Компјутерски вид

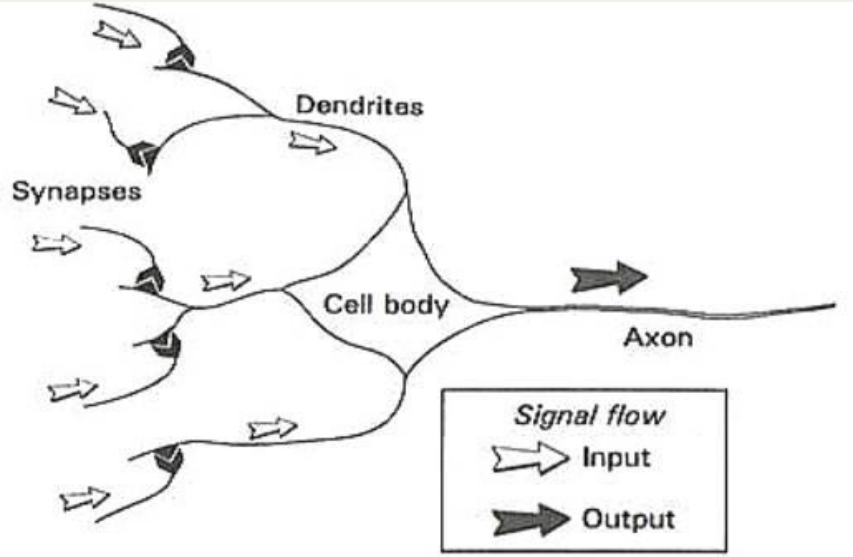
Школска 2022/2023

План рада

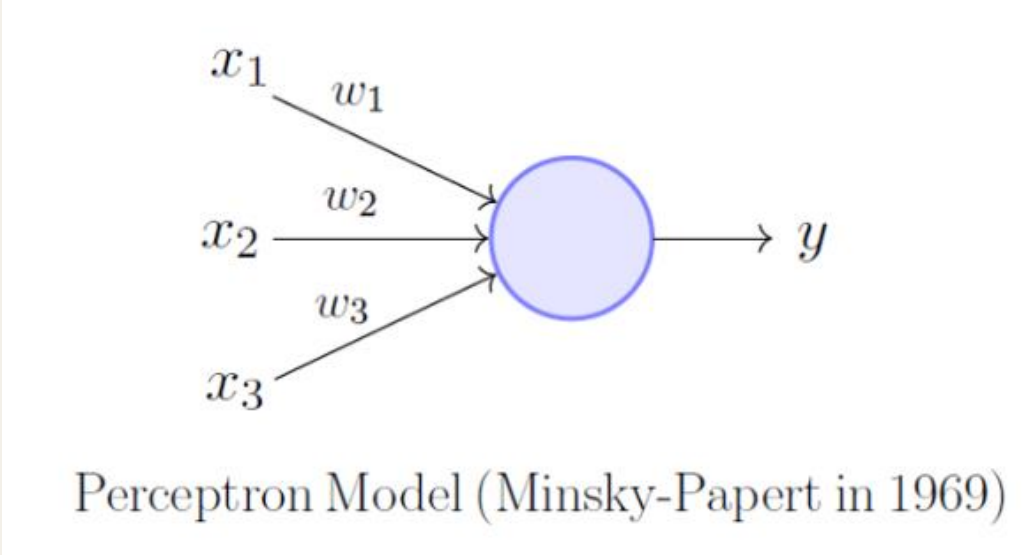
1. Неуронске мреже
2. Конволуционе неуронске мреже
3. Tensorflow – први кораци

Неуронске мреже - неурон

Неурон

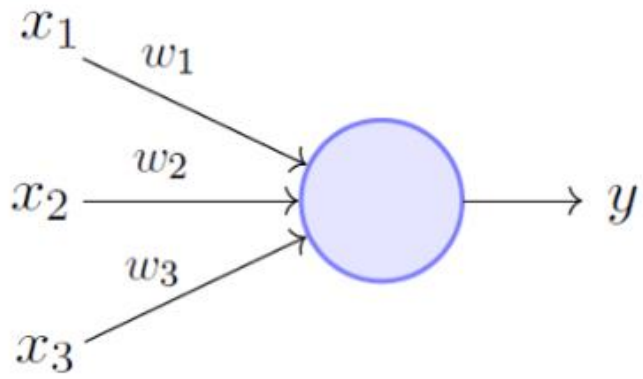


Перцептрон



Неуронске мреже - неурон

Перцептрон

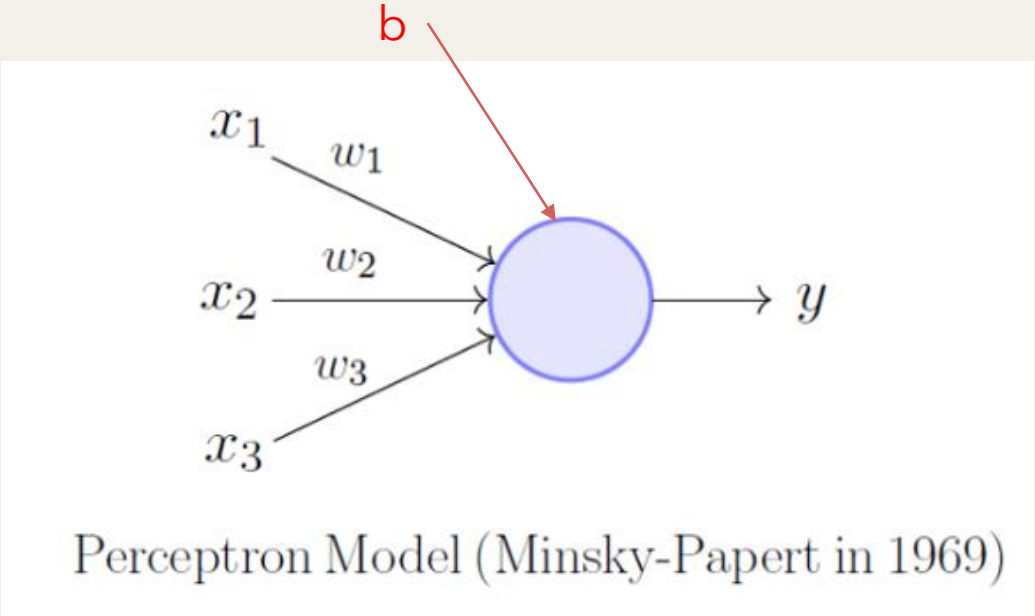


Perceptron Model (Minsky-Papert in 1969)

$$y = f(\sum_{i=0}^{3} w_i x_i)$$

Неуронске мреже - неурон

Перцептрон

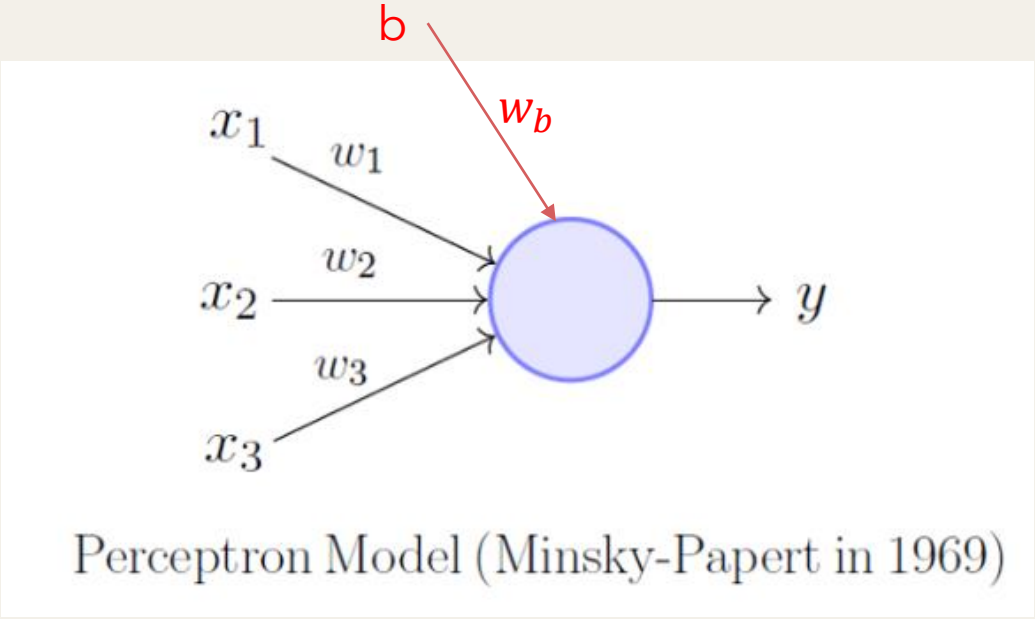


- Активациона функција f одређује да ли ће се неурон активирати или не (да ли је битан за предикцију или не)
- Активационом функцијом се постиже нелинеарност

$$y = f(\sum_{i=0}^{3} w_i x_i + b)$$

Неуронске мреже - неурон

Перцептрон

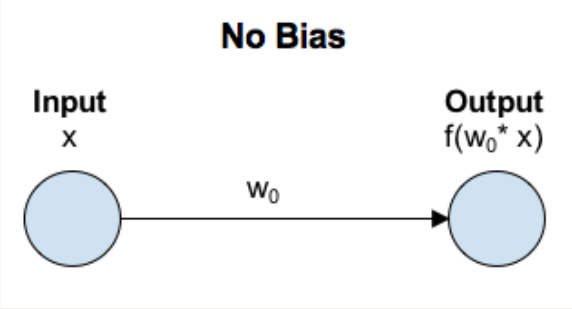


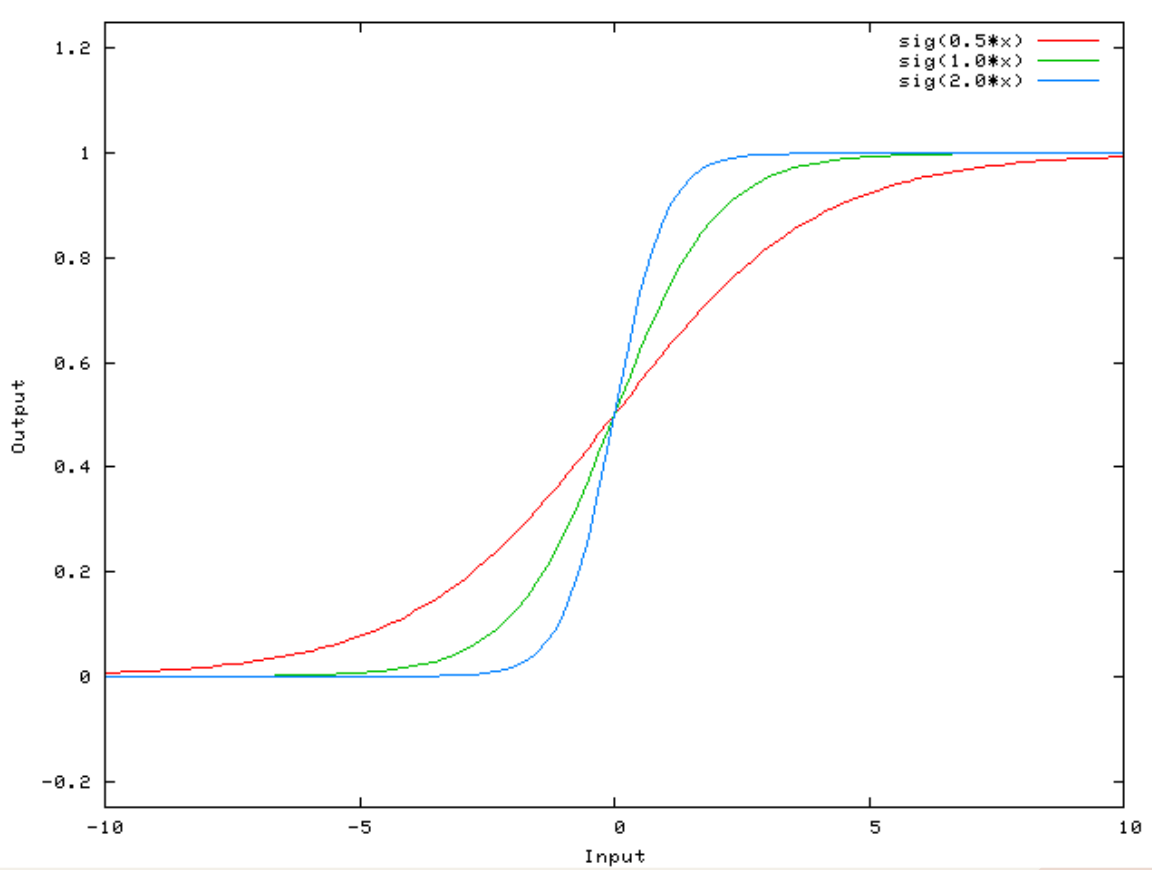
Perceptron Model (Minsky-Papert in 1969)

- Bias (b) омогућава транслирање вредности активационе функције

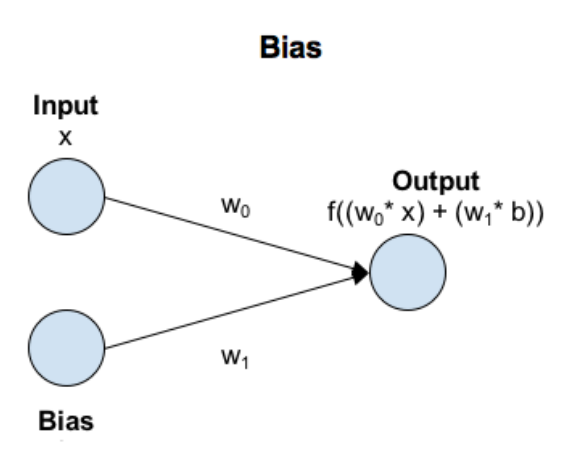
$$y = f(\sum_{i=0}^{3} w_i x_i + b w_b)$$

