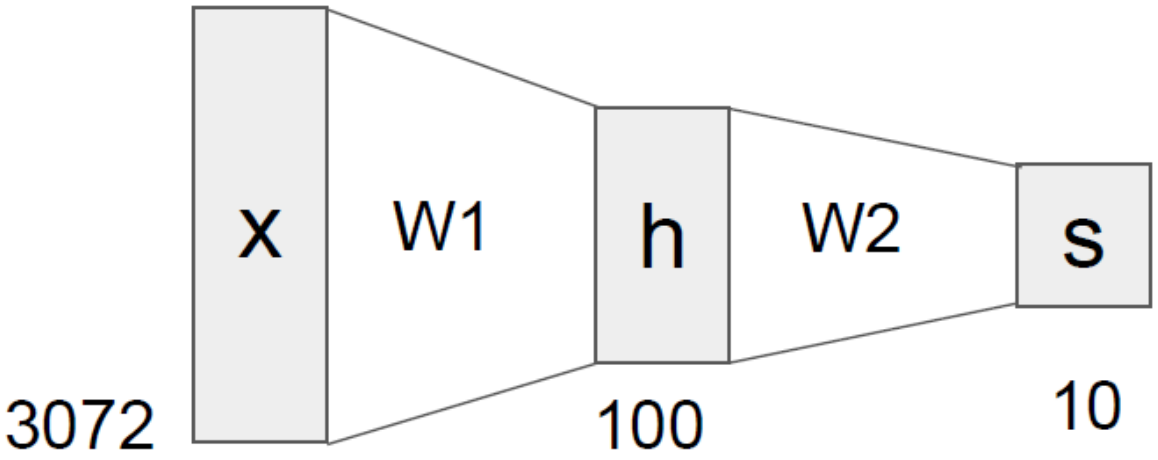
Predikcija modela (*feedforward*)

- Kako za dati ulaz x_i odrediti izlaz (predikciju $h(x_i)$)?

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$



Regresija

- U poslednjem sloju je samo jedan neuron
  - Ne koristimo aktivacionu funkciju na ovom neuronu
- Izlaz (naša hipoteza $h(x_i)$) je izlazni signal iz ovog neurona
- Greška predikcije je *Mean Squared Error* (MSE):
  - $(h(x_i) - y_i)^2$, gde y_i predstavlja stvarnu labelu primera x_i

Klasifikacija

- Broj neurona u poslednjem sloju je jednak broju kategorija (klasa) K
  - Obično ne koristimo aktivacione funkcije na ovim neuronima
  - Izlazni signali: $s_1, s_2, \ldots, s_K$

- Izlaz: *Softmax* funkcija

$$P(y_i = k | X = x_i) = \frac{\exp s_k}{\sum_{i=1}^{K} \exp s_i}$$

  - *Softmax* transformiše izlazne signale tako da su nenegativni i da se sumiraju na 1
  - Interpretacija: verovatnoće odgovarajućih klasa

- Greška: negativna vrednost logaritma verodostojnosti paremetara

$$L_i = -\log P(Y = y_i | X = x_i)$$

Klasifikacija

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

	unnormalized log probabilities	exp		normalize	probabilities	
cat	**3.2**	→	**24.5**	→	**0.13**	→ L_i = -log(0.13) = **0.89**
car	5.1	→	164.0	→	0.87	
frog	-1.7	→	0.18	→	0.00	

Obučavanje modela

- Dakle, za dat ulaz x_i i za date težine w možemo da izračunamo šta je predikcija modela $h_w(x_i)$
- Šta učimo (šta su parametri modela)?
  - Težine w u modelu (uključujući *bias*-e)
  - Obučavanje modela: pronalazak težina w tako da se predikcije $h_w(x_i)$ što bolje poklapaju sa stvarnim vrednostima izlaza y_i

- Kako ih učimo?
  - Izračunaćemo grešku predikcije
    - Razlika stvarnih vrednosti i onoga što naš model predviđa
    - Zavisi od parametara modela w
  - Minimizovaćemo grešku po w

Funkcija greške

- Za instancu x_i:
  - $h_w(x_i)$ - predikcija, y_i - stvarna labela
  - Greška predikcije na datoj instanci: $e(h_w(x_i), y_i)$
  - Npr. kod regresije, greška predikcije može biti MSE

$$e(h_w(x_i), y_i) = (h_w(x_i) - y_i)^2$$

- Za trening skup (N instanci):

$$E(w) = \frac{1}{N}\sum_{i=1}^{N} e(h_w(x_i), y_i)$$

(E je funkcija od w – obučavajući primeri $\{(x_i, y_i)\}$ su fiksni, ono što je promenljivo u modelu su težine w)

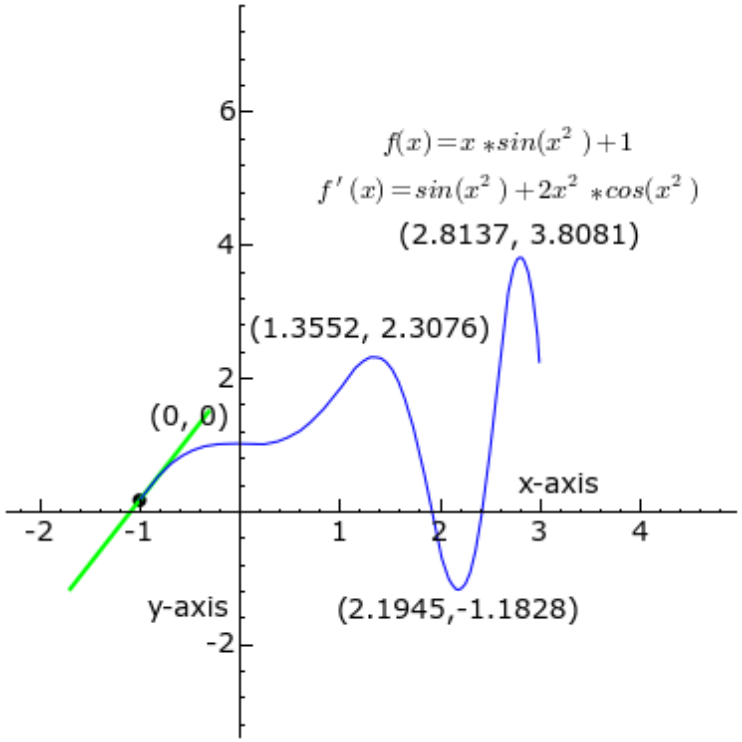
- Želimo da nađemo w za koje je $E(w)$ minimalno

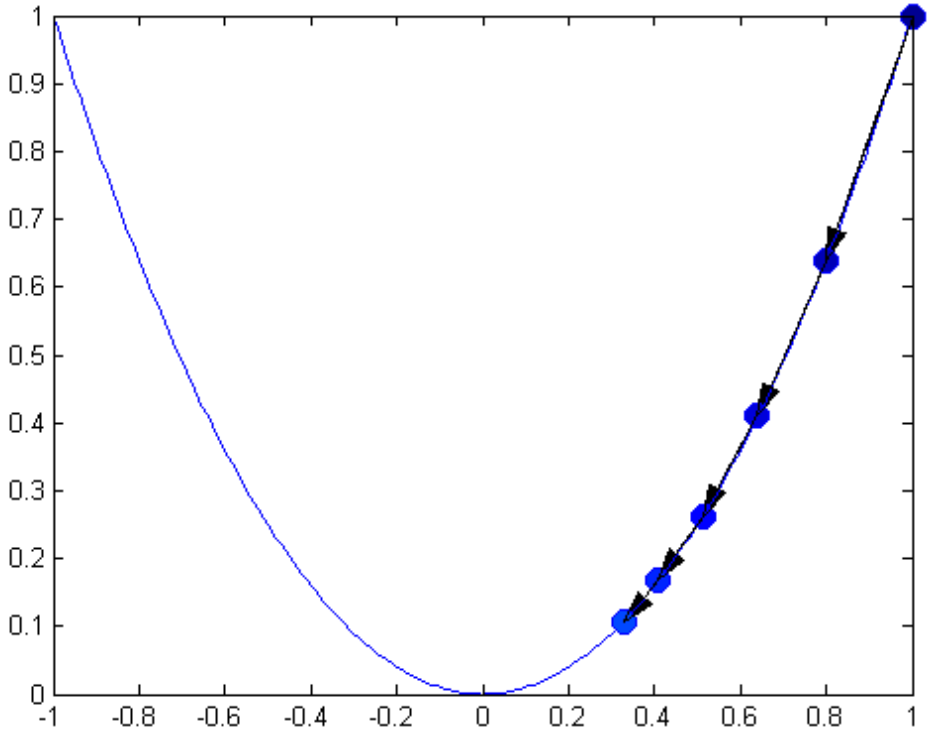
Funkcija greške

- Kako minimizovati funkciju greške?
  - Parametri bi se mogli varirati na razne načine kako bi se videlo u kom pravcu se može izvršiti pomak kako bi se dobila manja vrednost greške
  - Takav postupak je računski skup
  - Gradijent daje pravac pomeranja u kojem se greška *lokalno* najbrže povećava
  - Algoritam za optimizaciju: gradijentni spust

Gradijentni spust (*Gradient Descent*, GD)

- Izvod funkcije predstavlja nagib tangente na krivu funkcije
- Ideja: iterativno ćemo se pomerati ka minimumu
  - levo od minimuma: gradijent je negativan – pomeramo se u desno
  - desno od minimuma: gradijent je pozitivan – pomeramo se u levo





Gradijentni spust (*Gradient Descent*, GD)

- GD algoritam:
  - Inicijalizovati $w(0)$
  - Za $t = 1,2, \ldots$ (do konvergencije)
    - $w(t+1) = w(t) - \alpha \nabla_w E(w)$
  - Rezultat: optimalno w

GD_visualization.m

- Problem: $E(w) = \frac{1}{N} \sum_{i=1}^{N} e(h_w(x_i), y_i)$
  - Moramo proći kroz sve instance trening skupa: *Batch GD*
  - Loše za veliko N – mnogo računanja pre nego što se pomerimo
- Rešenje:
  - Aproksimiraćemo sumu korišćenjem podskupa trening skupa: *Minibatch GD*
  - Tipično: 32, 64 ili 128 primera

Obučavanje modela – *Mini-batch SGD*

- Uzmemo podskup primera trening skupa $\{x_i\}$
- Propustimo primere $\{x_i\}$ kroz neuronsku mrežu da dobijemo predikcije $\{h_w(x_i)\}$
- Koristimo funkciju greške da Izračunamo gubitak (*loss*)

$$E(w) = \frac{1}{|\{x_i\}|} \sum_i e(h_w(x_i), y_i)$$

- Ažuriramo w u pravcu negativnog gradijenta greške $E(w)$

Računanje gradijenta kod NN

- Dakle, ono što nam treba da primenimo GD jeste da odredimo gradijent funkcije greške $\nabla_w E(w)$
  - Ovo nije teško: neuronske mreže definišu kompleksnu funkciju greške, ali to nam je poznata funkcija
  - Npr. u slučaju MSE greške $E(W) = \sum_i (h_w(x_i) - y_i)^2$, a $h_w(x_i)$ računamo rekurzivno: to je izlazni funkcija poslednjeg sloja koji je funkcija sloja pre njega, a koji je funkcija sloja pre njega...
  - Dakle, analitička forma funkcije $E(W)$ je poznata i ne bi trebalo da predstavlja problem da odredimo gradijent
  - Problem je – kako da ovo uradimo *efikasno*?

- Algoritam propagacije u nazad (*backpropagation*)
  - Rekurzivna primena pravila pronalaska izvoda složene funkcije (*chain rule*)

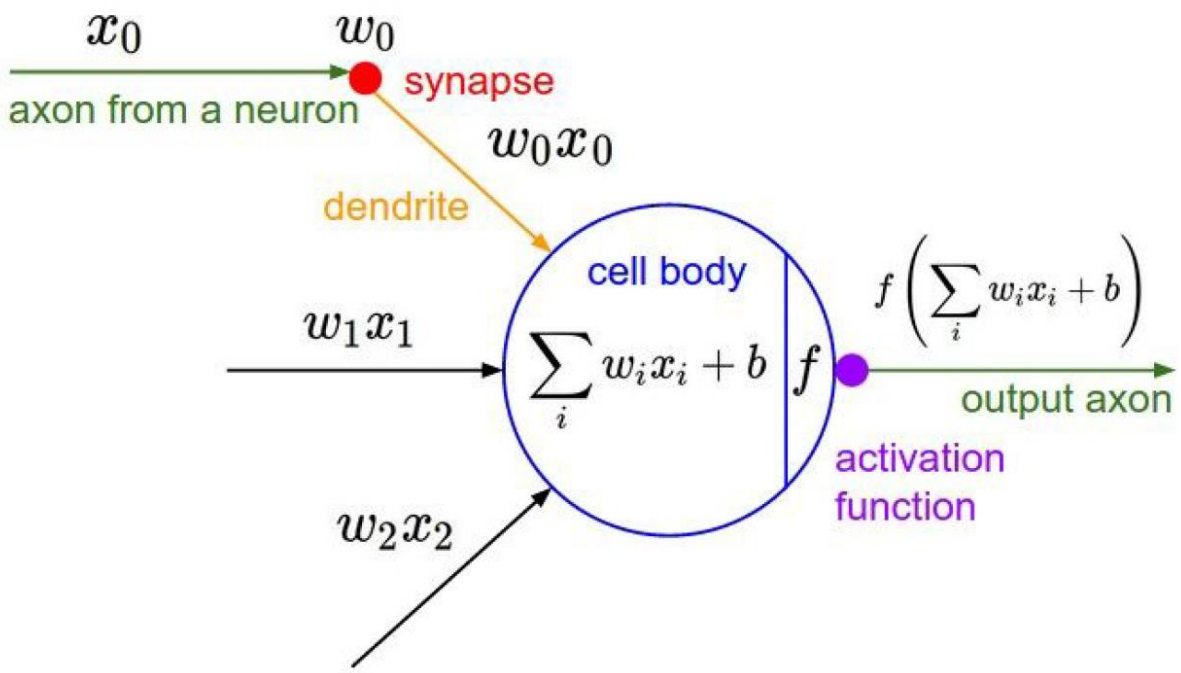
$$\big(f(g(x)\big)' = f'\big(g(x)\big)g'(x)$$

Backpropagation

- CS231N
  - Lecture 4: *Introduction to Neural Networks*
    - https://www.youtube.com/watch?v=d14TUNcbn1k&index=4&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv
- https://www.youtube.com/watch?v=Ilg3gGewQ5U

Aktivacione funkcije

Activation Functions



Zašto aktivaciona funkcija?

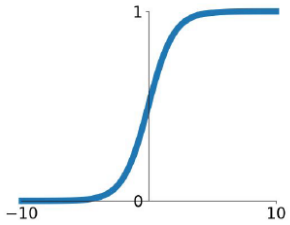
- Ako ne koristimo aktivacionu funkciju, izlazni signal će biti linearna funkcija
- Kombinacija linearnih funkcija je takođe linearna funkcija
- Bez aktivacione funkcije, cela neuronska mreža bi se svela na (veoma komplikovanu) linearnu regresiju

Aktivacione funkcije

Activation functions

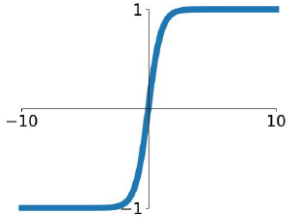
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



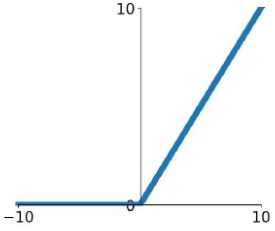
tanh

$$\tanh(x)$$



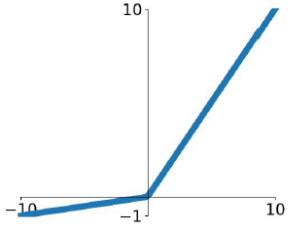
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

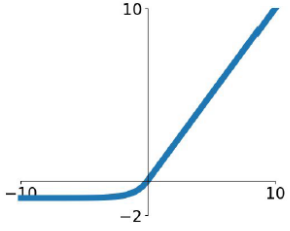


Maxout

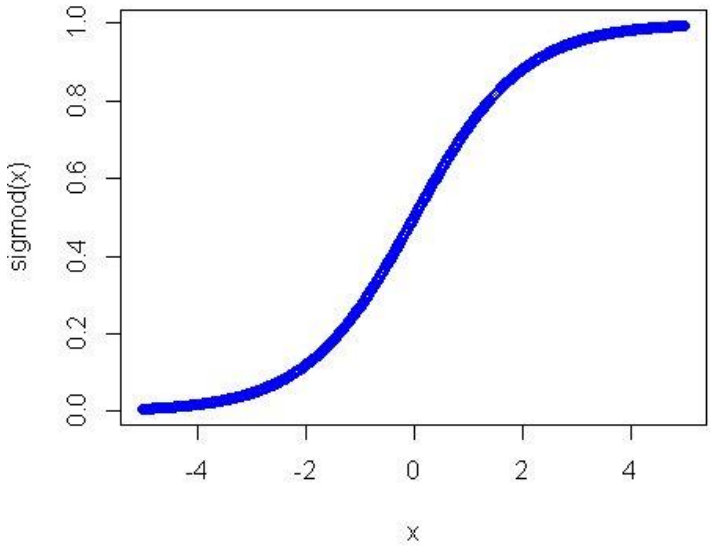
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



