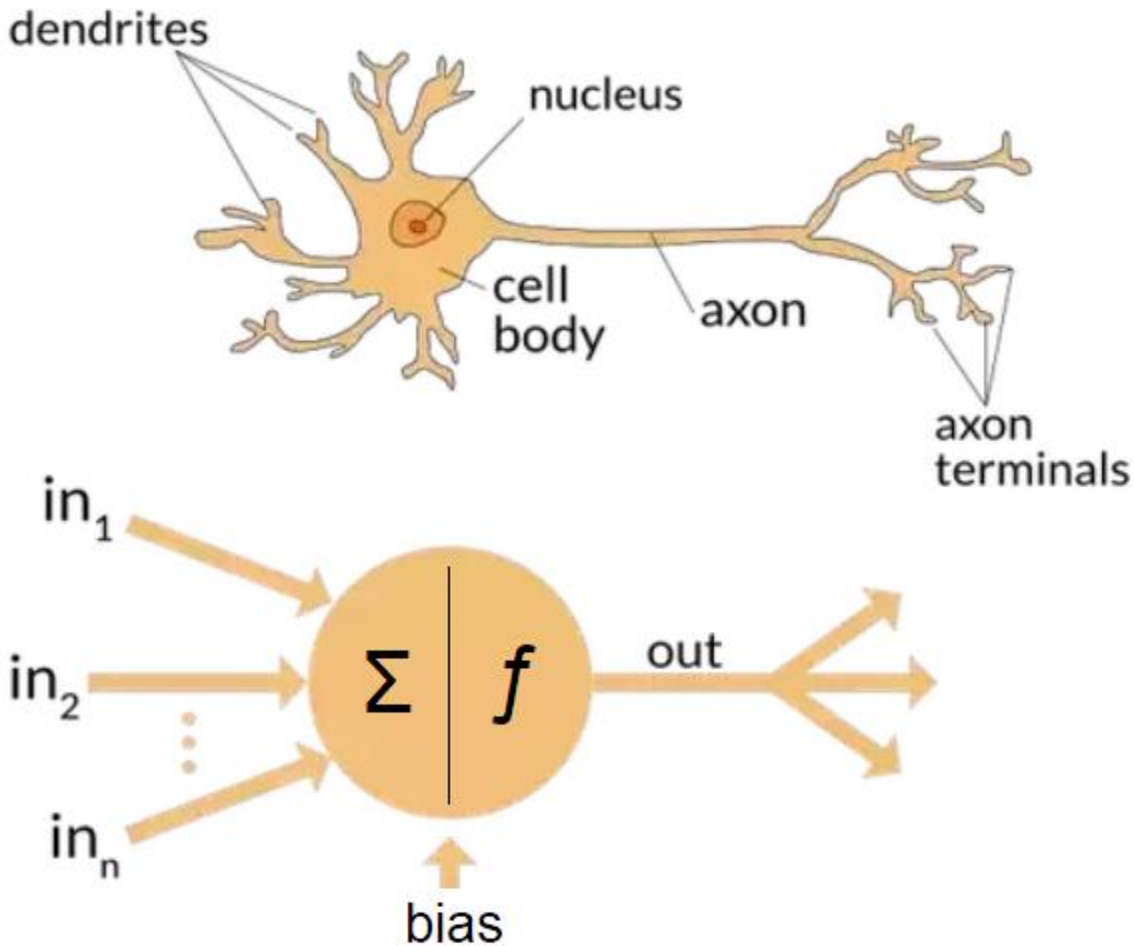
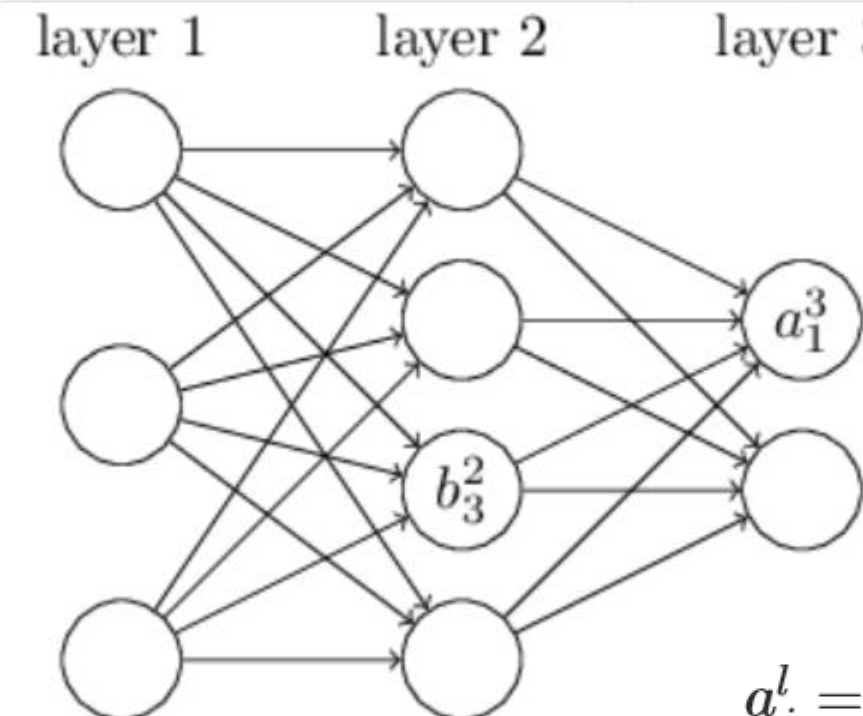
Image analysis using Deep Learning

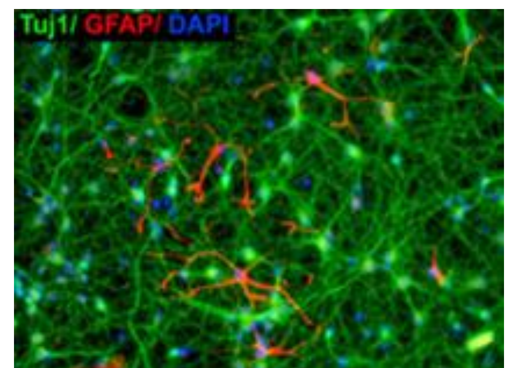
The artificial neuron



https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7



$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2$$



$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

- Define $\delta_j^l = \frac{\partial C}{\partial z_j^l}$: the rate of change of total cost with rspect to z_j^l

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial a_j^l}\frac{\partial a_j^l}{\partial z_j^l} = \left(t\arg et_j - a_j^l\right)a_j^l\left(1-a_j^l\right)$$

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

- Training the weights and biases

$$W_k = W_{k-1} - \epsilon \frac{\partial E(W)}{\partial W}$$

$$W_k = W_{k-1} - \epsilon \frac{\partial E^{p_k}(W)}{\partial W}$$

- One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE
- Batch: a subset of the dataset. After the end of the batch, the learnable parameters are adapted
- Iteration= 1 batch

1. **Input x:** Set the corresponding activation a^1 for the input layer.

2. **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.

3. **Output error δ^L:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.

5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

Output error

- Let Li the correct label and yi the actual prediction (ANN output)

- MSE

$$MSE = \frac{1}{N}\sum_i \left(y_i - L_i\right)^2$$

- Cross entropy

$$CE = \sum_i L_i \log\left(y_i - L_i\right)$$

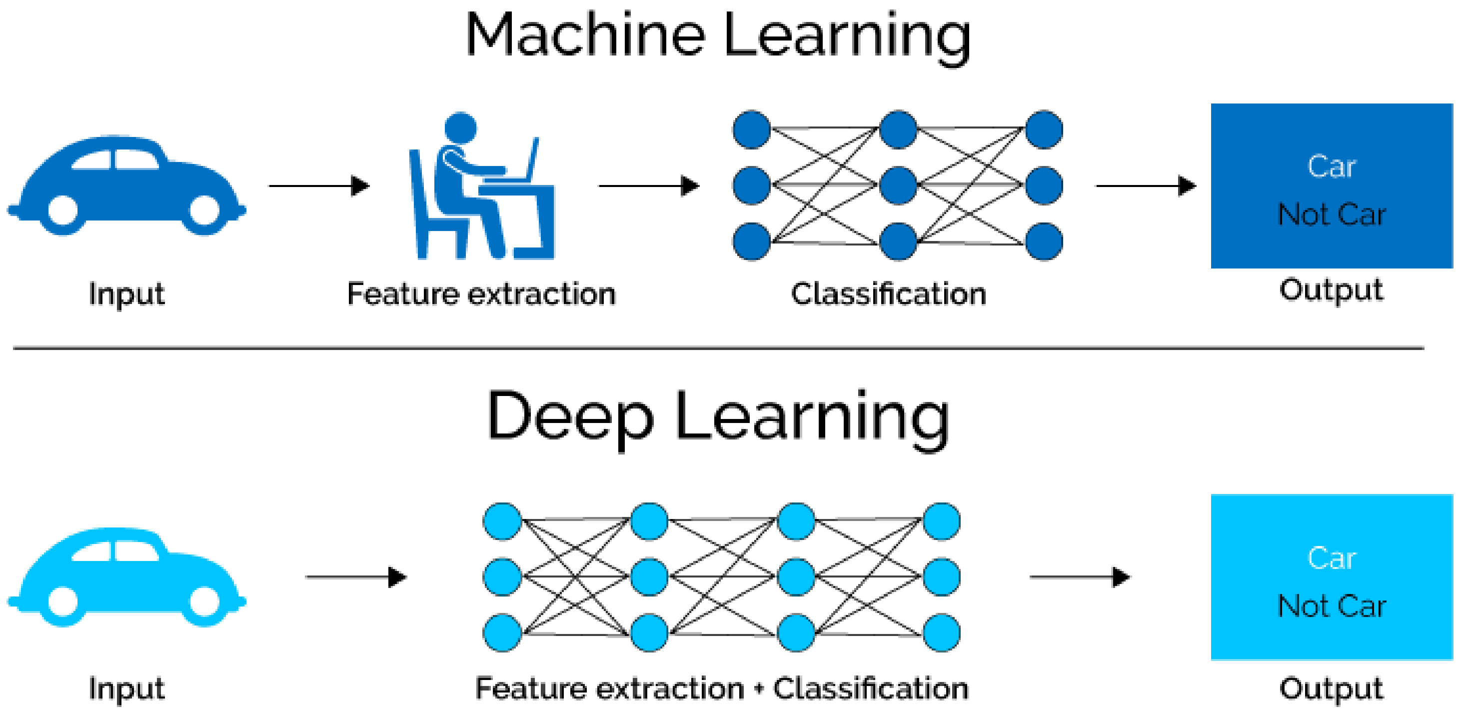
- Train error
- Validation error
- Test error
- When to terminate training
  - overfitting

What is Deep Learning (DL) ?