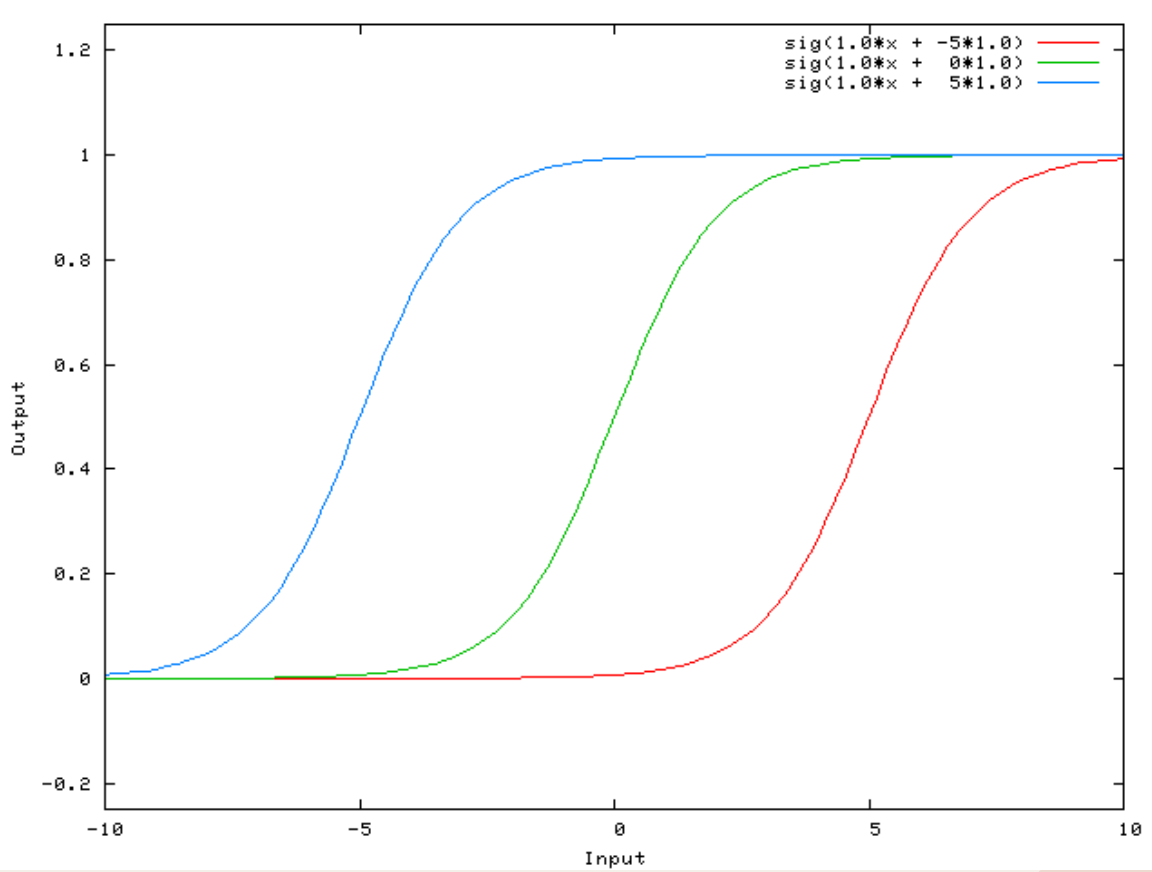
Неуронске мреже - неурон





Неуронске мреже - неурон



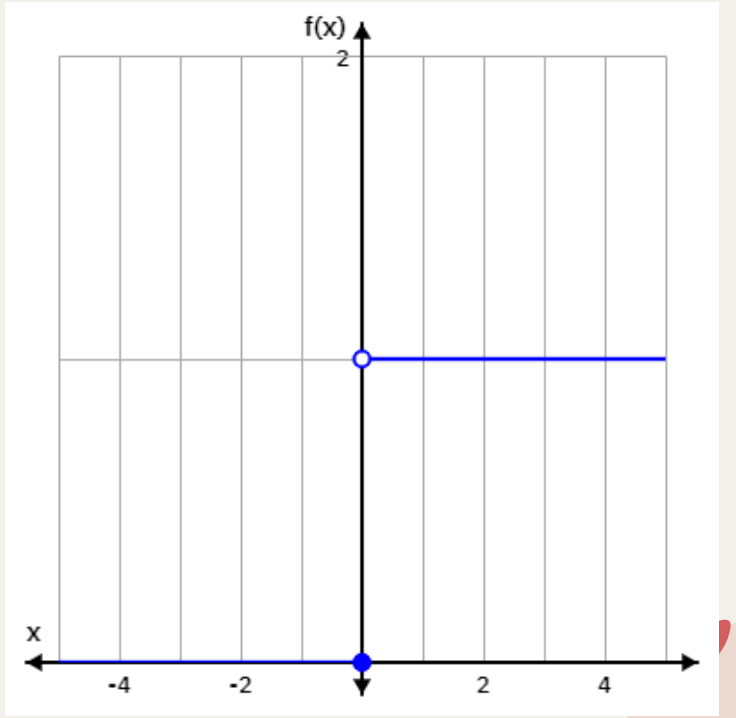


Неуронске мреже - шта је један перцептрон

$$f(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases}$$

- Може ли да имплементира основне логичке функције?

X	Y	NOT X	X AND Y	X OR Y
T	T	F	T	T
T	F	F	F	T
F	F	T	F	F
F	T	T	F	T



Неуронске мреже - шта је један перцептрон – NOT функција

x	F(x)
1	$f(-0.5) = 0$
0	$f(0.5) = 1$

X	Y	NOT X	X AND Y	X OR Y
1	1	0	1	1
1	0	0	0	1
0	0	1	0	0
0	1	1	0	1

$$y = f(wx + b)$$
$$w = -1$$
$$b = 0.5$$

Неуронске мреже - шта је један перцептрон – AND функција

X	Y	NOT X	X AND Y	X OR Y
1	1	0	1	1
1	0	0	0	1
0	0	1	0	0
0	1	1	0	1

$y = f(w_1 x_1 + w_2 x_2 + b)$
$w_1 = 1$
$w_2 = 1$
$b = -1.5$

x	y	F(x)
1	1	$f(0.5) = 1$
0	0	$f(-1.5) = 0$
1	0	$f(-0.5) = 0$
0	1	$f(-0.5) = 0$

Неуронске мреже - шта је један перцептрон – OR функција

X	Y	NOT X	X AND Y	X OR Y
1	1	0	1	1
1	0	0	0	1
0	0	1	0	0
0	1	1	0	1

$y = f(w_1x_1 + w_2x_2 + b)$
$w_1 = 1$
$w_2 = 1$
$b = -0.5$

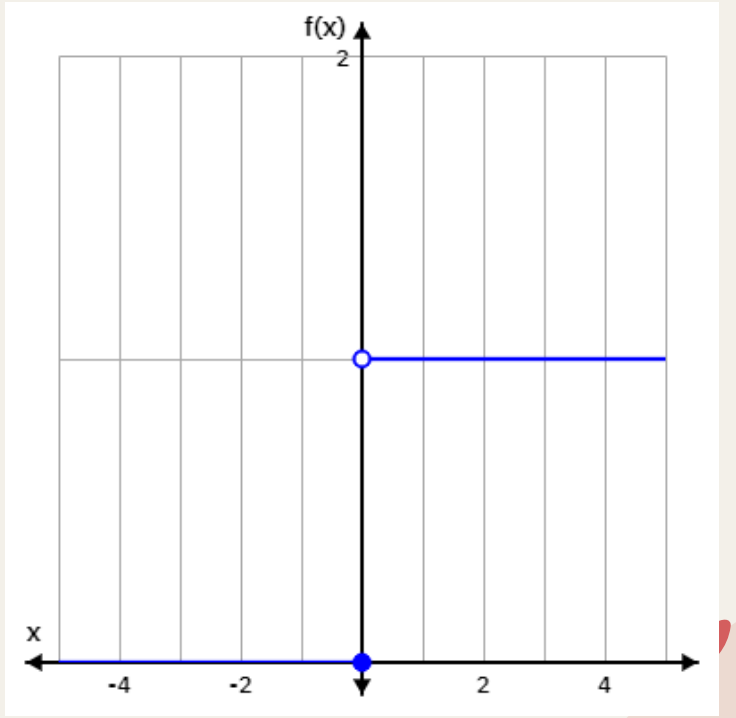
x	y	F(x)
1	1	$f(1.5) = 1$
0	0	$f(-0.5) = 0$
1	0	$f(0.5) = 1$
0	1	$f(0.5) = 1$

Неуронске мреже - шта је један перцептрон

$$f(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases}$$

- Може ли да имплементира основне логичке функције?
  - ДА!

X	Y	NOT X	X AND Y	X OR Y
T	T	F	T	T
T	F	F	F	T
F	F	T	F	F
F	T	T	F	T



Неуронске мреже - шта може један перцептрон

- Може ли да имплементира основне логичке функције?
  - ДА!
  - Може ли и XOR?

X	Y	NOT X	X AND Y	X OR Y	X XOR Y
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

Неуронске мреже - шта може један перцептрон – XOR функција

- Може ли да имплементира основне логичке функције?
  - ДА!
  - Може ли и XOR? – НЕ!

X	Y	NOT X	X AND Y	X OR Y	X XOR Y
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

Неуронске мреже - шта може један перцептрон – XOR функција

- Перцептроном се дефинише бинарни класификатор

Једначина линије!

$$w_1 x_1 + w_2 x_2 + b = 0$$

$$x_2 = \left(-\frac{w_1}{w_2}\right) x_1 + (-b \backslash w_2)$$

X	Y	NOT X	X AND Y	X OR Y	X XOR Y
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

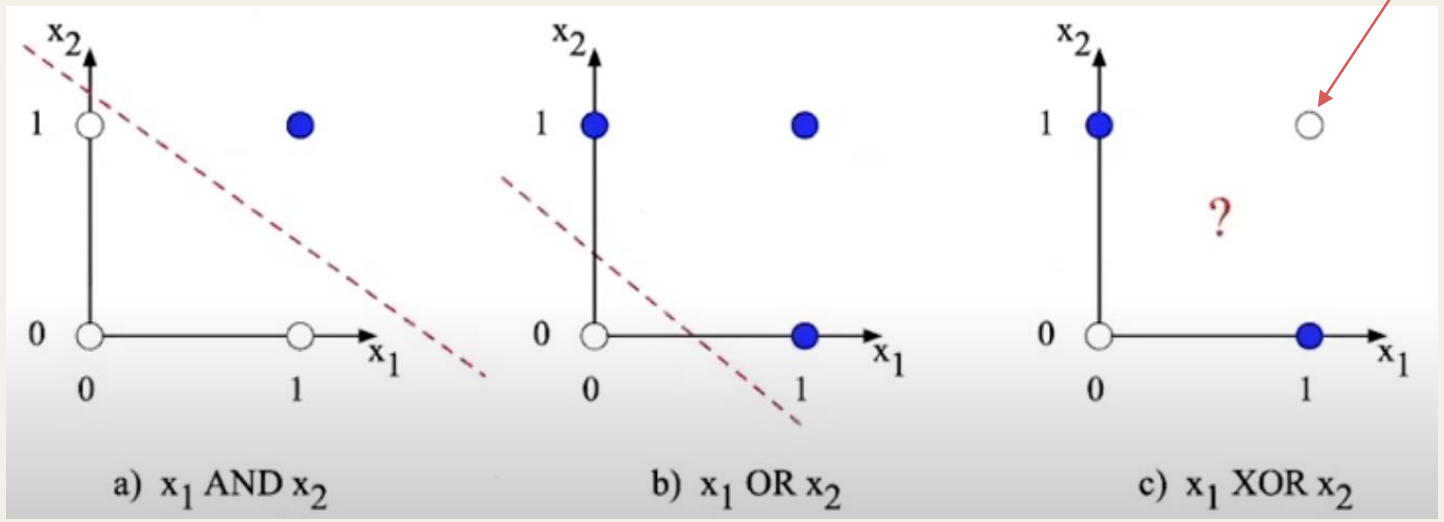
Неуронске мреже - шта је један перцептрон – XOR функција

- Перцептроном се дефинише бинарни класификатор

$$w_1 x_1 + w_2 x_2 + b = 0$$

$$x_2 = \left(-\frac{w_1}{w_2}\right) x_1 + (-b \backslash w_2)$$

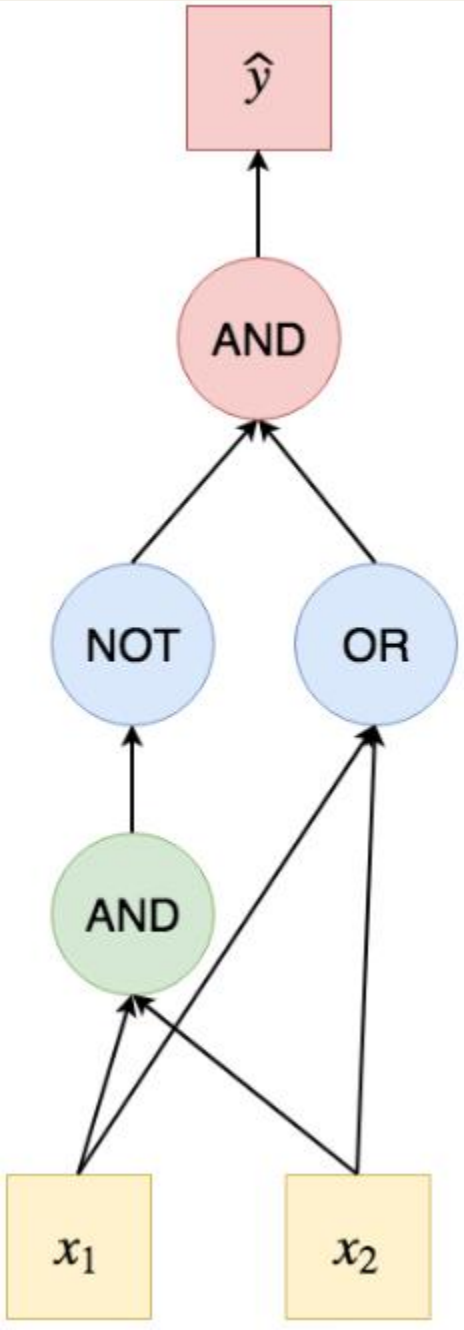
Није могуће раздвојити их једном линијом



Неуронске мреже - шта може један перцептрон – XOR функција

$$XOR(x_1, x_2) = AND\ (NOT(AND(x_1, x_2)), OR(x_1, x_2))$$

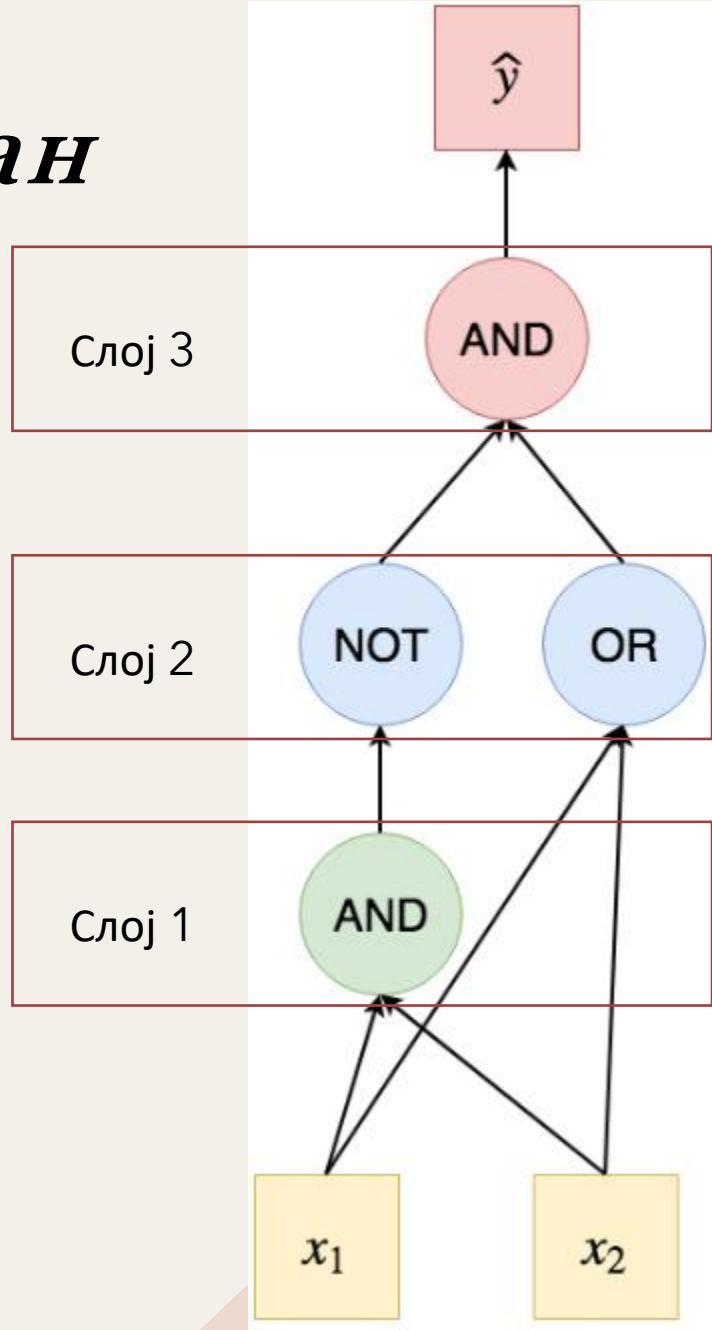
X	Y	X XOR Y
T	T	F
T	F	T
F	F	F
F	T	T



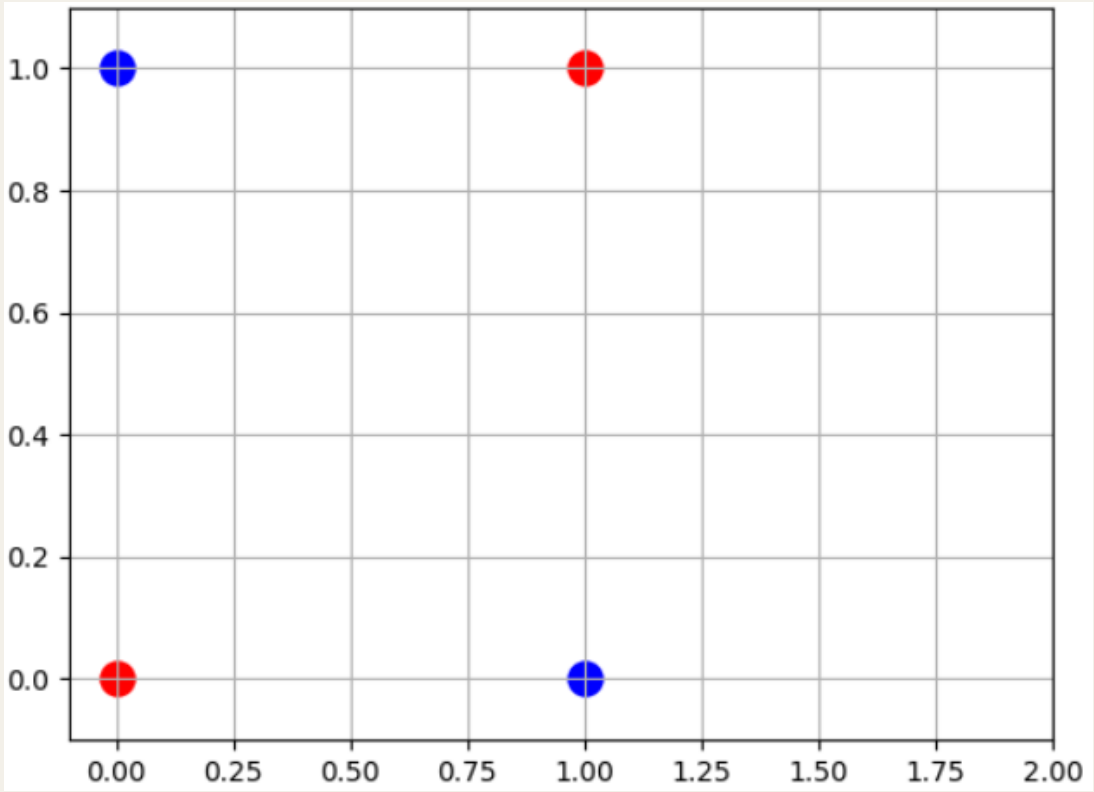
Неуронске мреже - шта може један перцептрон – XOR функција