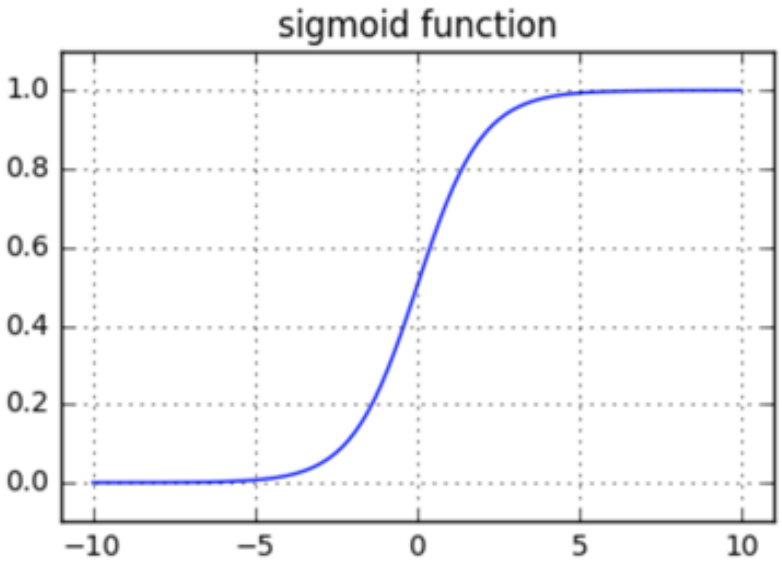
Sigmoid

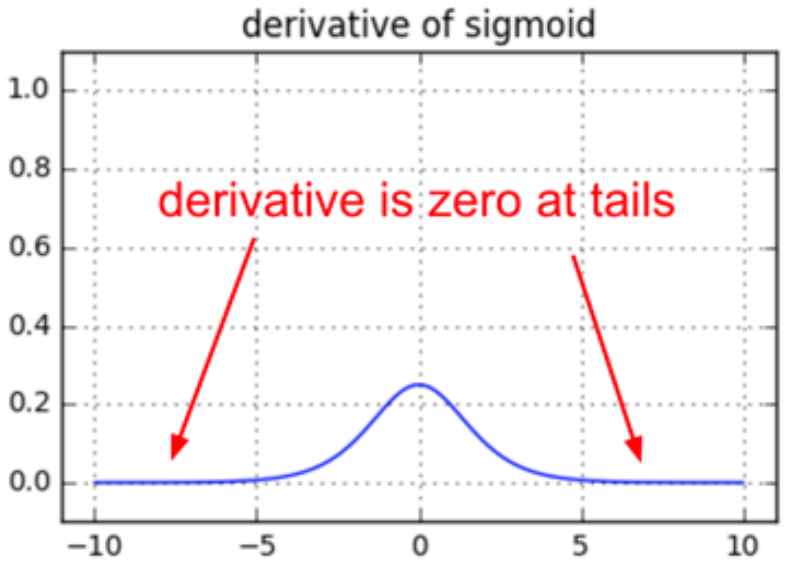


- $\sigma(x) = \frac{1}{(1+e^{-x})}$
- Mapira izlazni signal na interval $[0,1]$
- Istorijski popularna zbog lepe interpretacije

- „Ubija“ gradijente saturisanih neurona
- Nije centrirana oko nule
- Eksponencijalna funkcija je skupa za računanje

Sigmoid – problemi

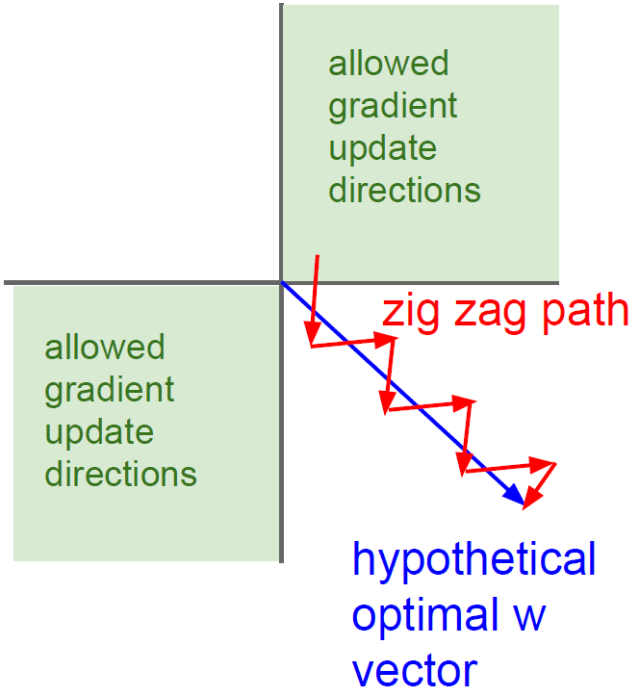




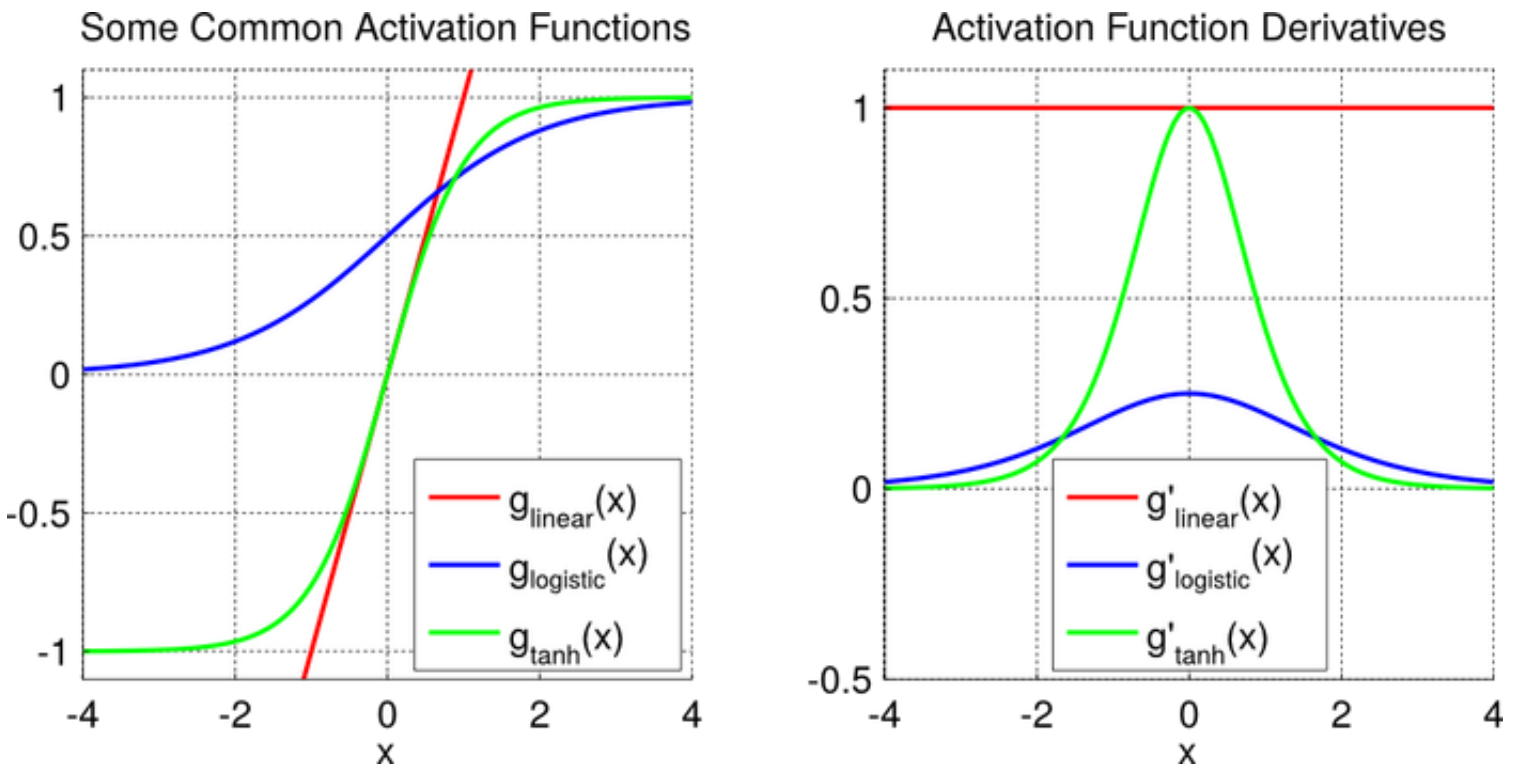
1. “Ubija“ gradijente saturisanih neurona
   - Tokom propagacije unazad, lokalni gradijent se množi sa gradijentom narednog sloja itd.
   - Ako je lokalni gradijent veoma mali – gotovo nikakav signal neće teći rekurzivno kroz slojeve
   - Zato se mora pažljivo birati početna vrednost – ako su vrednosti prevelike, većina neurona će postati saturisana i mreža će jedva da uči

Sigmoid – problemi

- Izlaz iz sigmoidne funkcije nije centriran oko nule
  - Ulaz u naredni neuron je uvek pozitivan
- $f(\sum_i w_i x_i + b)$ - ako je svako x_i pozitivno, lokalni gradijenti $\partial f / \partial w_i$ su svi pozitivni
- Gradijenti w su svi pozitivni (ili svi negativni)
  - zavisno od gradijenta koji nam se vraća sa izlaza
  - Zato će se svi gradijenti w kretati u istom smeru
  - Ovo rezultuje nepoželjnim oscilujućim efektom u ažuriranju gradijenata
- Zato centralizujemo x oko nule – kako bismo imali i pozitivne i negativne vrednosti

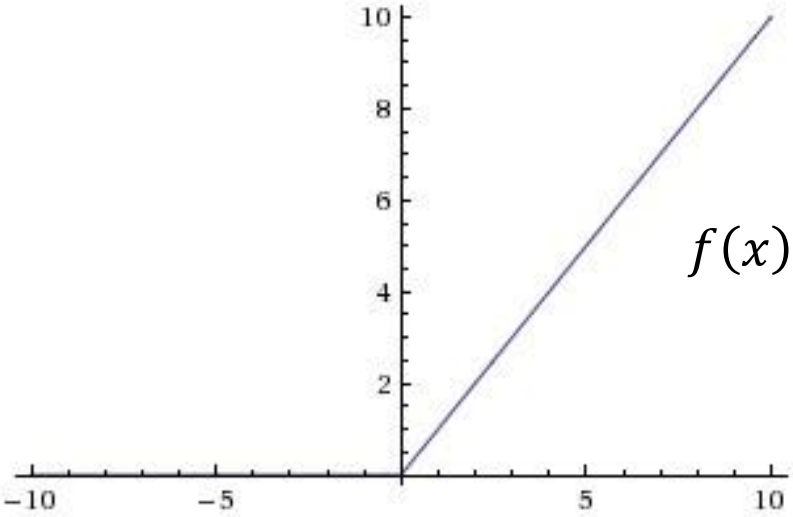


Tanh



- Mapira realan broj u opseg $[-1,1]$
- Kao i kod sigmoida, dolazi do saturacije aktivacija
- Za razliku od sigmoida je centriran oko nule
- Zato se u praksi uvek preferira naspram sigmoida

ReLU (*Rectified Linear Unit*)

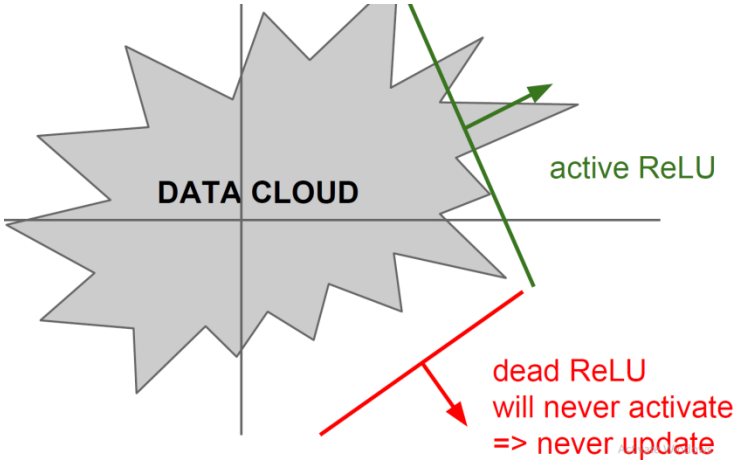


$$f(x) = \max(0, x)$$

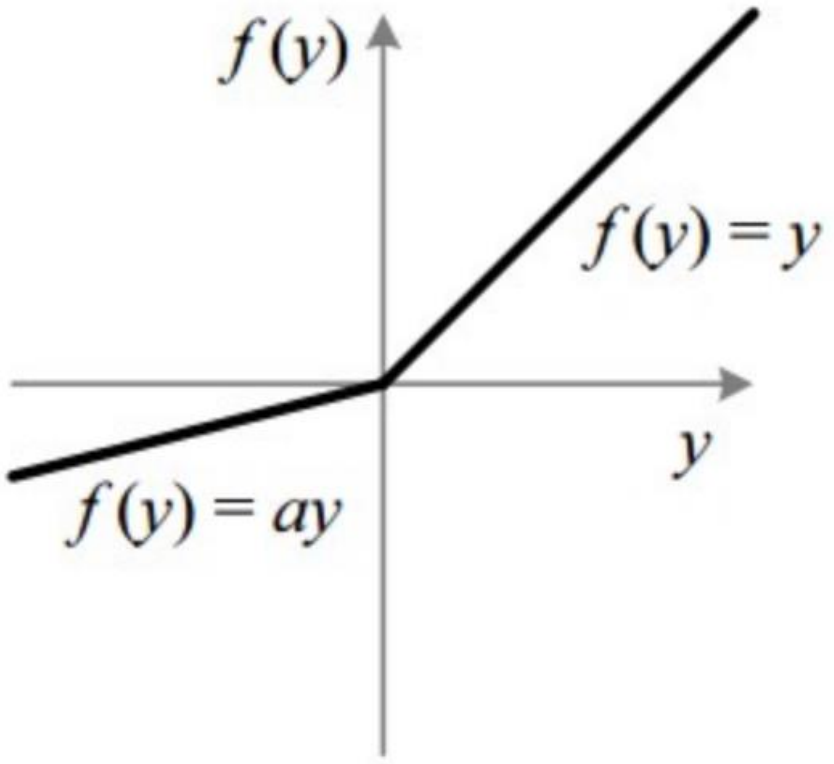
- Veoma popularan izbor
- Neće sauturisati u pozitivnoj regiji
- U praksi mnogo brže konvergira od sigmoida i tahn
- Bolja aproksimacija onog što se dešava kod bioloških neurona

ReLU – Mane

- Nije centriran oko nule
- Kod negativnog ulaza dolazi do saturacije
- Neuron može „umreti“ tokom treninga
  - Loša inicijalizacija – nikada ne dobijemo ulaz koji ih aktivira
    - U praksi, neuroni se inicijalizuju sa blago pozitivnim bijasima (npr. 0.01)
  - Veliki *learning rate*
  - 10-20% mreže tipično budu „mrtvi“ neuroni



Leaky ReLU



- Ne dolazi do saturacije
- Efikasno izračunavanje
- Konvergira mnogo brže od sigmoida i tanh
- Neuroni ne „umiru“

- Leaky ReLU

$$f(x) = \max(0.01x, x)$$

- Parametric Rectifier (PReLU)

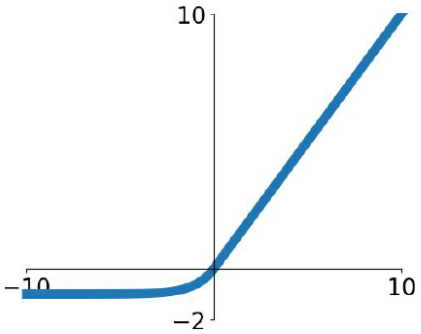
$$f(x) = \max(\alpha x, x)$$

ELU

Activation Functions

[Clevert et al., 2015]

Exponential Linear Units (ELU)



- All benefits of ReLU
- Closer to zero mean outputs
- Negative saturation regime compared with Leaky ReLU adds some robustness to noise

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

- Computation requires exp()

Maxout

Maxout "Neuron" [Goodfellow et al., 2013]

- Does not have the basic form of dot product -> nonlinearity
- Generalizes ReLU and Leaky ReLU
- Linear Regime! Does not saturate! Does not die!