A machine learning subfield of learning **representations** of data. Exceptional effective at **learning patterns**.

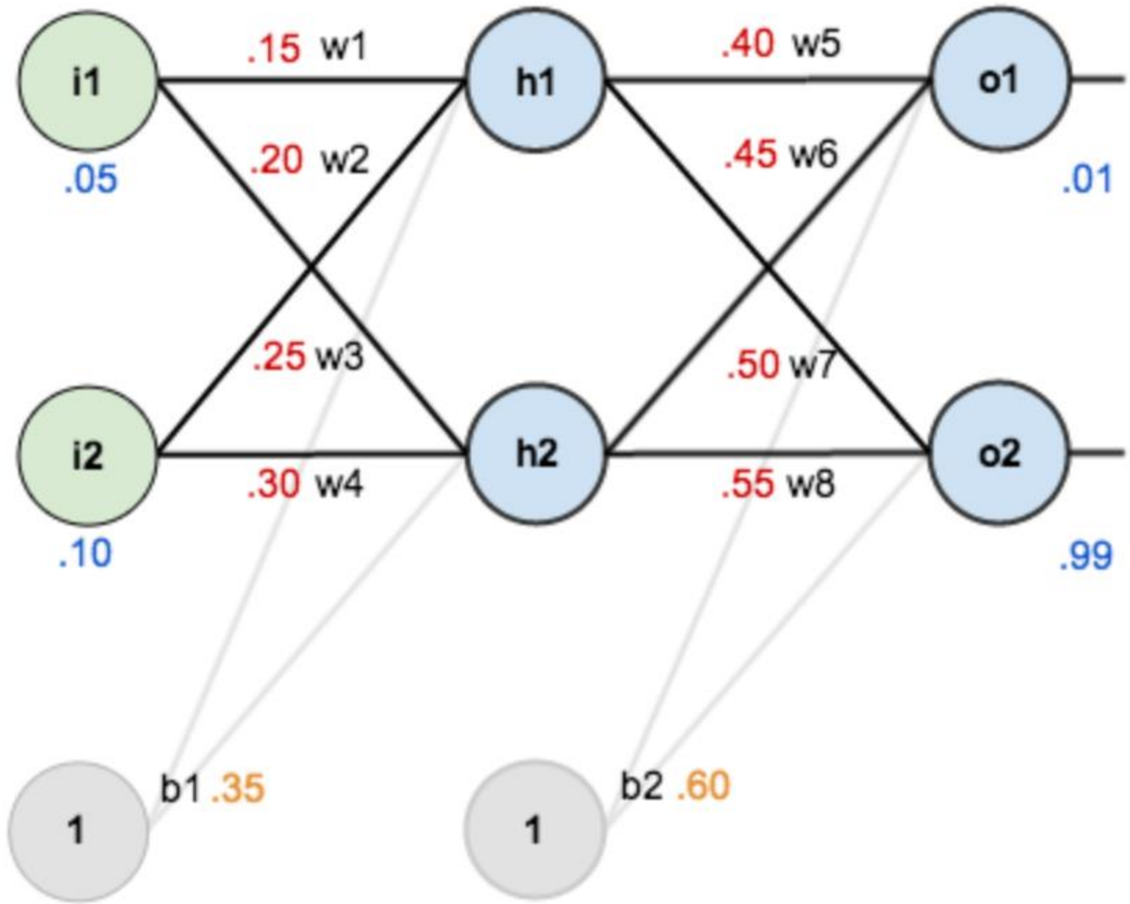
Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**

If you provide the system **tons of information**, it begins to understand it and respond in useful ways.



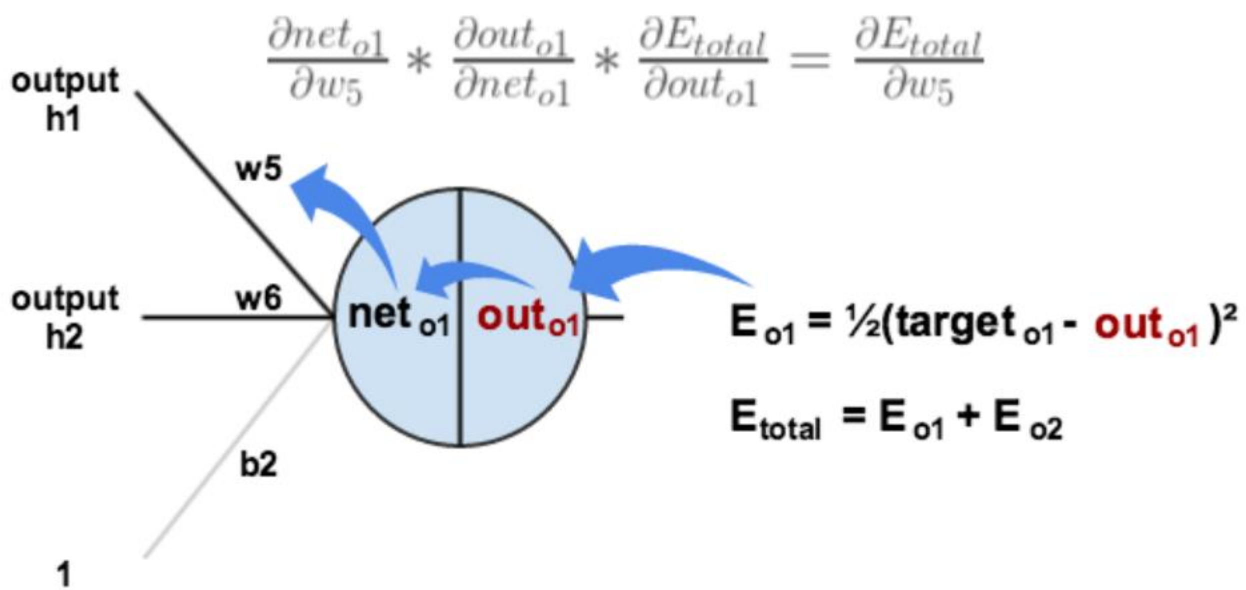
https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

Simple arithmetic example



$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

Visually, here's what we're doing:

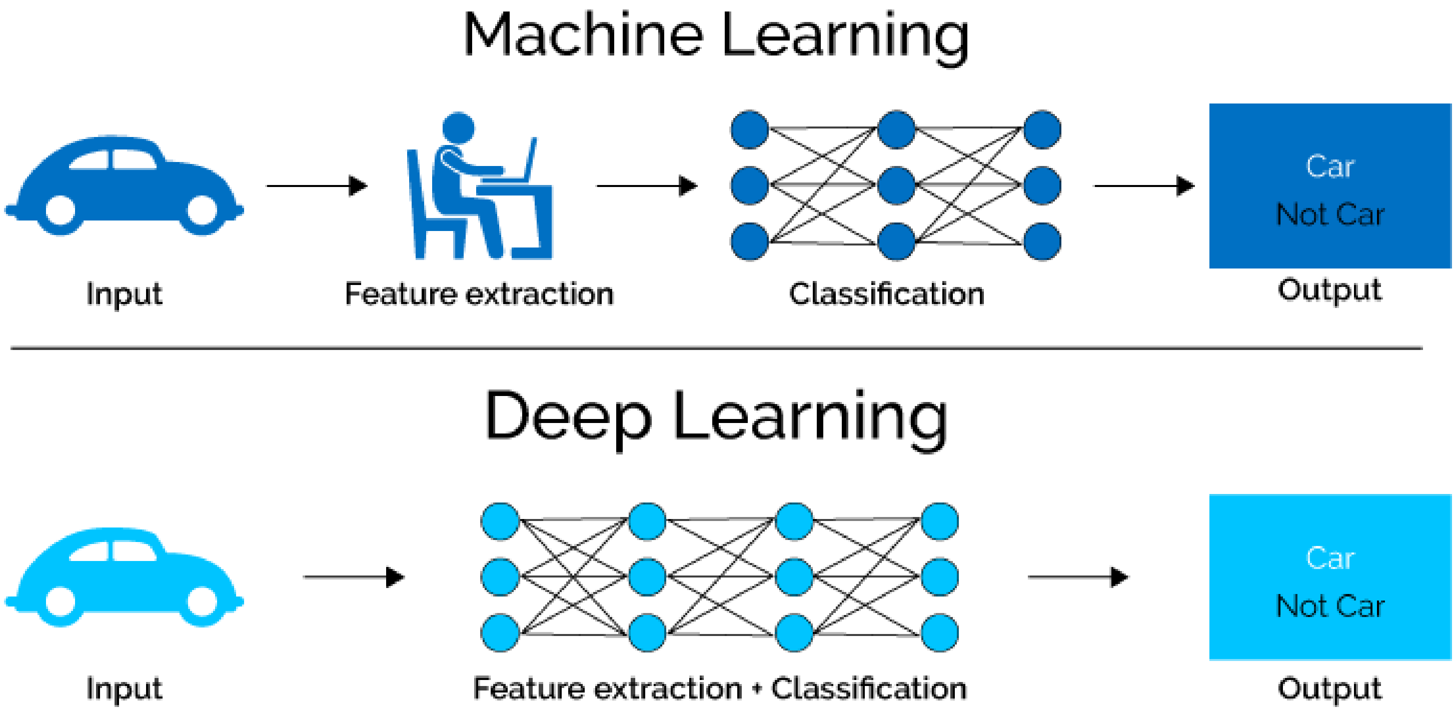


What is Deep Learning (DL) ?

A machine learning subfield of learning **representations** of data. Exceptional effective at **learning patterns**.

Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**

If you provide the system **tons of information**, it begins to understand it and respond in useful ways.