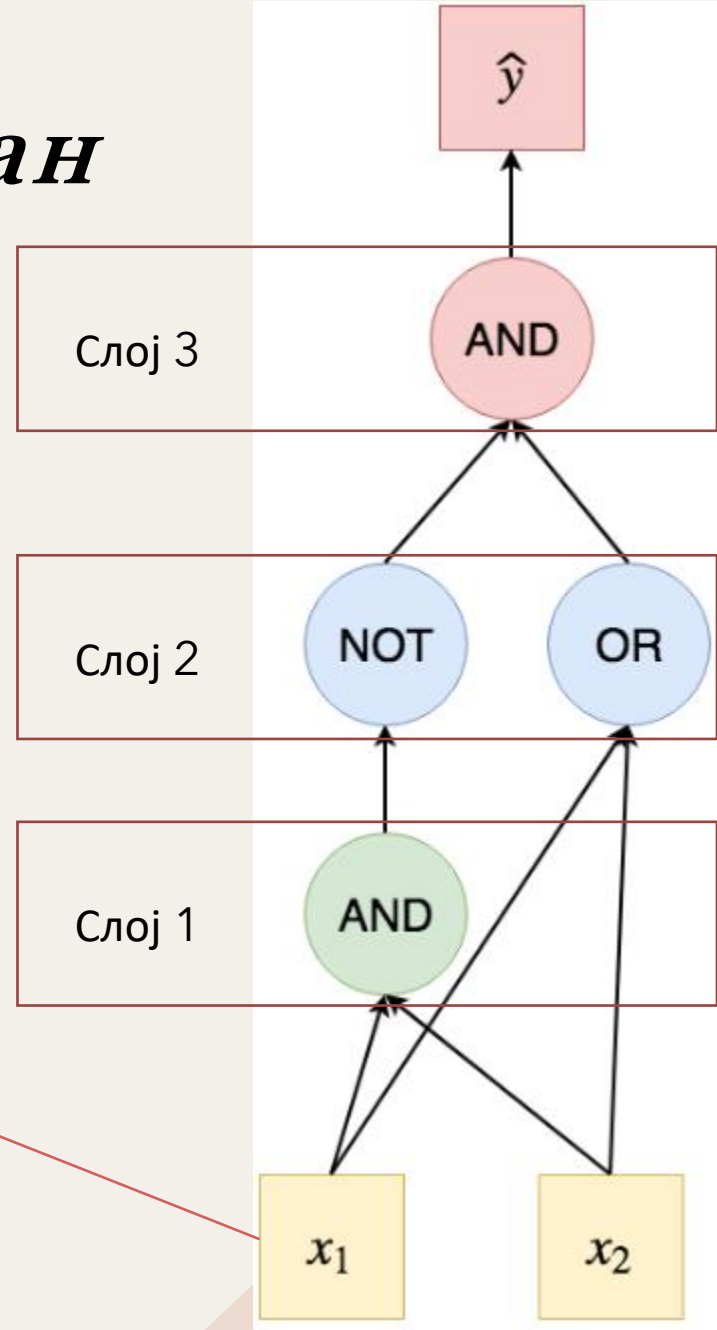
$$XOR(x_1, x_2) = AND\ (NOT(AND(x_1, x_2)), OR(x_1, x_2))$$

X	Y	X XOR Y
T	T	F
T	F	T
F	F	F
F	T	T

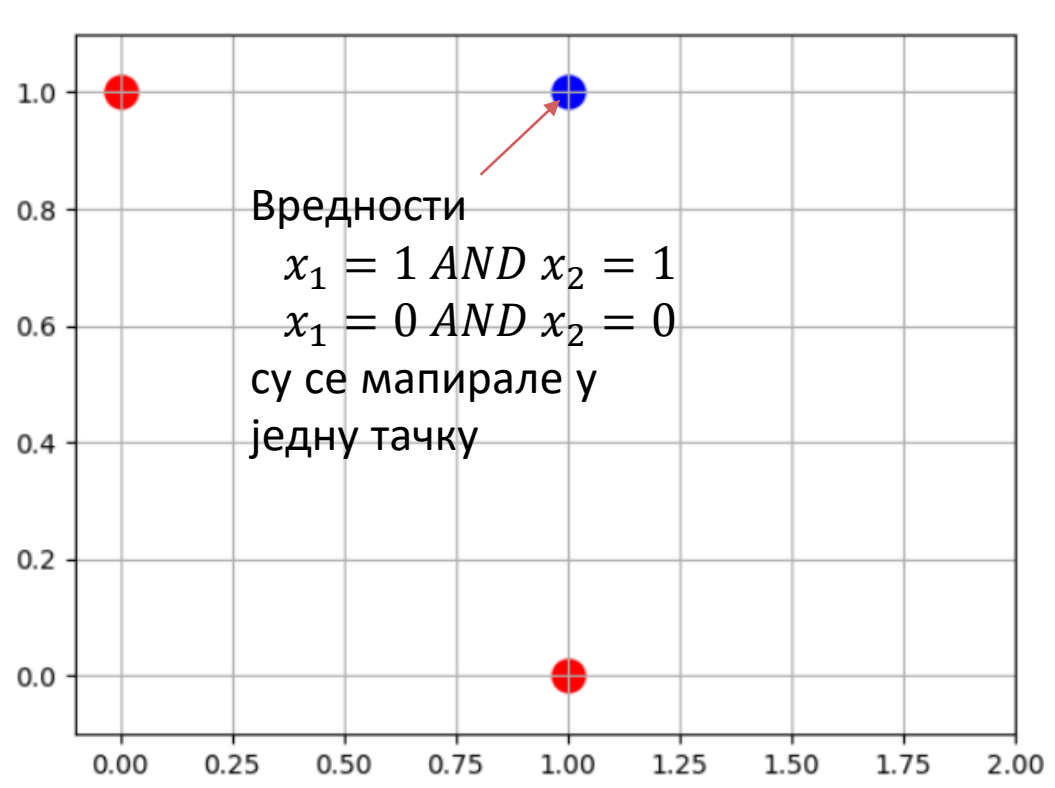


Неуронске мреже - шта може један перцептрон – XOR функција

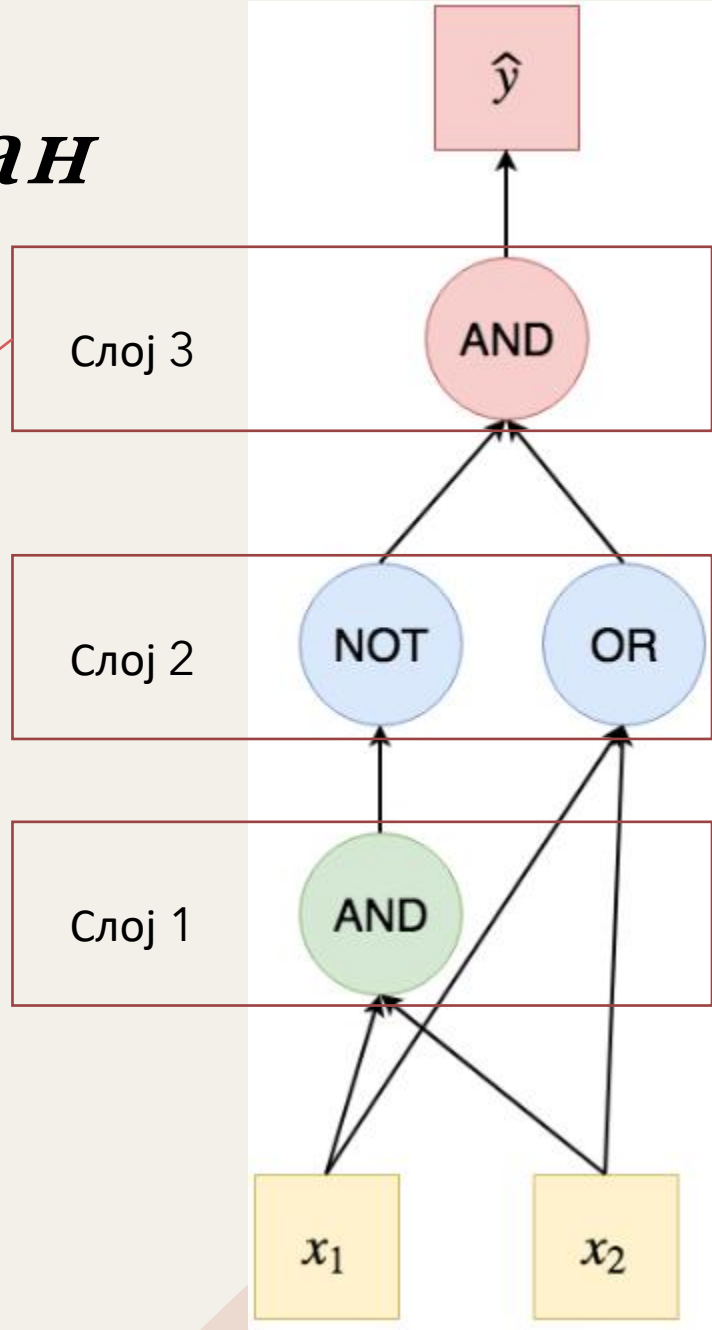




Неуронске мреже - шта може један перцептрон – XOR функција



Подаци су линеарно сепарабилни



Неуронске мреже - визуелизација

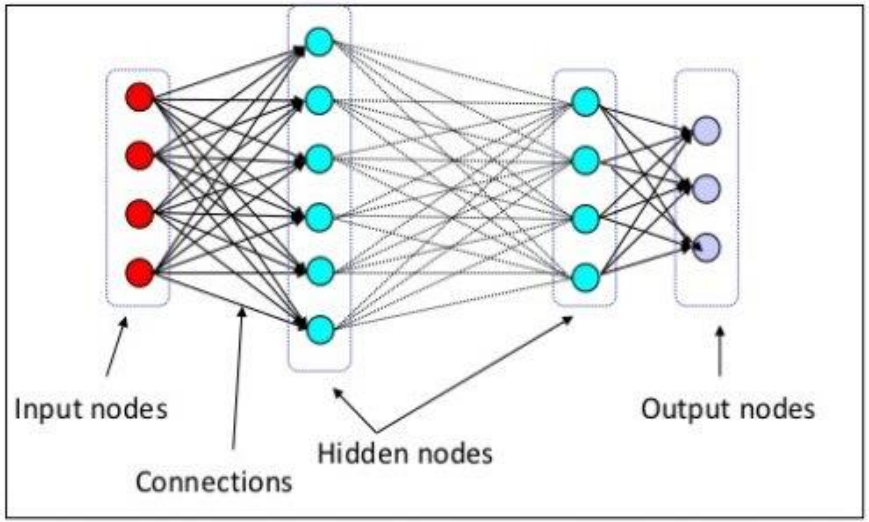
https://playground.tensorflow.org/

Неуронске мреже

- Слојевима је могуће формирати погоднију репрезентацију података
  - Као код XOR случаја, подаци нису били линеарно сепарабилни у првом слоју, али су низом трансформација постала сепарабилни у крајњем слоју
- Сваки слој мреже се састоји од више неурона
- Сваки неурон врши израчунавање на основу информација из претходног слоја
- Неурони унутар једног слоја израчунавање обављају паралелно

Дубоке неуронске мреже

- Deep forward networks = feedforward neural networks = multilayer perceptron
- Имају више од једног скривеног слоја
- Унутрашњи слојеви формирају репрезентацију података на начин који је погоднији за решавање проблема
- Укупан број слојева назива се **дубина мреже.**
- Последњи слој у мрежи назива се **завршни слој**



Дубоке неуронске мреже

- Циљ: апроксимација неке комплексне функције f^*
  - У случају класификације, $y = f^*(x)$, неуронска мрежа учи да мапира улаз x у класу y
- Ознака: $y = f(x;\theta)$ у значењу: функција f је одређена параметрима θ (ови параметри се добијају тренирањем)
- Слојеви мреже одређују директан ацикличан граф композиција више функција (видети пример код XOR функције)
  - Нпр. $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, где $f^{(1)}$ означава први слој мреже, $f^{(2)}$ други итд.

Дубоке неуронске мреже

- Тренинг подаци дефинишу емпиријску функцију $f^*(x)$ коју мреже покушава да апроксимира $f\ (x)$
- Тренинг пар (x, y) експлицитно одређује резултат завршног слоја
  - када се на улазу појави x мрежа треба да да за резултат y
- Остали (унутрашњи, скривени) слојеви немају јасно дефинисано понашање (на основу података)
  - Алгоритам обучавања проналази најбоље мапирање које се дешава у унутрашњим слојевима како би се у завршном слоју добила што боља апроксимација тренинг скупа

Дубоке неуронске мреже - обучавање

- Функција губитка: $$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

- Циљ: минимизација функција губитка -> промена вредности параметара у смеру смањивања вредности функције (супротно од градијената)
- Алгоритам: backpropagation

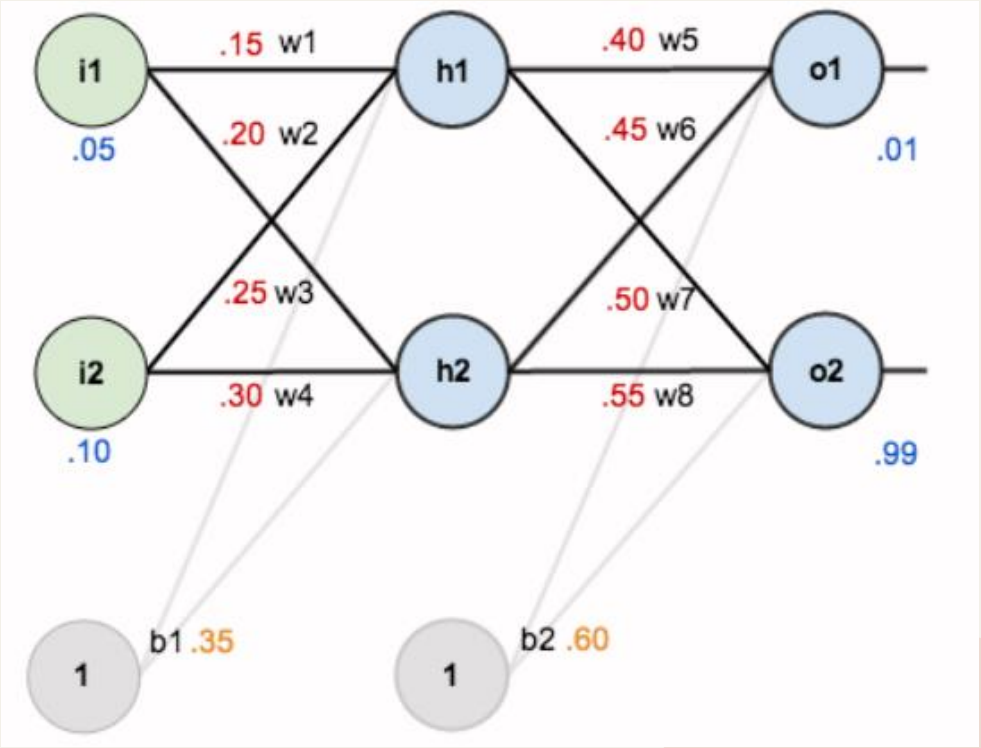
https://www.youtube.com/watch?v=d14TUNcbn1k&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=4