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

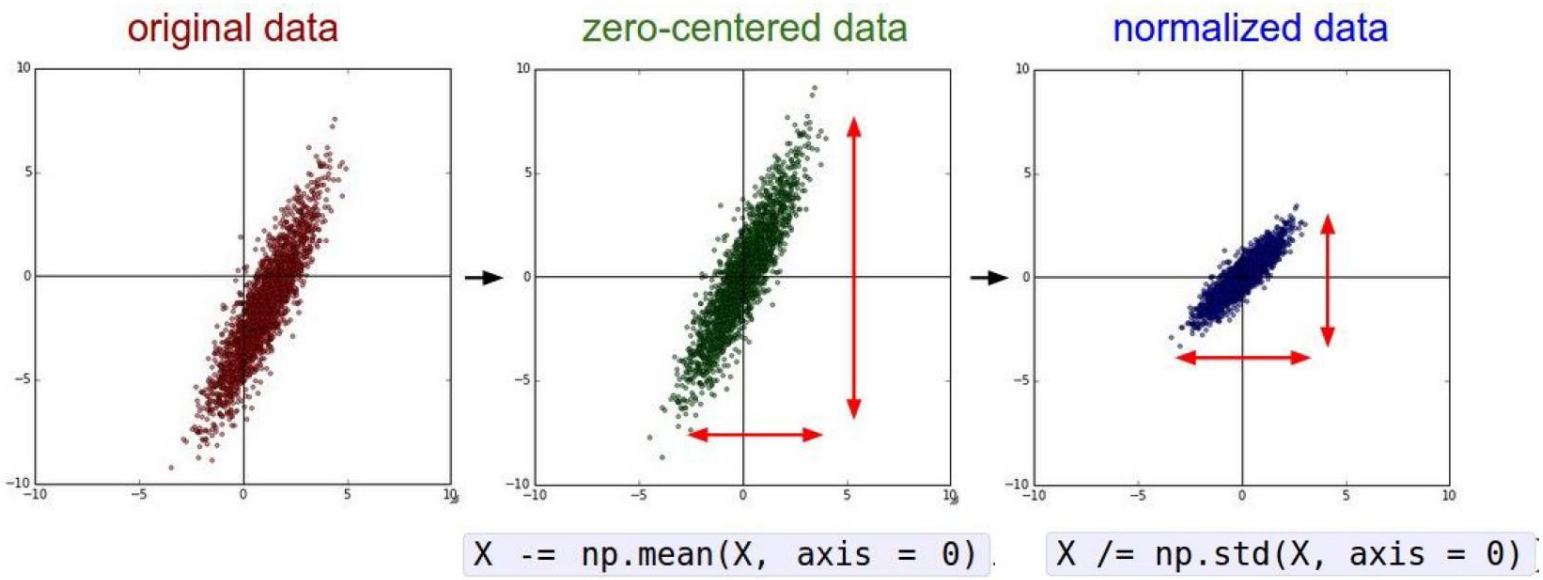
Problem: doubles the number of parameters/neuron :(

U praksi

- Koristite ReLU. Pažljivo odaberite *learning rate*
- Isprobajte Leaky ReLU/Maxout/ELU
- Isprobajte tanh (ali ne očekujte previše)
- Nemojte koristiti sigmoid

Pretprocesiranje podataka

- Za slike, centriramo oko 0, ali nije neophodna normalizacija – intenziteti piksela se kreću u uporedivim opsezima



(Assume X [NxD] is data matrix,
each example in a row)

Inicijalizacija težina

- Neka je aktivaciona funkcija tanh

- Šta ako postavimo $w = 0$?
  - Svi neuroni će imati istu izlaznu vrednost, dobiće isti gradijent i ažuriraće se svi na isti način
  - Ne možemo razbiti simetriju

- Mali slučajni brojevi
  - Gausijani sa srednjom vrednošću 0 u standardnom devijacijom 0.01
  - Radi za manje mreže, ali ne najbolje za duboke. Pošto u svakom sloju množimo sa malim w, aktivacije će u dubljim slojevima postati bliske 0

- Veći slučajni brojevi (1 umesto 0.01)
  - Gotovo svi neuroni su saturisani, ili -1 ili +1 – svi gradijenti će biti 0

Inicijalizacija težina

- Xavier initialization [Glorot et al., 2010]
  - `W = np.random.randn(fan_in, fan_out)/np.sqrt(fan_in)`
  - Ali ne radi za ReLU

- He et al. 2015
  - `W = np.random.randn(fan_in, fan_out)/np.sqrt(fan_in/2)`

- U praksi, inicijalizacija je veoma važna
  - Probajte Xavier, ako ne radi – isprobajte druge metode

Inicijalizacija težina

Proper initialization is an active area of research…

Understanding the difficulty of training deep feedforward neural networks by Glorot and Bengio, 2010

Exact solutions to the nonlinear dynamics of learning in deep linear neural networks by Saxe et al, 2013

Random walk initialization for training very deep feedforward networks by Sussillo and Abbott, 2014

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification by He et al., 2015

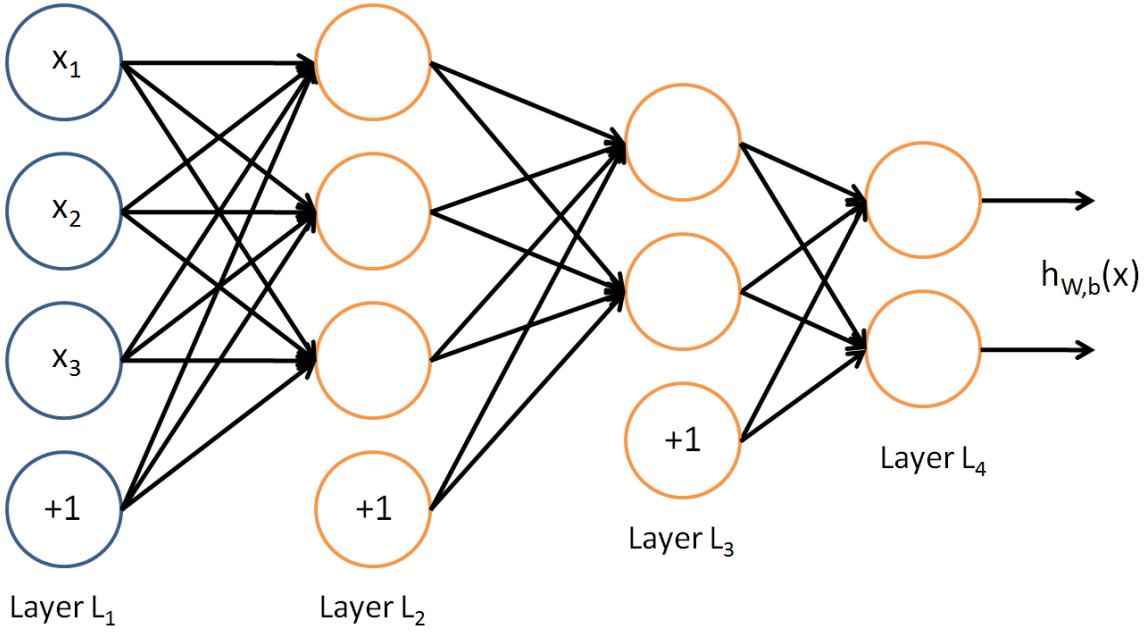
Data-dependent Initializations of Convolutional Neural Networks by Krähenbühl et al., 2015

All you need is a good init, Mishkin and Matas, 2015

…

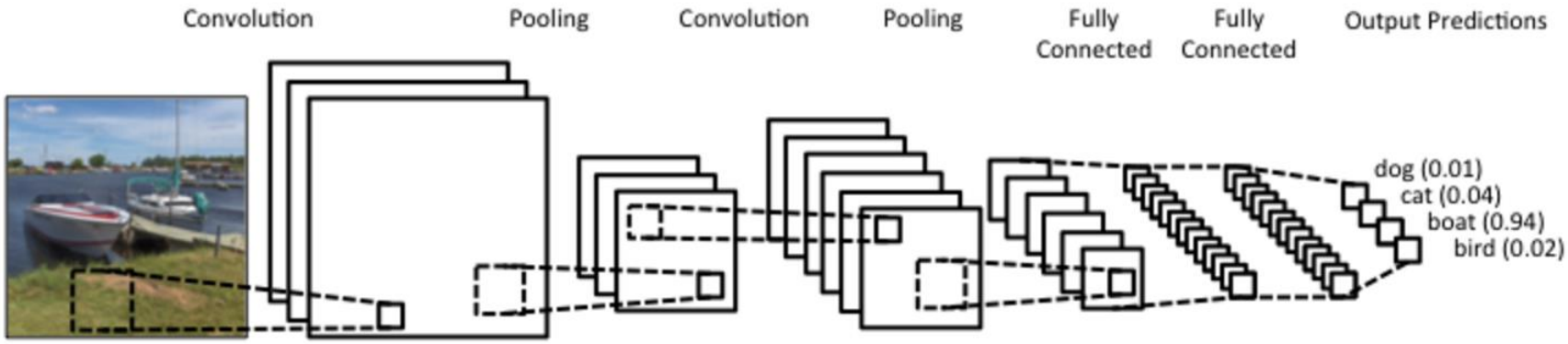
Problem slika visoke rezolucije

- Kod problema računarske vizije, srećemo se sa veoma velikim brojem ulaznih obeležja
  - Slika rezolucije 64×64 ima $64 \cdot 64 \cdot 3 = 12\,228$ ulazna obeležja
  - Slika rezolucije 1000×1000 ima $3 \cdot 10^6$ ulazna obeležja



Konvolucione neuronske mreže

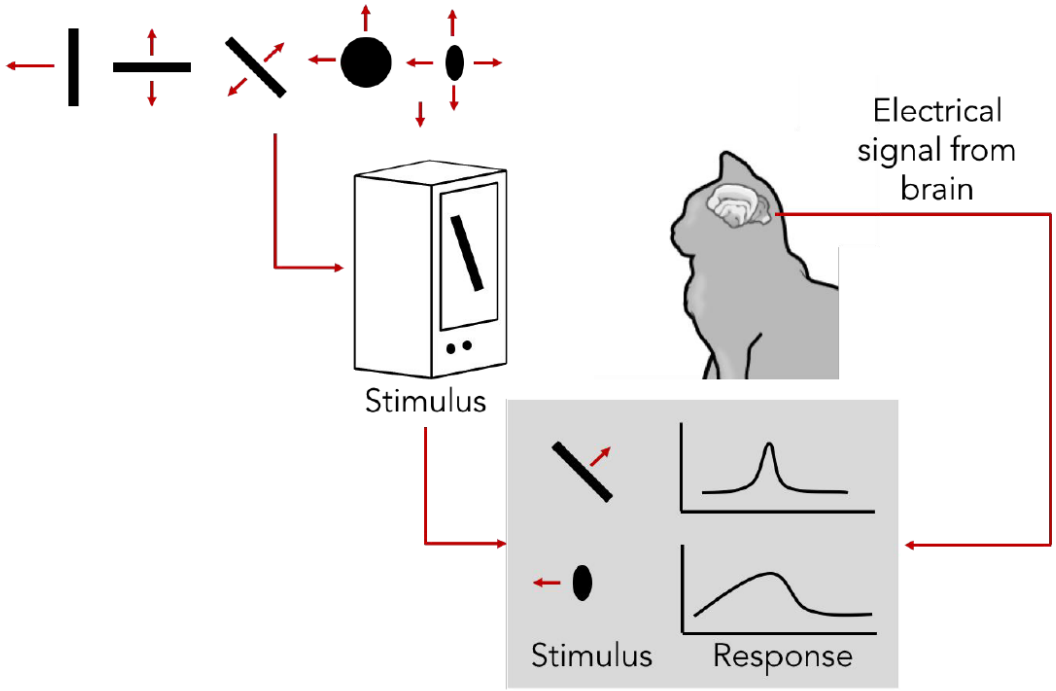
Convolutional Neural Networks (CNN)



Inspiracija za CNN

A bit of history:

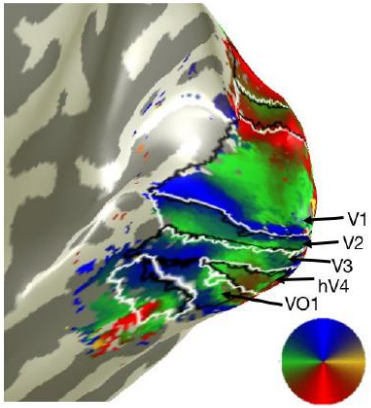
Hubel & Wiesel,
1959
RECEPTIVE FIELDS OF SINGLE NEURONES IN
THE CAT'S STRIATE CORTEX
1962
RECEPTIVE FIELDS, BINOCULAR INTERACTION
AND FUNCTIONAL ARCHITECTURE IN
THE CAT'S VISUAL CORTEX
1968...

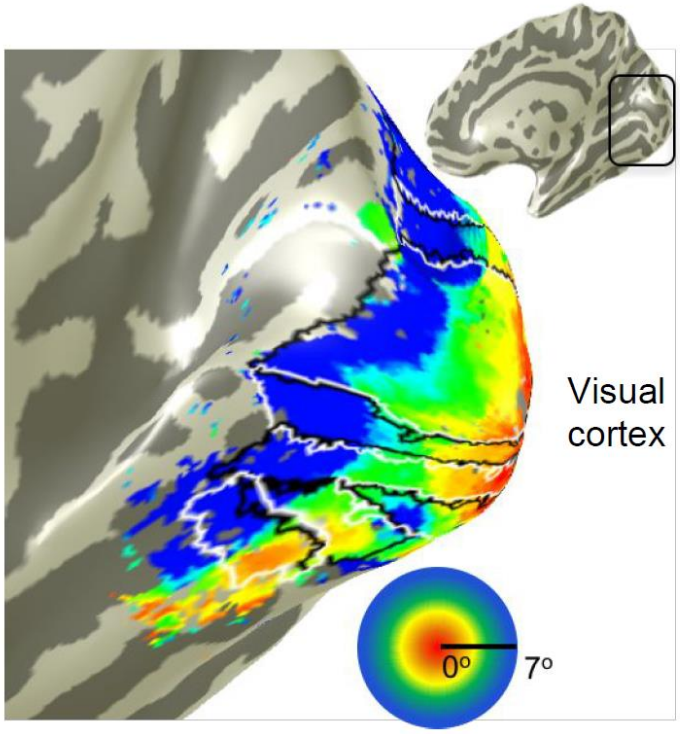


Topologija

A bit of history

Topographical mapping in the cortex:
nearby cells in cortex represent
nearby regions in the visual field





Retinotopy images courtesy of Jesse Gomez in the Stanford Vision & Perception Neuroscience Lab.

Hijerarhijska organizacija

Hierarchical organization

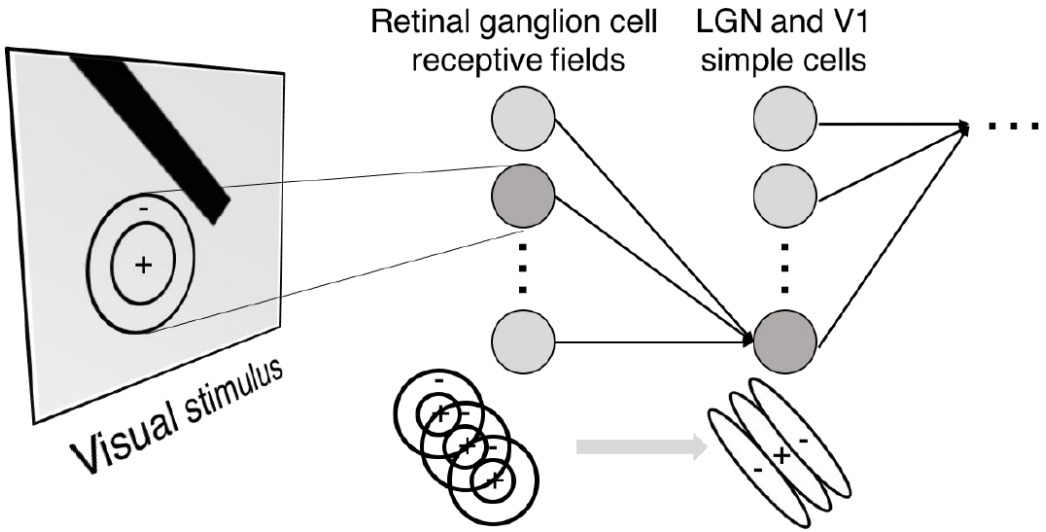
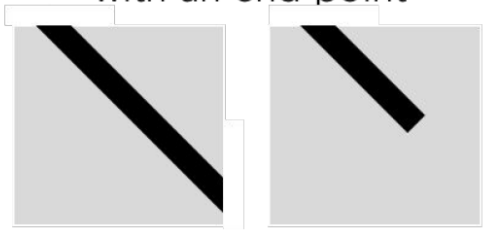


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point



No response

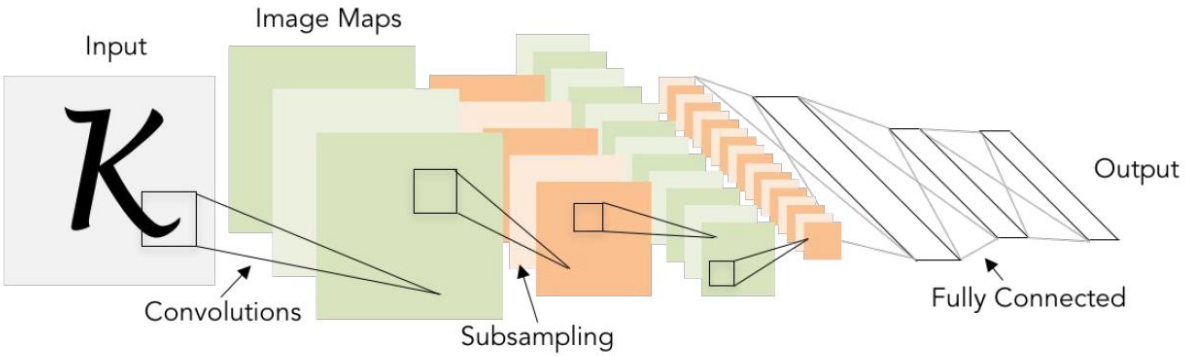
Response (end point)