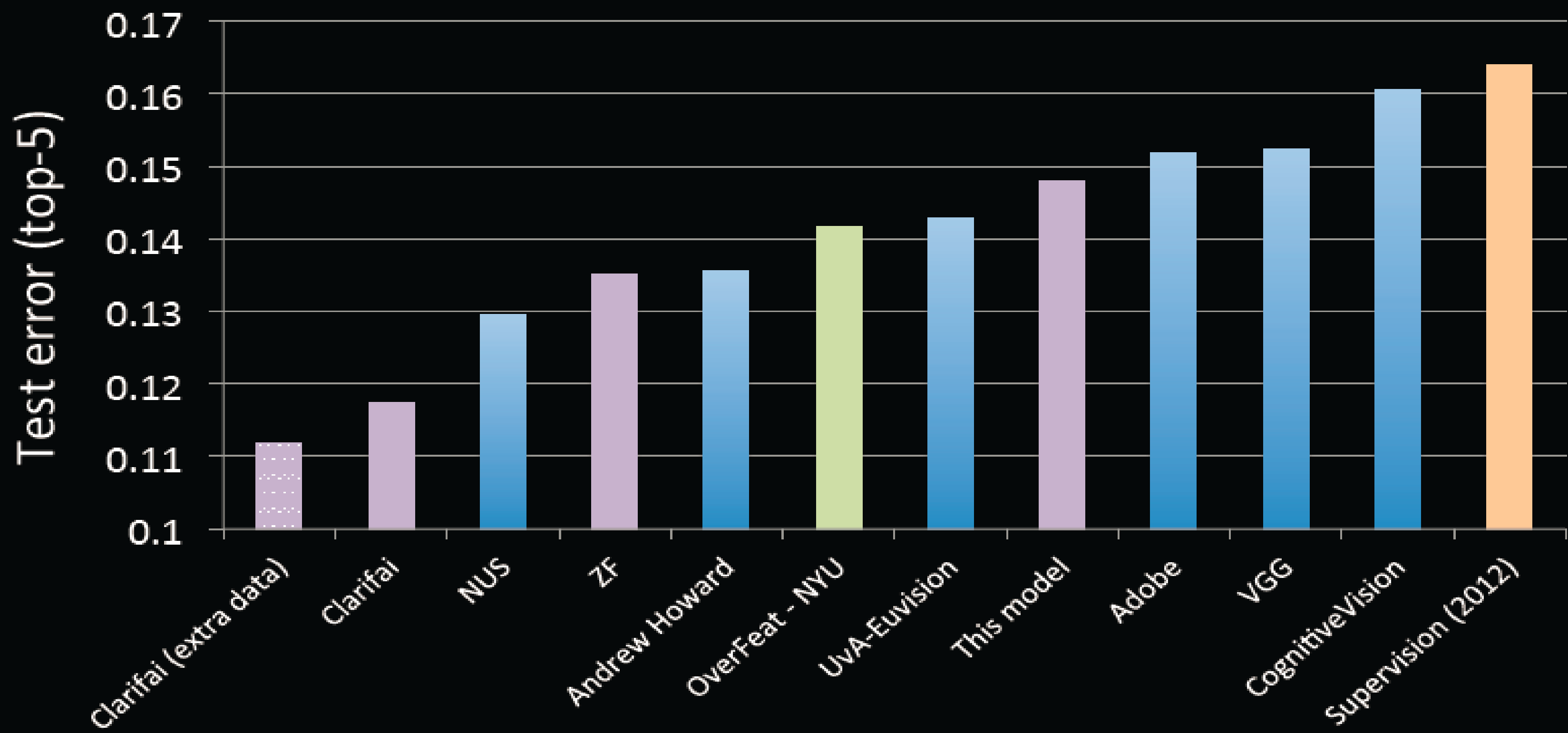
https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

- Alexnet 2012 [Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, ImageNet Classification with Deep Convolutional Networks]
  - 15 million annotated images from a total of over 22,000 categories
- ZF net 2013, Matthew Zeiler and Rob Fergus from NYU, error: 11.2 %
  - Very similar architecture to AlexNet, except for a few minor modifications.
  - ZF Net trained on only 1.3 million images.
  - used filters of size 7x7 and a decreased stride value.
  - Used ReLUs for their activation functions, cross-entropy loss for the error function, and trained using batch stochastic gradient descent.
  - Trained on a GTX 580 GPU for **twelve days**.
  - Developed a visualization technique **Deconvolutional Network**, to examine different feature activations and their relation to the input space

- VGG net 2014 (Karen Simonyan and Andrew Zisserman of the University of Oxford),[*https://arxiv.org/pdf/1409.1556v6.pdf*]
  - Simplicity and depth. 7.3% error rate.
  - 19 layer CNN that strictly used 3x3 filters with stride and pad of 1, along with 2x2 maxpooling layers with stride 2

Imagenet Classifications 2013

- http://www.image-net.org/challenges/LSVRC/2013/results.php



- Pre-2012: 26.2% error → 2012: 16.5% error → 2013: 11.2% error

Conv Net Topology

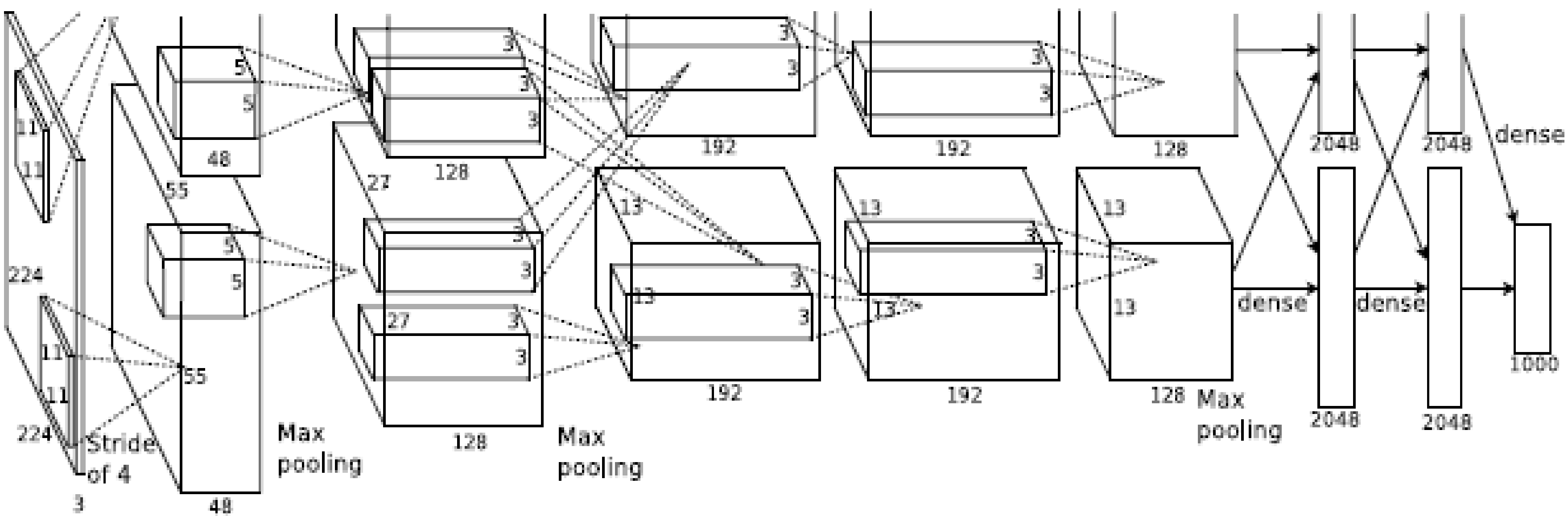
- 5 convolutional layers
- 3 fully connected layers + soft-max
- 650K neurons , 60 Mln weights

ImageNet Classification with Deep Convolutional Neural Networks

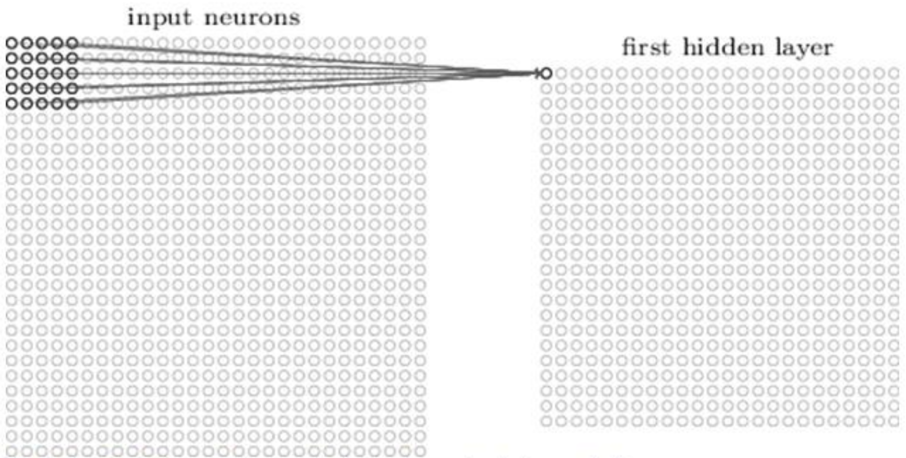
Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca



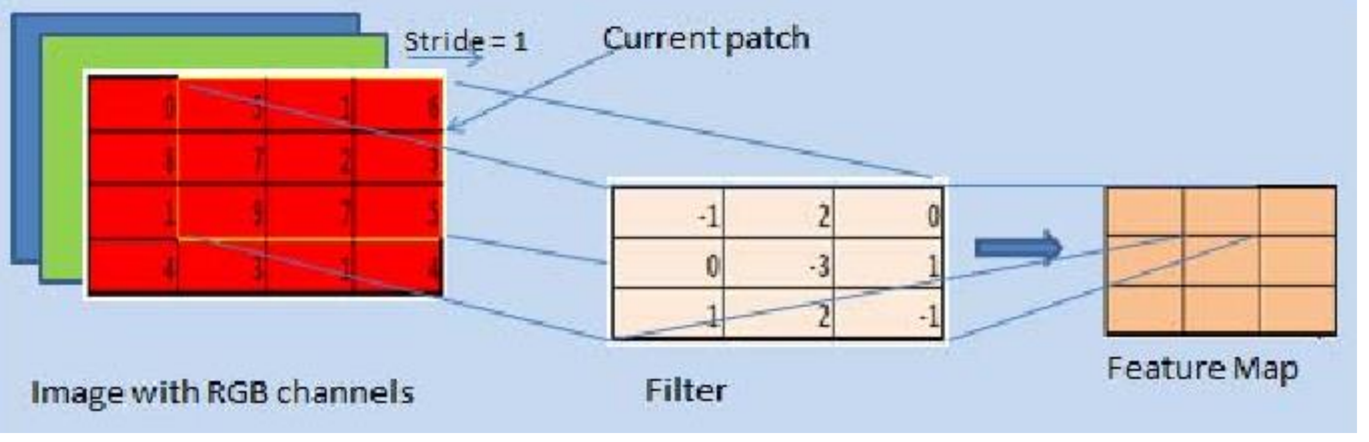
Convolutional layers

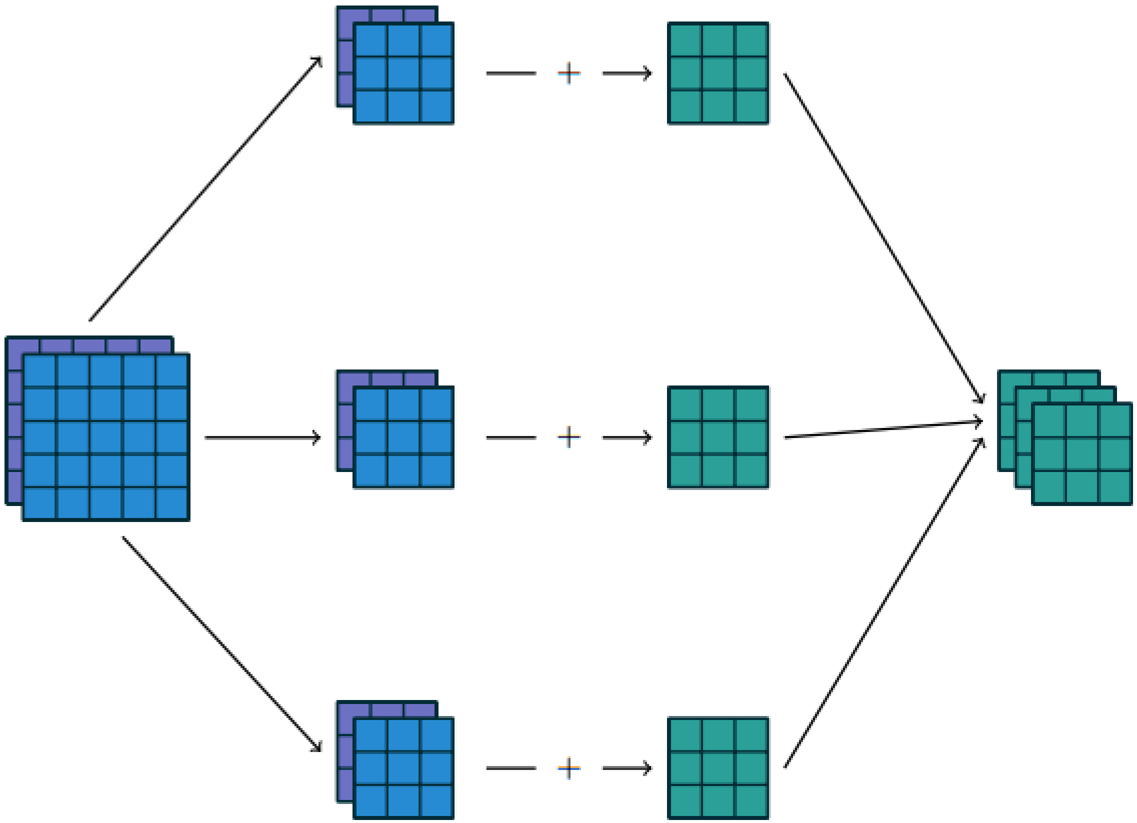


Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

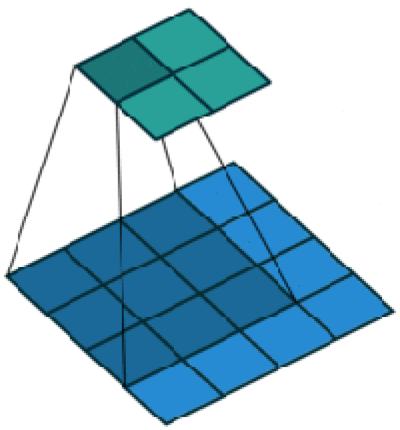
- Terms: **filter**, or **kernel**, or **neuron**, or **unit**
- If input NxNx3 (RGB)→kernel: KxKx3
- The contents of the filter are **weights**, or **parameters**
- **Receptive field**: the size of the kernel
- **activation maps**, (also called **feature maps**): the result of the convolution

- In the case of RGB images: each neuron has 3 kernels: one for each channel
- The outputs of each channel are summed to generate 1 activation (feature) map

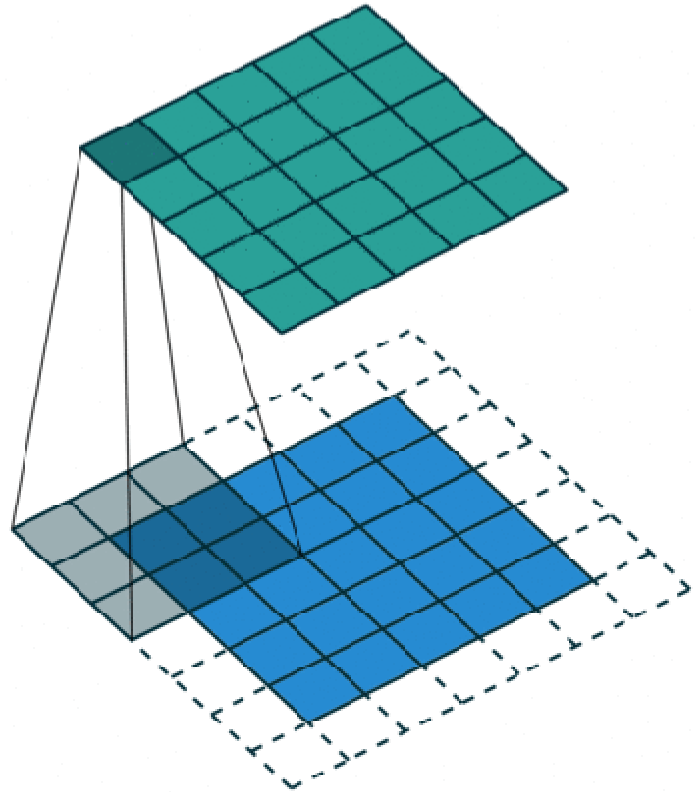




- Definition of Zero-padding, stride
- Zero_padding=(K-1)/2
- Out=(in – K +2P)/s +1



Convolution with no zero-padding

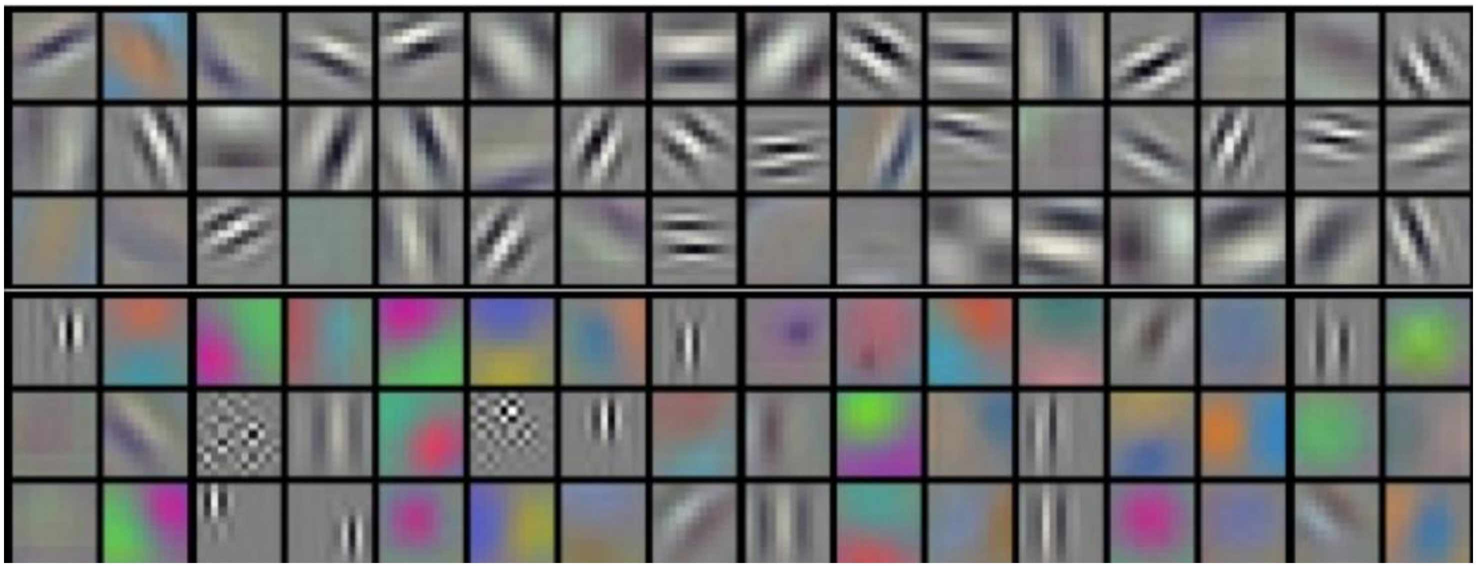


Convolution with zero-padding

- *real-world example*. [Krizhevsky et al.] architecture that won the ImageNet challenge in 2012
- accepted images of size [227x227x3].
- On the first Convolutional Layer,
  - receptive field size *F*=11x11, stride *S*=4x4 and no zero padding *P*=0.
  - Since (227 - 11)/4 + 1 = 55, and since the Conv layer had a depth of *K*=96, the Conv layer output volume had size [55x55x96].
  - Each of the 55*55*96 neurons in this volume was connected to a region of size [11x11x3] in the input volume. Moreover, all 96 neurons in each depth column are connected to the same [11x11x3] region of the input, but of course with different weights

Summary of the convolutional layer

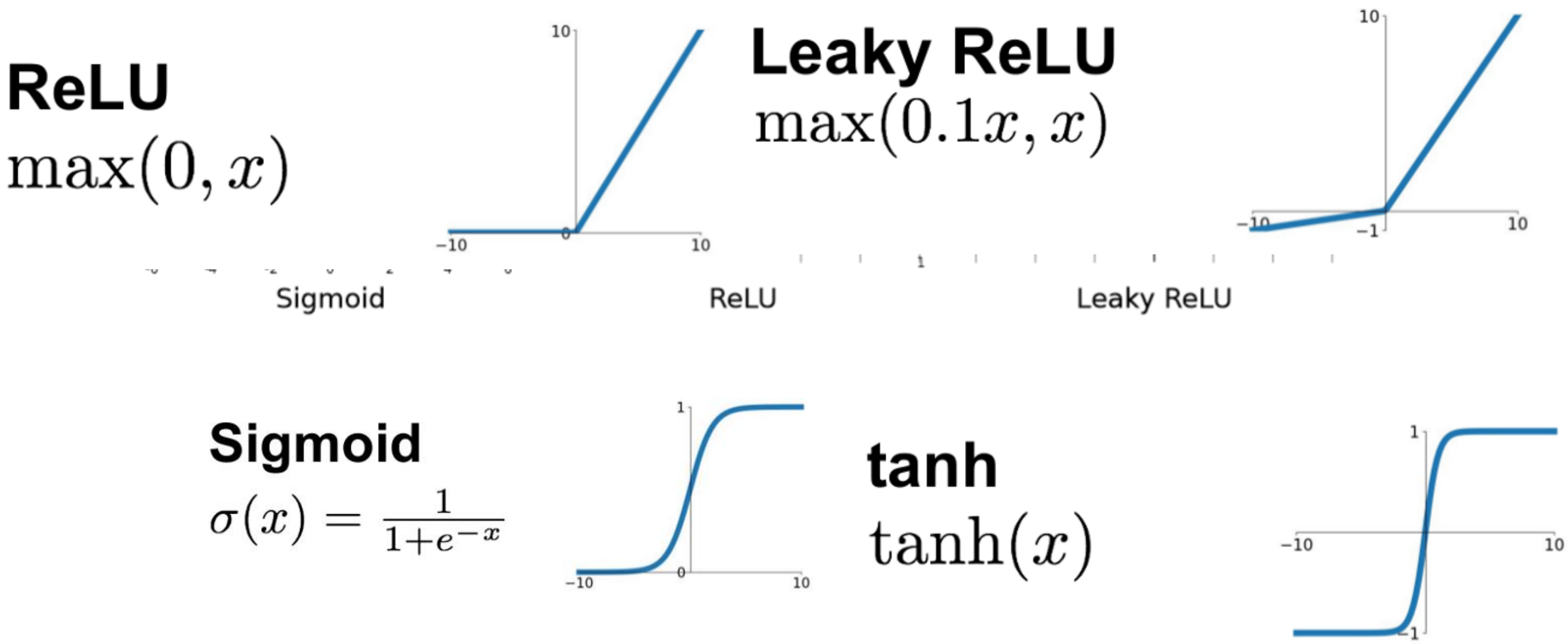
- Accepts a volume of size $W1 \times H1 \times D1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2=(W1-F+2P)/S+1$
  - $H2=(H1-F+2P)/S+1$ (i.e. width and height are computed equally by symmetry)
  - $D2=K$