Дубоке неуронске мреже - обучавање

Backpropagation:

- Проналазе се вредности тежина које доводе до најмање вредности функције губитка, на основу тренинг података

1. Forward pass – рачунање излаза за дати улаз (са тренутним вредностима тежина) и израчунавање функције губитка
2. Backward pass - одређивање вредности градијената и промена вредности параметара



https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Градиејент ни спуст

Израчунати градијенти

Backpropagation: a simple example

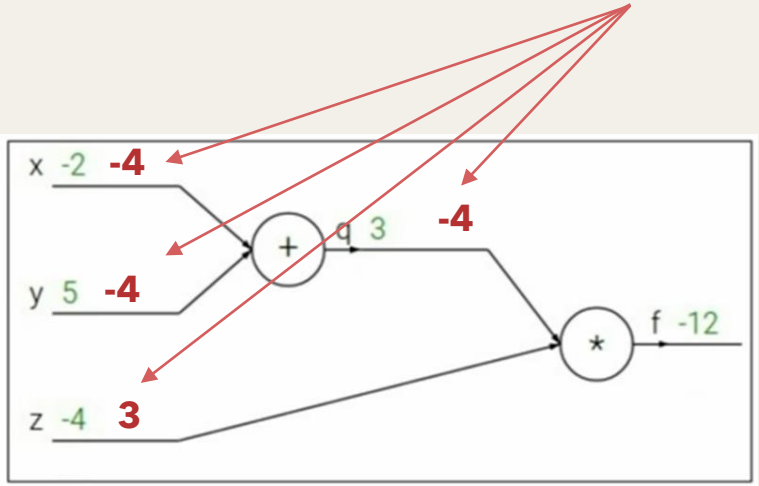
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

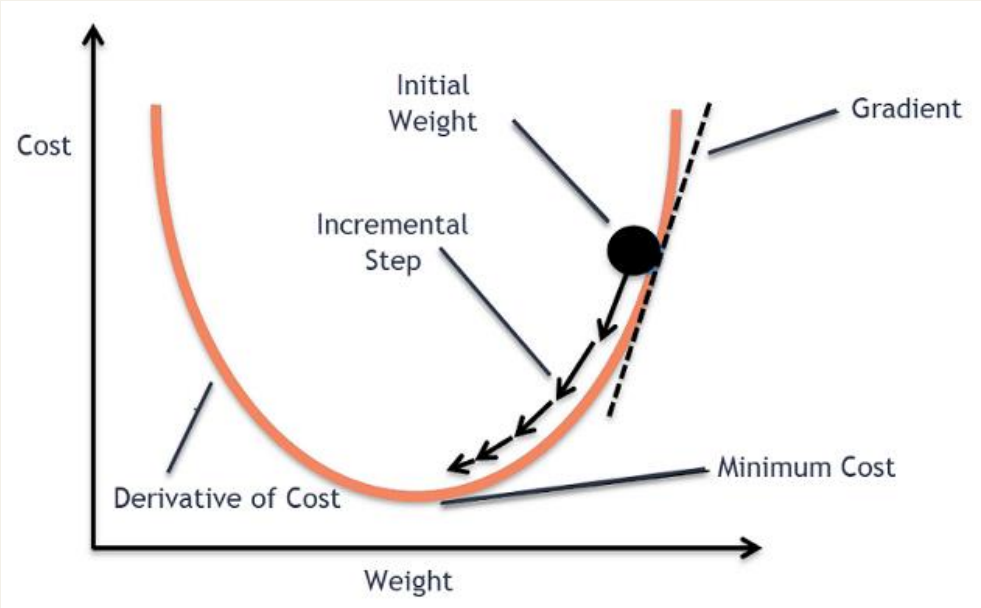
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

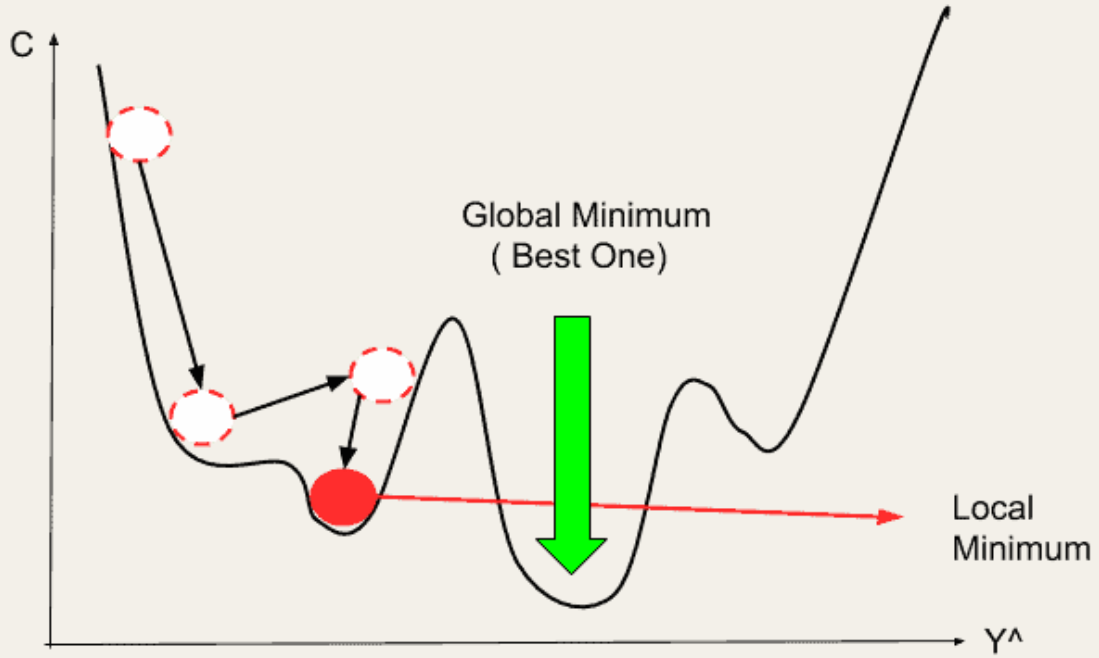
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



https://www.youtube.com/watch?v=d14TUNcbn1k&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=4

Градиејент ни спуст





https://medium.com/@ugurozcan108/batch-gradient-descent-in-python-4d3b16d40755

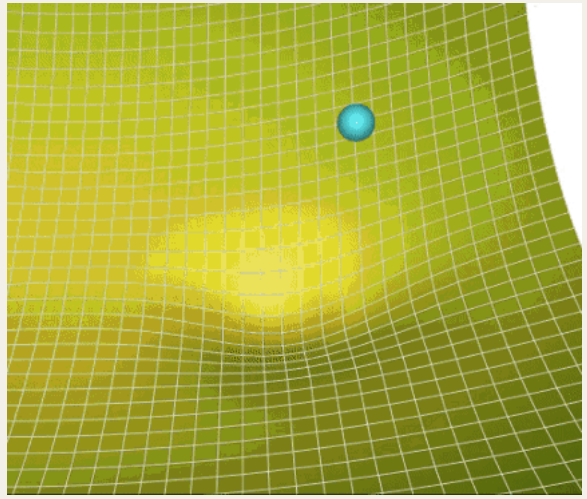
https://www.mltut.com/stochastic-gradient-descent-a-super-easy-complete-guide/

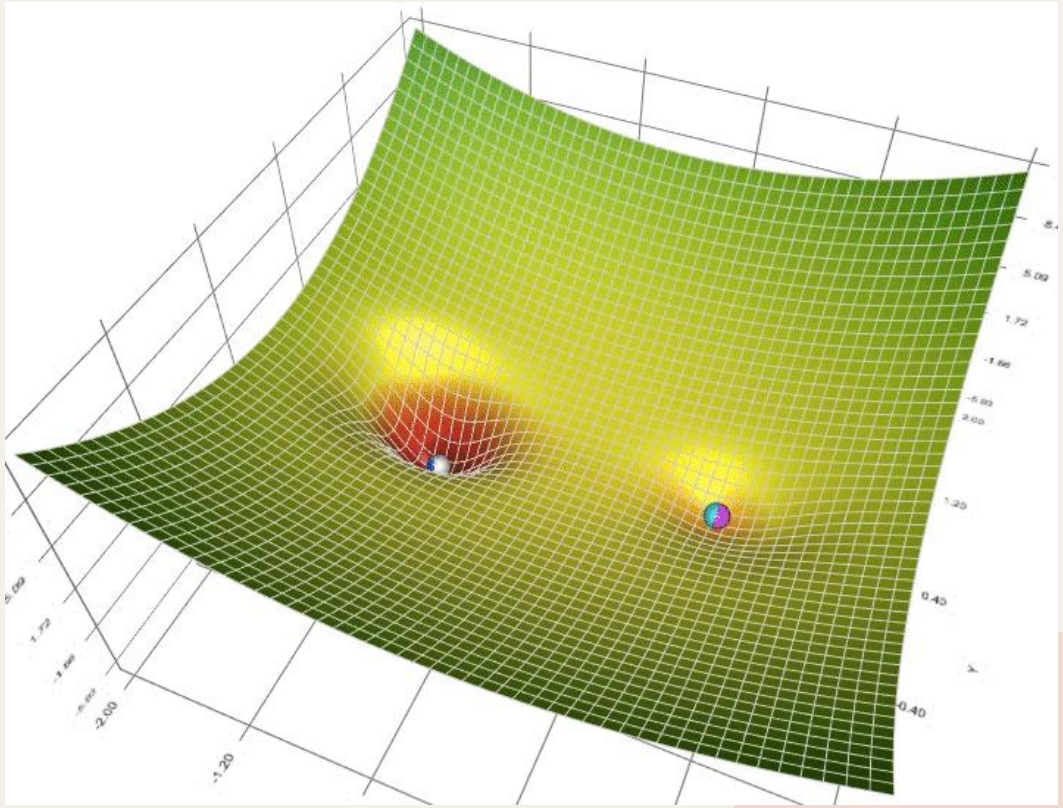
Алат за визуелизацију, врло корисно!

https://github.com/lilipads/gradient_descent_viz

Градиејент ни спуст

У зависности од полазне тачке и облика функције, оптимизација може да заврши у локалном минимуму.





https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Градиејент ни спуст

Израчунати градијенти

Backpropagation: a simple example

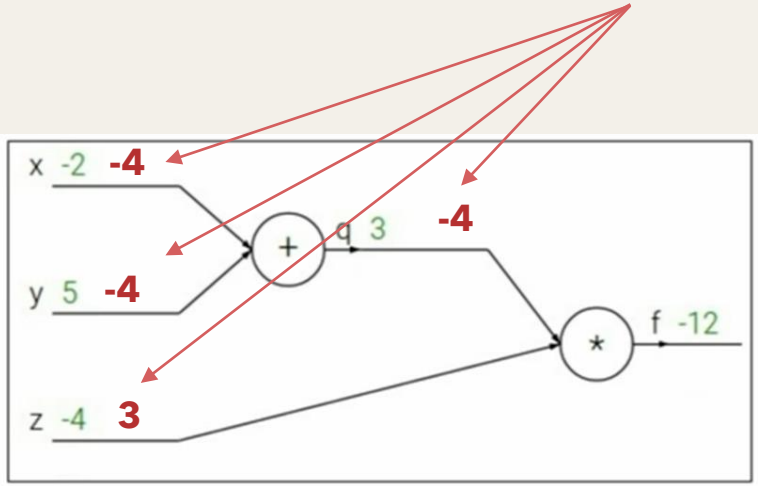
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



… практичне вежбе за имплементацију у Tensorflow-у

https://www.youtube.com/watch?v=d14TUNcbn1k&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=4

Дубоке неуронске мреже - обучавање

- Функција губитка: $$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

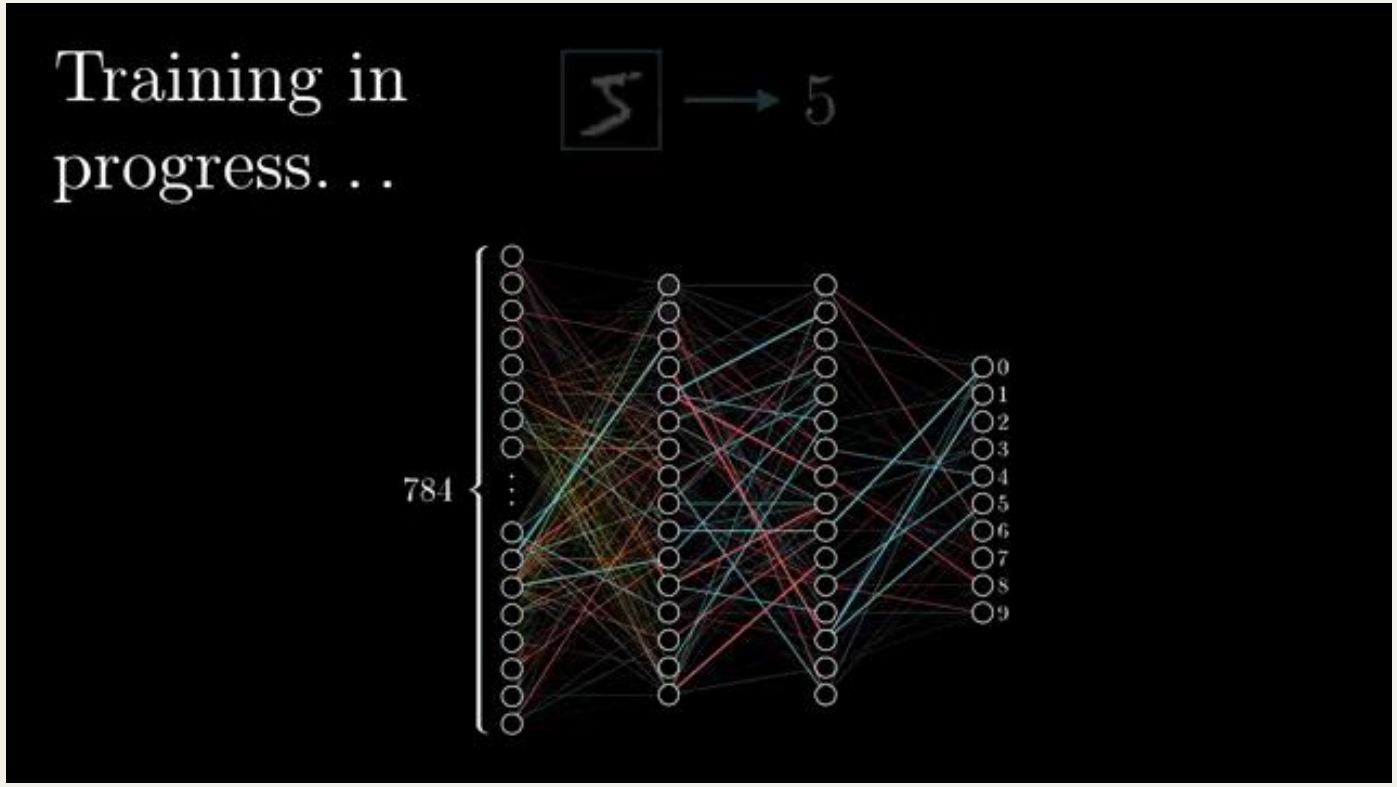
- Циљ: минимизација функција губитка -> промена вредности параметара у смеру смањивања вредности функције (супротно од градијената)