CNN istorija

Prvi primer primene backpropagation za treniranje CNN

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

CNN istorija

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

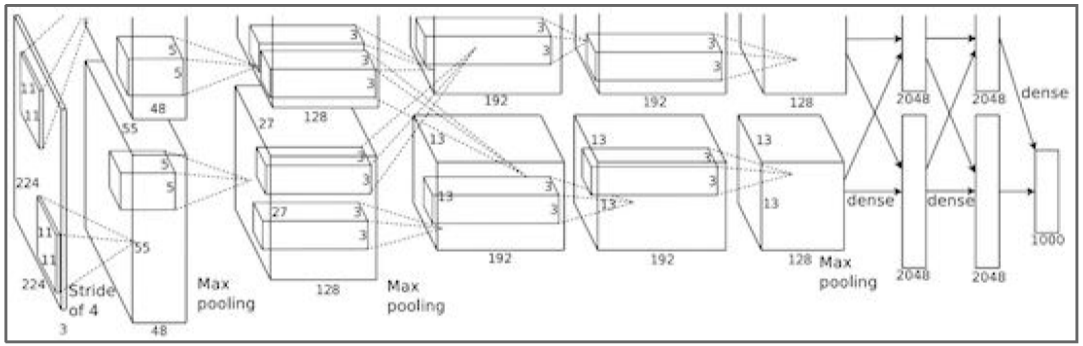
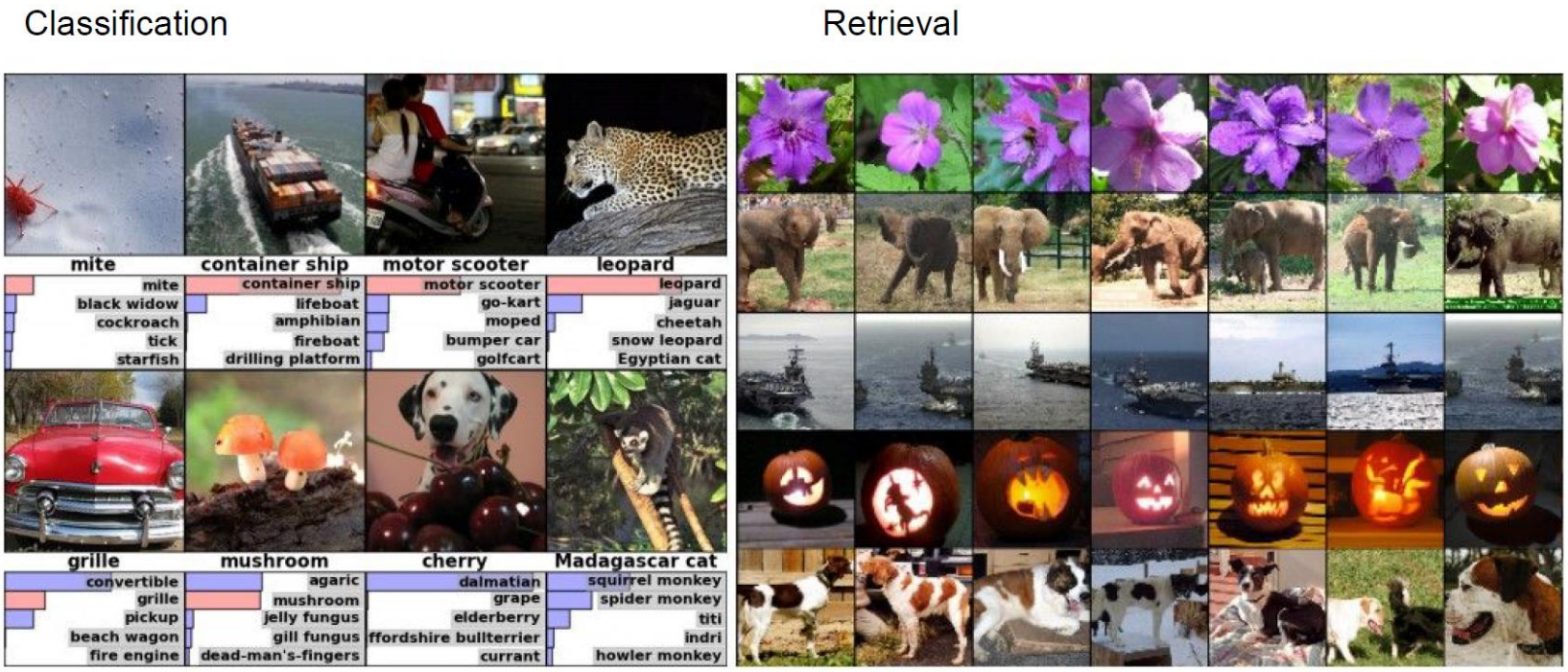


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

CNN danas

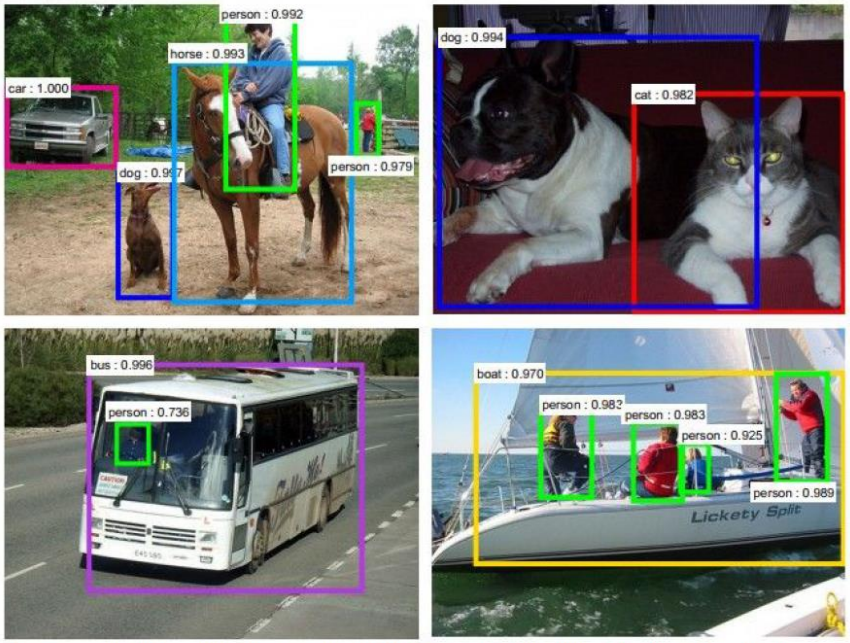
Klasifikacija i dobavljanje slika



CNN danas

Lokalizacija (gde se objekat nalazi na slici) i segmentacija

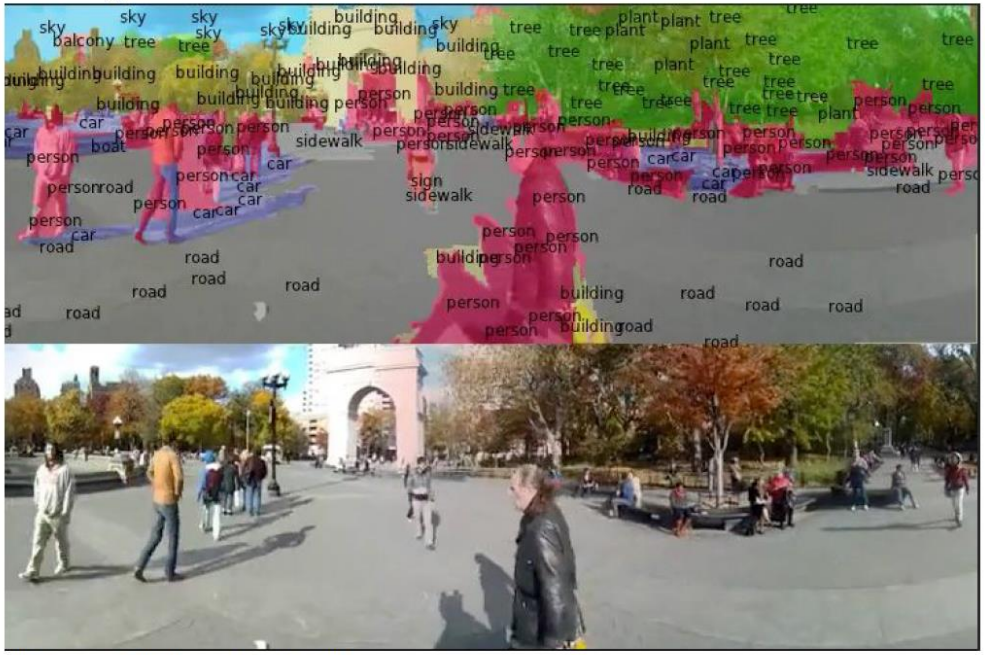
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

CNN danas

Samovozeća kola

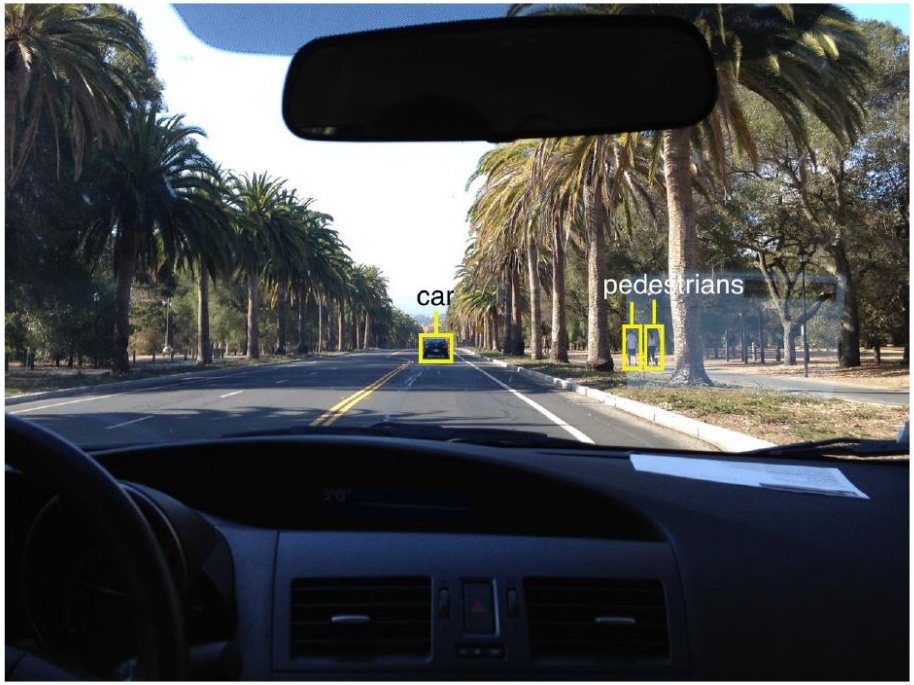


Photo by Lane McIntosh. Copyright CS231n 2017.

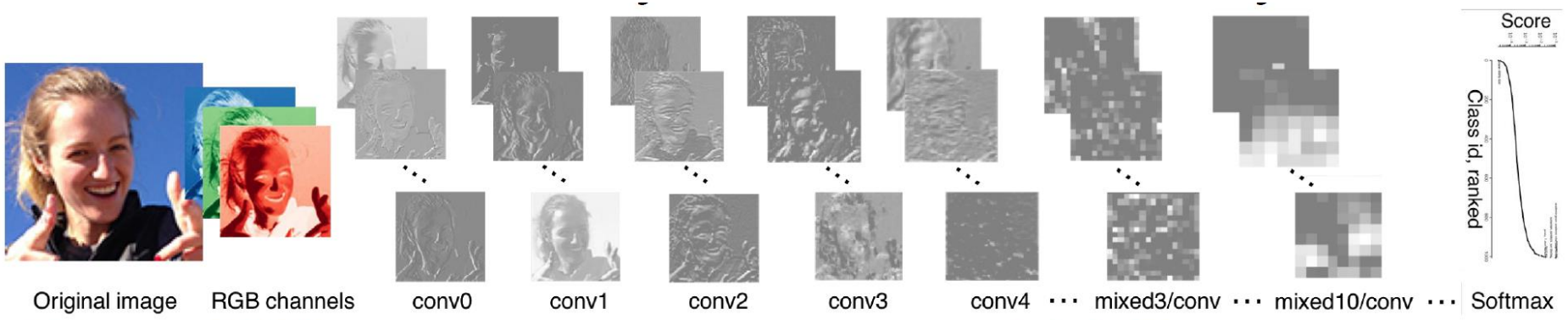
self-driving cars

NVIDIA Tesla line
(these are the GPUs on rye01.stanford.edu)

Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

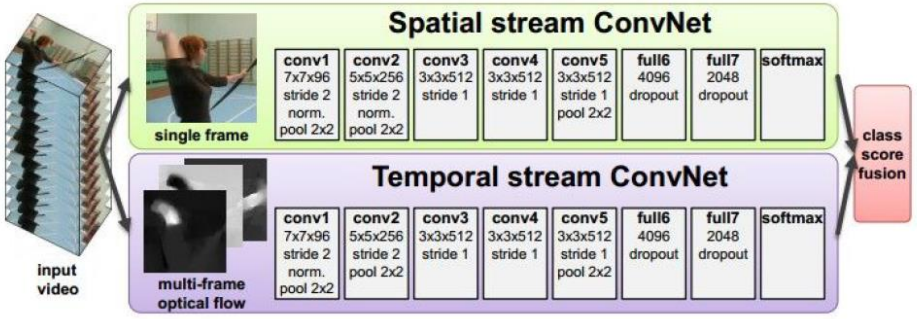
CNN danas

Prepoznavanje lica, klasifikacija videa



[Taigman et al. 2014]

Activations of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.



[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

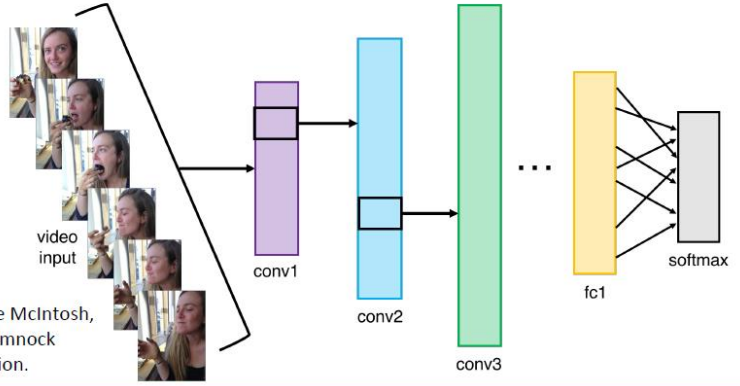


Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

CNN danas

Prepoznavanje poze, igranje igara

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

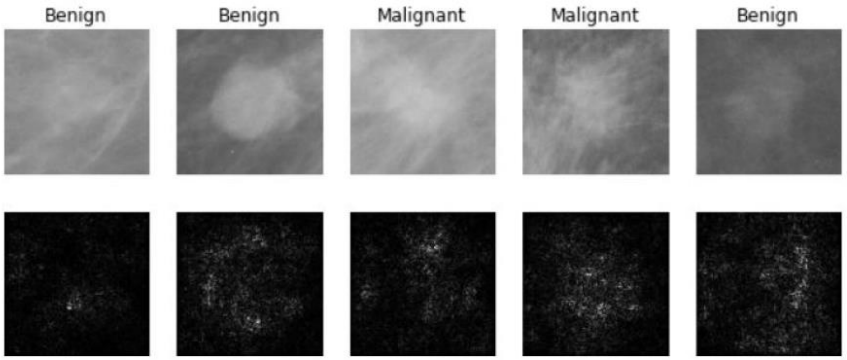
[Toshev, Szegedy 2014]

[Guo et al. 2014]

Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

CNN danas

Interpretacija i dijagnoza medicinskih slika, klasifikacija galaksija, prepoznavanje saobraćajnih znakova



[Levy et al. 2016]

Figure copyright Levy et al. 2016.
Reproduced with permission.

[Dieleman et al. 2014]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

[Sermanet et al. 2011]
[Ciresan et al.]

Photos by Lane McIntosh.
Copyright CS231n 2017.

CNN danas

This image by Christin Khan is in the public domain and originally came from the U.S. NOAA.

Whale recognition, Kaggle Challenge

Photo and figure by Lane McIntosh; not actual example from Mnih and Hinton, 2010 paper.

Mnih and Hinton, 2010

CNN danas

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

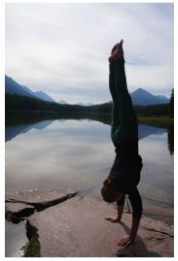
Minor errors

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Somewhat related

A woman is holding a cat in her hand



A woman standing on a beach holding a surfboard

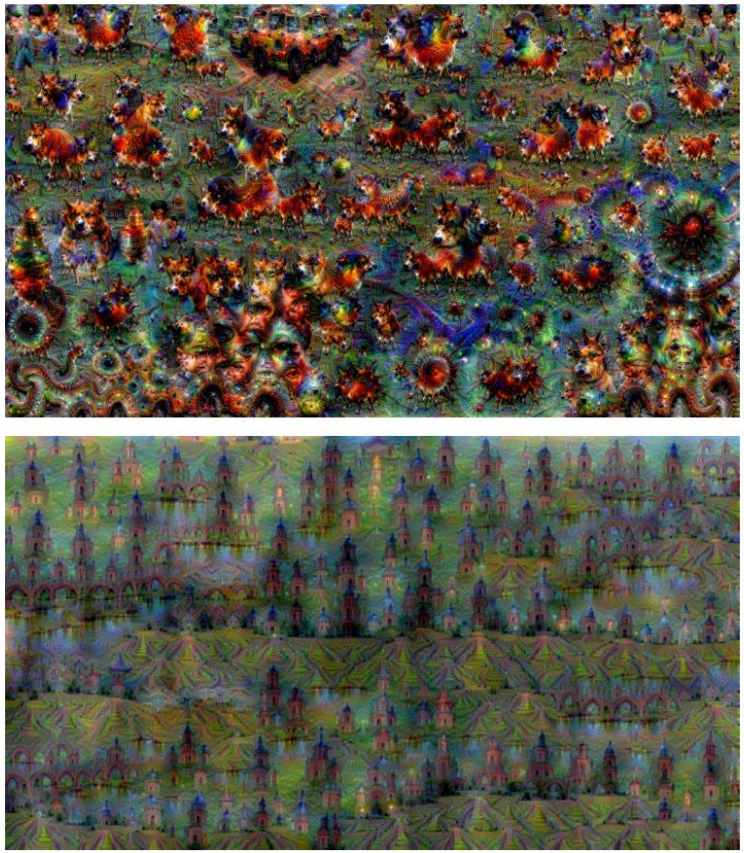
Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

Captions generated by Justin Johnson using Neuraltalk2

CNN danas

Umetnička dela, transfer stila



Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.