- Example filters learned by Krizhevsky et al. Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the 55*55 neurons in one depth slice

Activation functions

- The output of the convolutional layer is passed through an non-linear activation function



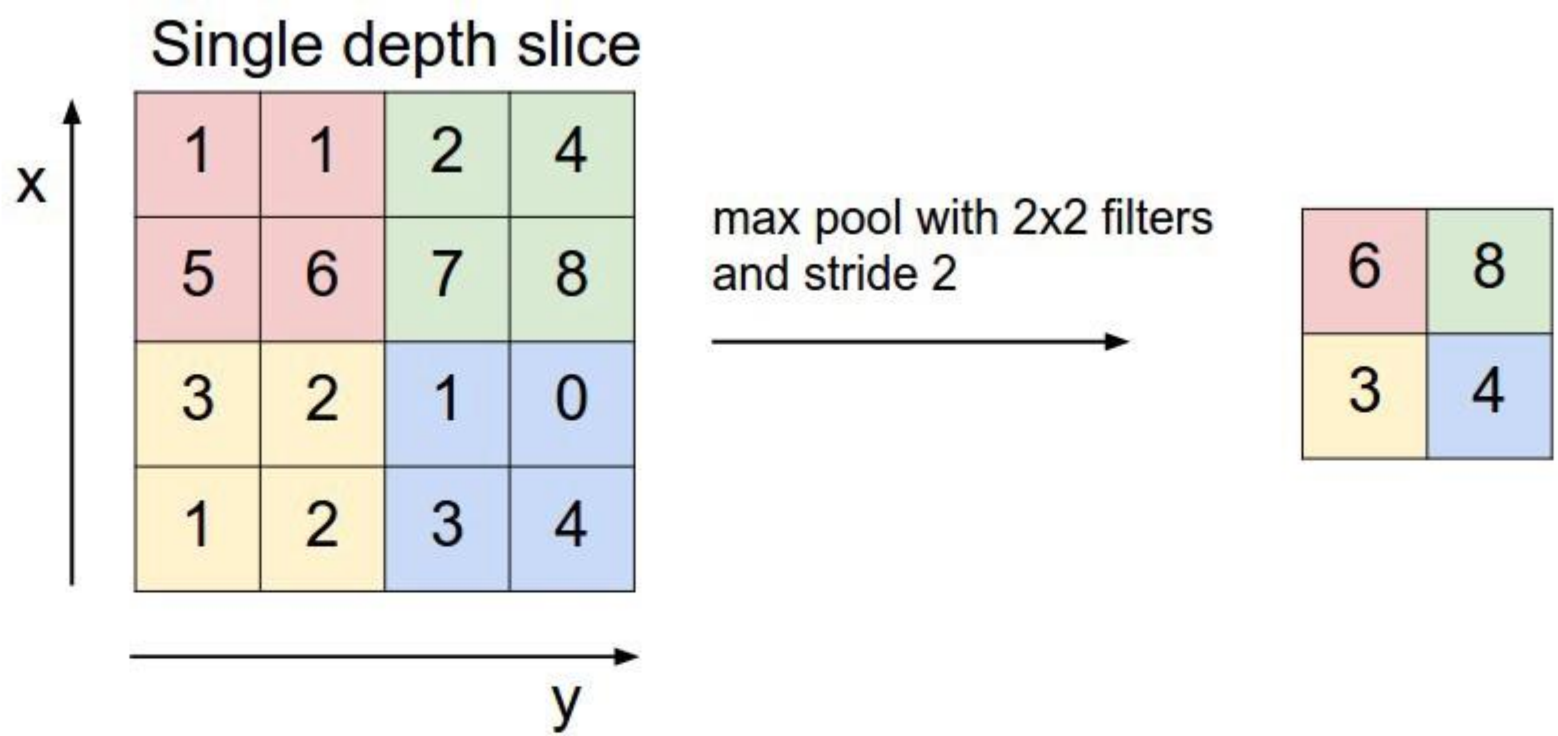
ReLU Layer

Filter 1 Feature Map

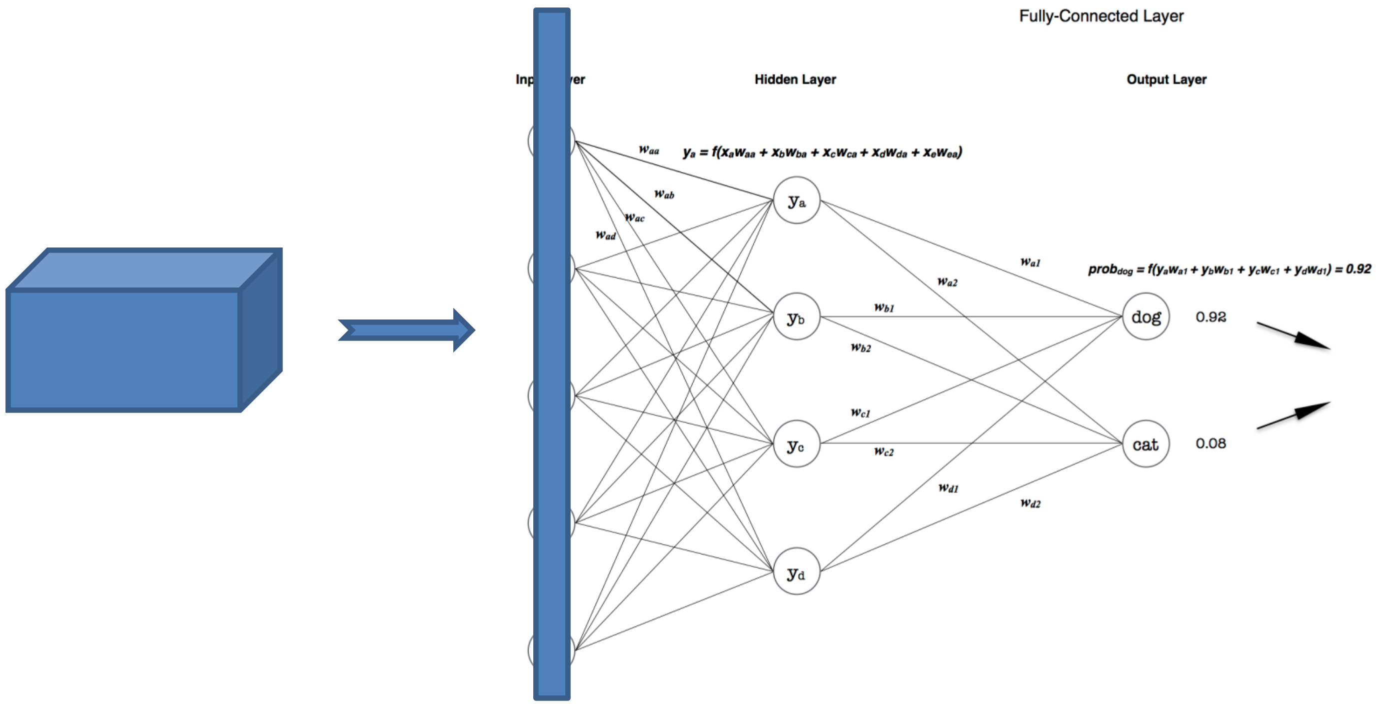
9	3	5	-8
-6	2	-3	1
1	3	4	1
3	-4	5	1

9	3	5	0
0	2	0	1
1	3	4	1
3	0	5	1

Pooling layer



Fully connected layers

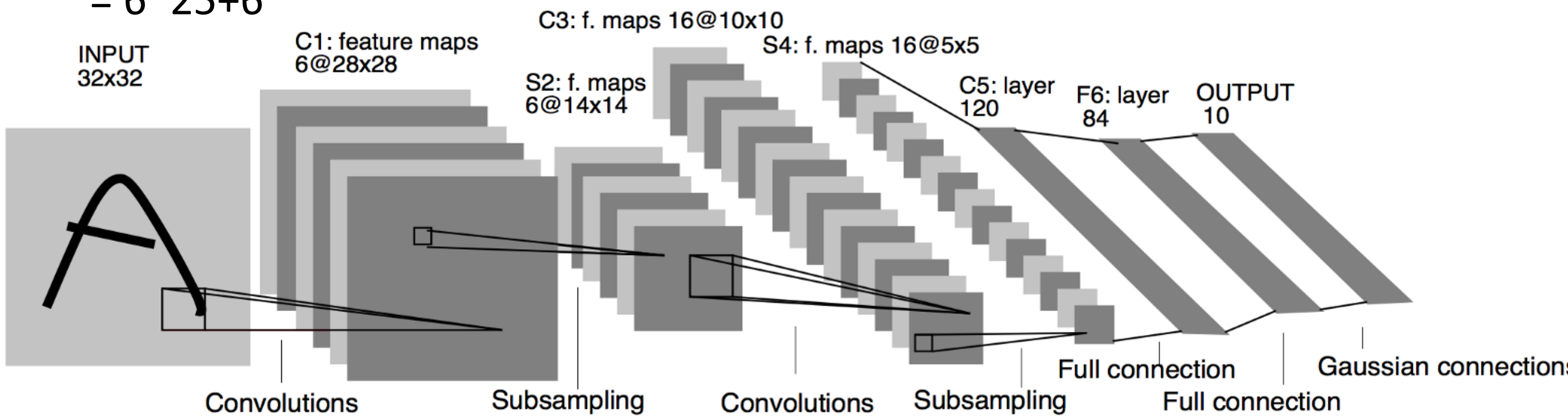


- Softmax layer: logit to probability distribution
- Transfer Learning
- Data augmentation