$$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\boldsymbol{g},$$

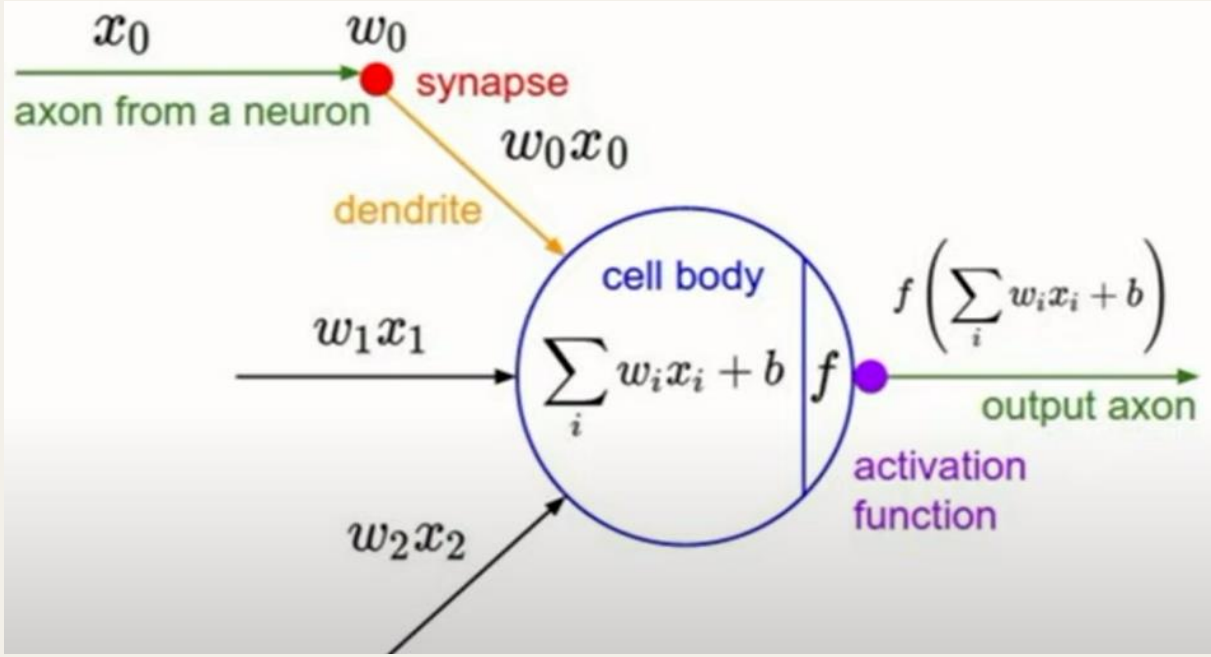
ϵ је learning rate

Дубоке неуронске мреже - обучавање



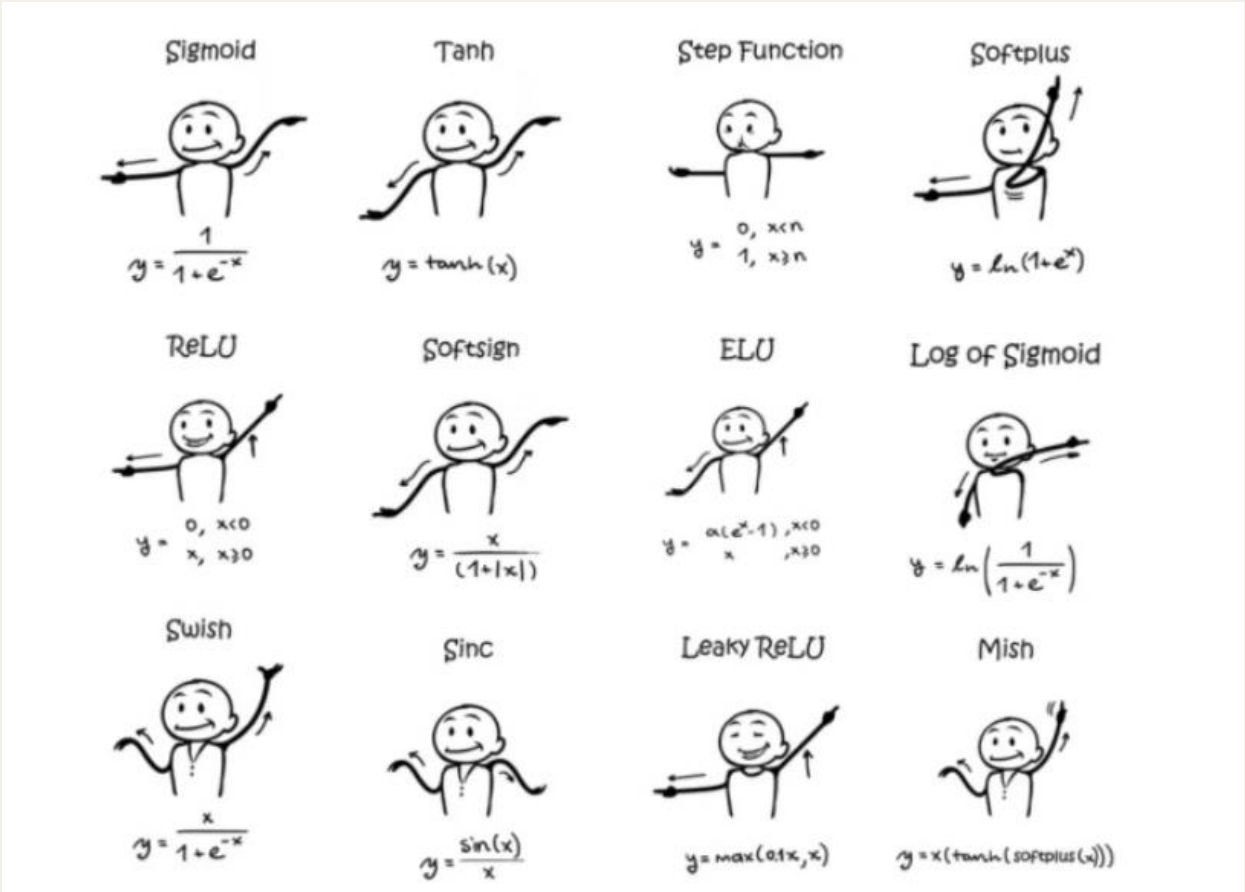
https://gfycat.com/

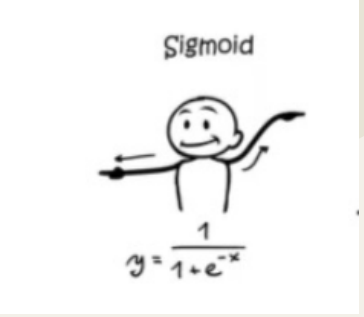
Неуронске мреже - активационе функције



https://www.youtube.com/watch?v=d14TUNcbn1k&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=4

Неуронске мреже - акт ивационе функције

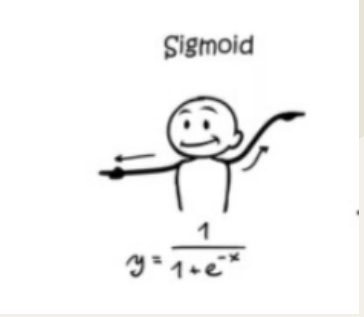




Неуронске мреже - активационе функције

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

- Диференцијабилна
- Излаз може да се тумачи као вероватноћа (између 0 и 1)

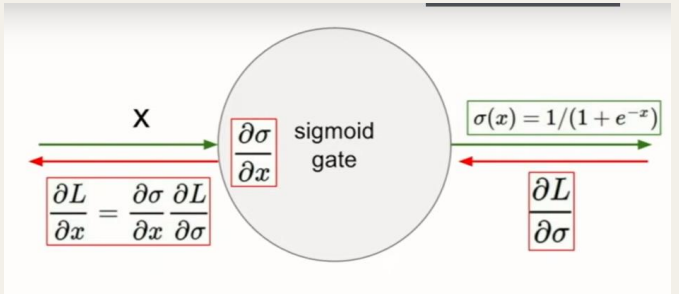


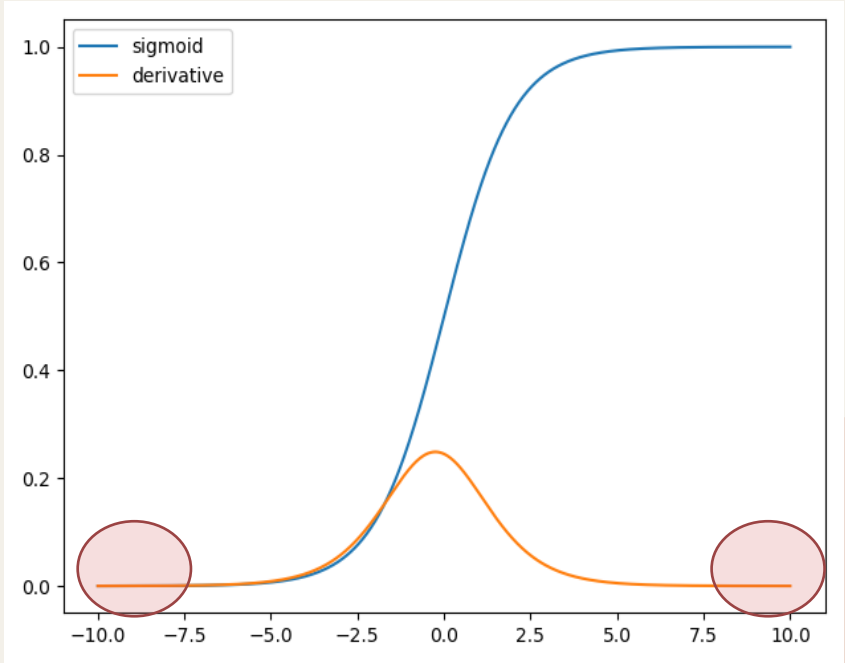
Неуронске мреже - активационе функције

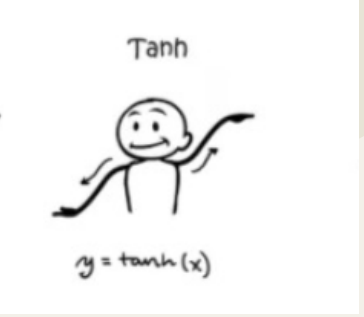
$$f(x) = \frac{1}{1 + e^{-x}}$$

- Диференцијабилна
- Излаз може да се тумачи као вероватноћа (између 0 и 1)

- Може имати мале градијенте
- Проблем нестајућих градијената
- Неефикасно учење, све тежине у једном неурону се мењају у исту страну (позитивно или негативно)
  - ➢ подаци треба да буду центрирани око 0
- Експоненцијална функција је „скупа“ операција





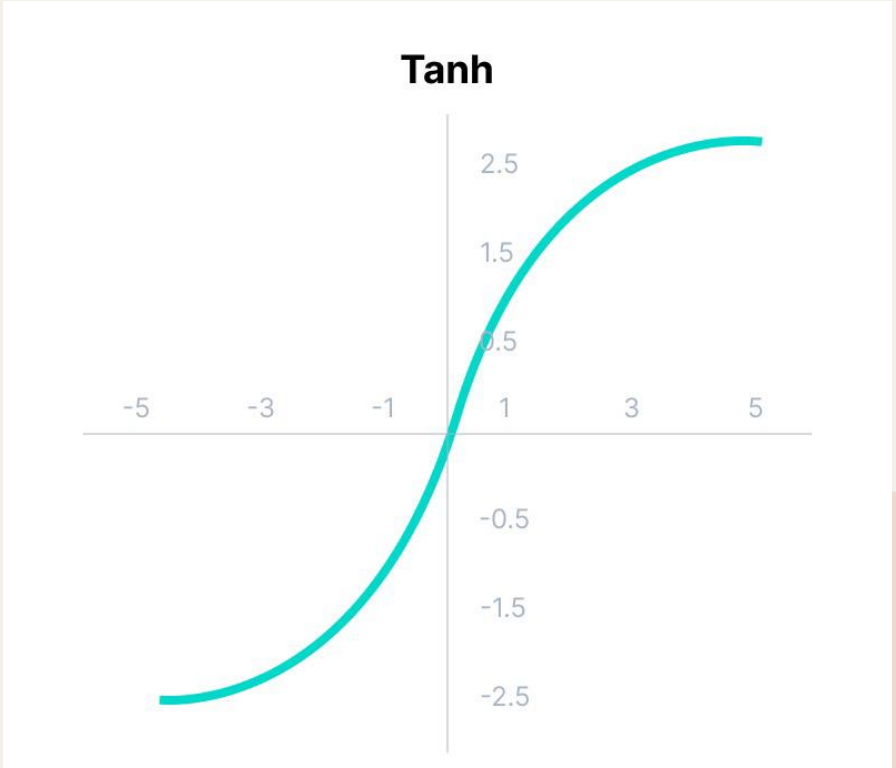


Неуронске мреже - активационе функције

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

- Диференцијабилна
- Излаз може да се тумачи као вероватноћа (између -1 и 1)
- Центрирана око 0

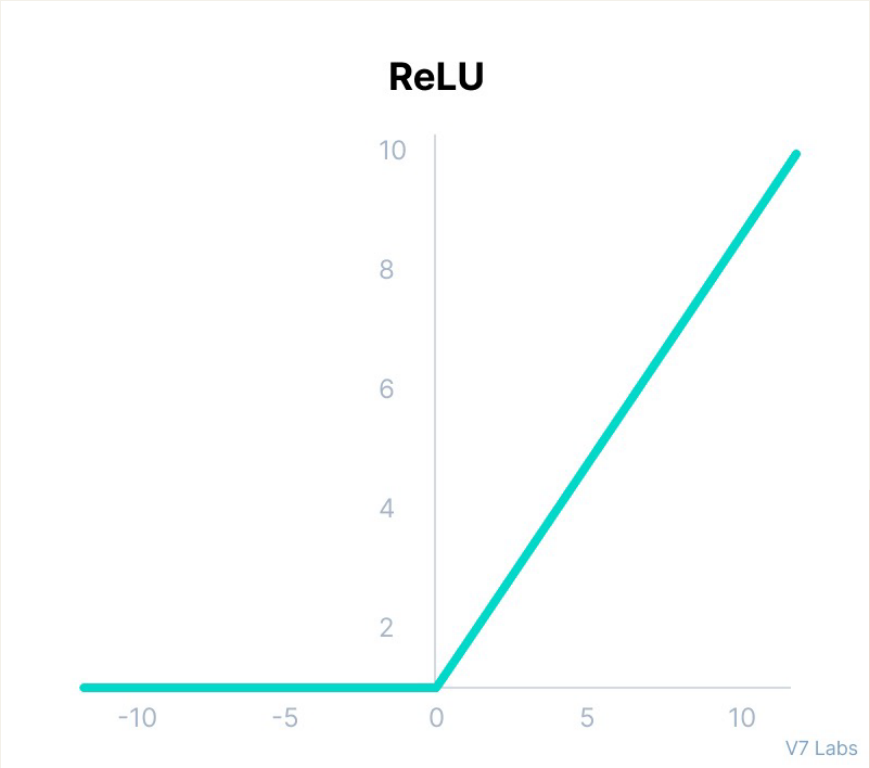
- Може имати мале градијенте
  - Проблем нестајућих градијената



Неуронске мреже - активационе функције

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

- Активна само за позитивне вредности
- Мањи број неурона ће бити активан
- Нема нестајуће градијенте за позитивне вредности
- „брзо“ се израчунава, ближа апроксимација биолошком неурону
- Бржа конвергенција (у пракси) од sigmoid/tanh

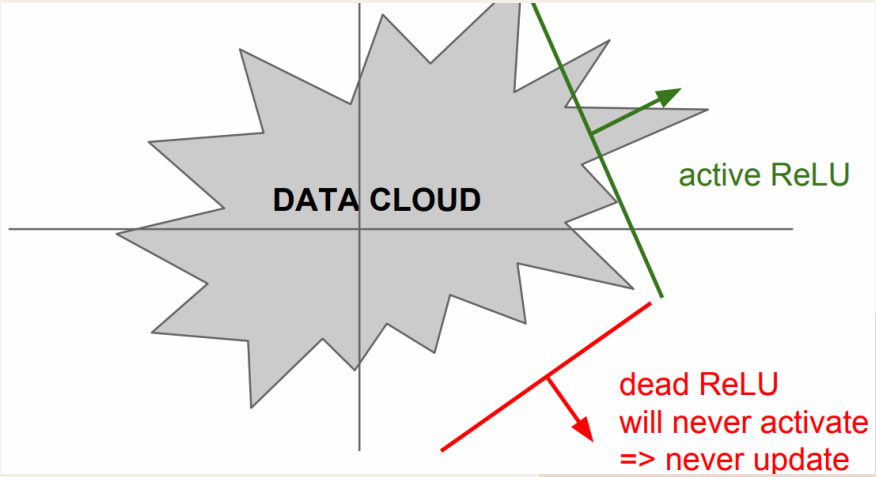


Неуронске мреже - активационе функције

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

- Активна само за позитивне вредности
- Мањи број неурона ће бити активан
- Нема нестајуће градијенте за позитивне вредности
- „брзо“ се израчунава, ближа апроксимација биолошком неурону
- Бржа конвергенција (у пракси) од sigmoid/tanh