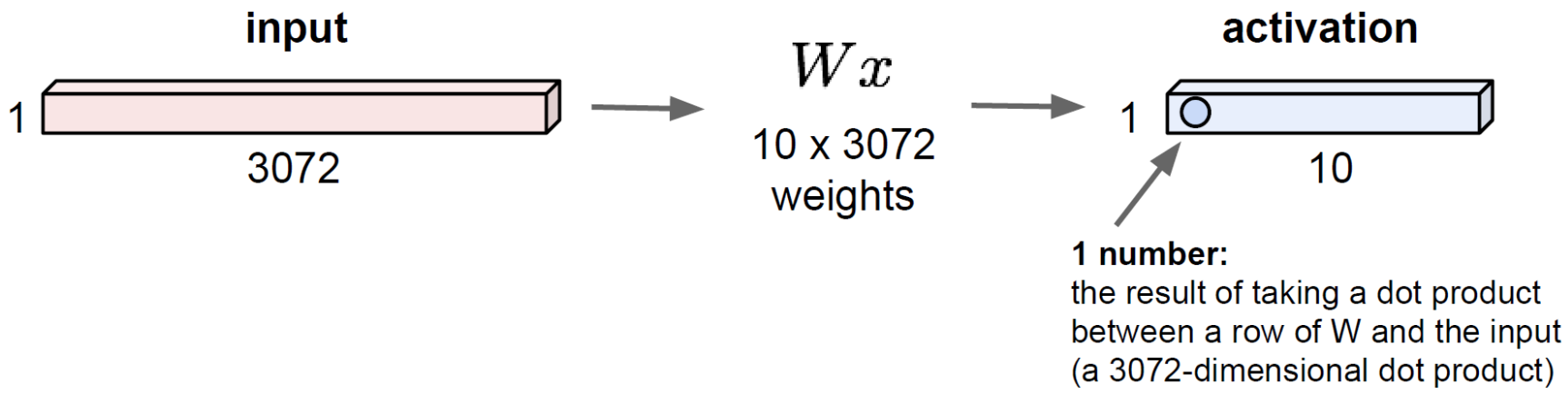
Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Potpuno povezani sloj

Fully Connected Layer

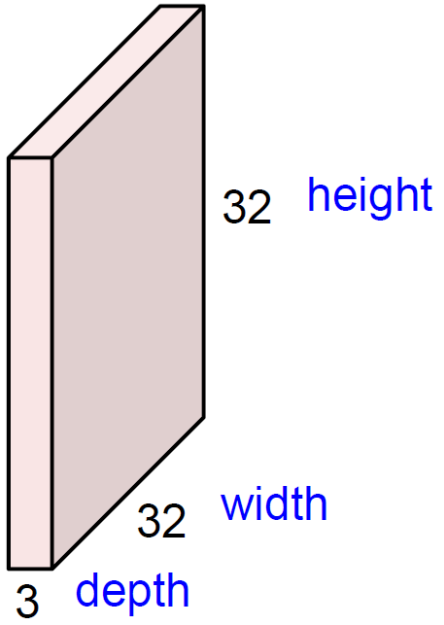
32x32x3 image -> stretch to 3072 x 1



Konvolucioni sloj

Convolution Layer

32x32x3 image -> preserve spatial structure

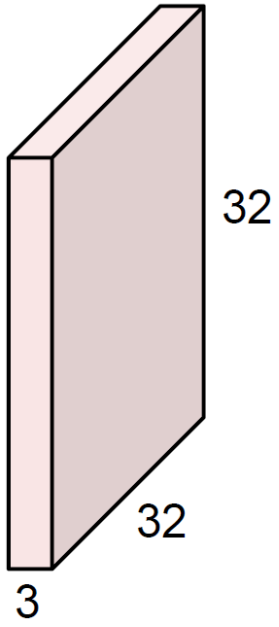


Konvolucioni sloj

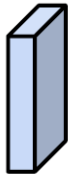
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



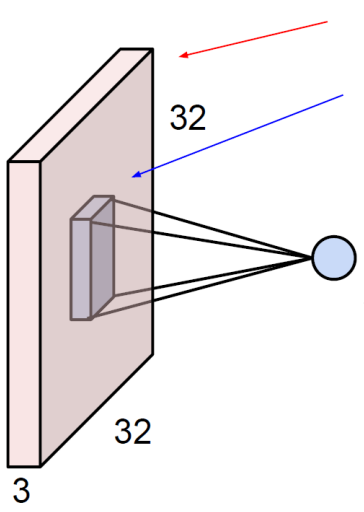
5x5x3 filter



Težine koje treba da naučimo

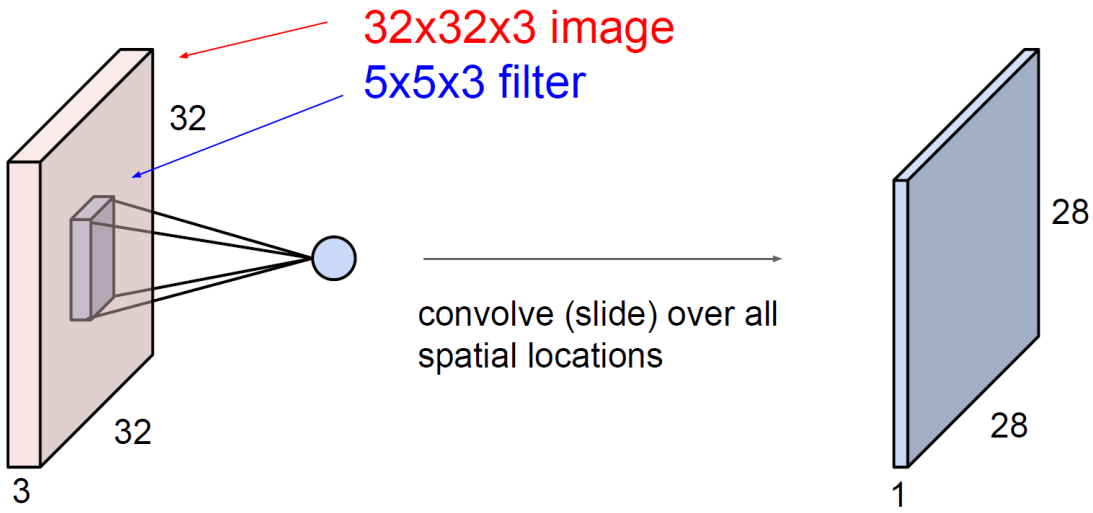
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Konvolucioni sloj



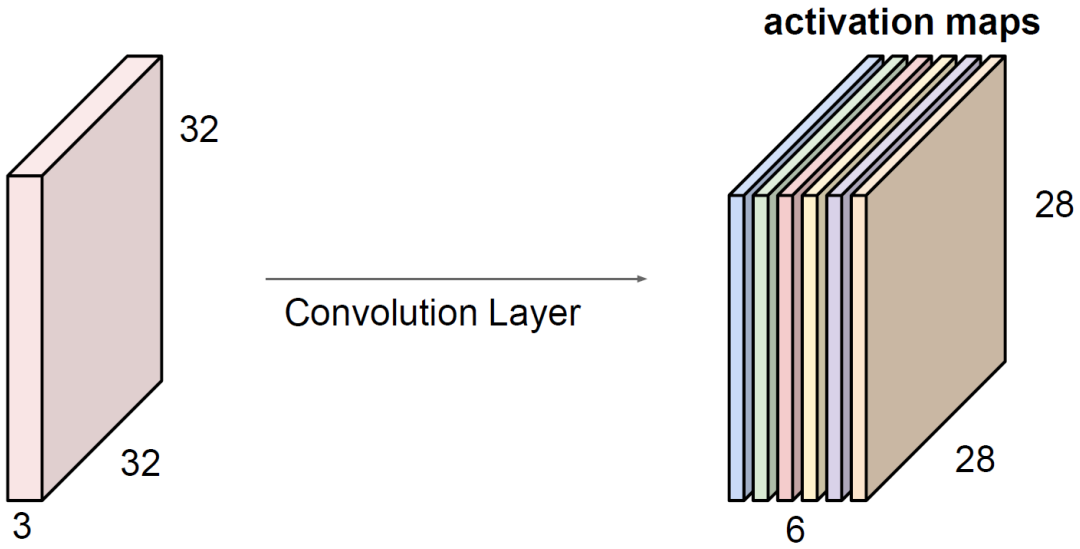
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$



Konvolucioni sloj

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



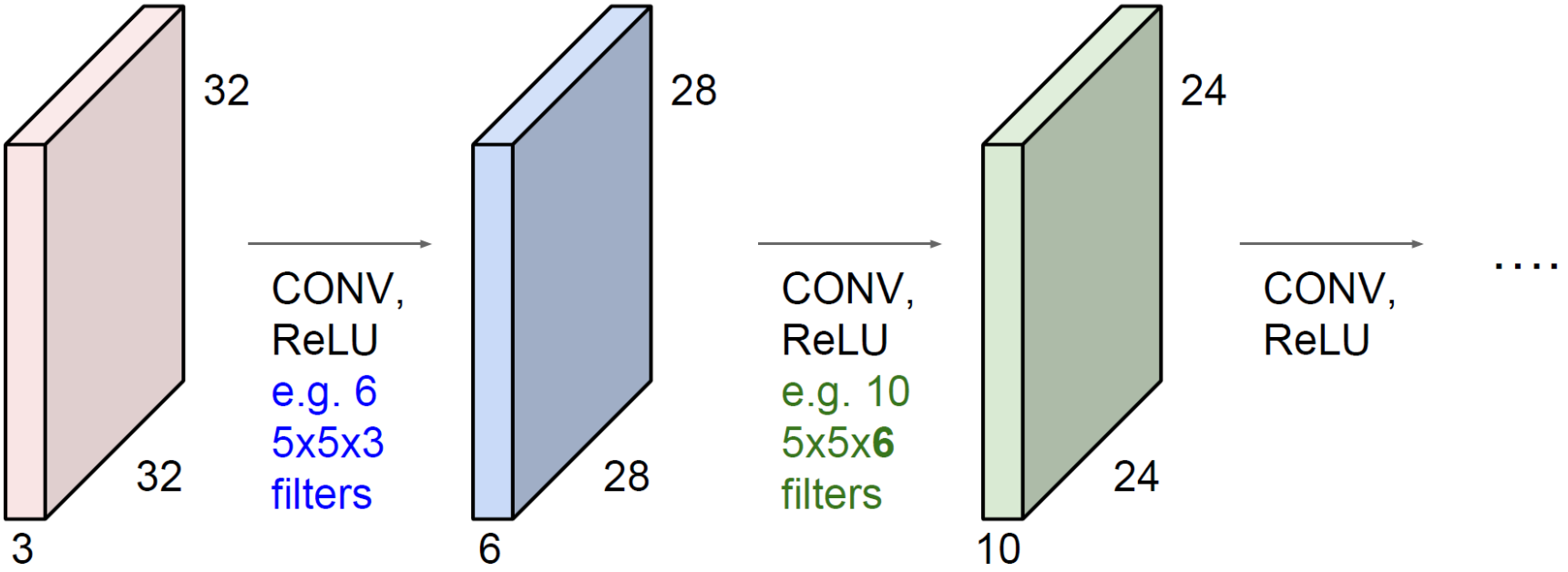
Možemo imati više filtera

- Svaki traži određeni koncept (šablon) u ulazima
- Svaki rezultuje jednom aktivacionom mapom

We stack these up to get a "new image" of size 28x28x6!

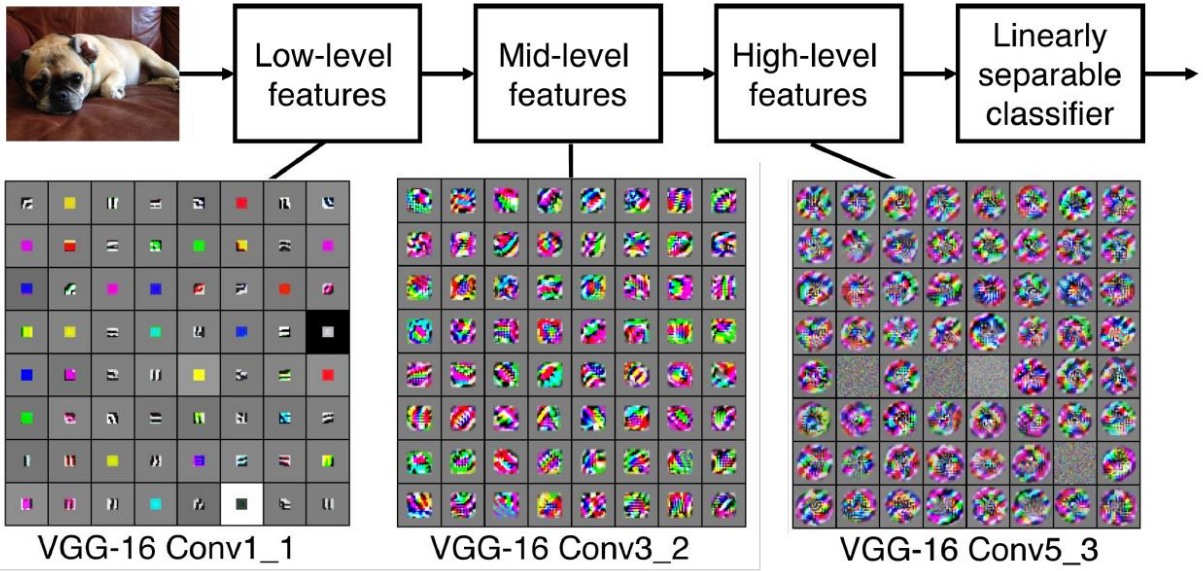
Više konvolucionih slojeva

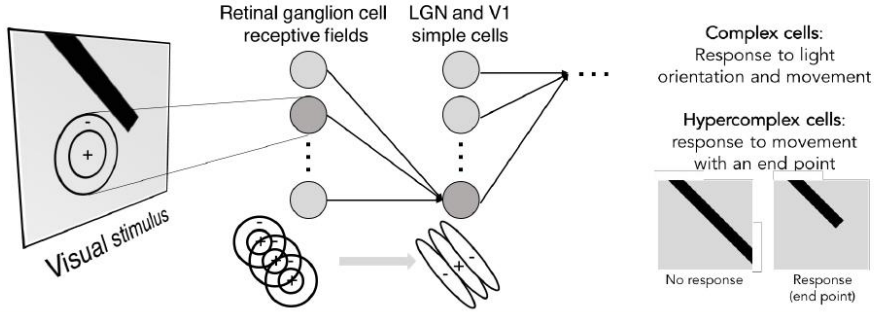
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



