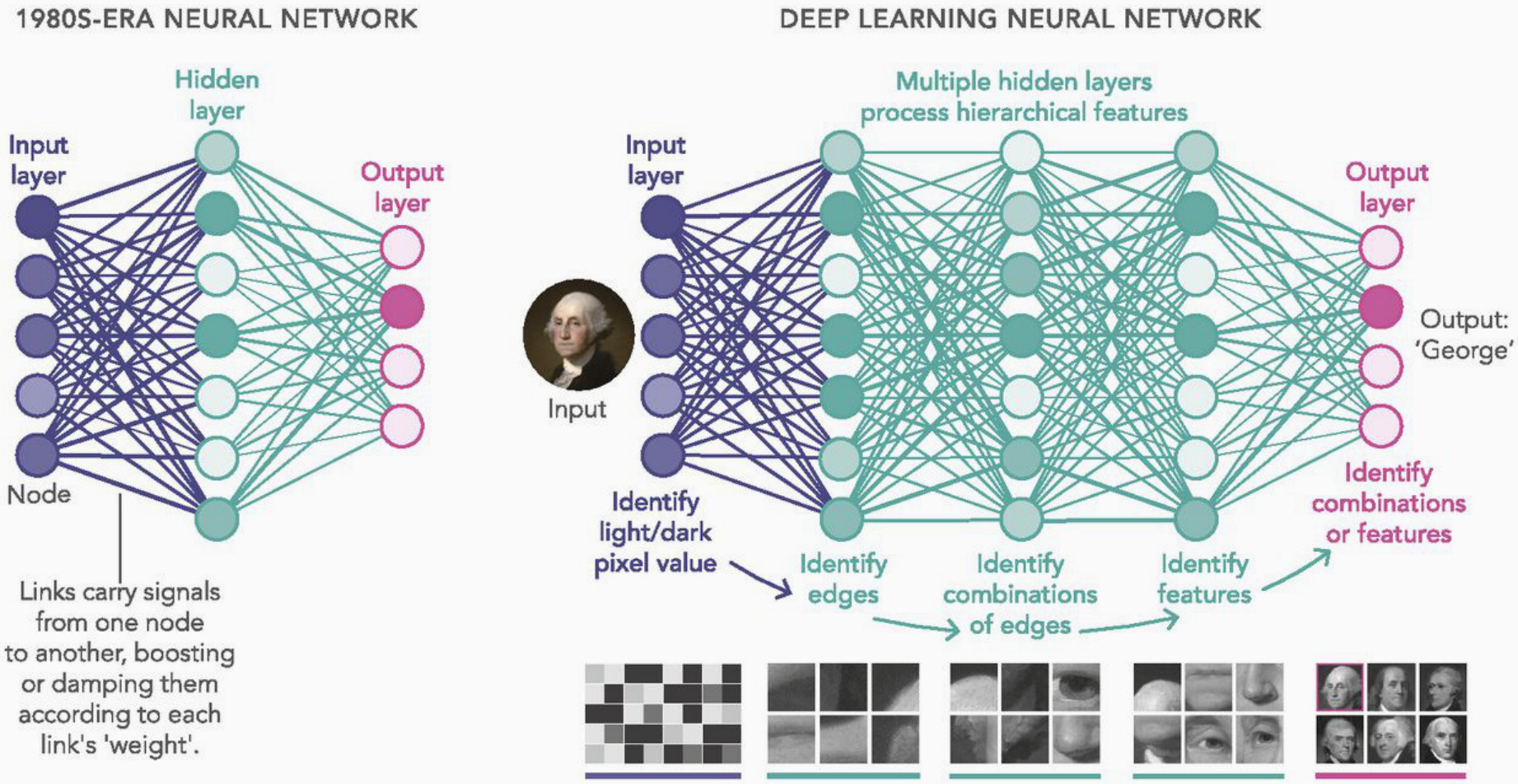
Konvolúciós neuronhálók (CNN)

Deep learning



https://www.pnas.org/content/116/4/1074

Alkalmazás: képek/szövegek osztályozása, képek szegmentálása, szöveg fordítása, hang átírása szöveggé, képek generálása, stb

Konvolúciós operátor

- diszkrét konvolúció két egydimenziós jel között:

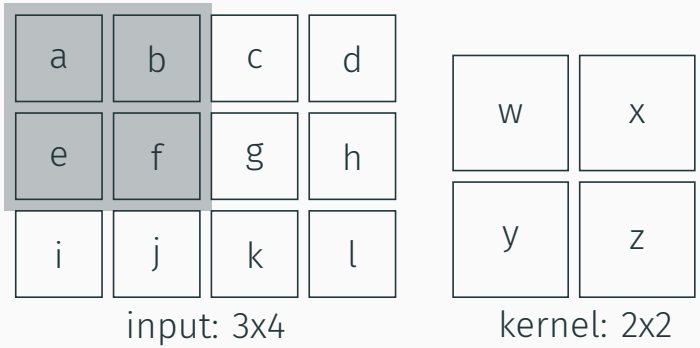
$$(f*g)(n)=\sum_t f(t)g(n+t)$$

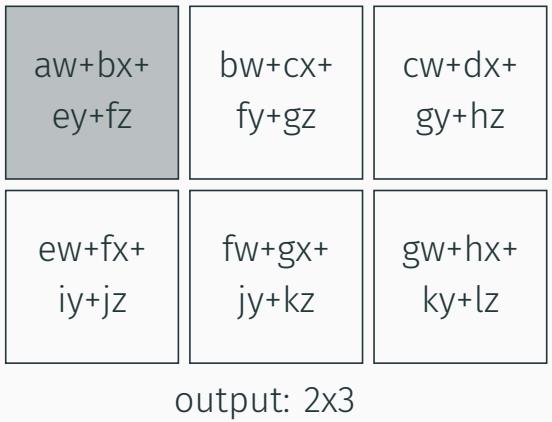
példa: $f=[a,b], \quad g=[w,x,y,z]$
$f*g=[aw+bx, ax+by, ay+bz]$

- diszkrét konvolúció két kétdimenziós jel között:

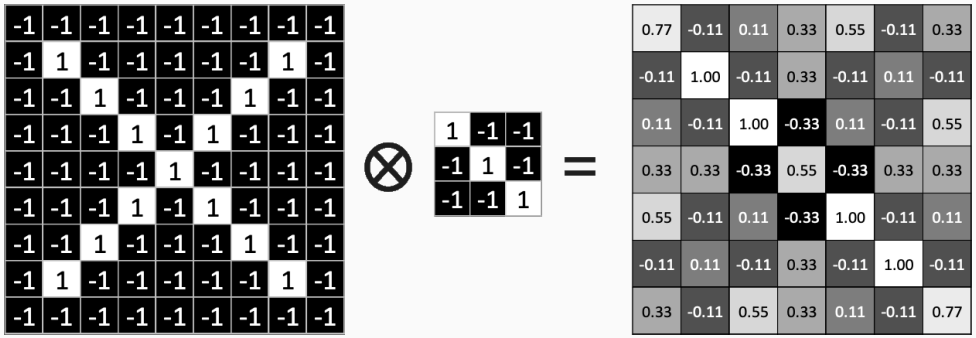
$$(K*I)(i,j)=\sum_{m,n} K(m,n)I(i+n,j+m)$$

jelentés: a K konvolúciós kernel/filter/szűrő alkalmazása az I jelre/képre





Konvolúciós operátor, miért?

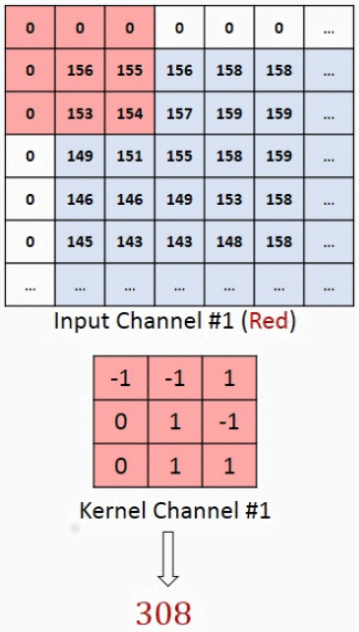


https://brohrer.github.io/how_convolutional_neural_networks_work.html

JS demo: https://setosa.io/ev/image-kernels/

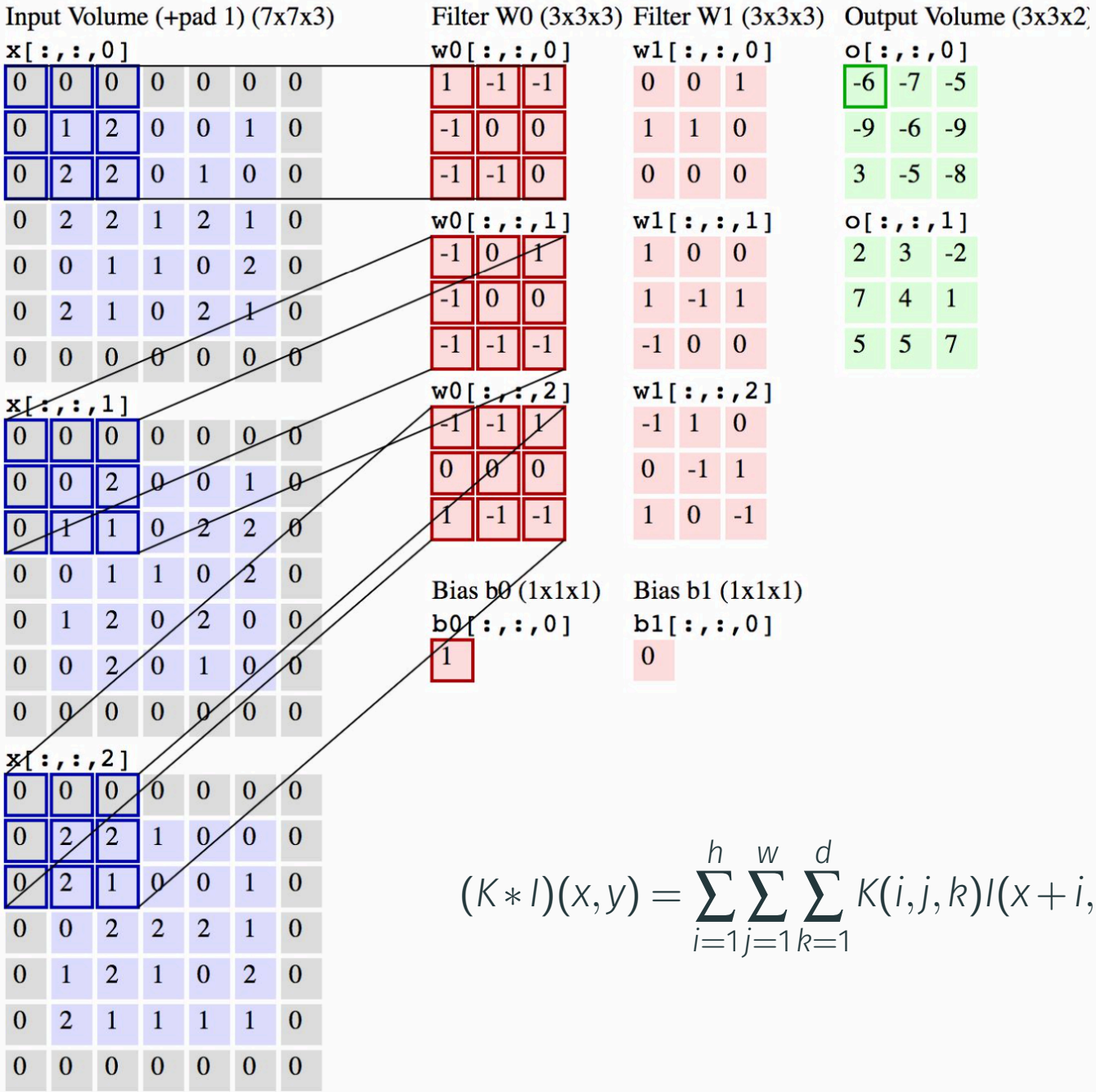
Padding, stride, channel, volume

- padding: az kép (rendszerint nullákkal való) szegélyezése
- stride: mennyivel toljuk el egy iterációban a kernelt
- channel: rendszerint R, G, B
- volume: a konvolúciós réteg (>2D) bemenete/kimenete