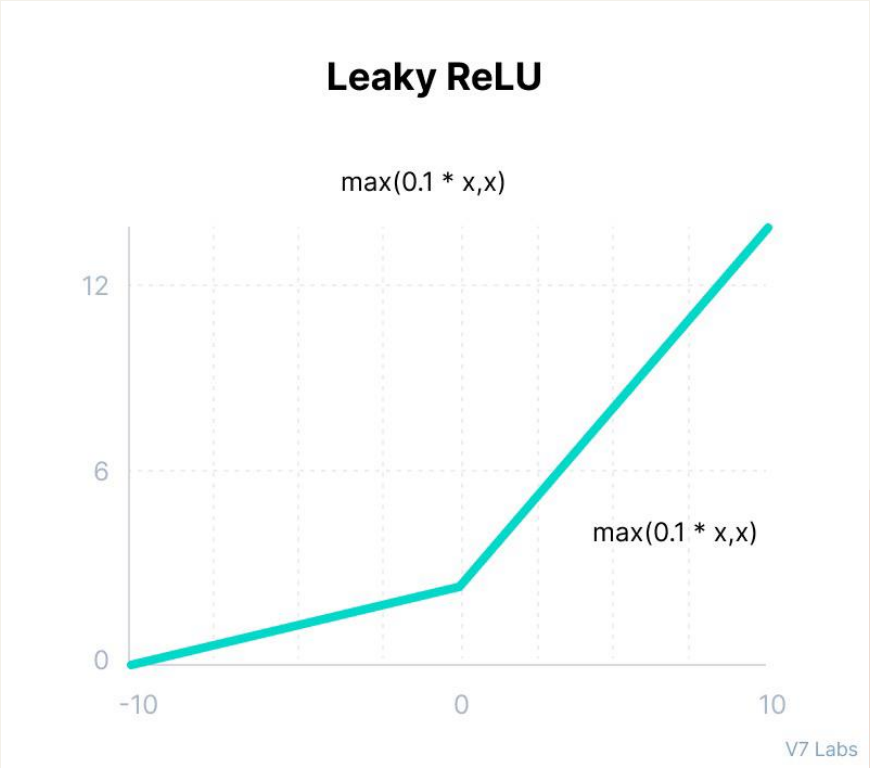
- Није центрирана око 0
- Градијенти негативних вредности су 0
- Dead ReLU -резултат је увек нула, за сваки улаз
  - Узрок: лоша иницијализација, превелики LR
  - Обично нема опоравка; у пракси 10-20% ReLU ће бити „мртве“



Неуронске мреже - акт ивационе функције

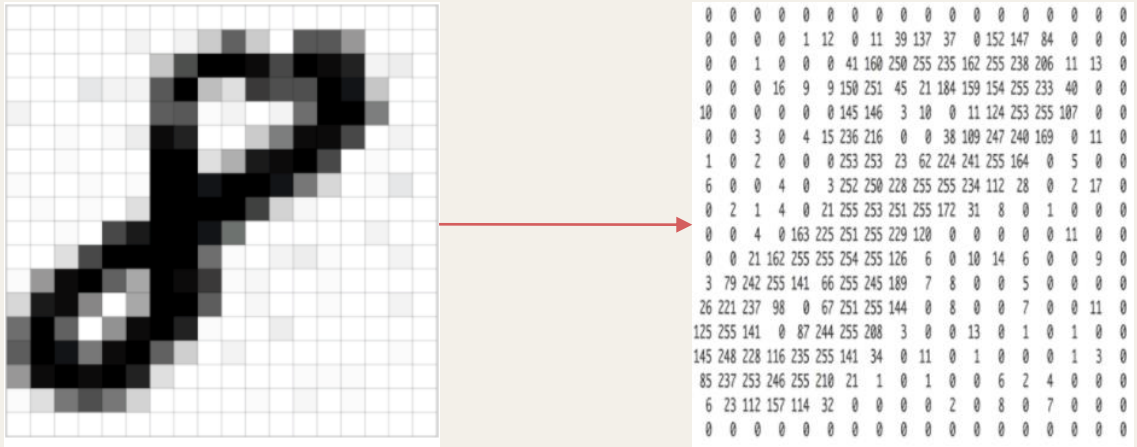
$f(x) = \max(0.1x, x)$ или параметарска верзија $f(x) = \max(\alpha x, x)$, где се α учи током тренирања

- Нема нестајућих градијената
- Мали градијенти за негативне вредности
- „брзо“ се израчунава
- Бржа конвергенција (у пракси) од sigmoid/tanh



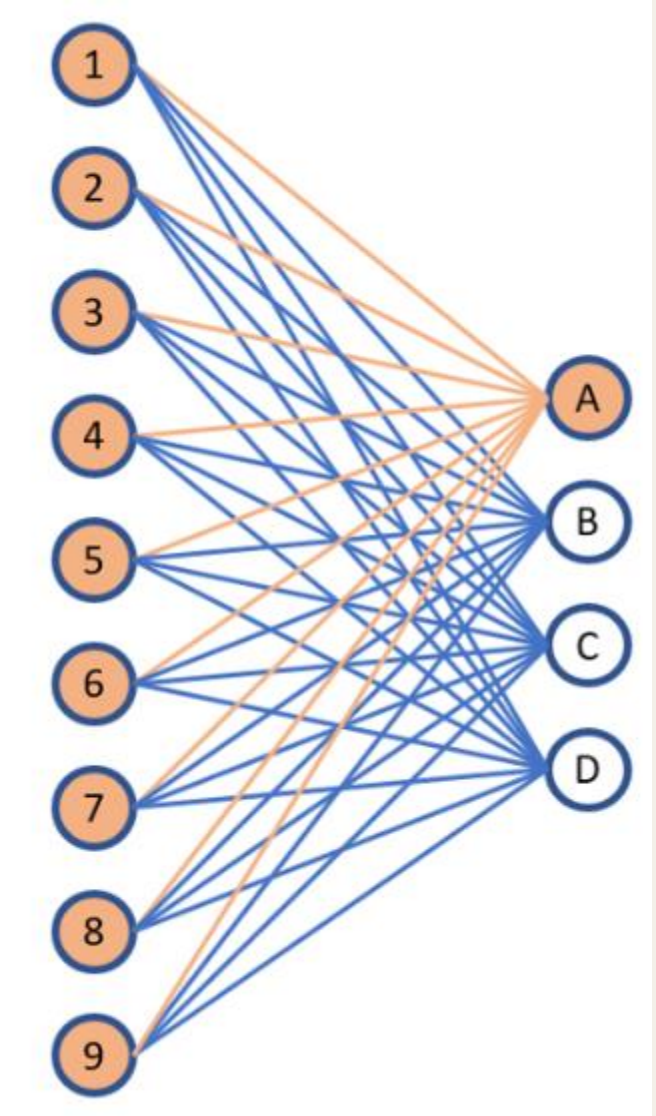
Слика као улаз у неуронску мрежу

- Слика је матрица бројева
- Сваки пиксел је представљен је једном вредношћу (у случају RGB са три)
- Сваки пиксел је информација која представља улаз у мрежу
- Слика димензија $800 \times 600 \times 3$ резултује са 1440000 улазних чворова



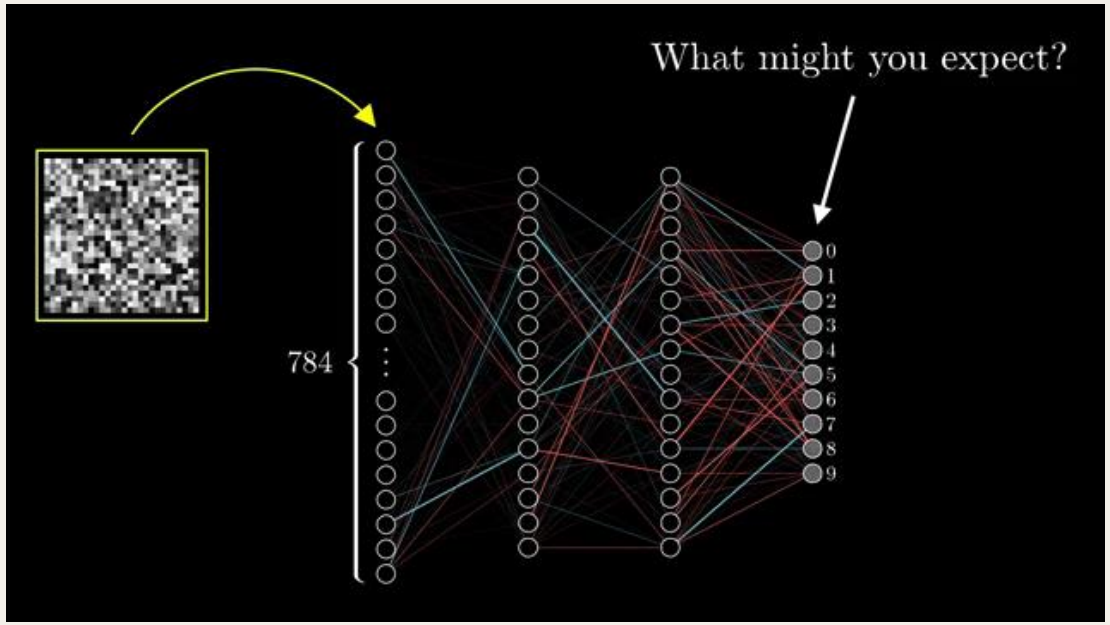
Слојеви у неуронским мрежама

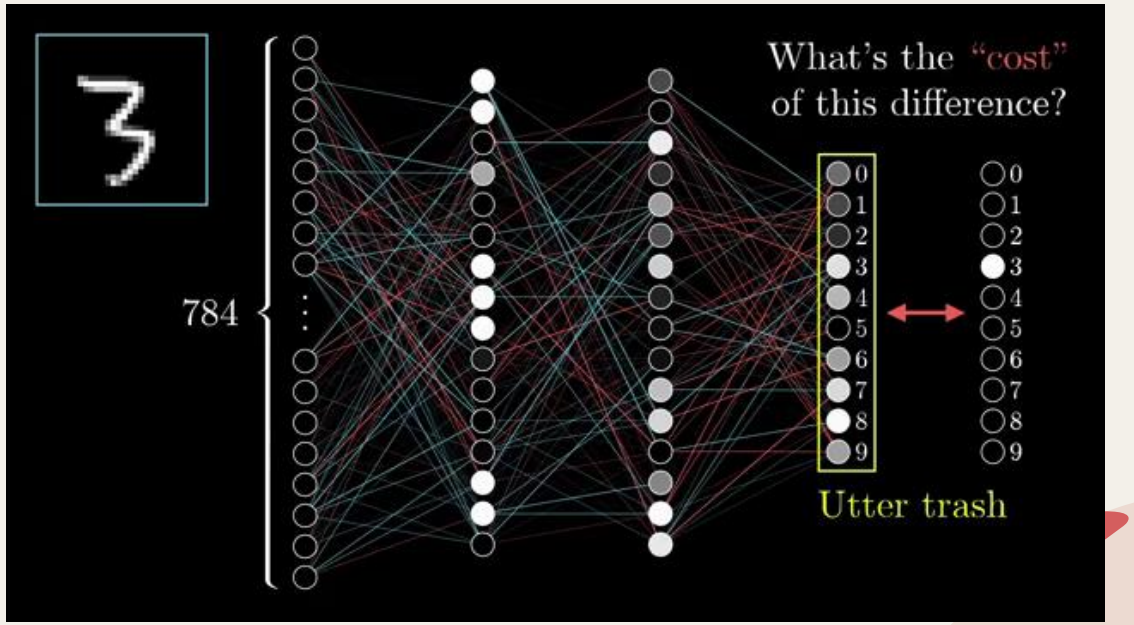
- Потпуно повезани слојеви (Fully connected, Dense)
  - Сваки неурон је повезан са сваким неуроном из претходног слоја
  - Тренингом се одређују тежине грана које повезују један слој са другим
- Када је улаз слика, број неурона (и тежина) постаје брзо јако велики
  - Нпр. слика димензије $800 \times 600 \times 3$ представља 1440000 улазних чворова
  - Ако први скривени слој има 10 неурона, само у том слоју има $14\,400\,000$ параметара које треба одредити