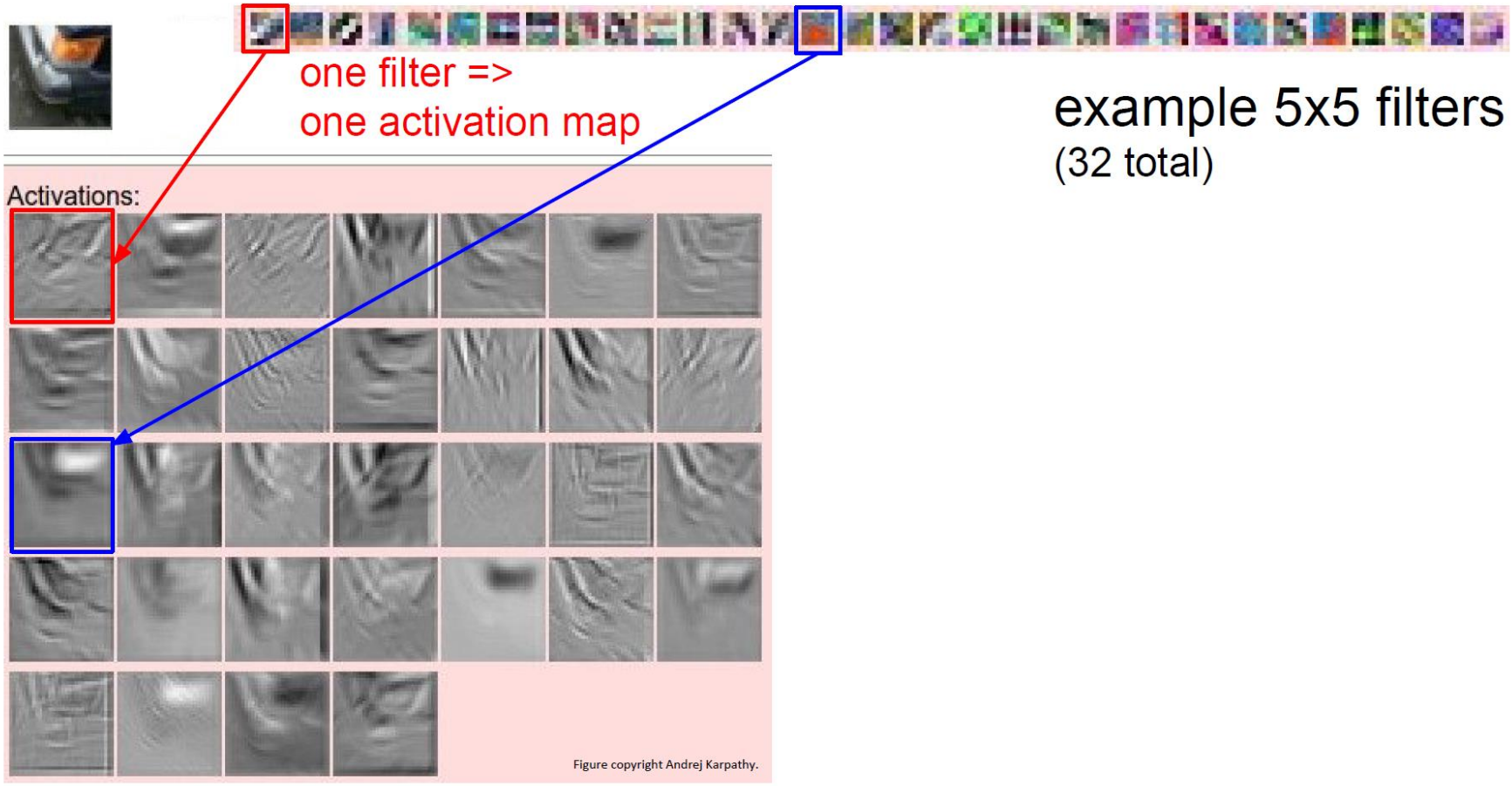
Hijerarhijska organizacija

Preview



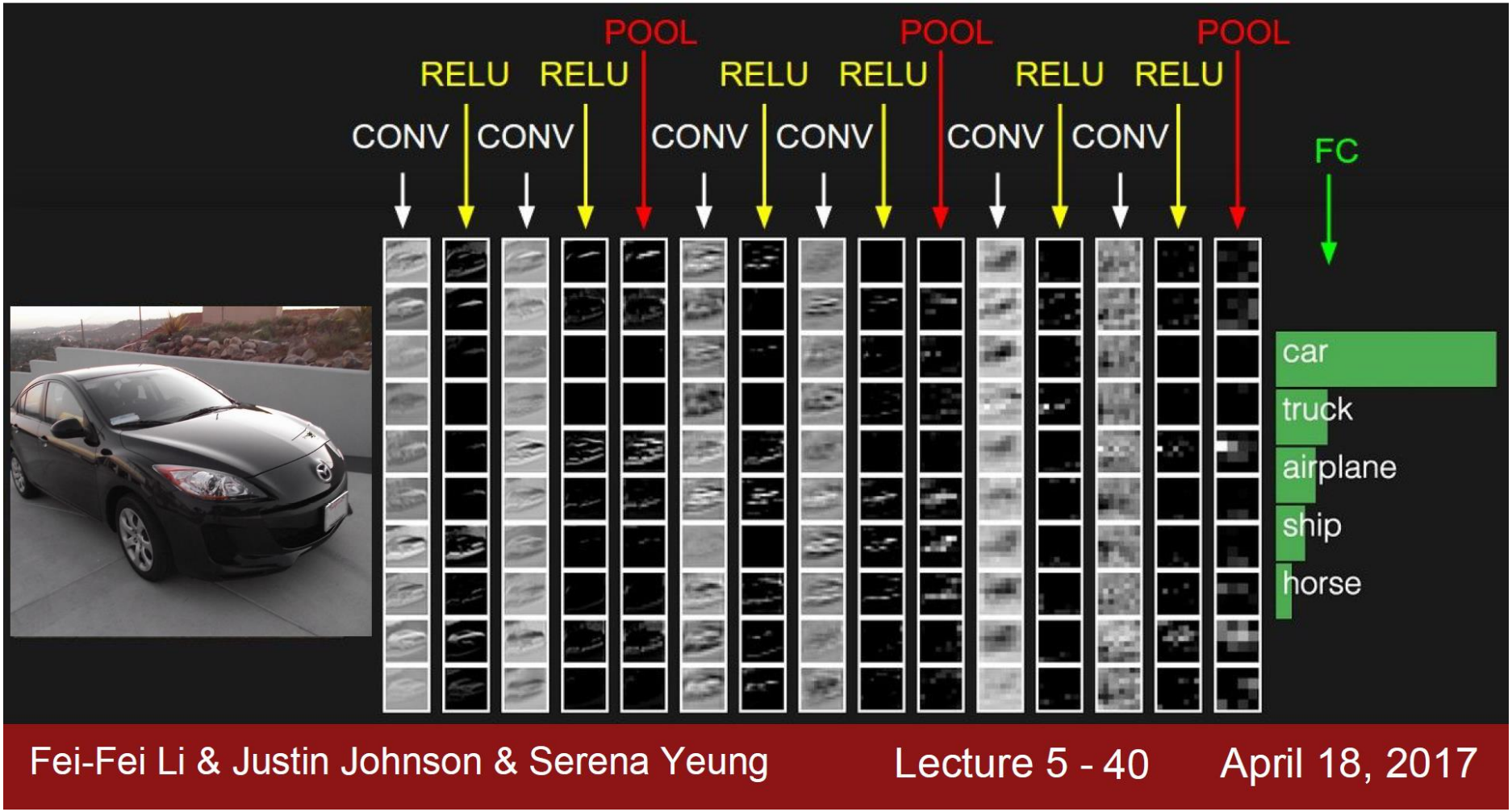


Vizuelizacija



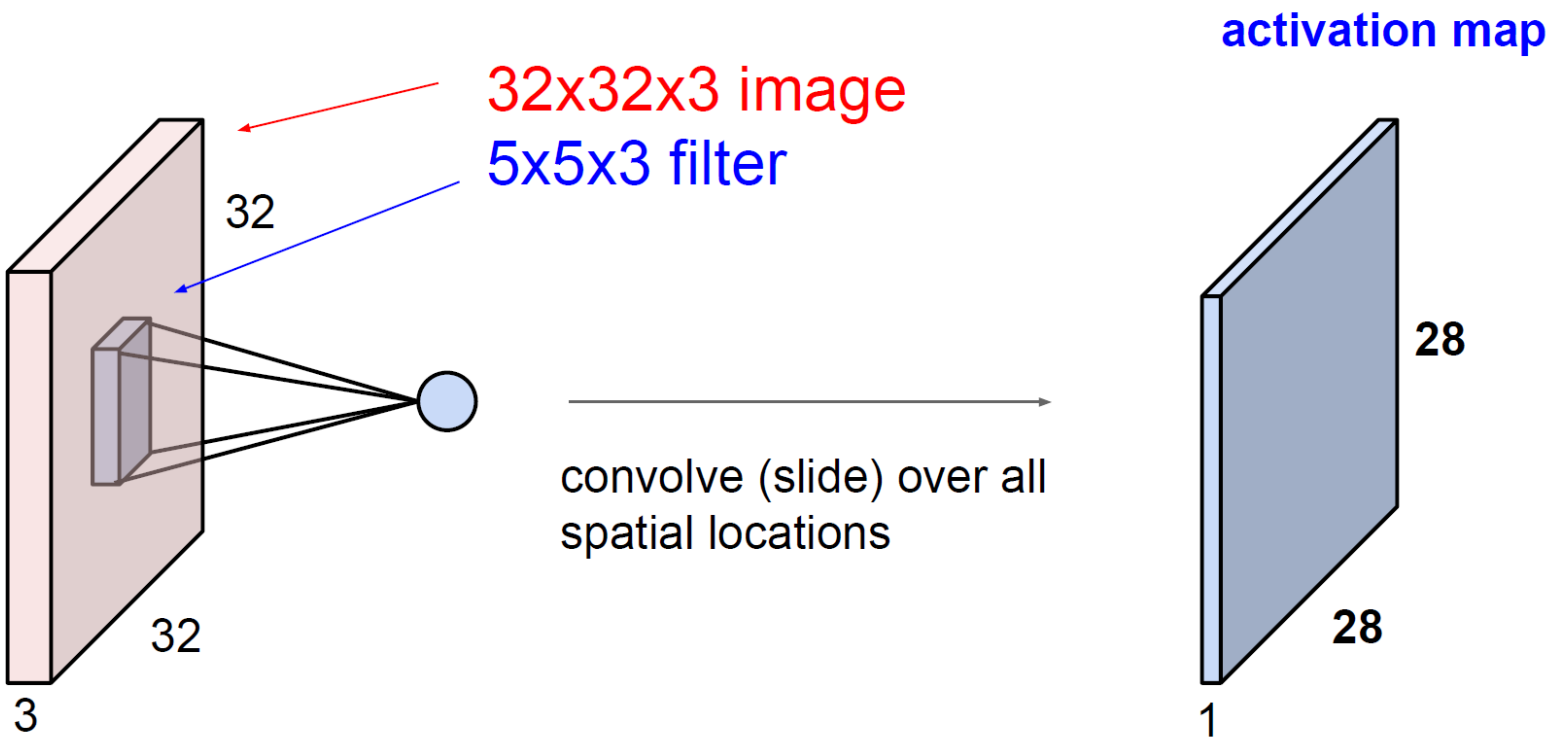
Arhitektura CNN

preview:



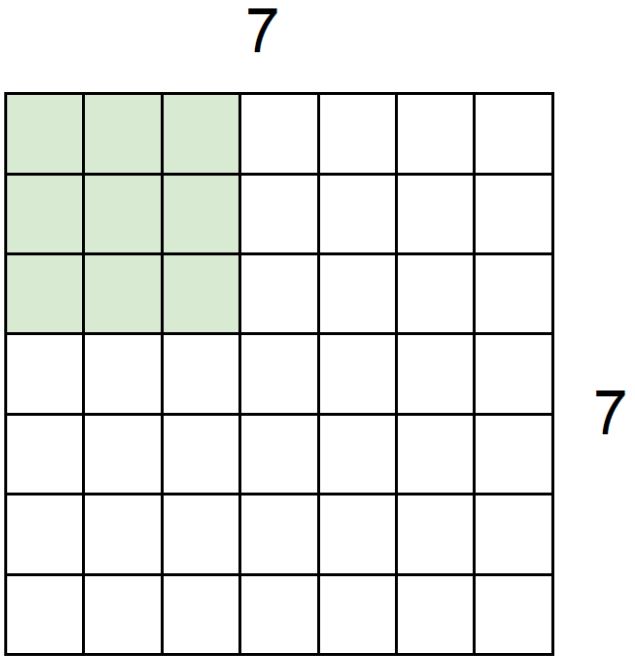
Dimenzije

A closer look at spatial dimensions:



Dimenzije

A closer look at spatial dimensions:

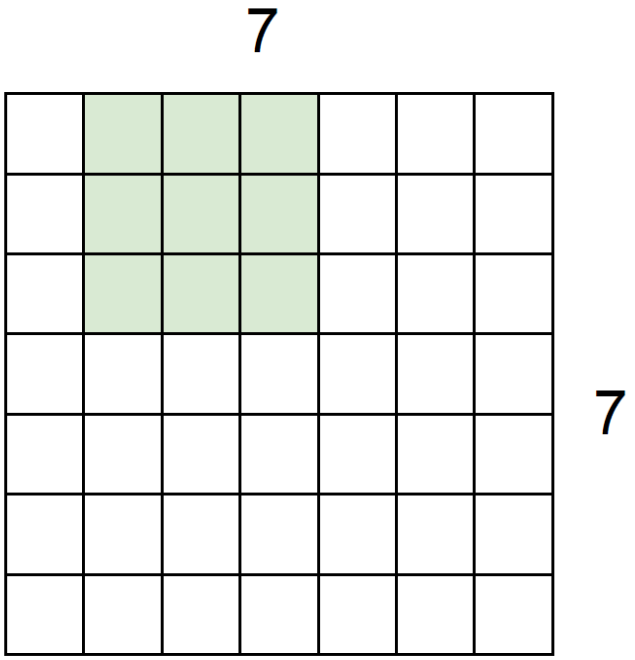


7x7 input (spatially)
assume 3x3 filter

Korak (*stride*) = 1

Dimenzije

A closer look at spatial dimensions:

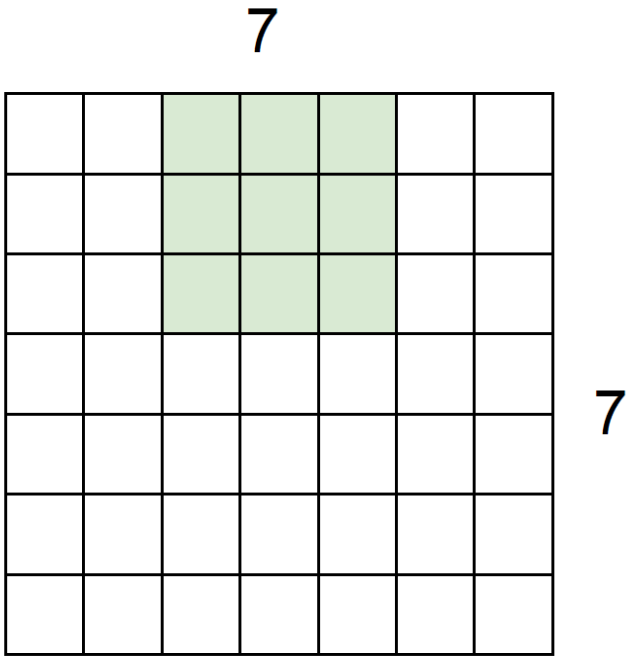


7x7 input (spatially)
assume 3x3 filter

Korak (*stride*) = 1

Dimenzije

A closer look at spatial dimensions:

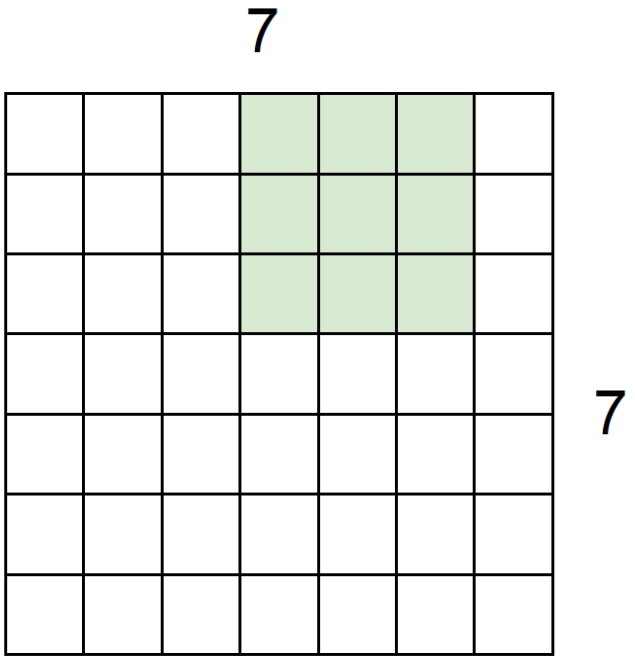


7x7 input (spatially)
assume 3x3 filter

Korak (*stride*) = 1

Dimenzije

A closer look at spatial dimensions:

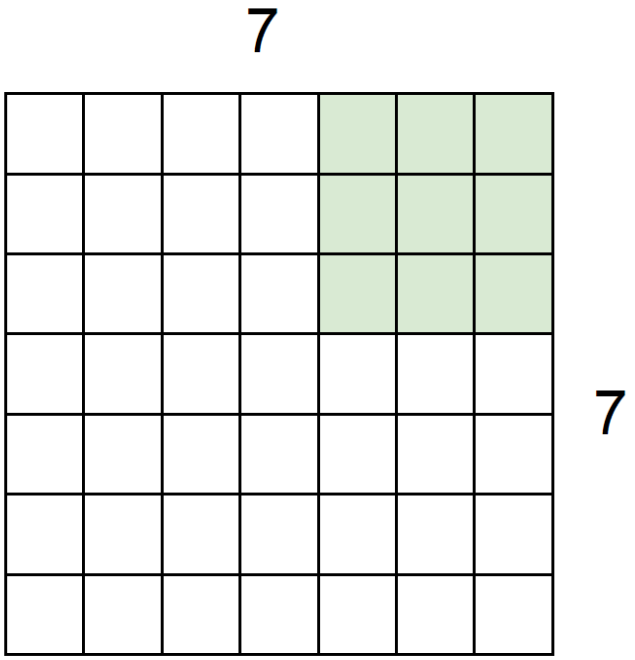


7x7 input (spatially)
assume 3x3 filter

Korak (*stride*) = 1

Dimenzije

A closer look at spatial dimensions:



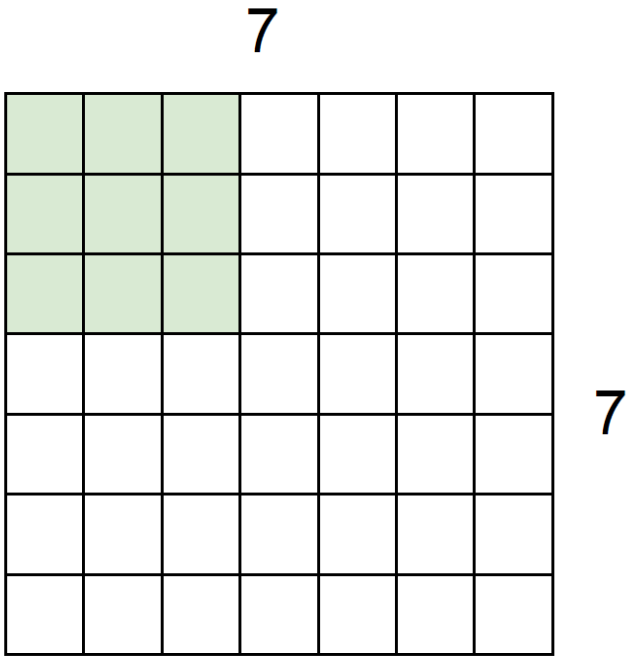
7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Korak (*stride*) = 1

Dimenzije

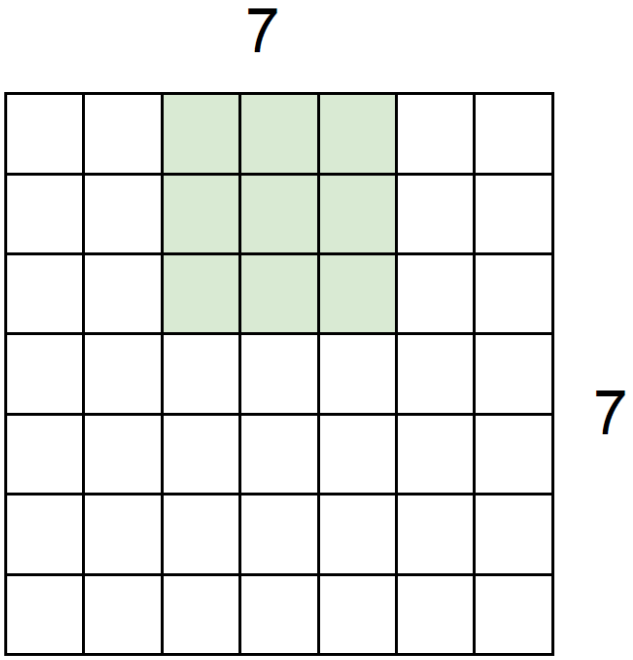
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Dimenzije