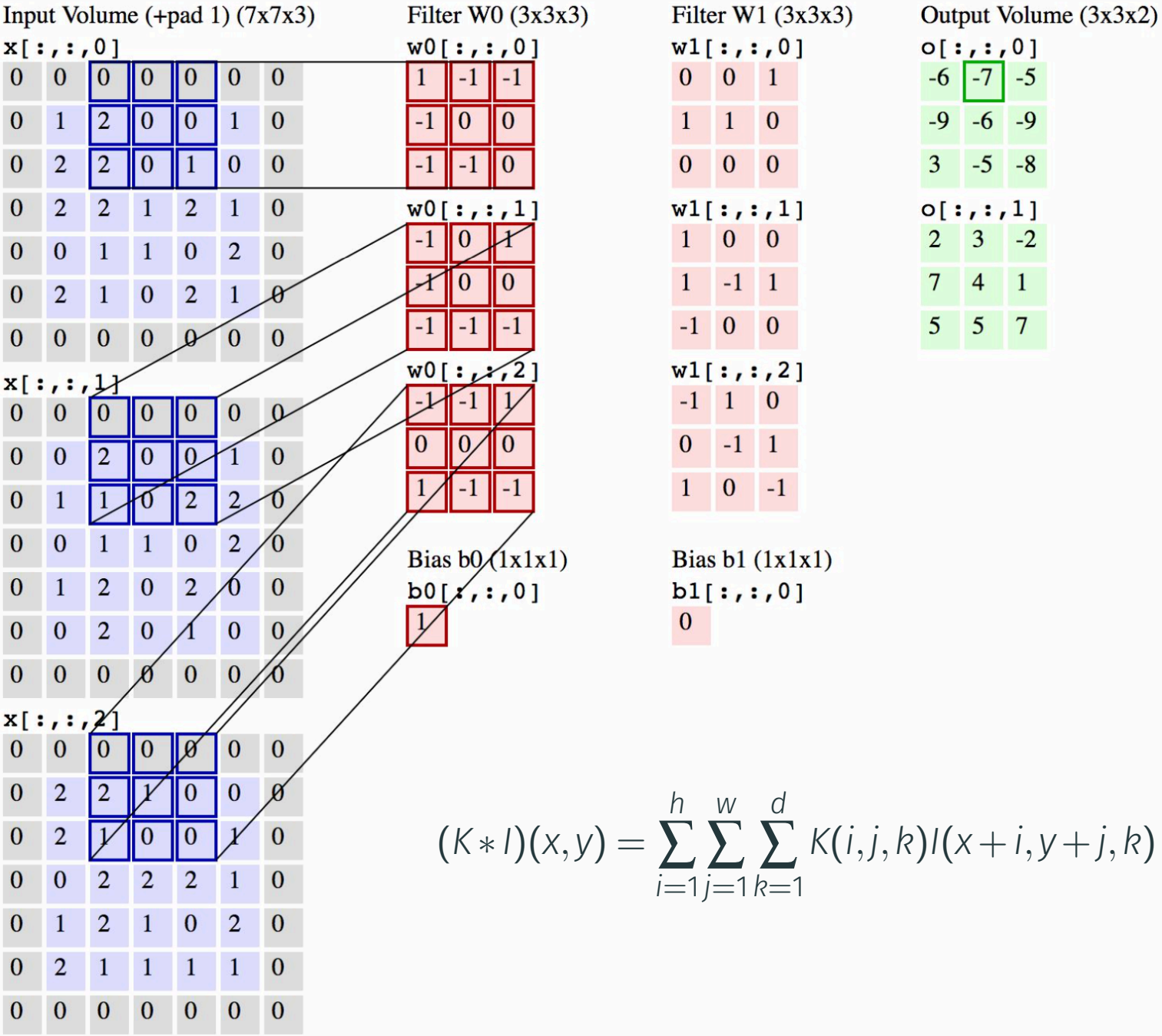
https://lipyeow.github.io/ics491f17/morea/deepnn/ImageClassification-CNN.pdf

3D kernel



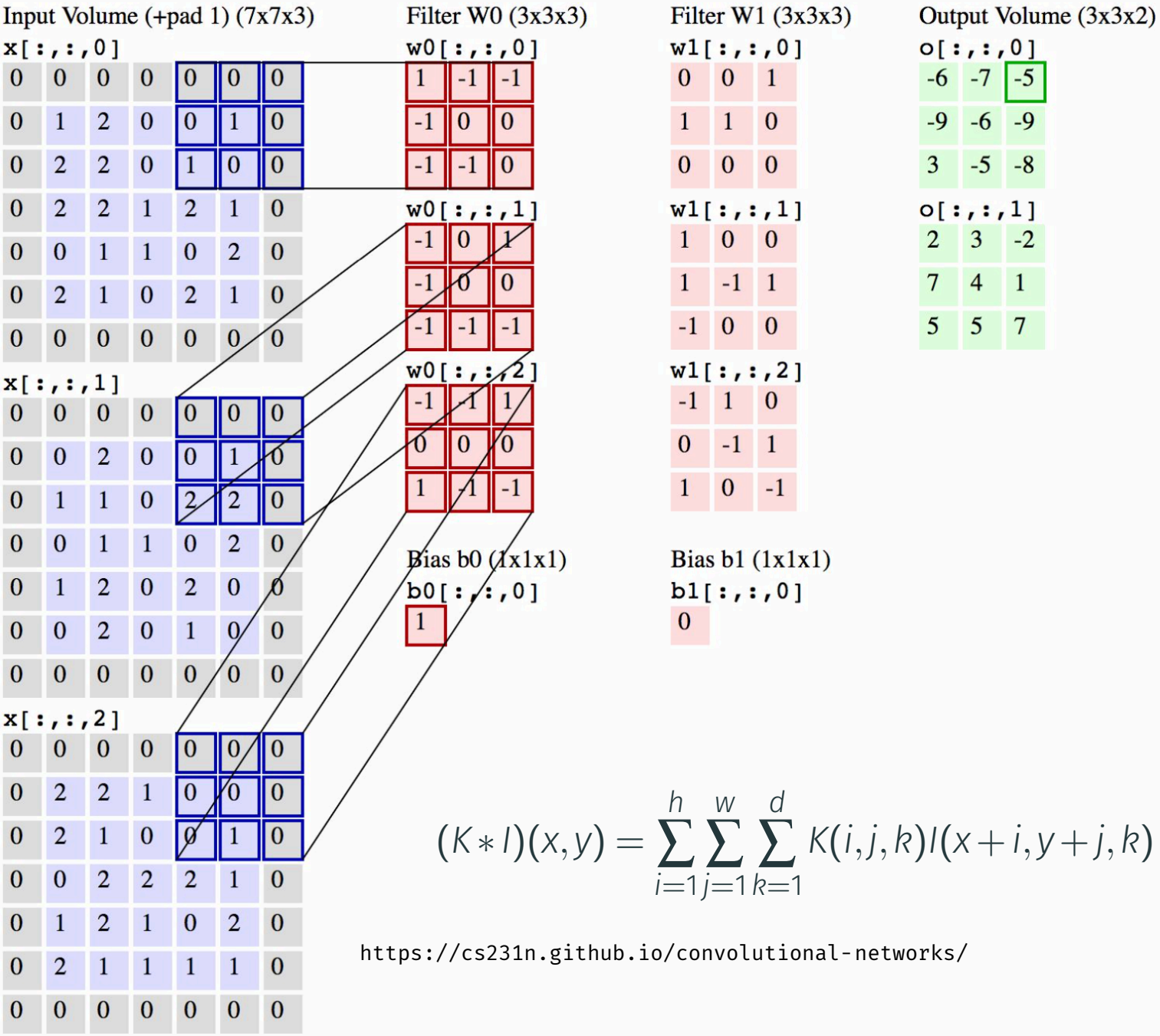
$$(K * I)(x,y) = \sum_{i=1}^{h} \sum_{j=1}^{w} \sum_{k=1}^{d} K(i,j,k) I(x+i, y+j, k)$$

3D kernel



$$(K * I)(x,y) = \sum_{i=1}^{h} \sum_{j=1}^{w} \sum_{k=1}^{d} K(i,j,k) I(x+i, y+j, k)$$

3D kernel

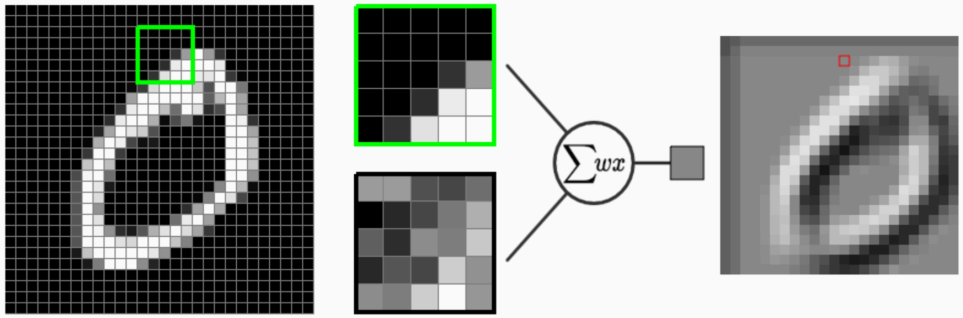


$$(K * I)(x,y) = \sum_{i=1}^{h} \sum_{j=1}^{w} \sum_{k=1}^{d} K(i,j,k) I(x+i, y+j, k)$$

https://cs231n.github.io/convolutional-networks/

Konvolúciós réteg

- cél: térbeli jellemzők kinyerése
- bemenet: $W_1 \times H_1 \times D_1$
- kernel: $F \times F \times K$
  - hiperparaméter: stride (S)
  - hiperparaméter: padding (P)
- kimenet: $W_2 \times H_2 \times D_2$
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$
  - $D_2 = K$
- a kernelek a megtanulandó paraméterek közé tartoznak
- alkalmazás: kép, hang, idősor, stb feldolgozása

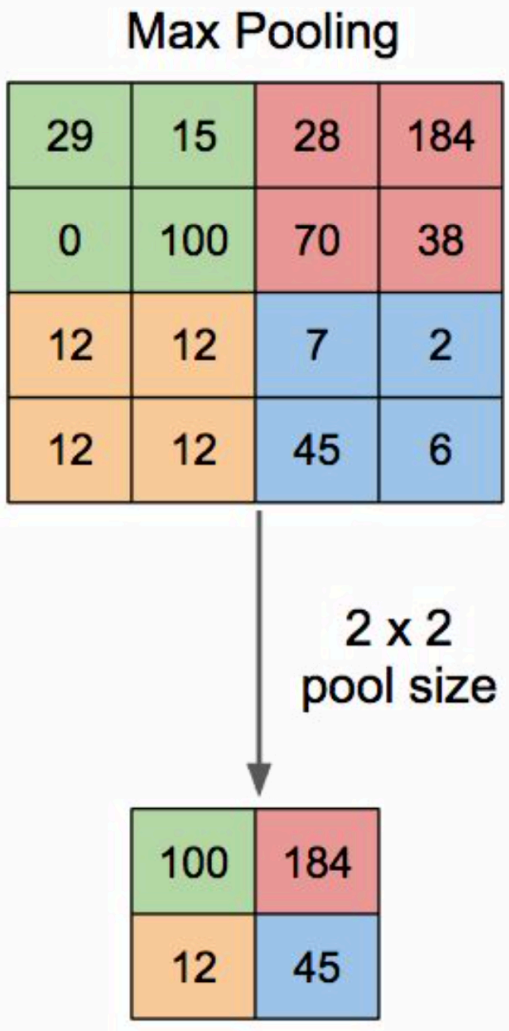


ReLU

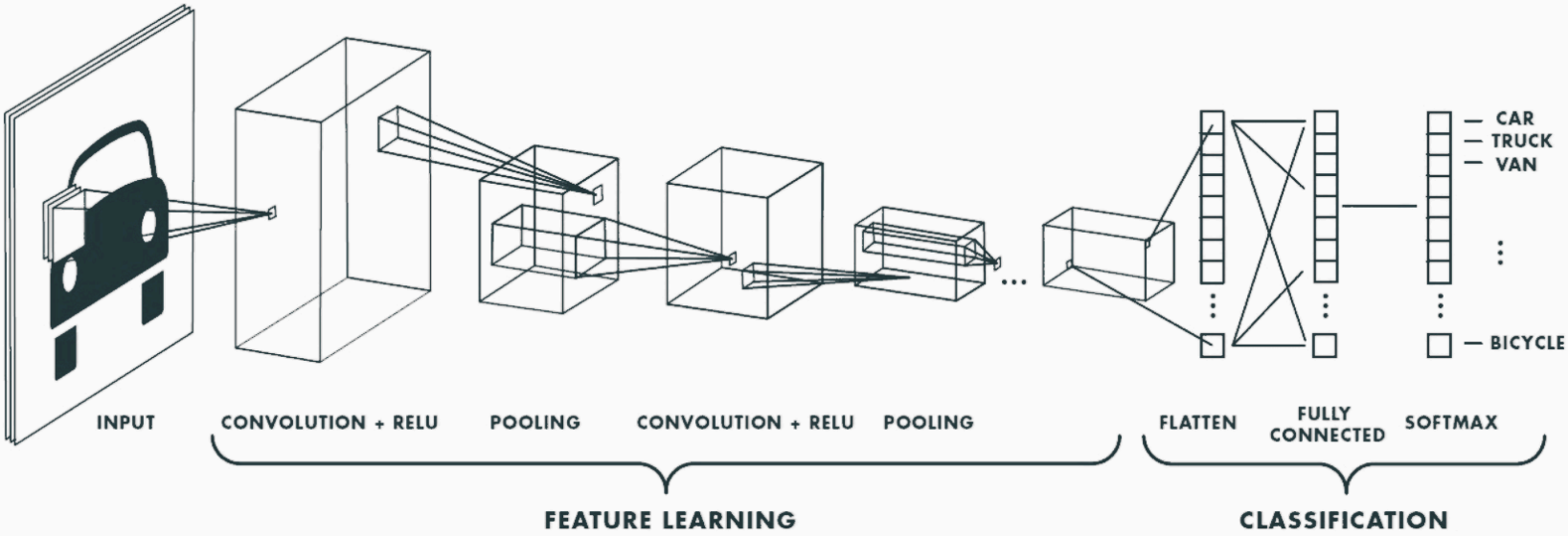
- cél: nonlinearitás
- $ReLU(x) = \max(0, x)$
- alternatívák: LeakyReLU, Param ReLU, Rand ReLU, exponential linear unit, stb.
- a nonlináris réteg kimenete: activation map

Mintavételező réteg

- cél: a bemenet méretének csökkentése
- bemenet: $W_1 \times H_1 \times D_1$
- subsampling/pooling:
  - hiperparaméter: méret ($F \times F$)
  - hiperparaméter: stride (S)
  - pooling függvény: max, vagy avg
- kimenet: $W_2 \times H_2 \times D_2$
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$



Generikus CNN



Demo:

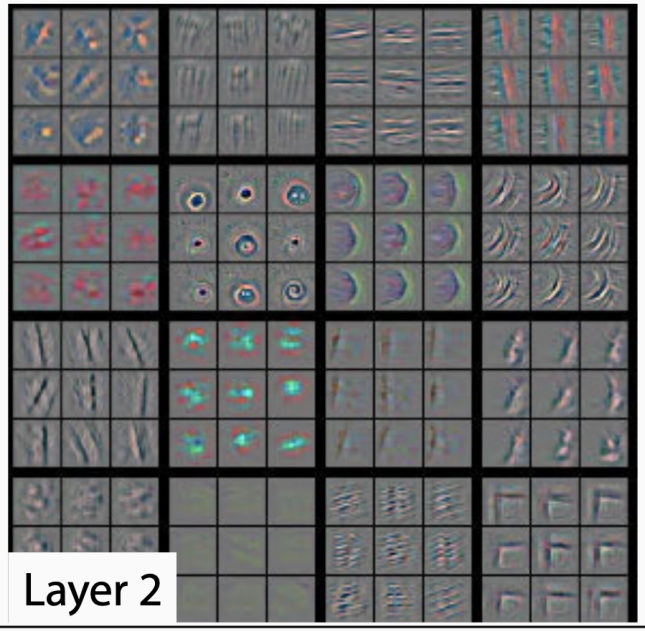
https://cs.stanford.edu/~karpathy/convnetjs/demo/mnist.html

https://www.cs.ryerson.ca/~aharley/vis/conv/flat.html

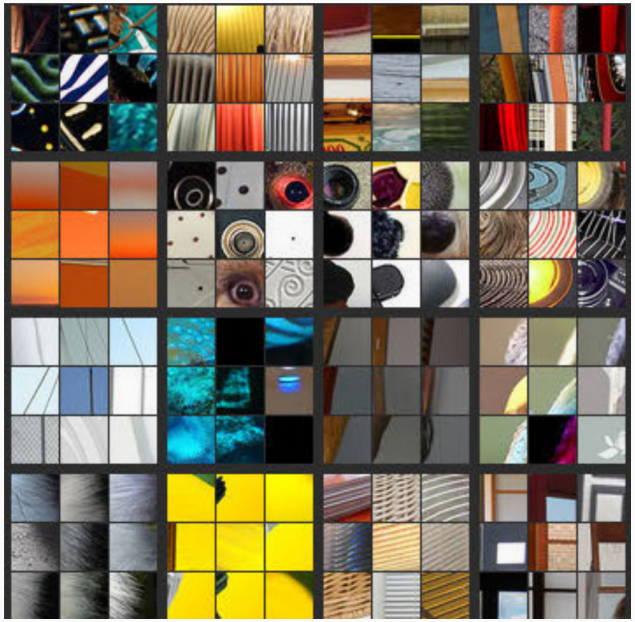
https://www.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html

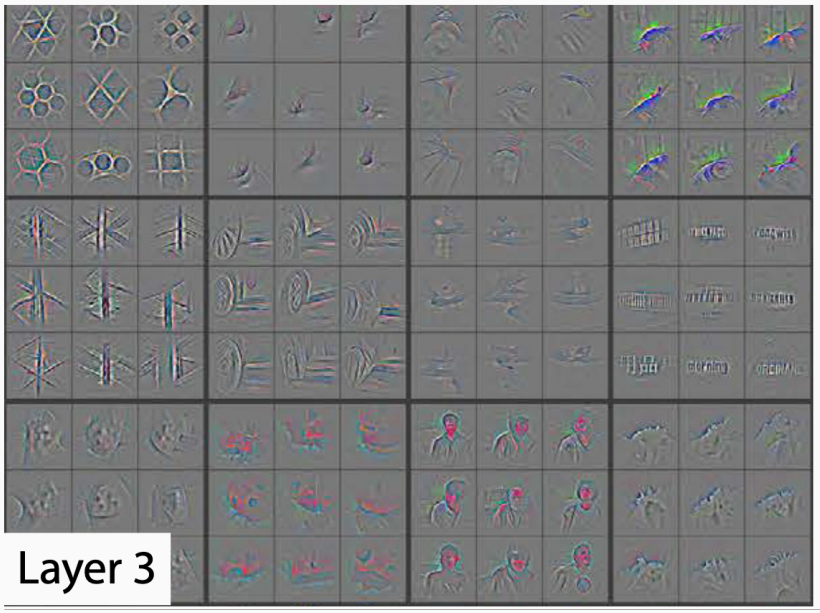
A konvolúciós rétegek vizualizációja

Layer 1

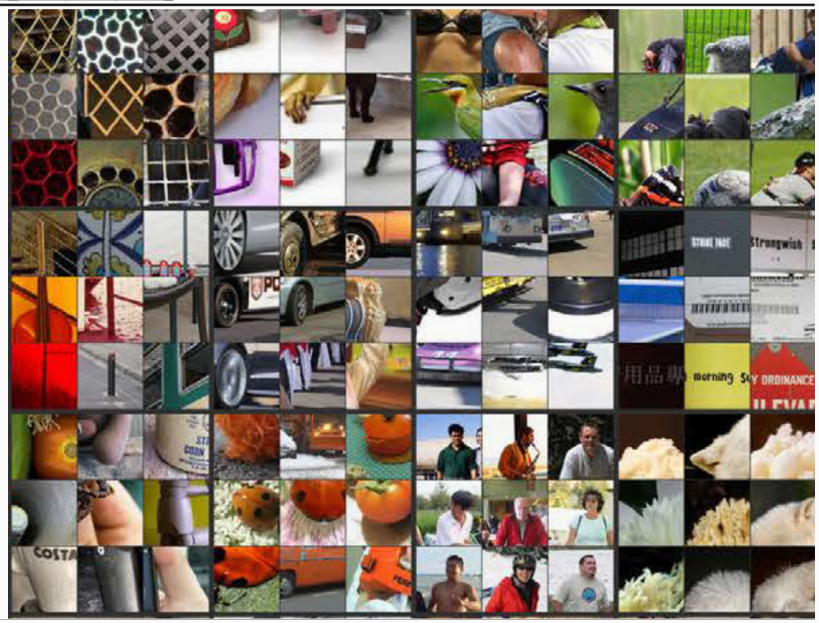


Layer 2



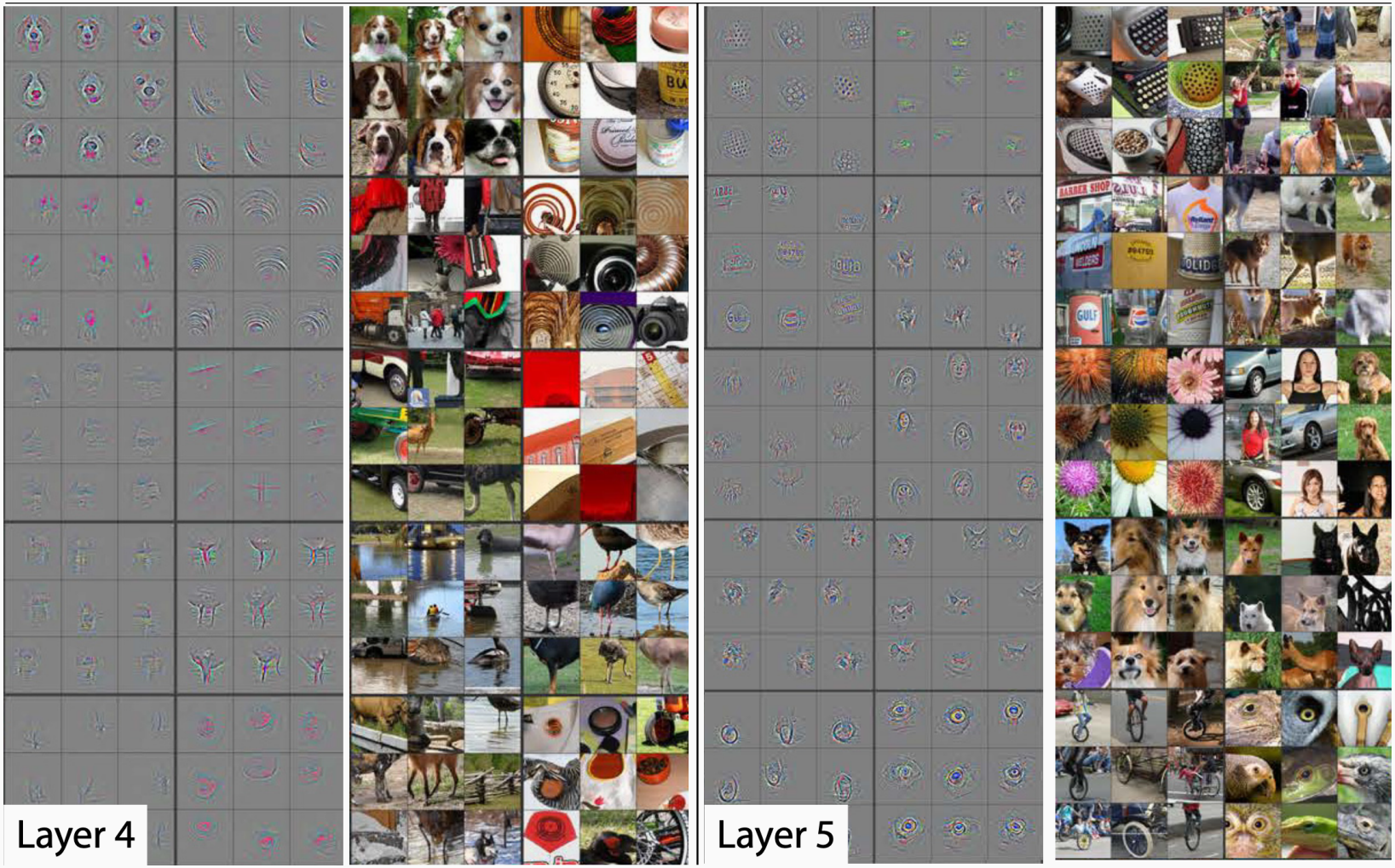


Layer 3



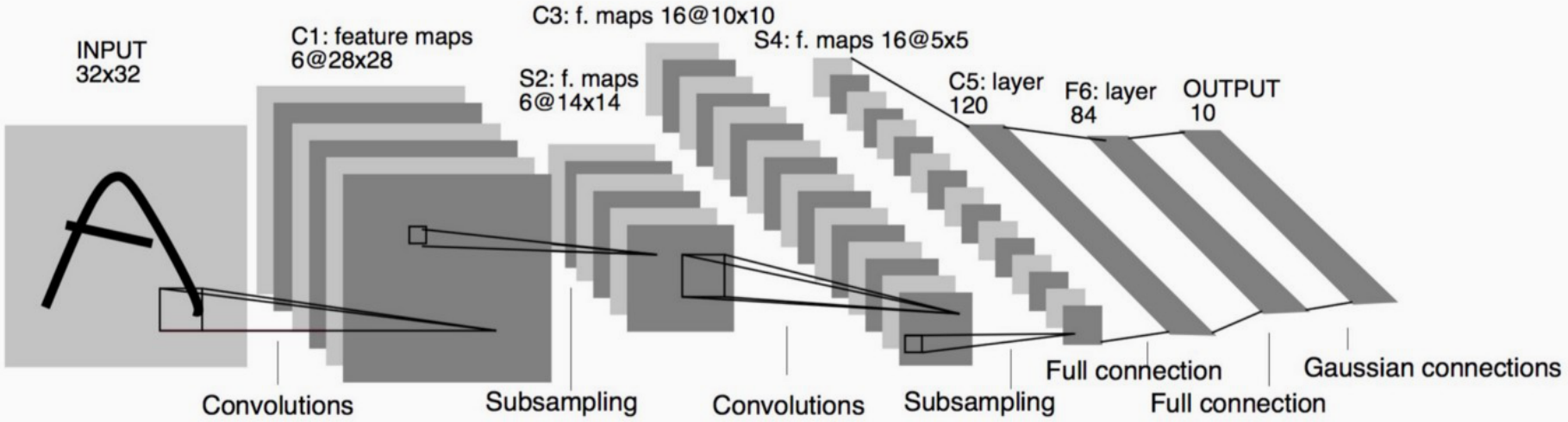
https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

A konvolúciós rétegek vizualizációja



https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

Architektúrák: LeNet

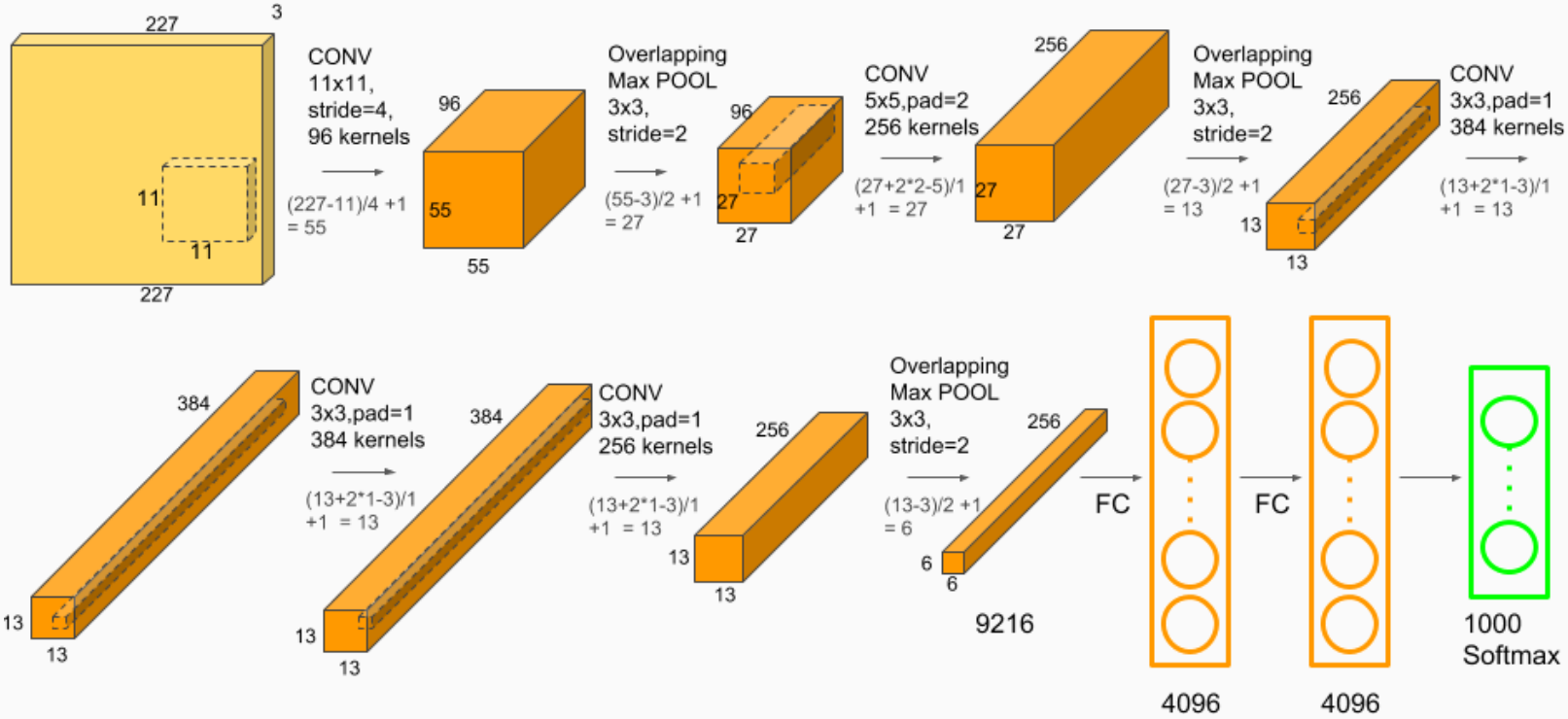


`http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf`

- Yan LeCunn, 1989: konvolúciós rétegek + backpropagation
- irányítószámok, banki csekkek feldolgozása

Architektúrák: AlexNet

- IMAGENET Large Scale Visual Recognition Challenge, `http://image-net.org/challenges/LSVRC/2012/`
- GPU + sok adat + backprop
- Alex Krizhevsky, 2012, AlexNet, ≈ 60M paraméter, 6 nap tanítás 2 drb GTX 580 3GB GPUn

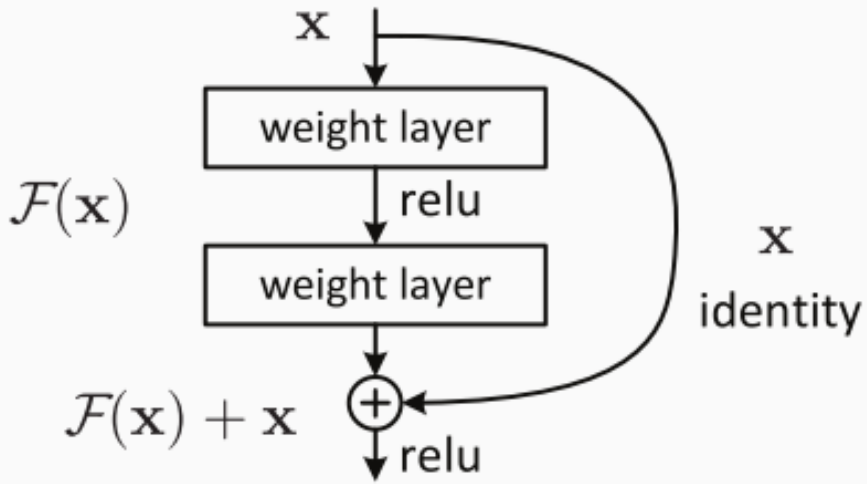


`https://www.learnopencv.com/understanding-alexnet/`

`demo: objclass / matlab`

Architektúrák: ResNet

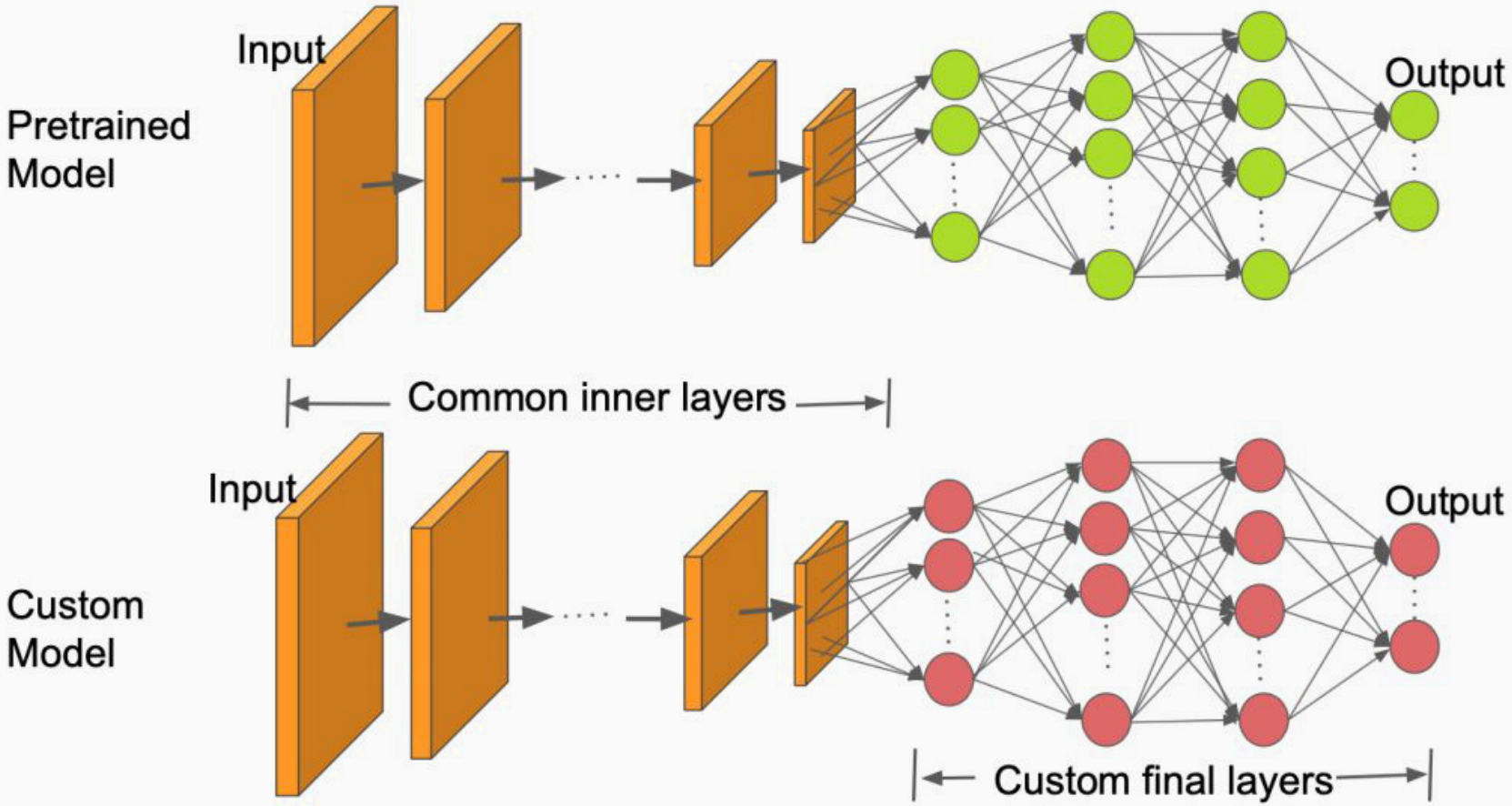
- a rétegek egyszerű felfűzése nem vezet feltétlenül jobb eredményekhez
- residual block:



https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

- ”konvexebbé” teszi a hibafüggvényt https://arxiv.org/pdf/1712.09913.pdf

Transfer learning



https://www.learnopencv.com/image-classification-using-transfer-learning-in-pytorch/

Python demo:

https://github.com/fastai/course-v3/blob/master/nbs/dl1/lesson1-pets.ipynb

Könyvtárak

- Python: Pytorch (Facebook)
- Python: Tensorflow (Google)
- Matlab: Deep Learning Toolbox (Mathworks)
- `https://en.wikipedia.org/wiki/Comparison_of_deep-learning_software`
- átjárás: Open Neural Network Exchange (ONNX) formátum