https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b

Слојеви у неуронским мрежама - Fully connected, Dense



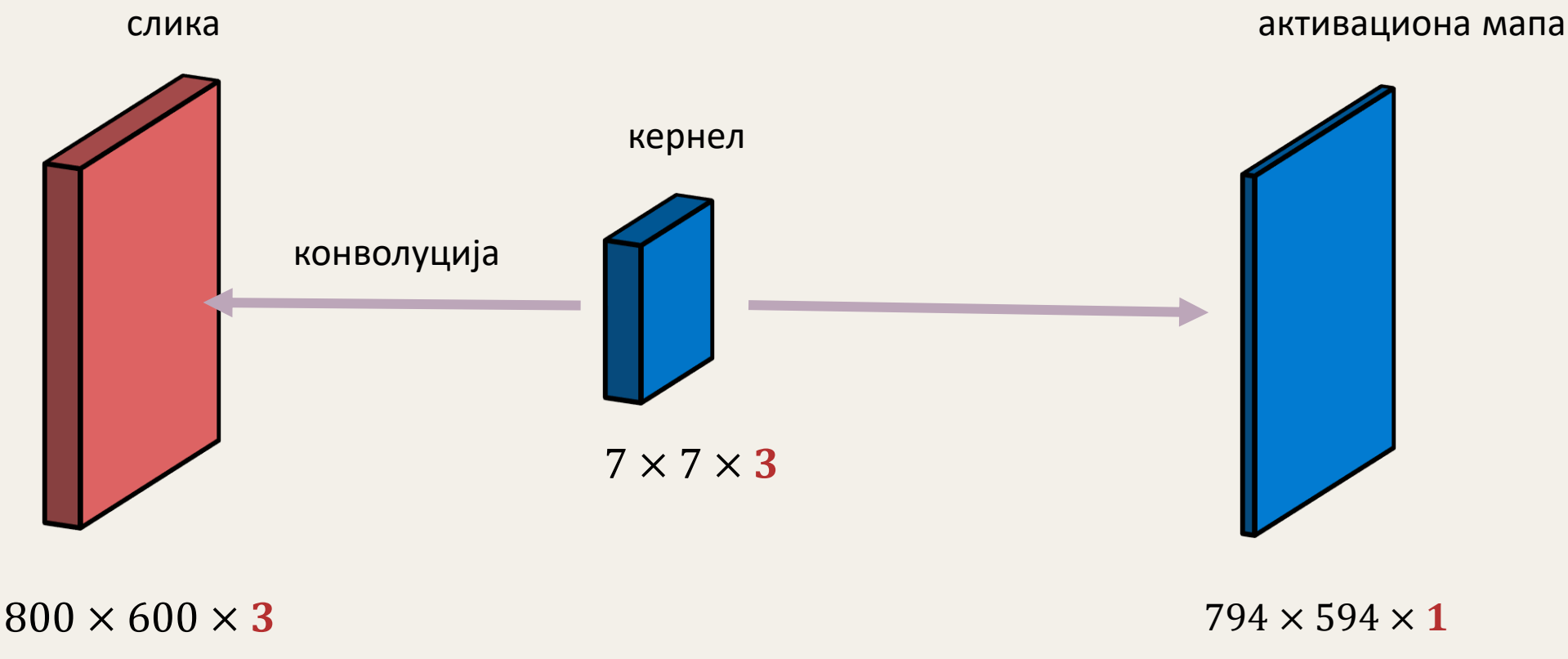


https://gfycat.com/

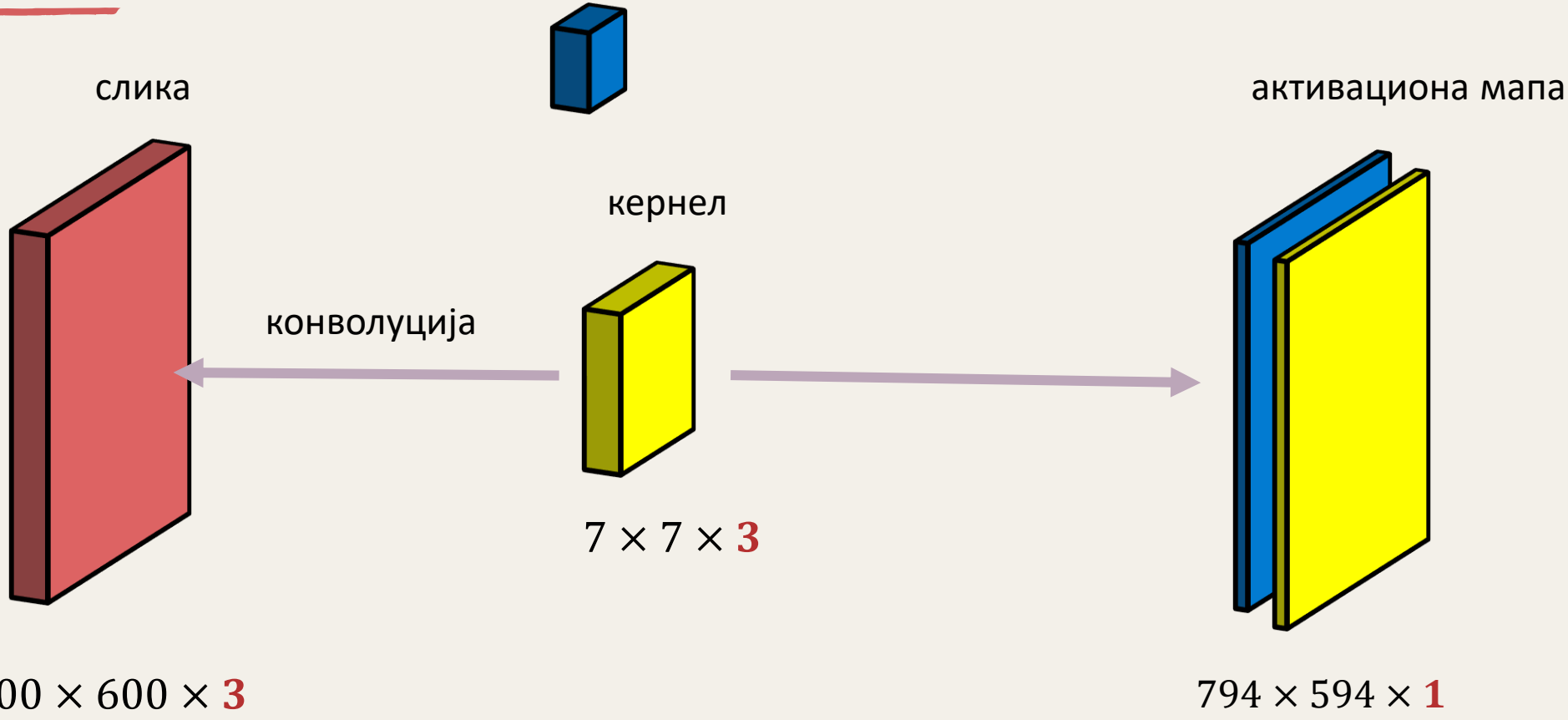
Слојеви у неуронским мрежама
Конволуциони слој

- Конволуциони слојеви (Convolutional Layers)
  - Третирају слику као тродимензионалну структуру (код потпуно повезаних то није случај јер су пиксели слике поређани у низ)
  - Сваки слој представља колекцију конволуционих филтера којима се прелази преко слике – резултат је активациона мапа филтера

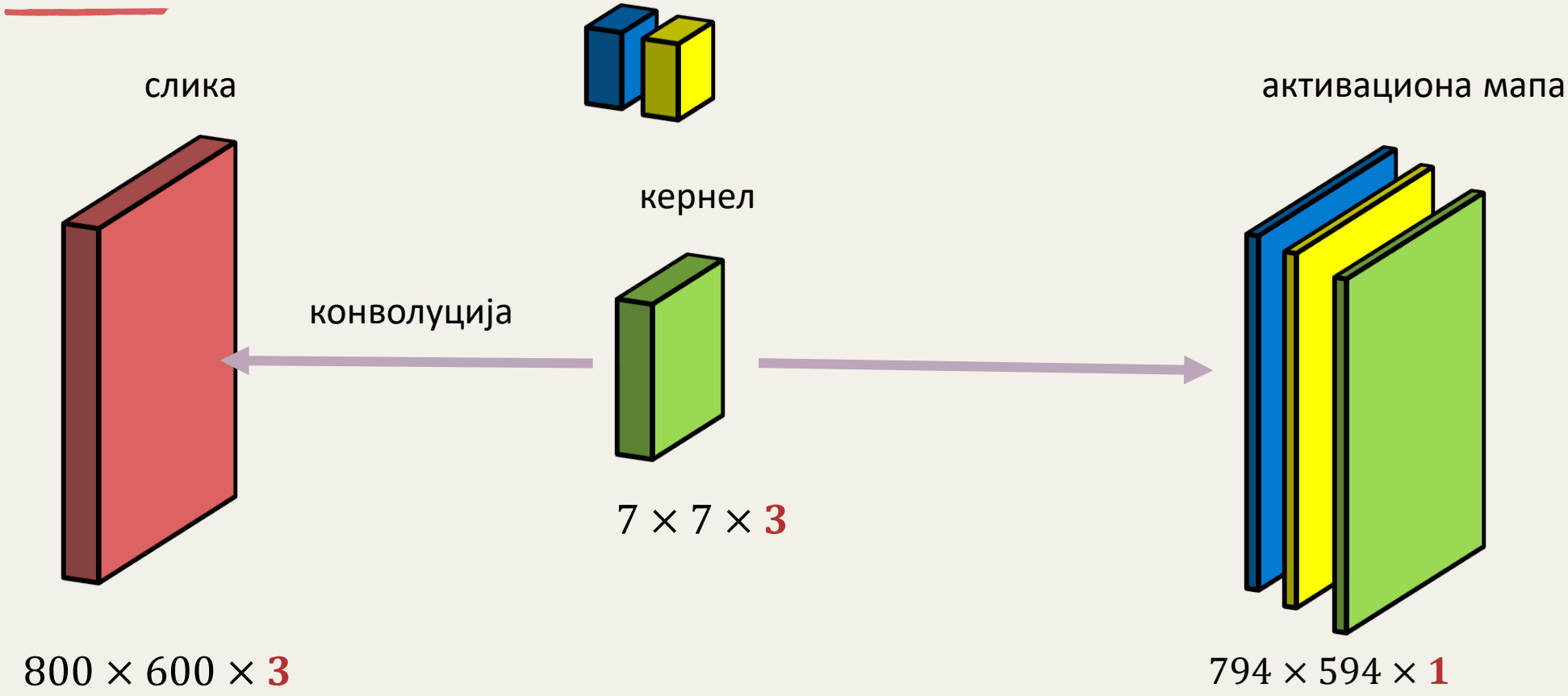
Слојеви у неуронским мрежама
Конволуциони слој



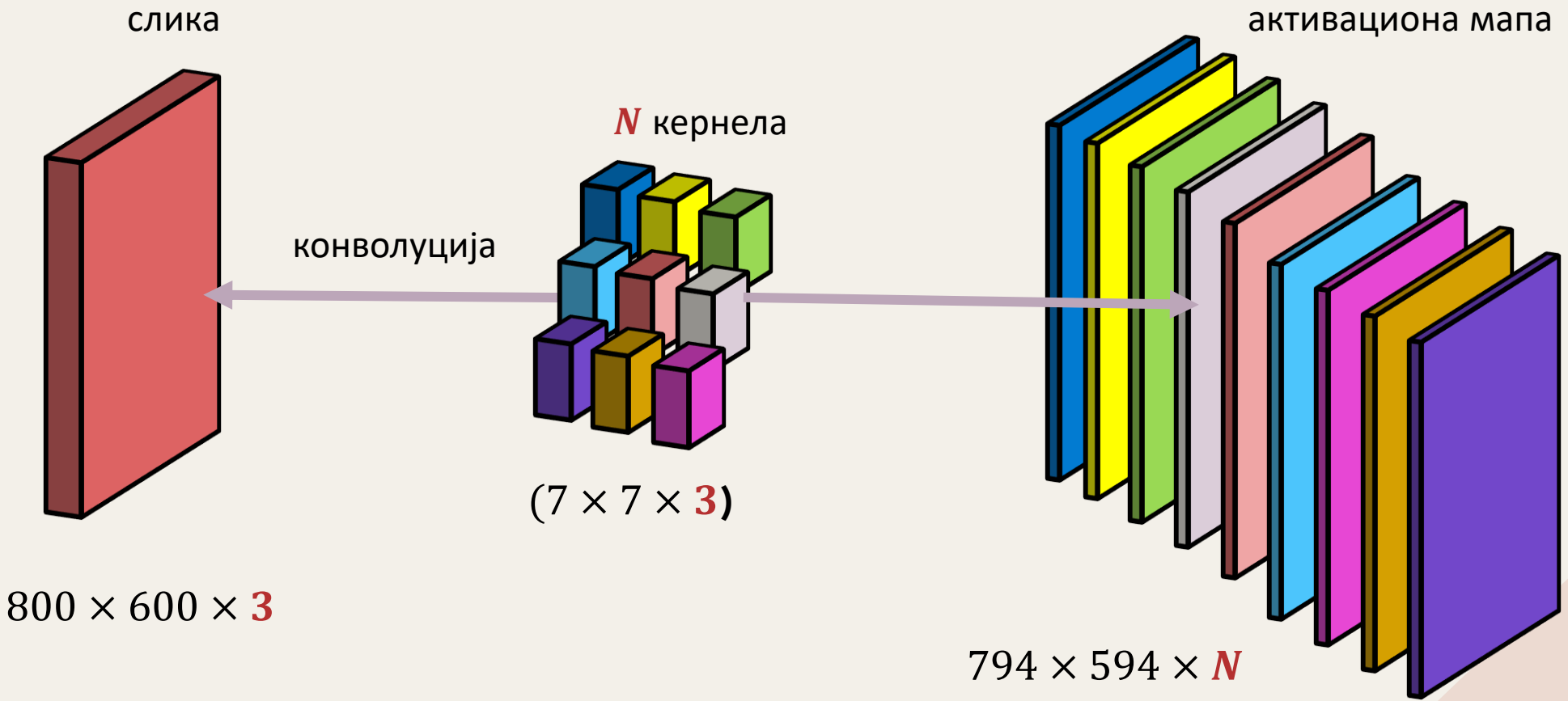
Слојеви у неуронским мрежама
Конволуциони слој



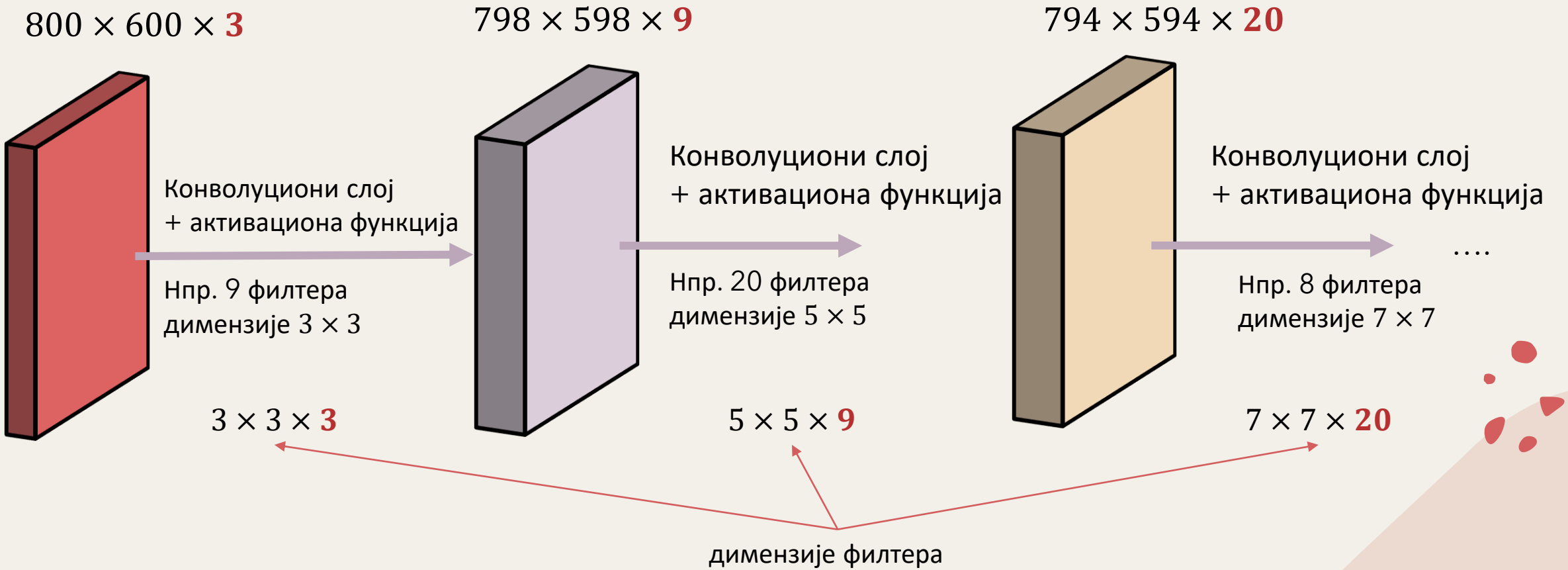
Слојеви у неуронским мрежама
Конволуциони слој



Слојеви у неуронским мрежама
Конволуциони слој

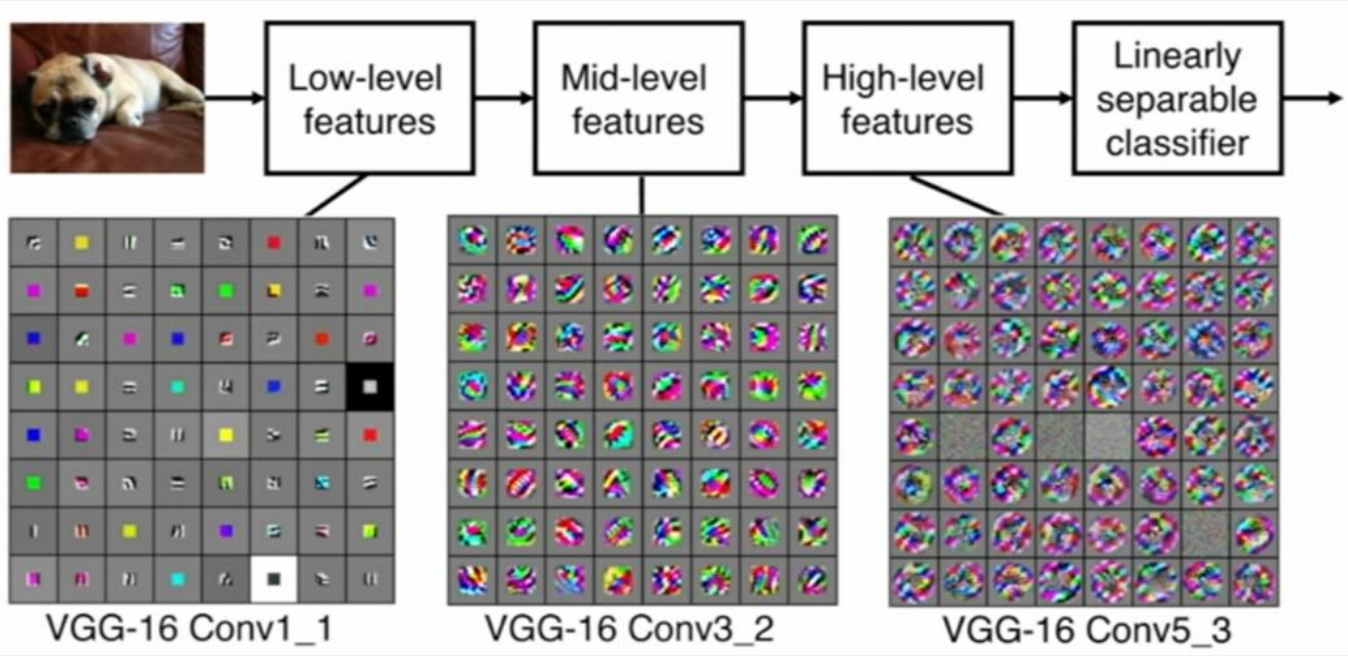


Слојеви у неуронским мрежама
Конволуциони слој



Слојеви у неуронским мрежама
Конволуциони слој

- Тренирањем мреже одређују се вредности филтера
- Које филтере мрежа научи?



https://www.youtube.com/watch?v=d14TUNcbn1k&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=4

Слојеви у неуронским мрежама
Конволуциони слој

- Конволуциони слој прихвата улазни тензор димензије $W_1 \times H_1 \times D_1$
  - K – Број филтера (обично степен 2)
  - F – Димензију филтера (обично је квадратна, 3,5,7,…,)
    - Није потребно наводити дубину филтера јер је могуће израчунати је на основу претходног слоја
  - S – Stride - за колико се помера кернел приликом рачунања конволуције
  - P – Padding - за колико се проширује улазни тензор пре конволуције
- Резултат еје тензор димензија $W_2 \times H_2 \times D_2$, где је
  - $W_2 = \frac{W_1 - F + 2P}{S} + 1$
  - $H_2 = \frac{H_1 - F + 2P}{S} + 1$
  - $D_2 = K$

https://www.youtube.com/watch?v=d14TUNcbn1k&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=4

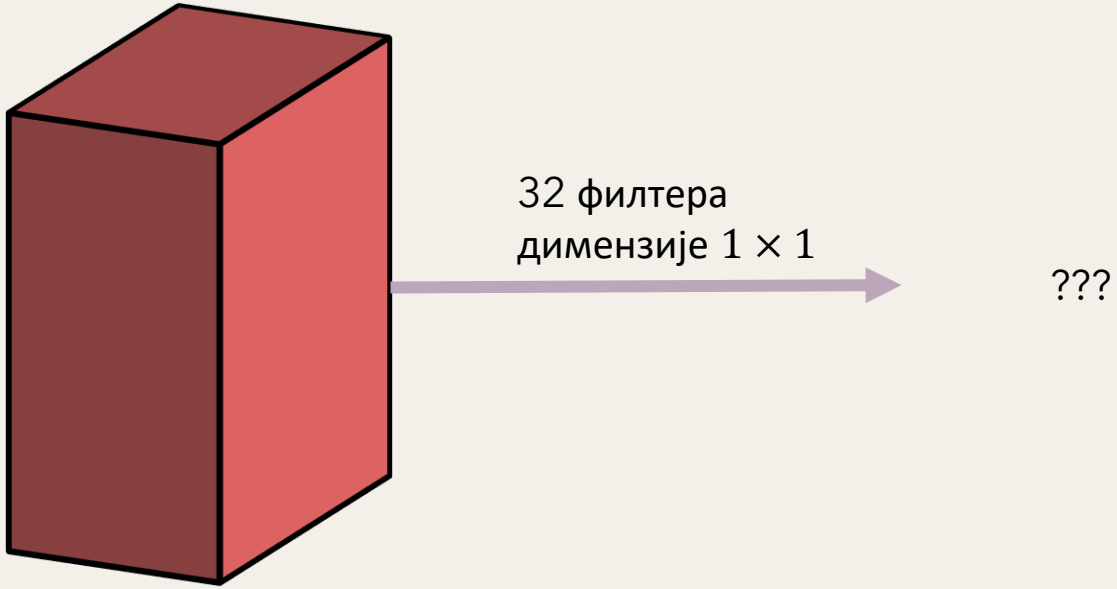
Слојеви у неуронским мрежама
Конволуциони слој

- Број параметара
  - У једном слоју, укупан број параметара је $(F \times F \times D_1) \times K + K$ (за $bias$)
  - Не зависи од димензије улазне слике

https://www.youtube.com/watch?v=d14TUNcbn1k&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=4

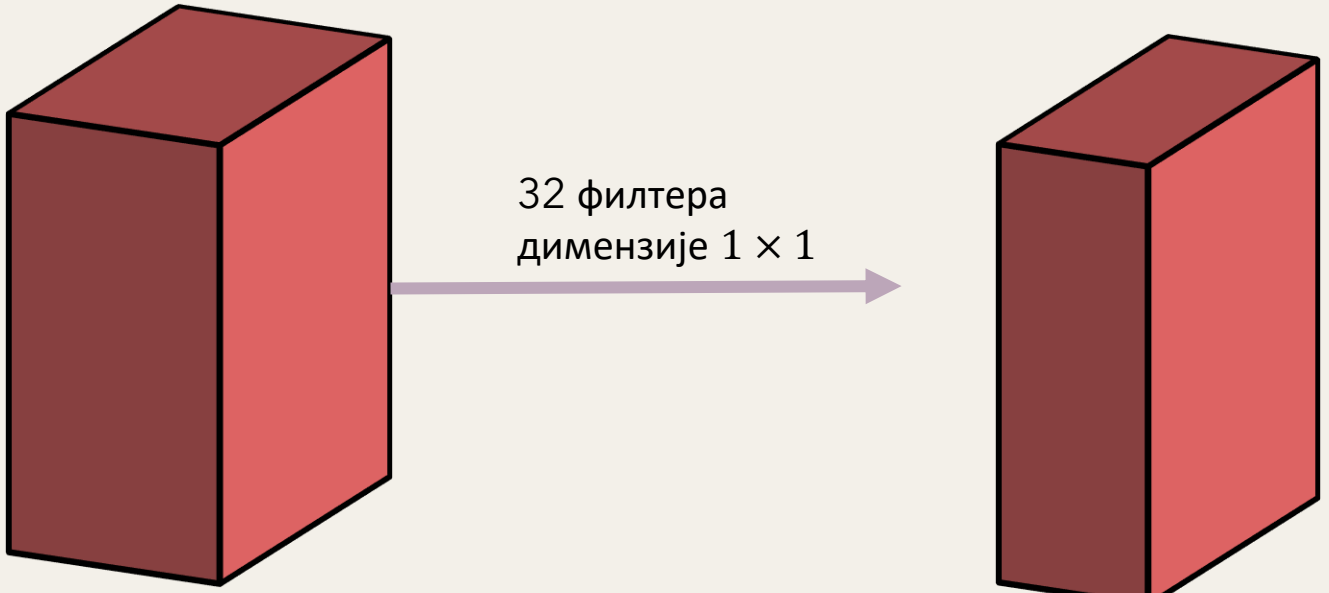
Слојеви у неуронским мрежама
Конволуциони слој - 1×1 *конволуција*

$100 \times 100 \times \mathbf{64}$



Слојеви у неуронским мрежама
Конволуциони слој - 1×1 *конволуција*

$100 \times 100 \times \mathbf{64}$

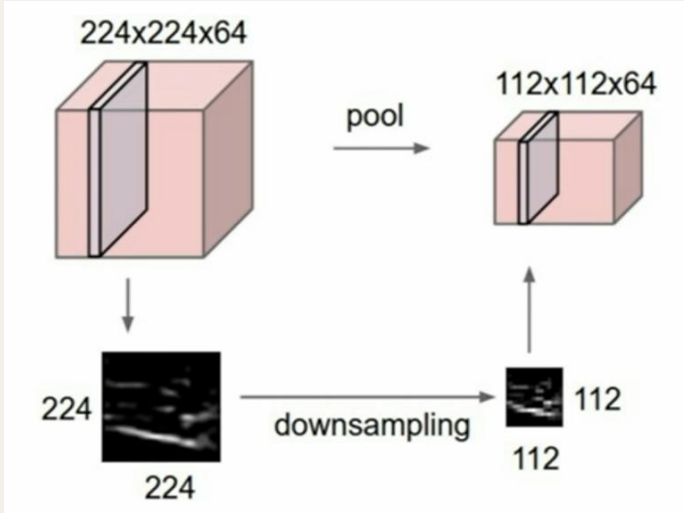


$100 \times 100 \times \mathbf{32}$

Резултат једног филтера је скаларни производ два 64-димензионална вектора (један је кернел)

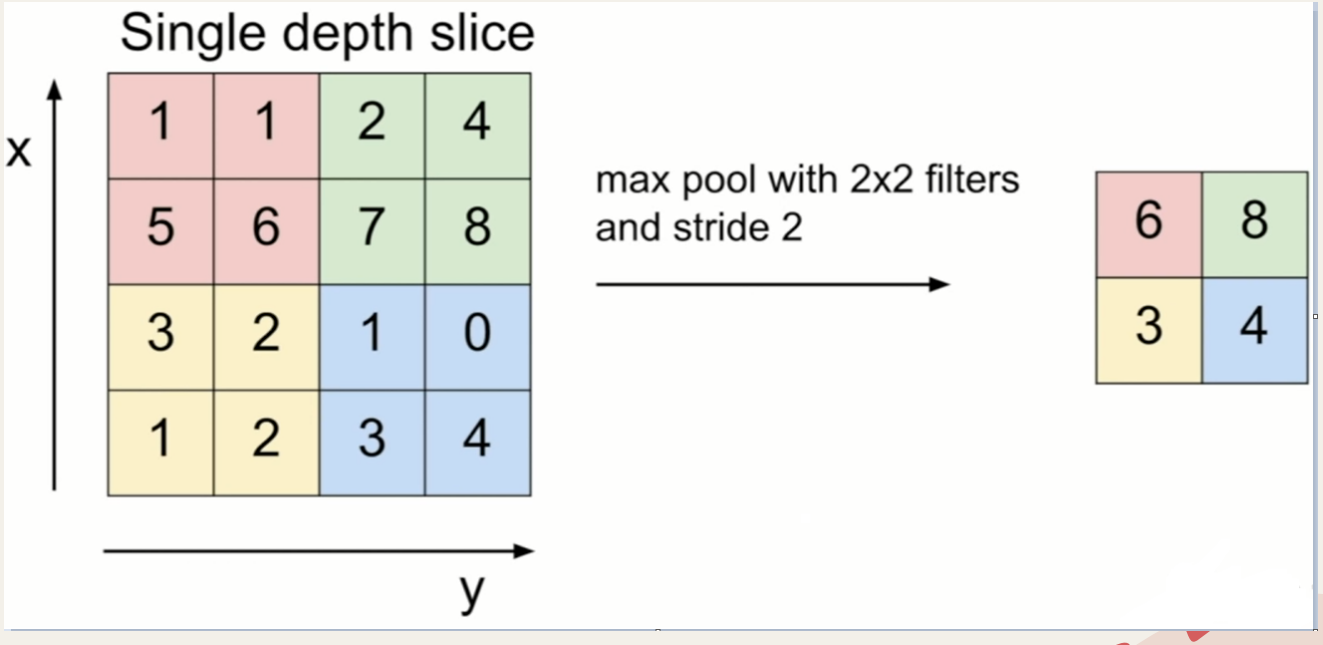
Слојеви у неуронским мрежама
Pooling слој

- Резултат конволуционог слоја је и даље (просторно) доста велики
  - Смањивање димензионалности је могуће постављањем одговарајућег stride-a, међутим то утиче и на резултат активационе мапе
- Циљ Pooling слоја је да смањи димензионалност репрезентације што омогућава лакшу манипулацију
- Не утиче на дубину, само на „дужину“ и „ширину“ тензора



Слојеви у неуронским мрежама
Pooling слој – MAX Pooling

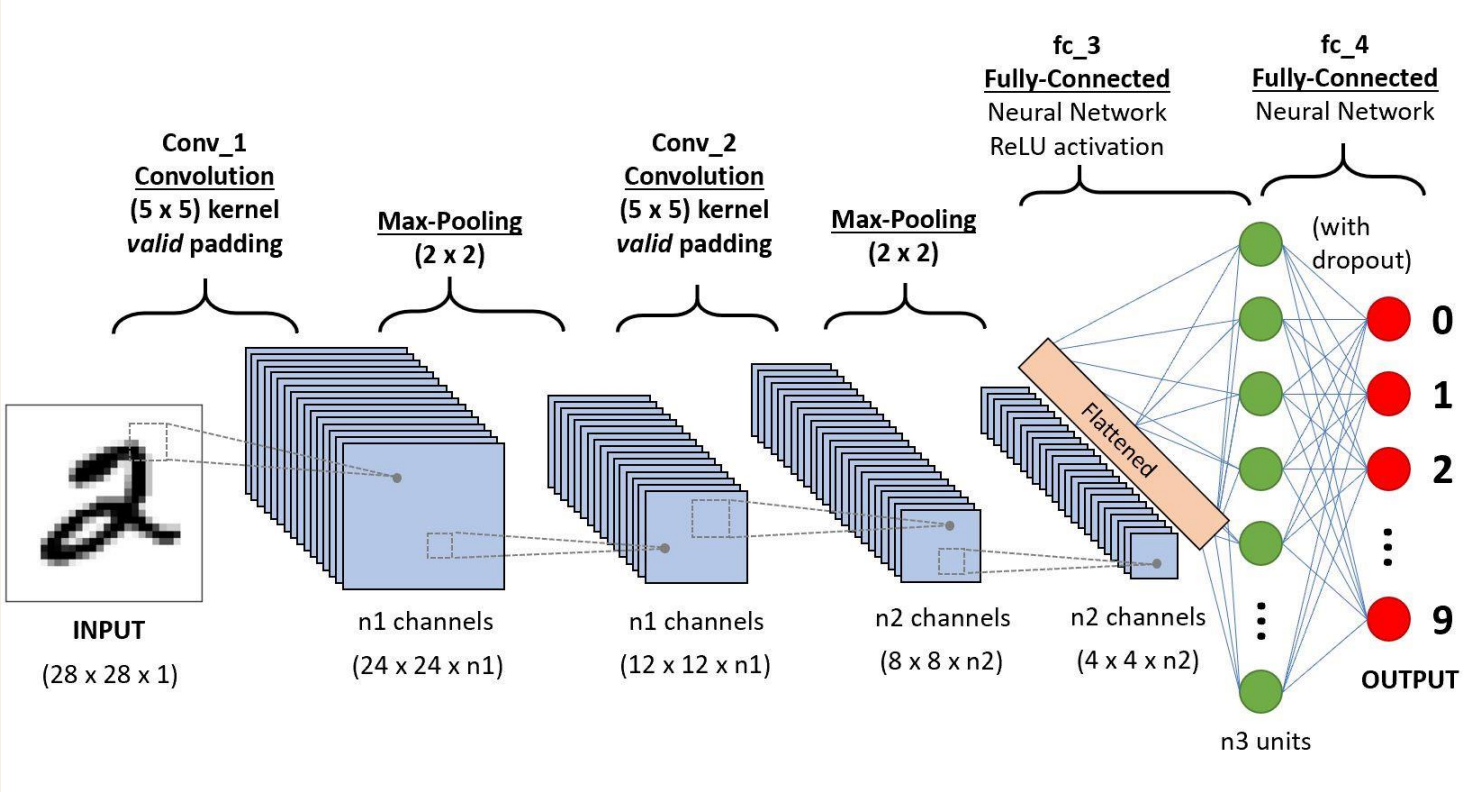
Интуиција: Један број у левој матрици представља активацију филтера на одређеној локацији (нпр. филтер који детектује појаву „ока“). У коначном резултату, битна је информација да ли се „око“ појавило на слици или не. Максимална вредност са суседних локација може да се протумачи као „око се појавило (или не) у овом региону слике“



Слојеви у неуронским мрежама
Pooling слој

- Pooling операција: најчешће max или average
- Pooling слој прихвата улазни тензор димензије $W_1 \times H_1 \times D_1$ и
  - F – Димензију филтера (обично 2)
  - S – Stride - за колико се помера кернел приликом рачунања конволуције (обично је исте димензије као и филтер)
- Резултат еје тензор димензија $W_2 \times H_2 \times D_2$, где је
  - $W_2 = \frac{W_1 - F}{S} + 1$
  - $H_2 = \frac{H_1 - F}{S} + 1$
  - $D_2 = D_1$
- Не уводи додатне параметре јер рачуна фиксну функцију (нпр. максимум)

Типична архитектура конволуционе мреже



https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/

Практичан пример

1. XOR имплементација
2. Класификација слика, MLP vs. CNN