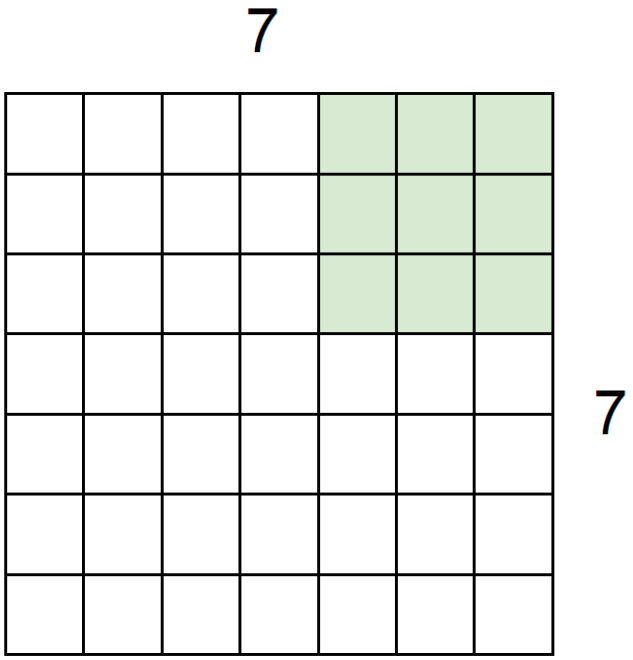
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Dimenzije

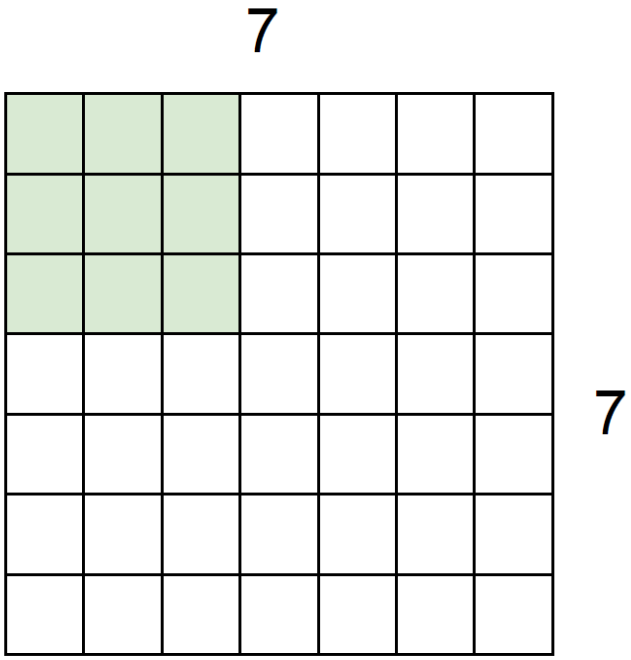
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Dimenzije

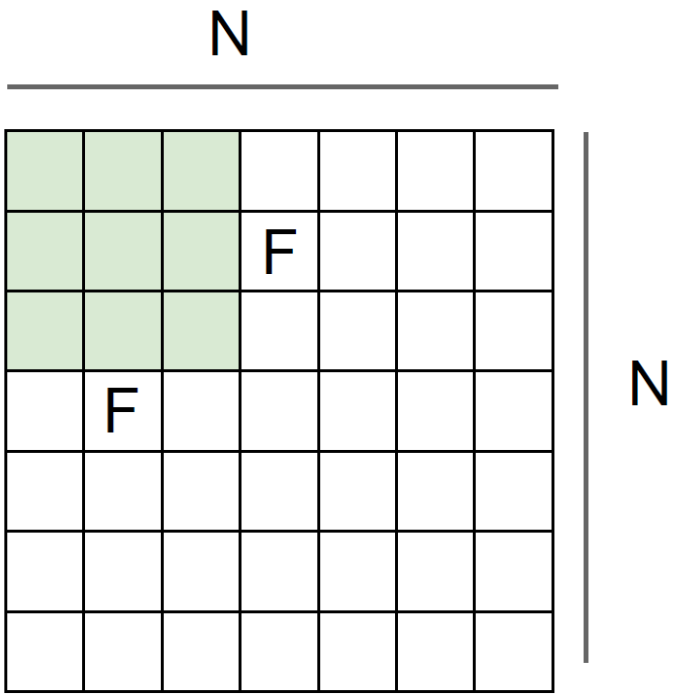
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Dimenzije

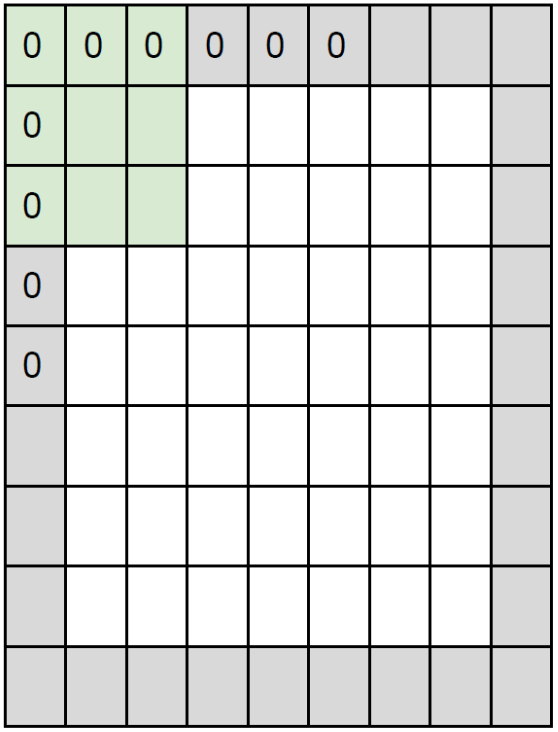


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Proširivanje ivica nulama (*zero padding*)

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

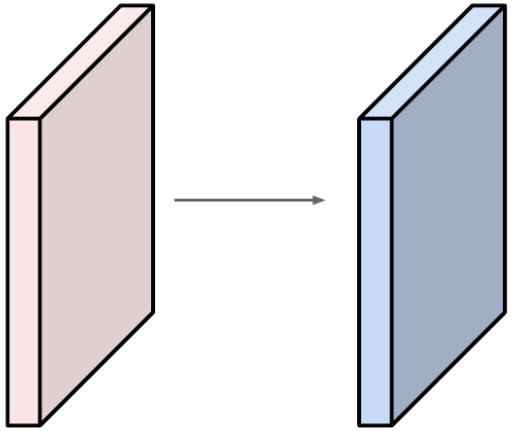
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Hiperparametri

- Veličina filtera
  - Obično se koriste filteri $3 \times 3, 5 \times 5, 7 \times 7$
- *Zero padding*
  - Cilj je da veličina izlaza bude jednaka veličini ulaza
    - Za 3×3, bira se 1
    - Za 5×5, bira se 2
    - …
  - Bez *zero padding*, veličina aktivacionih mapa bi se brzo smanjivala sa brojem slojeva, što je nezgodno ako imamo zaista duboke mreže
- Veličina koraka

Dimenzije

Examples time:

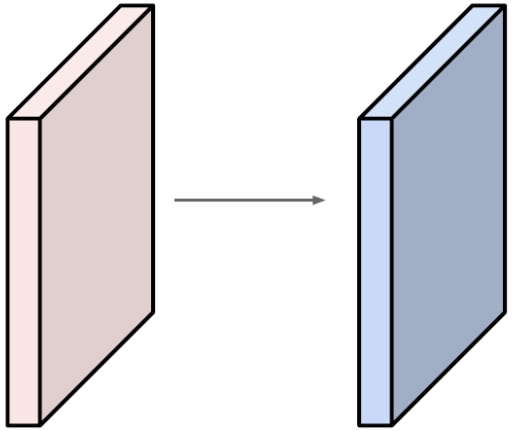


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Broj parametara

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Konvolucioni sloj – sumarizacija

Summary. To summarize, the Conv Layer:

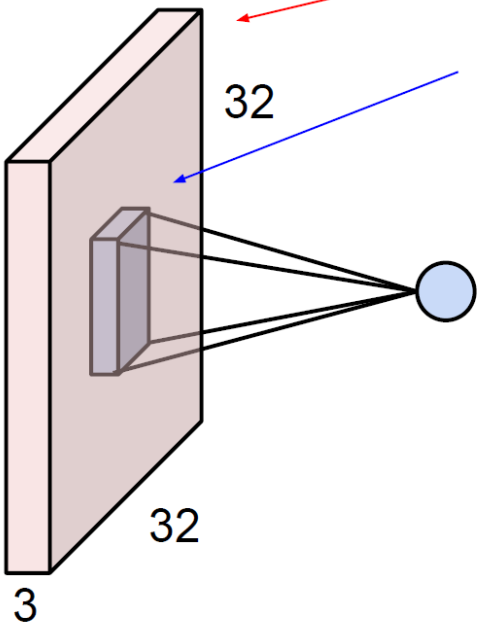
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Common settings:

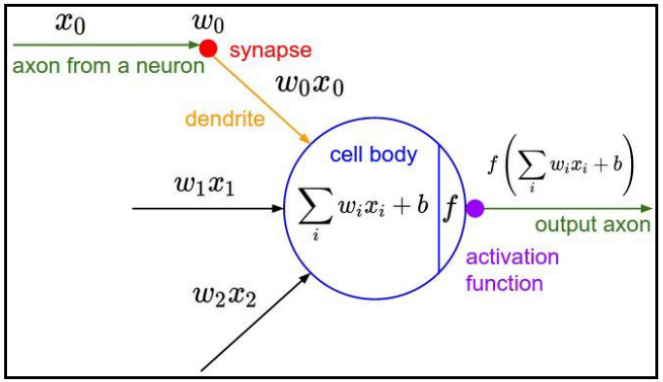
K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Neuron u CNN

The brain/neuron view of CONV Layer



32x32x3 image
5x5x3 filter

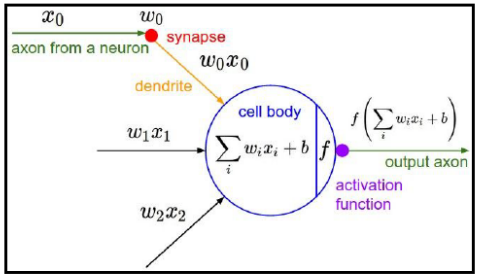


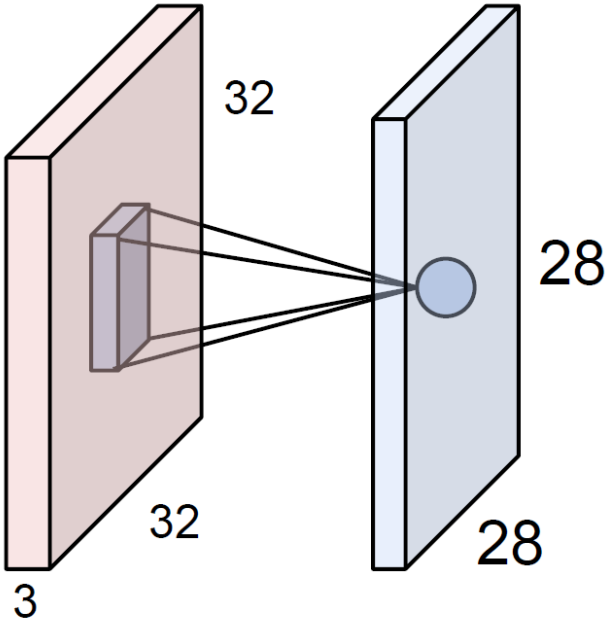
It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

Neuron u CNN

The brain/neuron view of CONV Layer





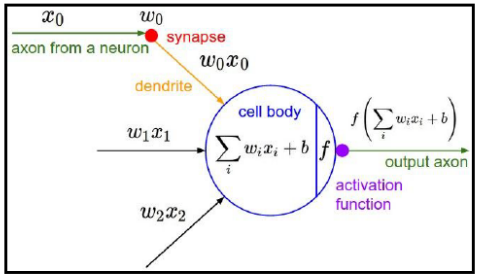
An activation map is a 28x28 sheet of neuron outputs:

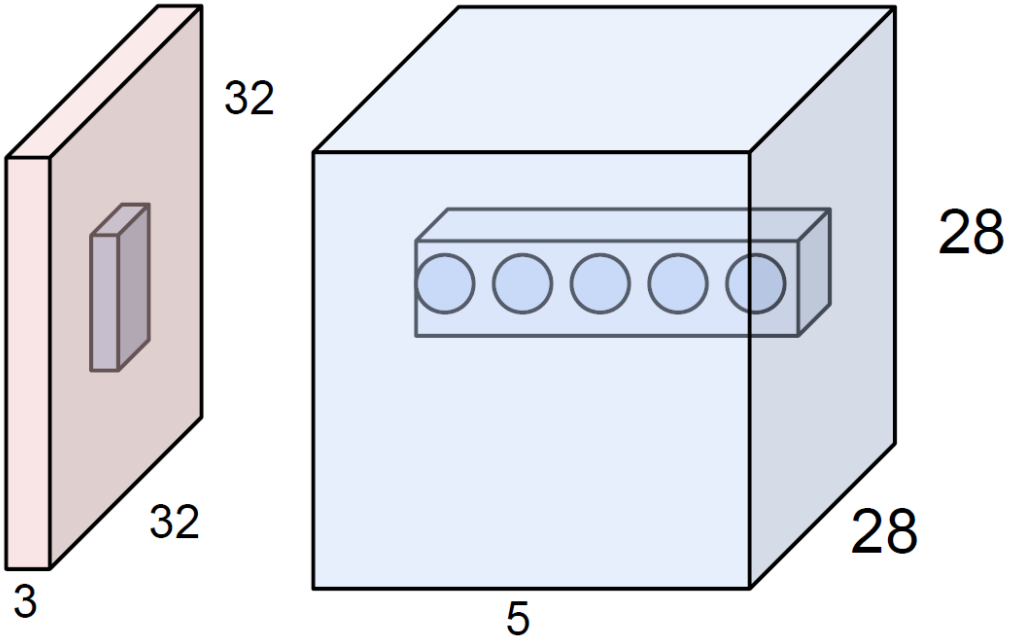
1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

Neuron u CNN

The brain/neuron view of CONV Layer





E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

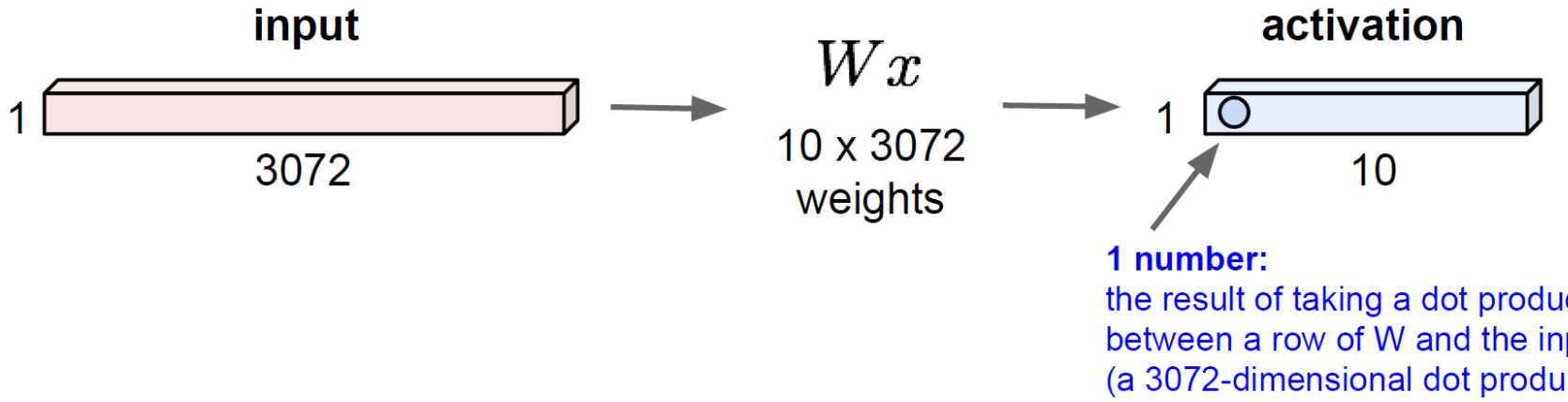
There will be 5 different neurons all looking at the same region in the input volume

Podsetnik: potpuno povezana mreža

Reminder: Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

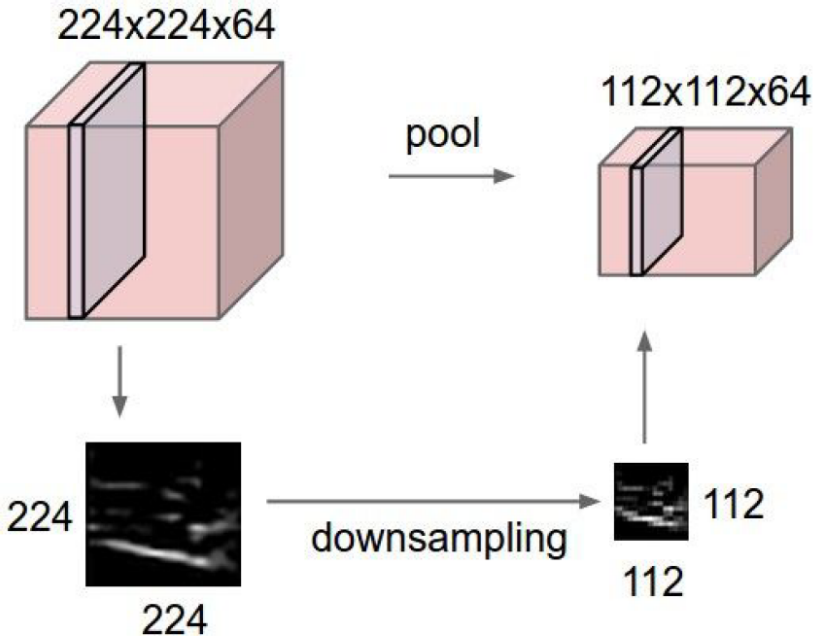
Each neuron looks at the full input volume