LeNet,19

C3: 16 kernels, 5x5, convolution
Pad:0, Stride: 1
Trainable params: 1516=
6x (5x5x3 +1)
+ 9x (5x5x4 +1)
+ 1x (5x5x6 +1)
Not all kernels see all input channels

C1: 6 kernels 5x5,
Pad:0
Stride:1
Trainable params: 156
= 6*25+6

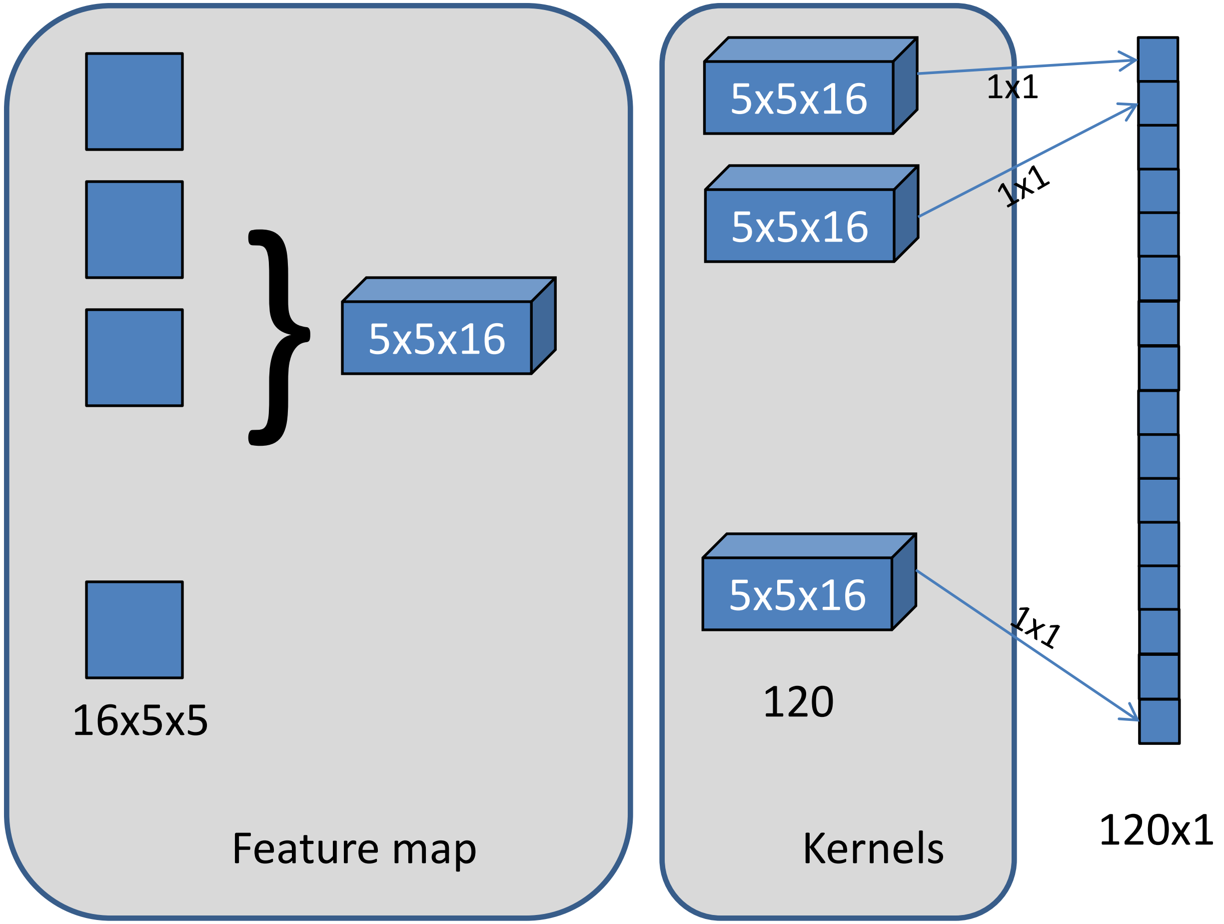


S2: 6 kernels, 2x2, not convolution
Sum pooling + bias
Pad:0, Stride: 2
Trainable params: 12
=6 multipl + 6 bias

S4: 16 kernels, similar to **S2**
Trainable params: 12
=6 multipl + 6 bias

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998).
Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324
>32000 citations

Lenet C5 layer



- Trainable parameters: 120x(5x5x16 +1bias)=42120

LeNet Input and Output layer

- Grayscale 28x28 cropped out of 32x32.
- Normalization: 0-mean, 1-std

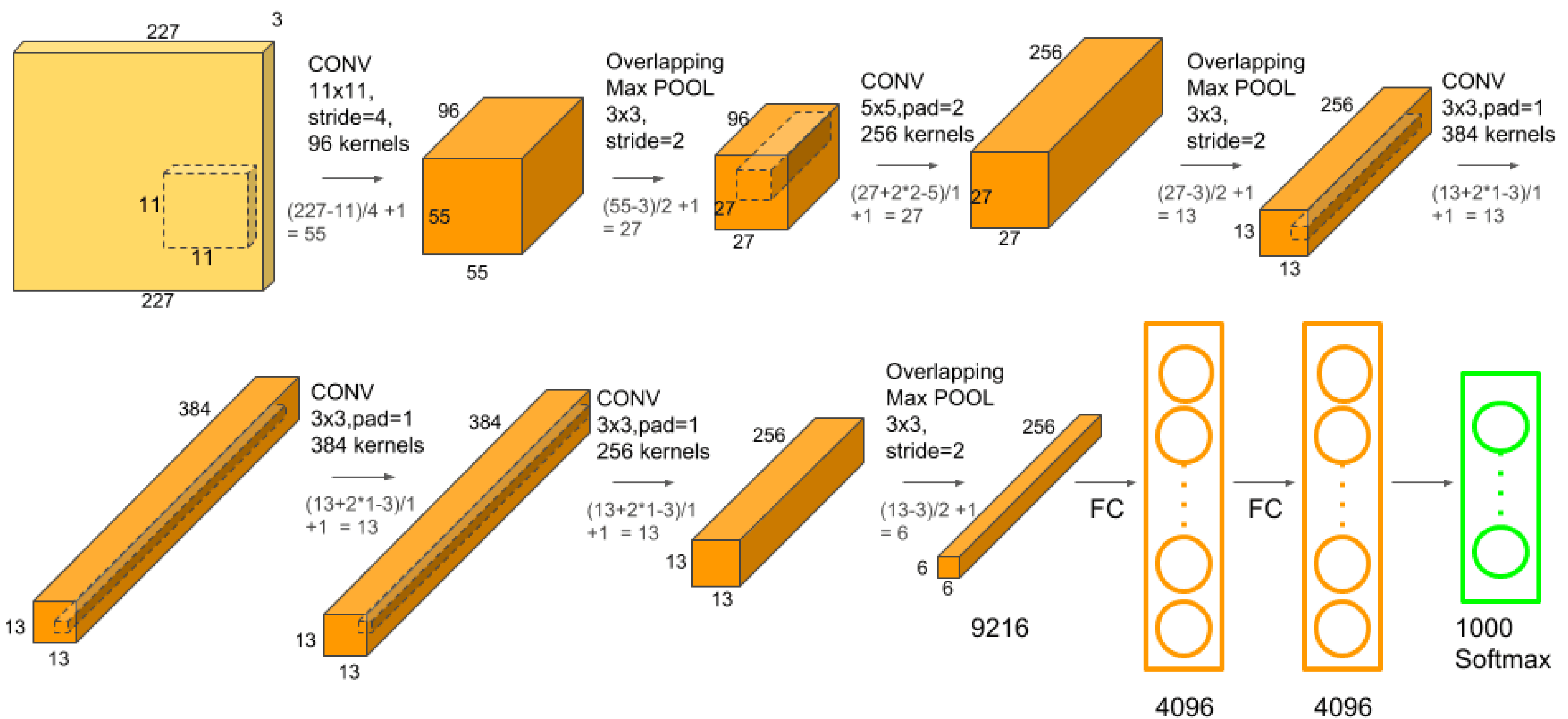
$$I_i = \frac{I_i - \mu_i}{\sigma_i}$$

- Η έξοδος του νευρώνα j (w_{ij} η σύνδεση με τον i του προηγούμενου Layer, x_j η είσοδος στον i από τον j)

$$y_i = \sum_j \left(x_j - w_{ij} \right)^2$$

Alexnet 2012

- Overall architecture



https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90.
Cited >75000

- The kernels of the 2nd , 4th , and 5th convolutional layers are connected only to the previous layer kernel maps in the same GPU.
- The kernels of the 3rd are connected to all kernel maps 2nd conv. layer.
- The neurons in the FC layers are connected to all neurons in the previous layer
- Learning rule

$$
\begin{aligned}
v_{i+1} &:= 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i} \\
w_{i+1} &:= w_i + v_{i+1}
\end{aligned}
$$

- ReLU activations after every convolutional and fully-connected layer instead of Tanh. It accelerates the speed by 6 times at the same accuracy
- Use dropout instead of regularisation to deal with overfitting.
  - the training time is doubled with the dropout rate of 0.5
- 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs
- ImageNet is a dataset of 15 million labeled high-resolution images of 22,000 categories (human labelers)
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) uses a subset of ImageNet:
  - 1000 images in each of 1000 categories. In all,
  - 1.2 million training images,
  - 50,000 validation images, and
  - 150,000 testing images

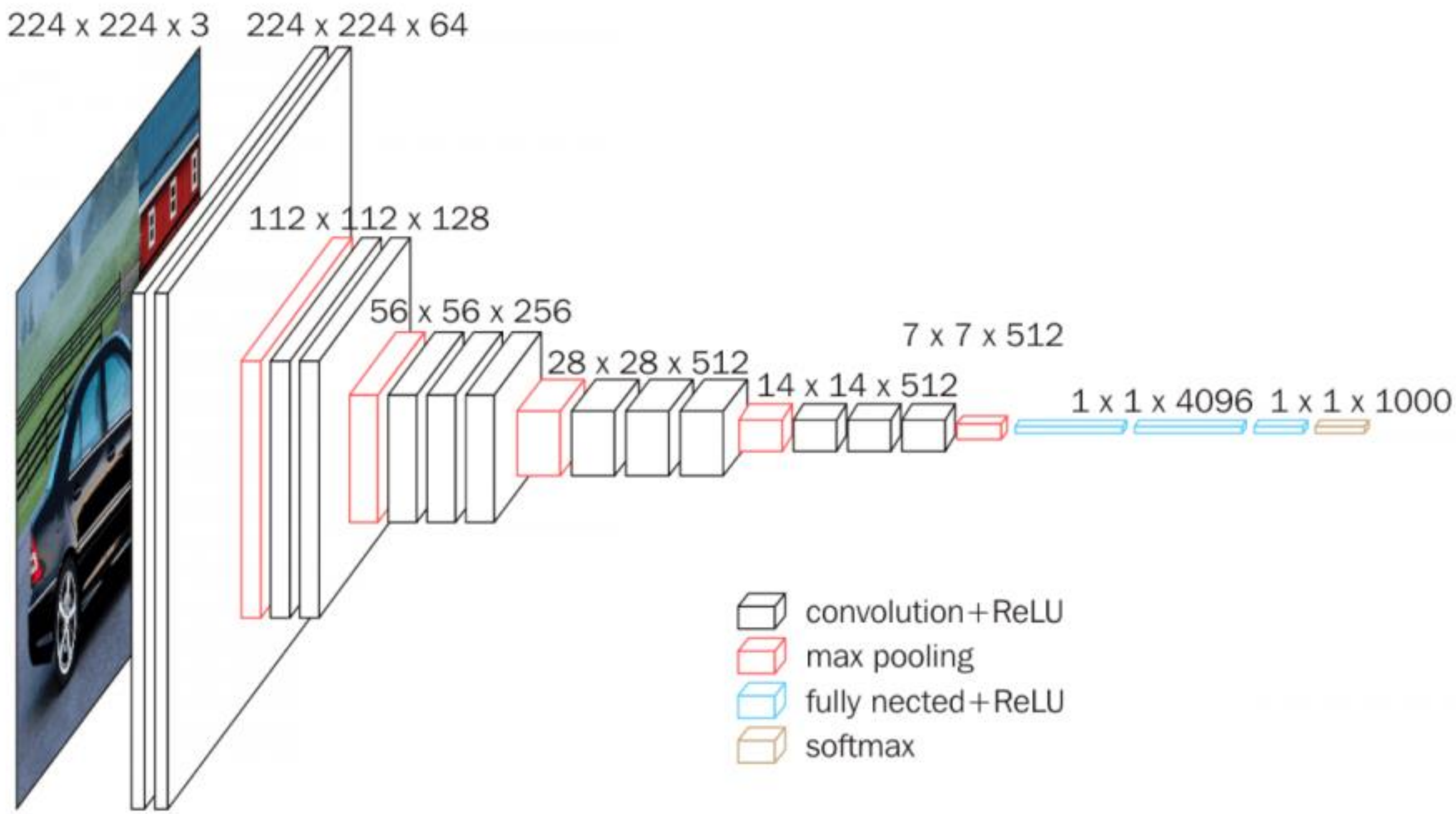
AlexNet output layer

- Softmax is implemented just before the output layer.
- The Softmax layer must have the same number of nodes as the output layer

$$\sigma\left(z_i\right) = e^{bz_i} \; / \sum_j e^{bz_i}$$

LeNet Output layer

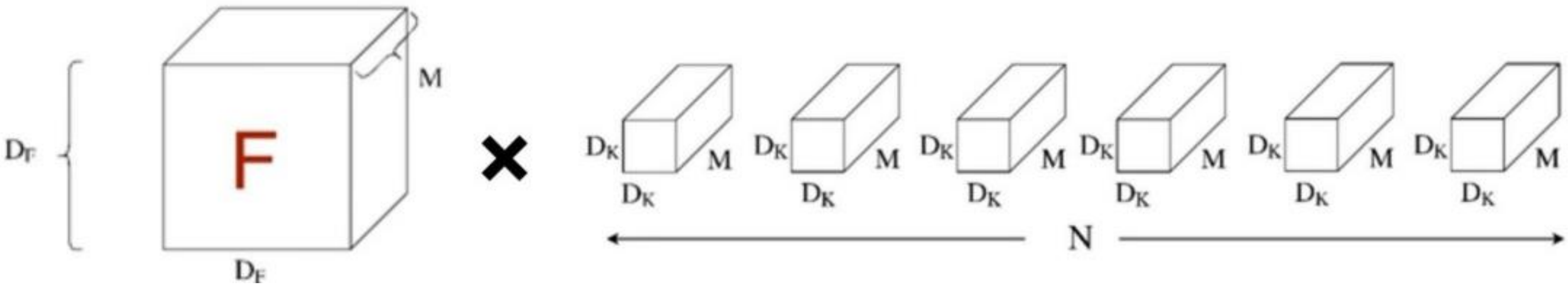
To CNN VGG16


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

VGG16 - Structural Details													
#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
Total													**138,423,208**

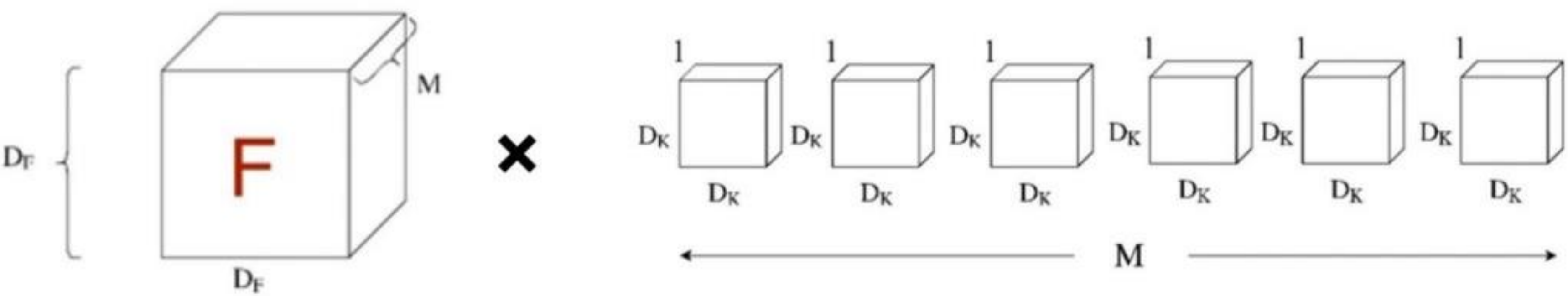
https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

Mobile net



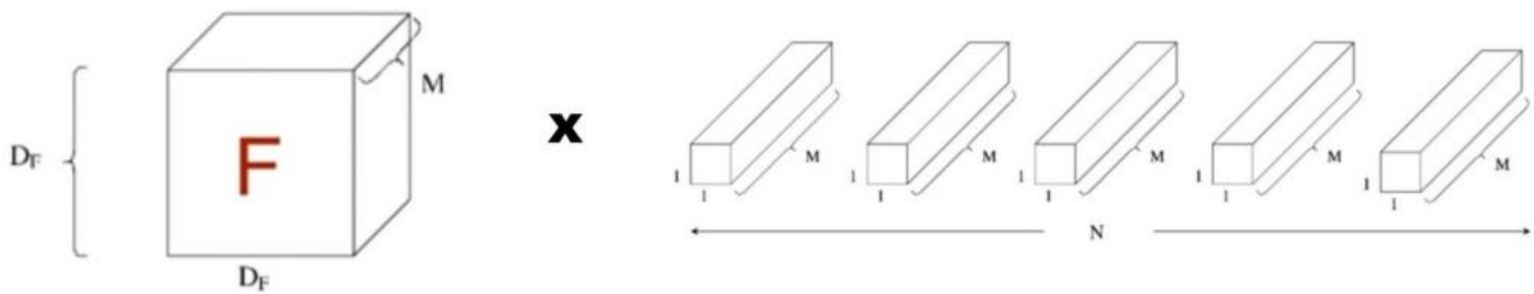
Normal convolution layer

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$



Depth-wise convolution

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F$$



Point-wise convolution

$$M \cdot N \cdot D_F \cdot D_F$$

https://medium.com/@godeep48/an-overview-on-mobilenet-an-efficient-mobile-vision-cnn-f301141db94d

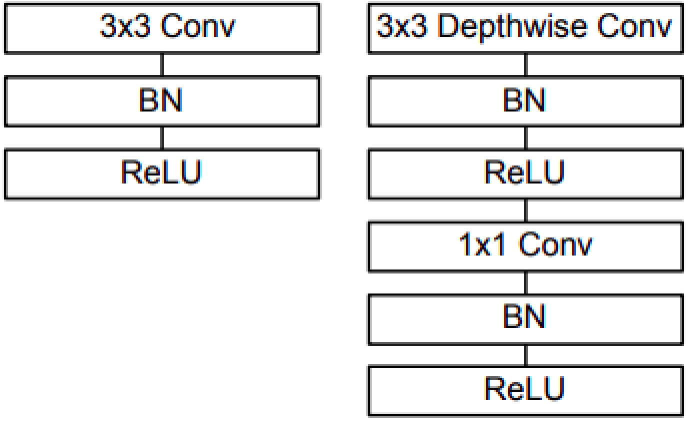


Table 8. MobileNet Comparison to Popular Models

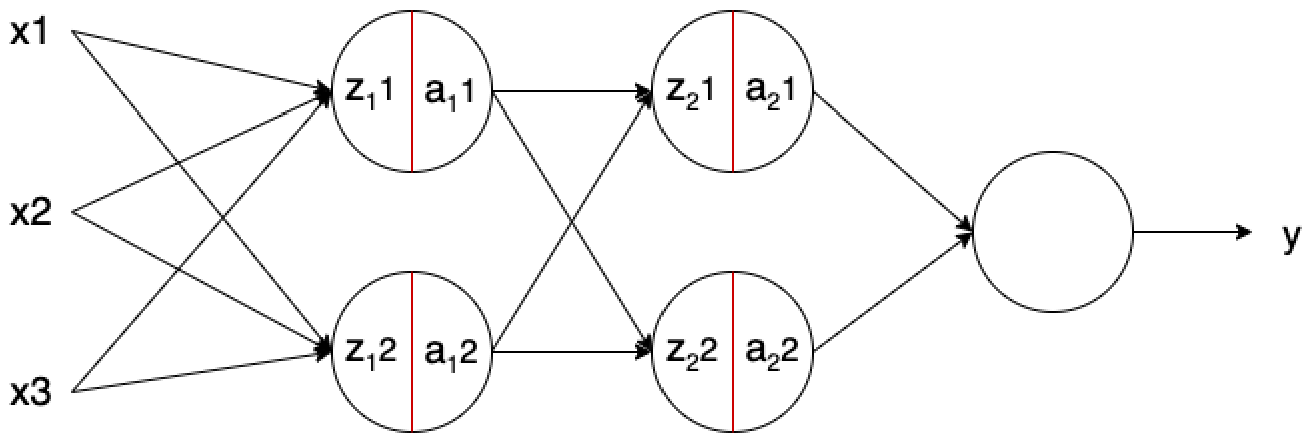
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Batch normalization

- Fully-connected (FC) layers: the input to the FC by x, the Wx+b (weight W and bias b), and the activation function by φ, we can express the computation of a batch-normalization-enabled, fully-connected layer output h as follows



$$z = g(w, x) + b; \qquad a = f(z)$$

$$z = g(w, x); \qquad z^N = \left(\frac{z - m_z}{s_z}\right) \cdot \gamma + \beta; \qquad a = f(z^N)$$

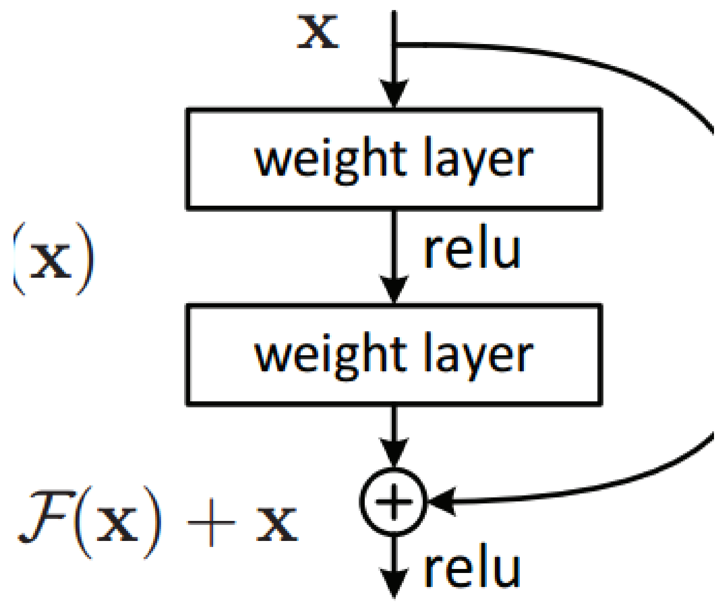
Batch normalization

- convolutional layers, apply BN after the convolution and before the nonlinear activation function.
- BN for *each* of the outputs of these channels, and each channel has its own scale and shift parameters, both scalars

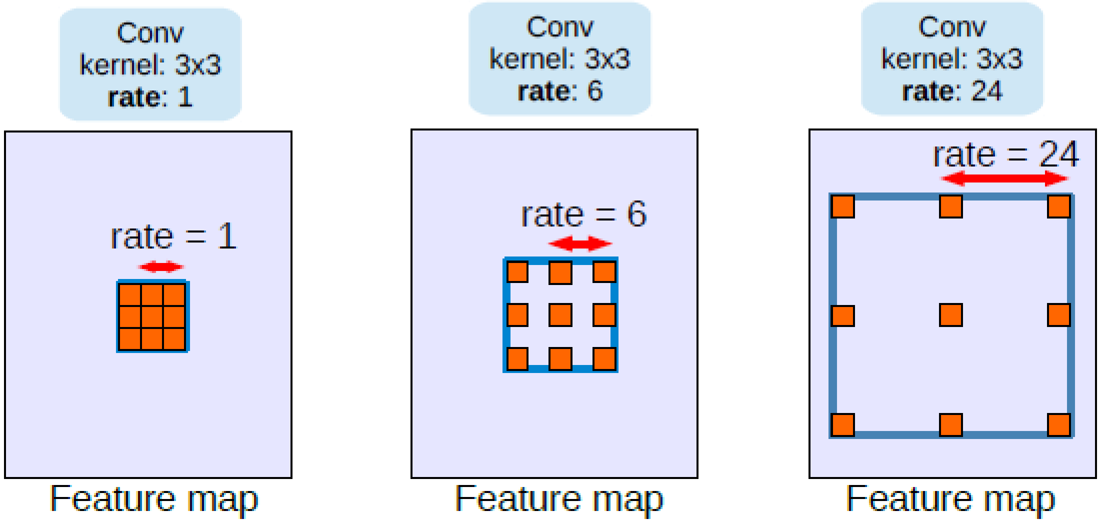
Residual networks

- Experiments have shown that the accuracy decreases by adding more layers to the network
- the vanishing gradient problem. As we make the CNN deeper, the derivative when back-propagating to the initial layers becomes almost insignificant in value
- ResNet addresses this network by introducing two types of ‘shortcut connections’: *Identity shortcut* and *Projection shortcut*

- **shortcut connection**: Instead of learning the mapping from x →F(x), the network learns the mapping from x → F(x)+G(x).
- *Identity connection*: the dimension of the input x and output F(x) is the same.
- The identical mapping is learned by zeroing out the weights in the intermediate layer during training.
- *Projection connection*: the dimensions of F(x) differ from x (due to stride length>1 in the CONV layers in between).
  - The function G(x) changes the dimensions of input x to that of output F(x). Two kinds of mapping were considered in the original paper

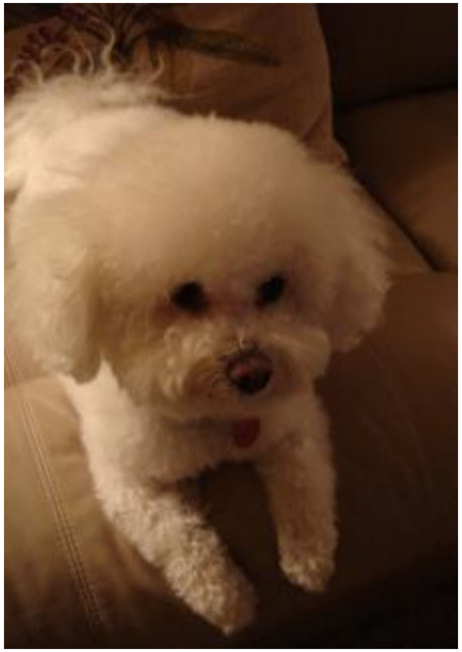


Atrous or Dilated Convolution



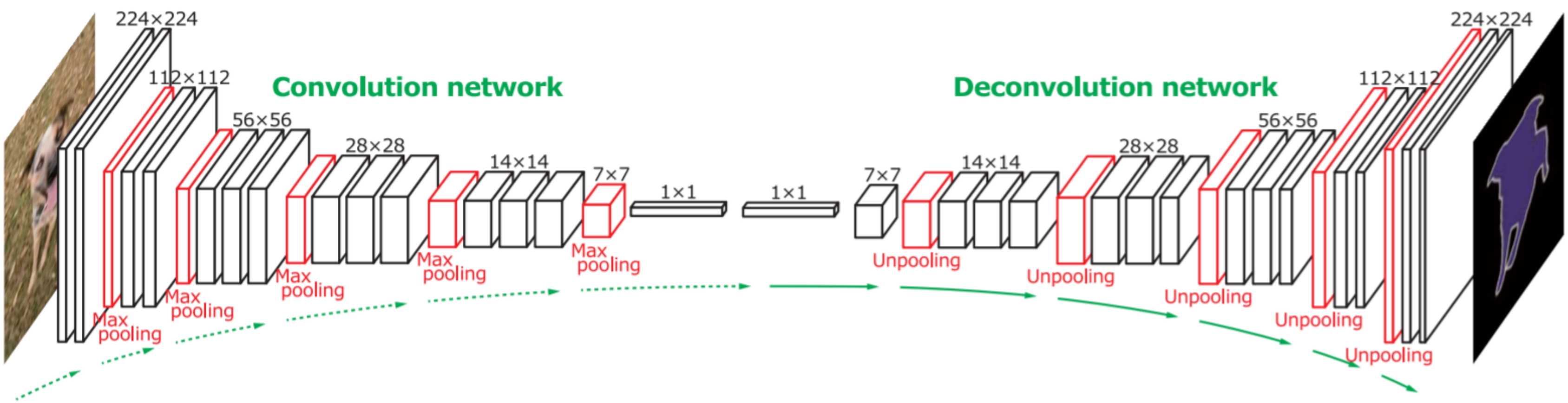
Pixel-based image segmentation

- An architecture that outputs an image of size equal to the input

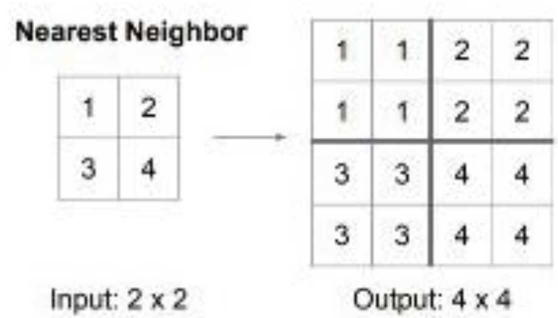


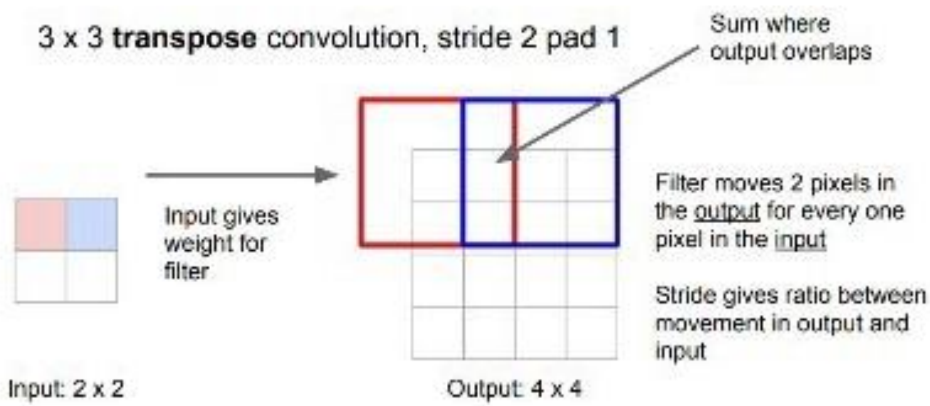
New operators

- Unpooling
- Deconvolution



Unpooling


Nearest Neighbor
1 2
3 4
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4
Input: 2 x 2
Output: 4 x 4


3 x 3 **transpose** convolution, stride 2 pad 1
Sum where output overlaps
Input gives weight for filter
Filter moves 2 pixels in the output for every one pixel in the input
Stride gives ratio between movement in output and input
Input: 2 x 2
Output: 4 x 4

Transfer Learning

- **Transfer learning** is the process of taking a pre-trained model on a large dataset and "fine-tuning" the model with your own dataset
- Freeze the kernels and fine tune the lower layers of the network.

Yosinski J, Clune J, Bengio Y, and Lipson H. How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems 27 (NIPS '14), NIPS Foundation, 2014.