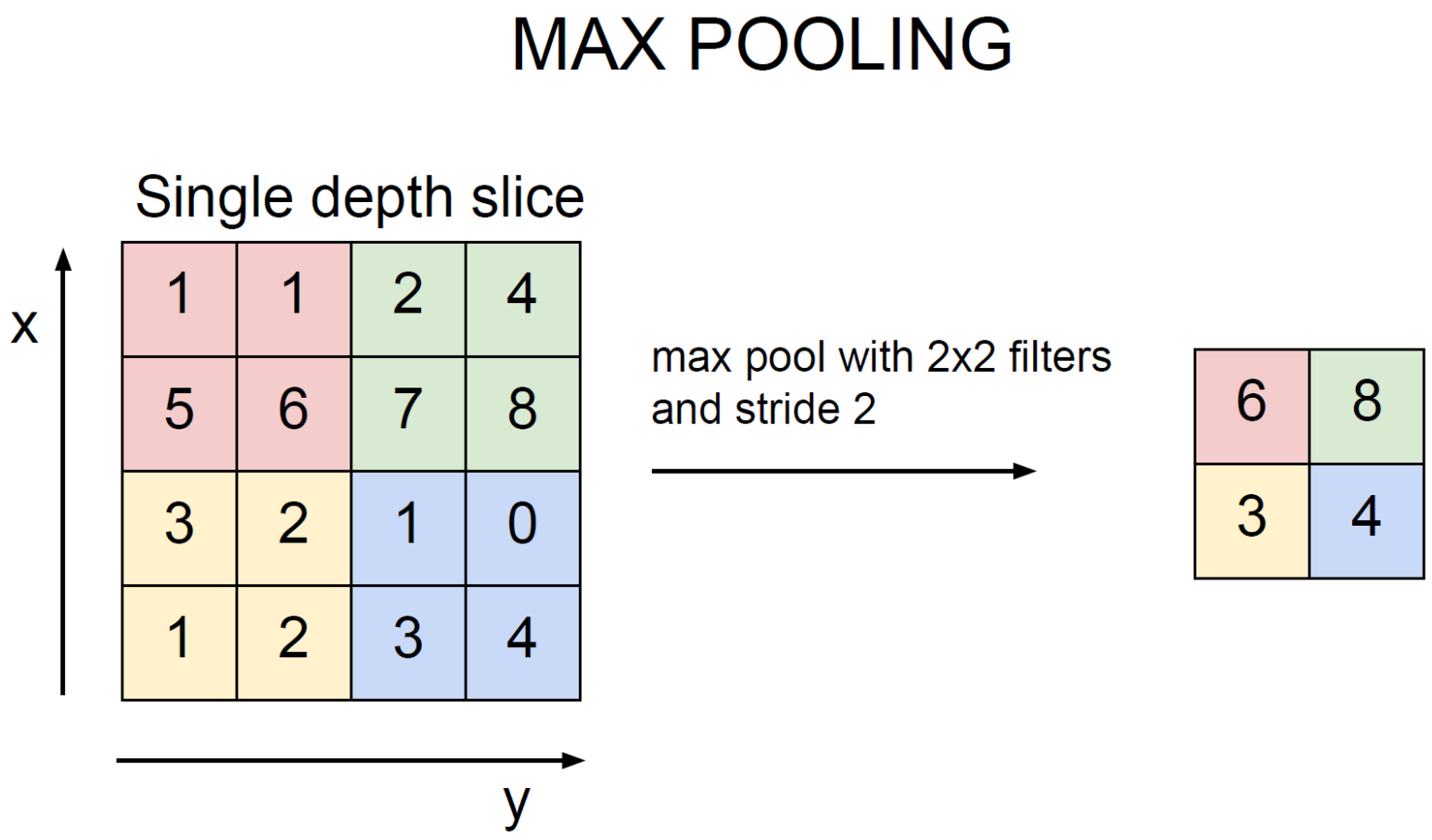
Slojevi sažimanja (*pooling*)

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Često korišćen: *Max Pooling*



Hiperparametri

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

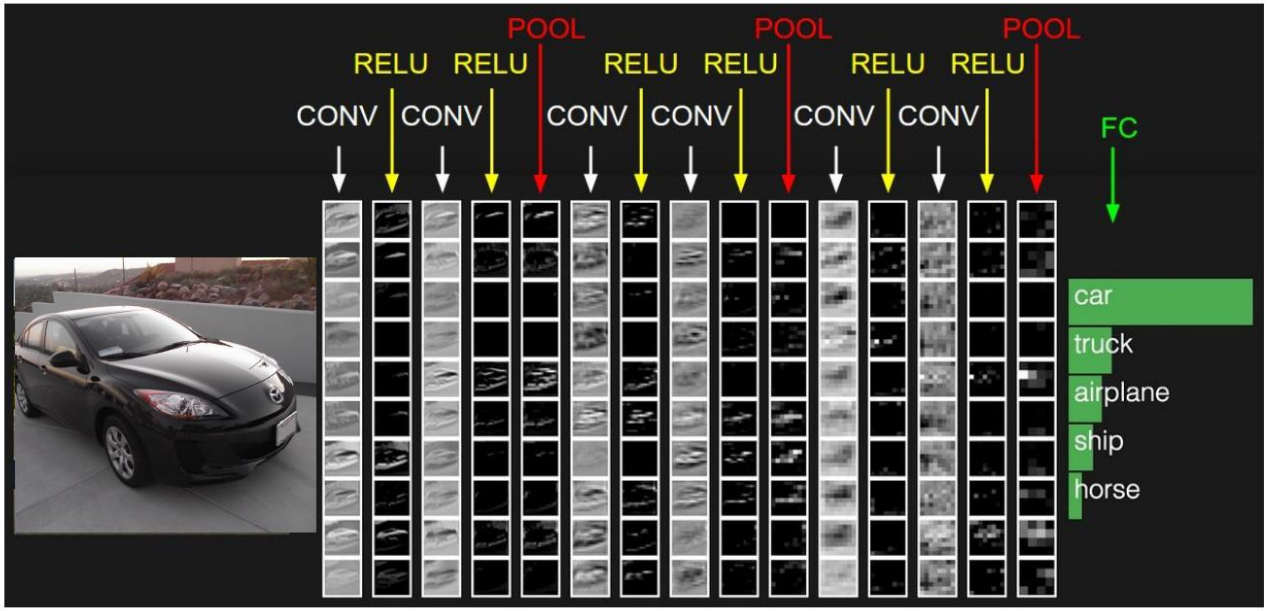
Common settings:

F = 2, S = 2
F = 3, S = 2

Potpuno povezan sloj

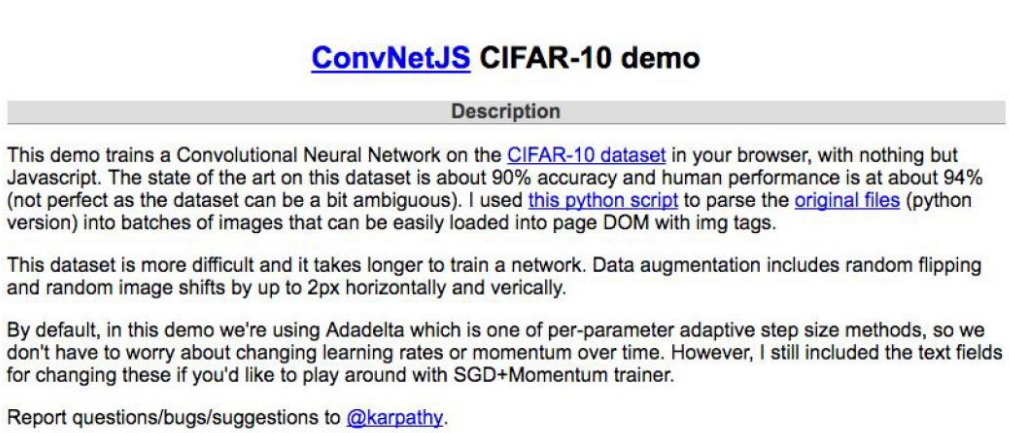
Fully Connected Layer (FC layer)

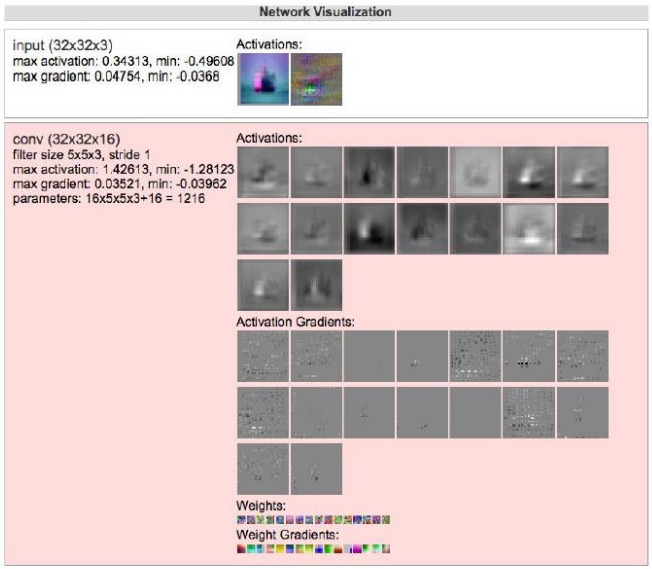
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Demo

[ConvNetJS demo: training on CIFAR-10]





http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Transfer learning

Transfer Learning

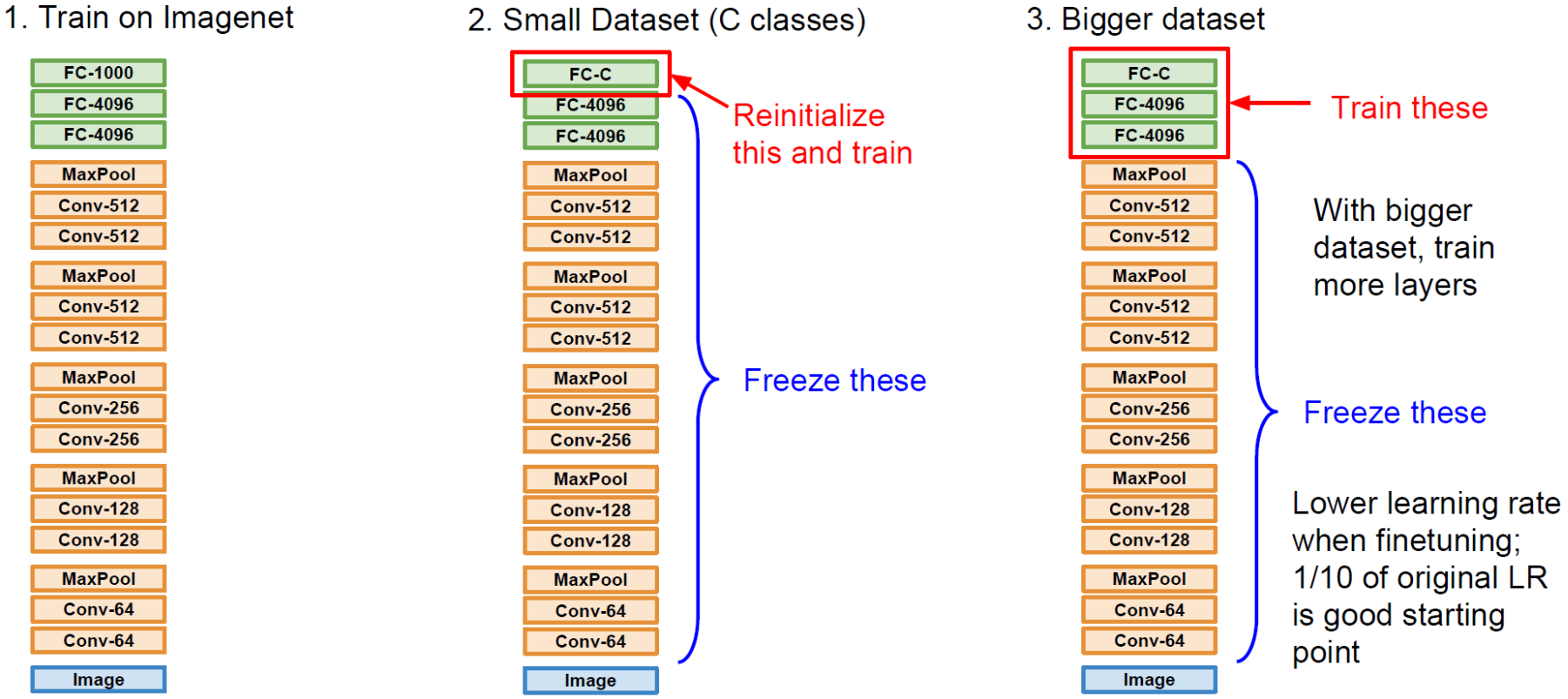
"You need a lot of a data if you want to train/use CNNs"

BUSTED

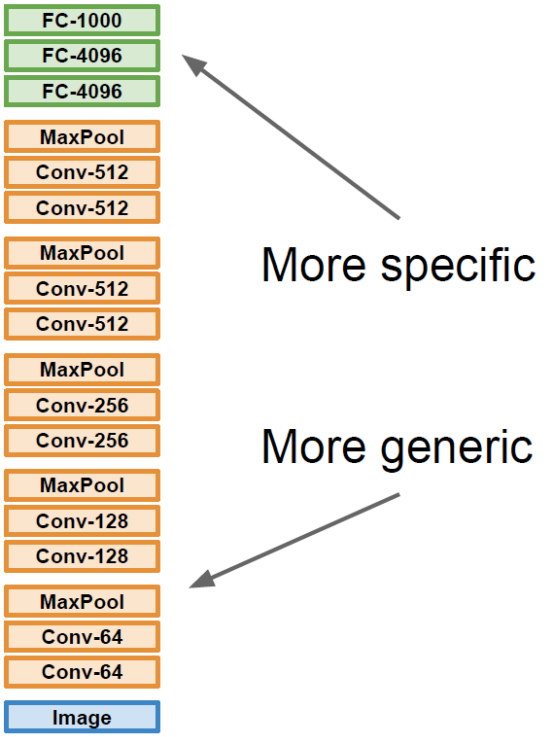
Transfer learning

Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014



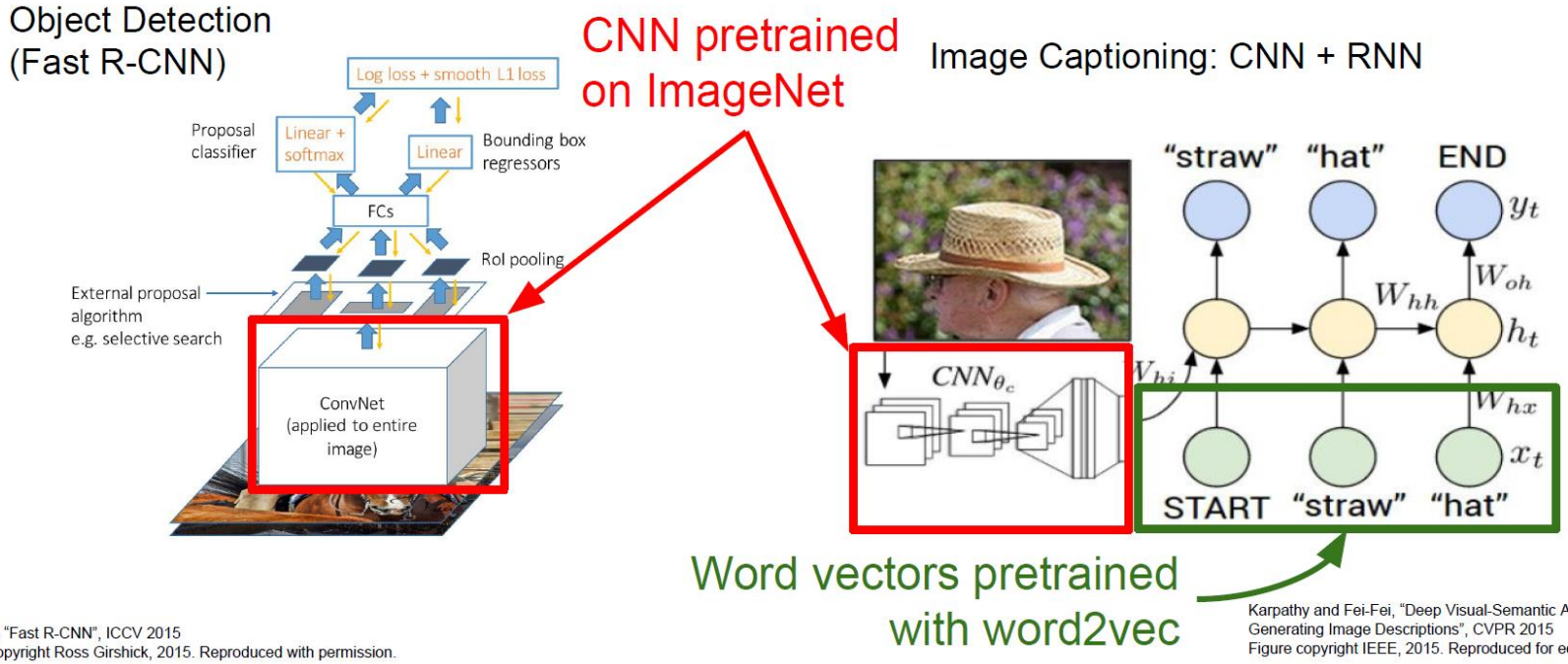
Transfer learning



	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Transfer learning

Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)



Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

Transfer learning

Takeaway for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a "Model Zoo" of pretrained models so you don't need to train your own

Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision