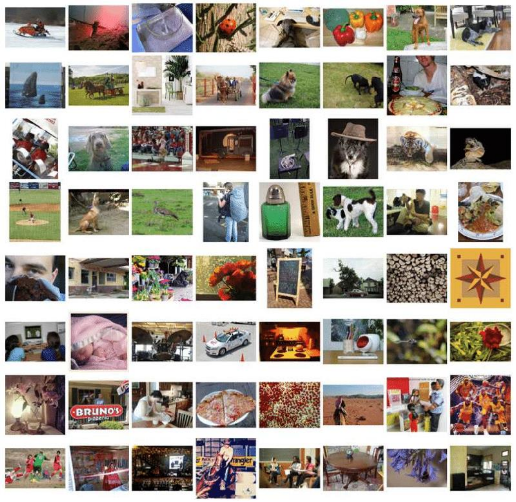
Convolutional Neural Networks (CNNs / ConvNets)

Key terms

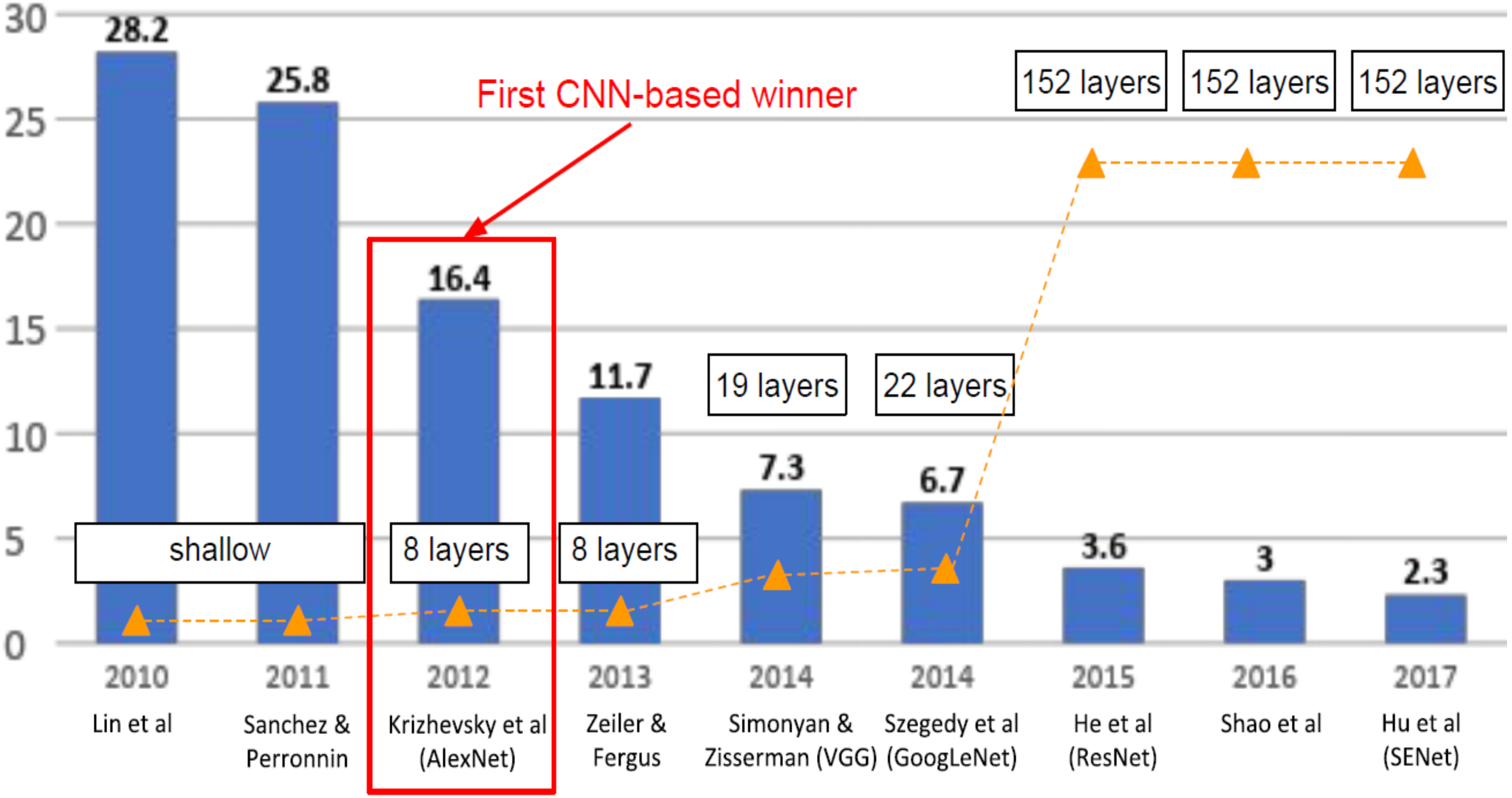
- Convolution
- Local receptive field
- Shared weights
- Pooling
- Channel
- Filter
- Kernel
- Stride
- Padding
- Feature map

- Data Augmentation
- Transfer learning



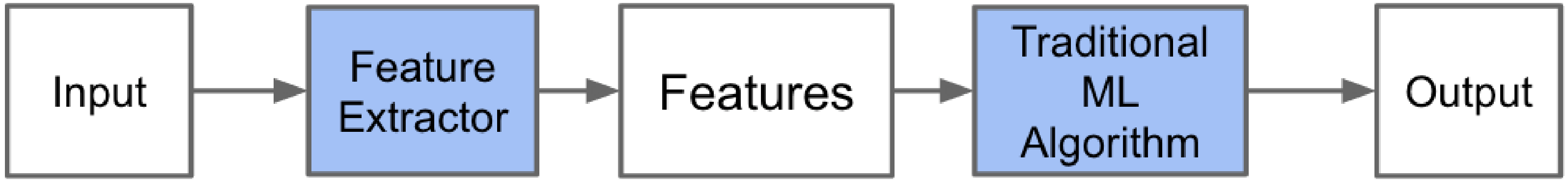
ILSVRC ImageNet challenge

- The images are large (256x256 pixels high) and there are 1,000 classes, some of which are really subtle (try distinguishing 120 dog breeds)
- The top-5 error rate for image classification fell from over 26% to barely over 3% in just five years (human error rate: 5.1%)

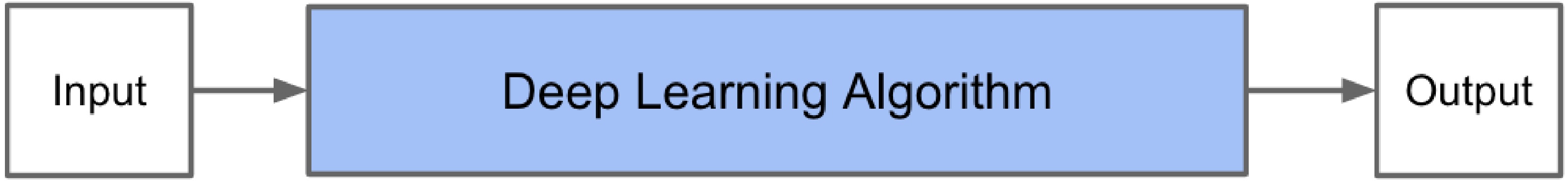


source: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture09.pdf

Traditional ML vs. Deep learning



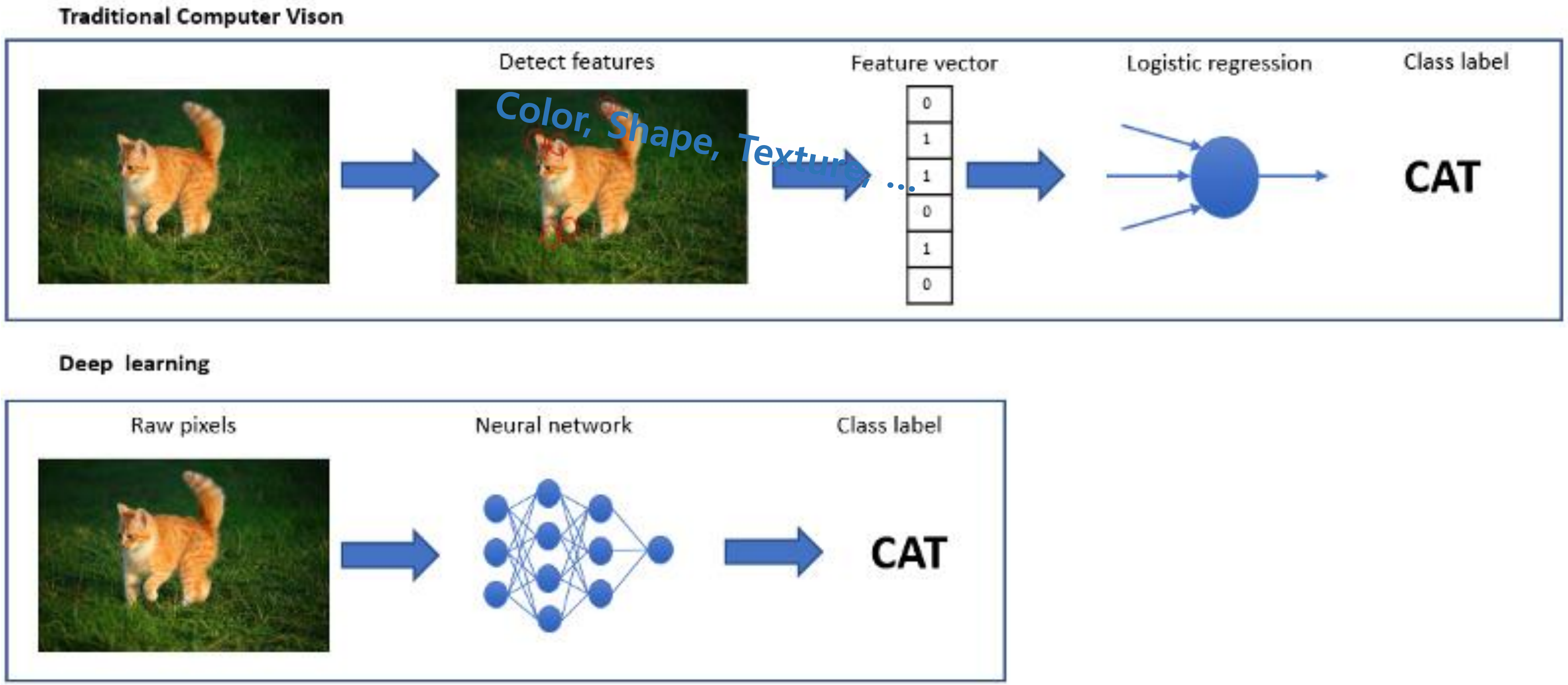
Traditional Machine Learning Flow



Deep Learning Flow

Coming up with features is difficult, time-consuming, requires expert knowledge.“
Applied machine learning" is basically feature engineering. *- Andrew Ng*

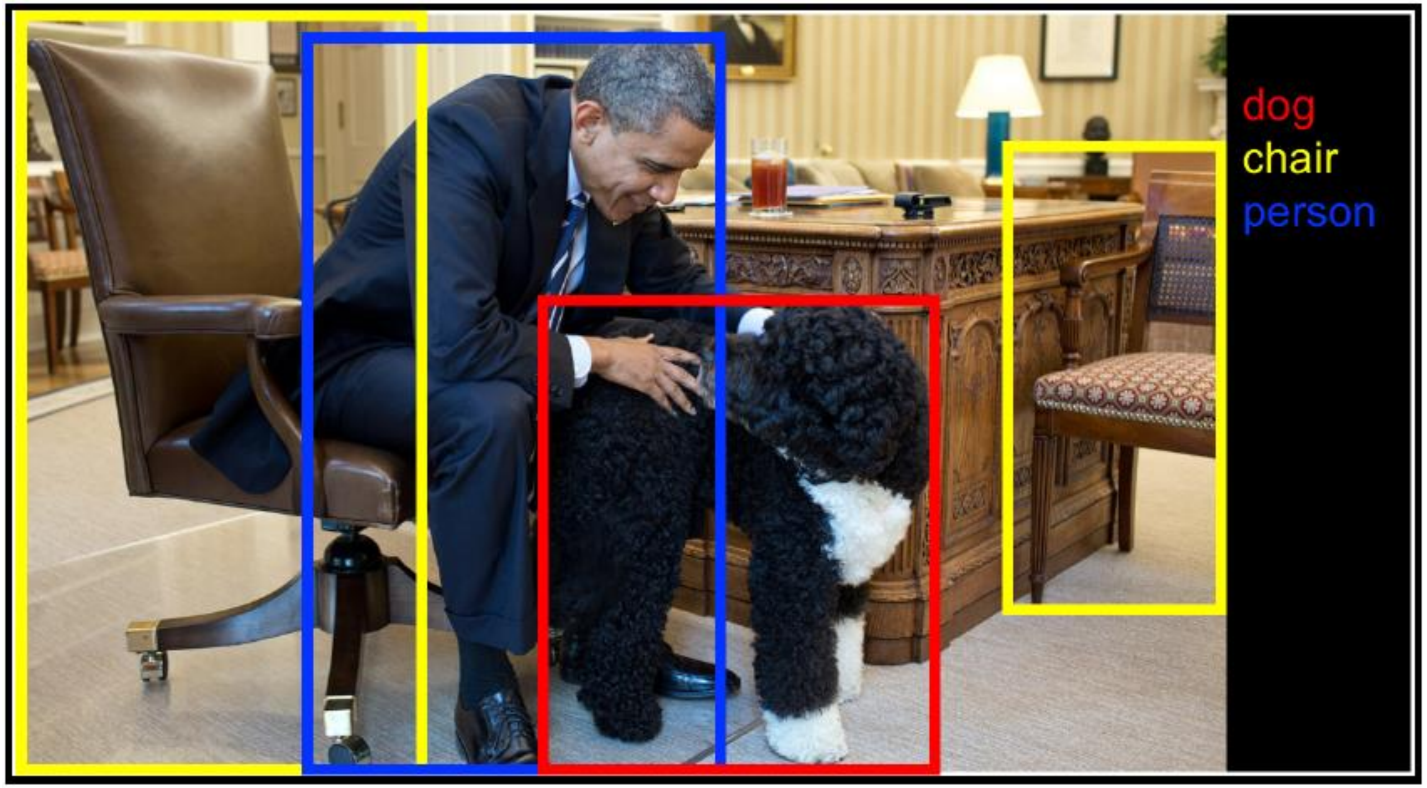
Traditional ML vs. Deep learning



Color, Shape, Texture, …

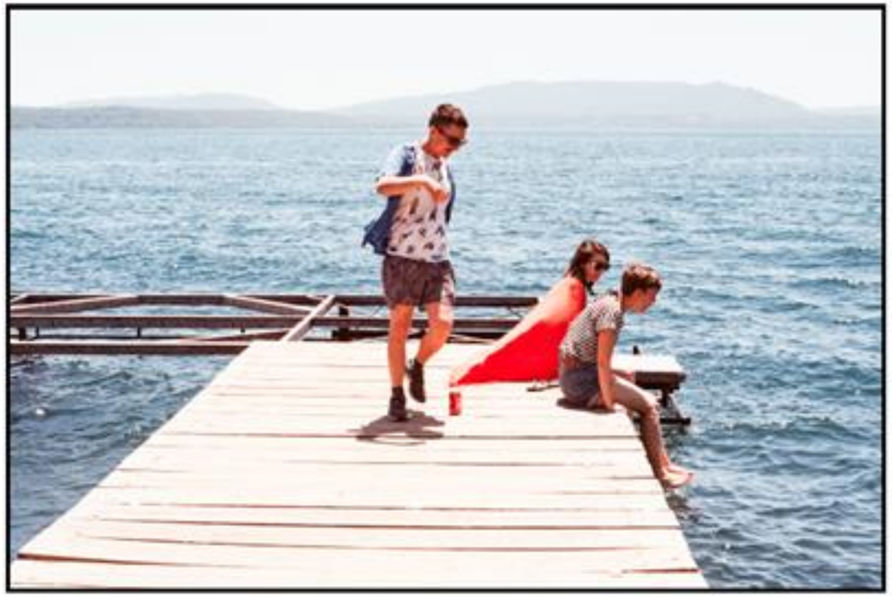
Source: https://blog.esciencecenter.nl/mcfly-time-series-classification-made-easy-e47de8d29838

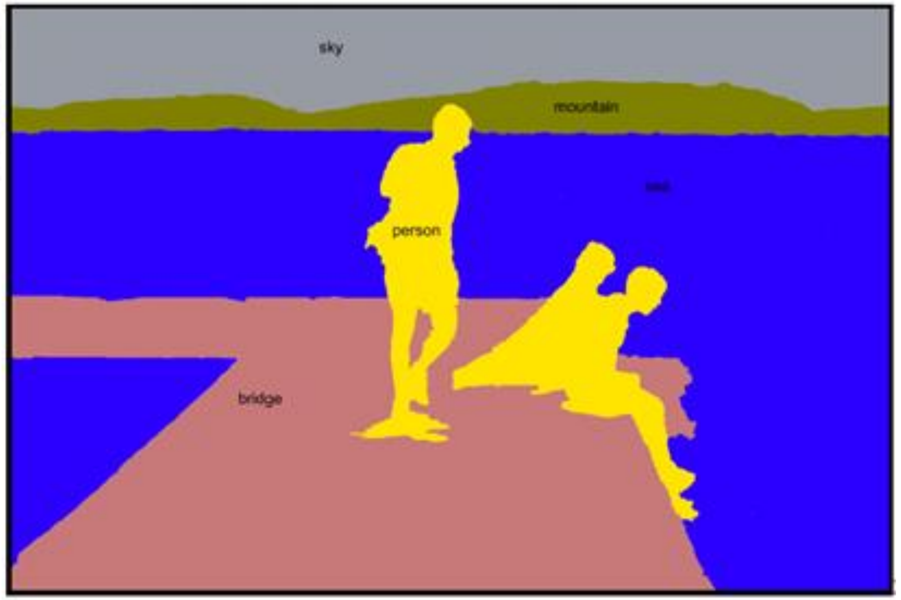
Deep Learning for Computer Vision - Object Detection



Source: "Deep Learning for Computer Vision", Shanmugamani

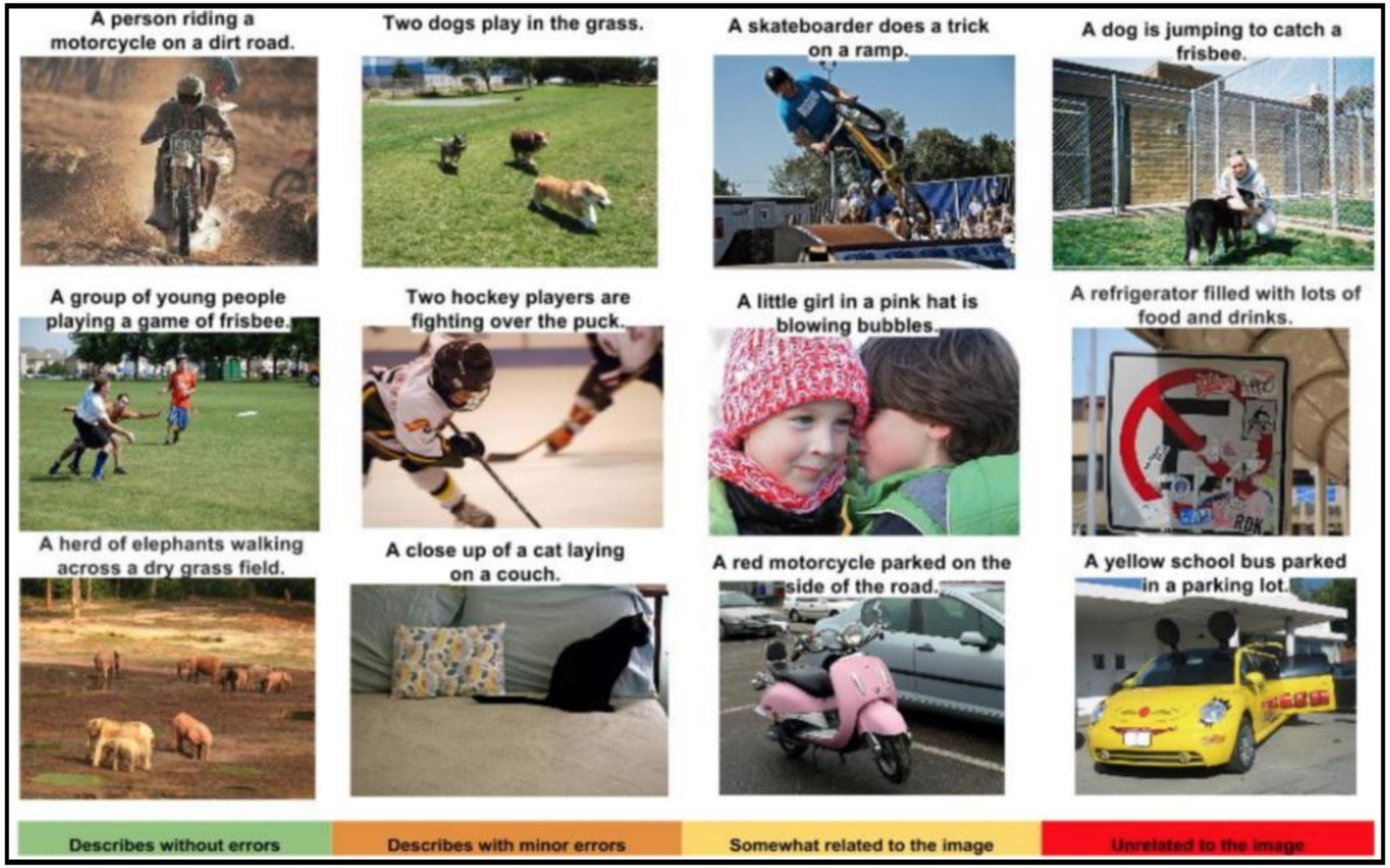
Deep Learning for Computer Vision - Semantic Segmentation





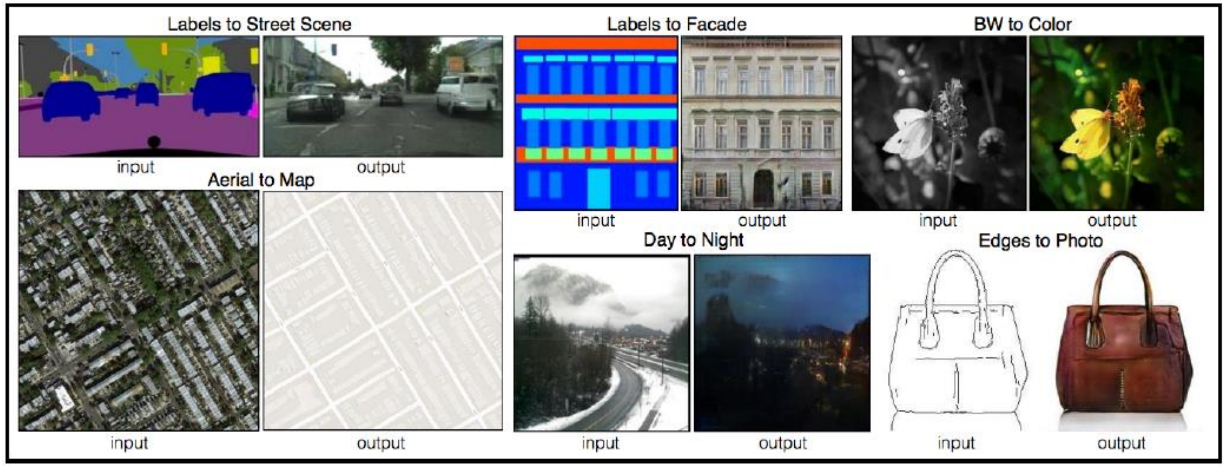
Source: "Deep Learning for Computer Vision", Shanmugamani

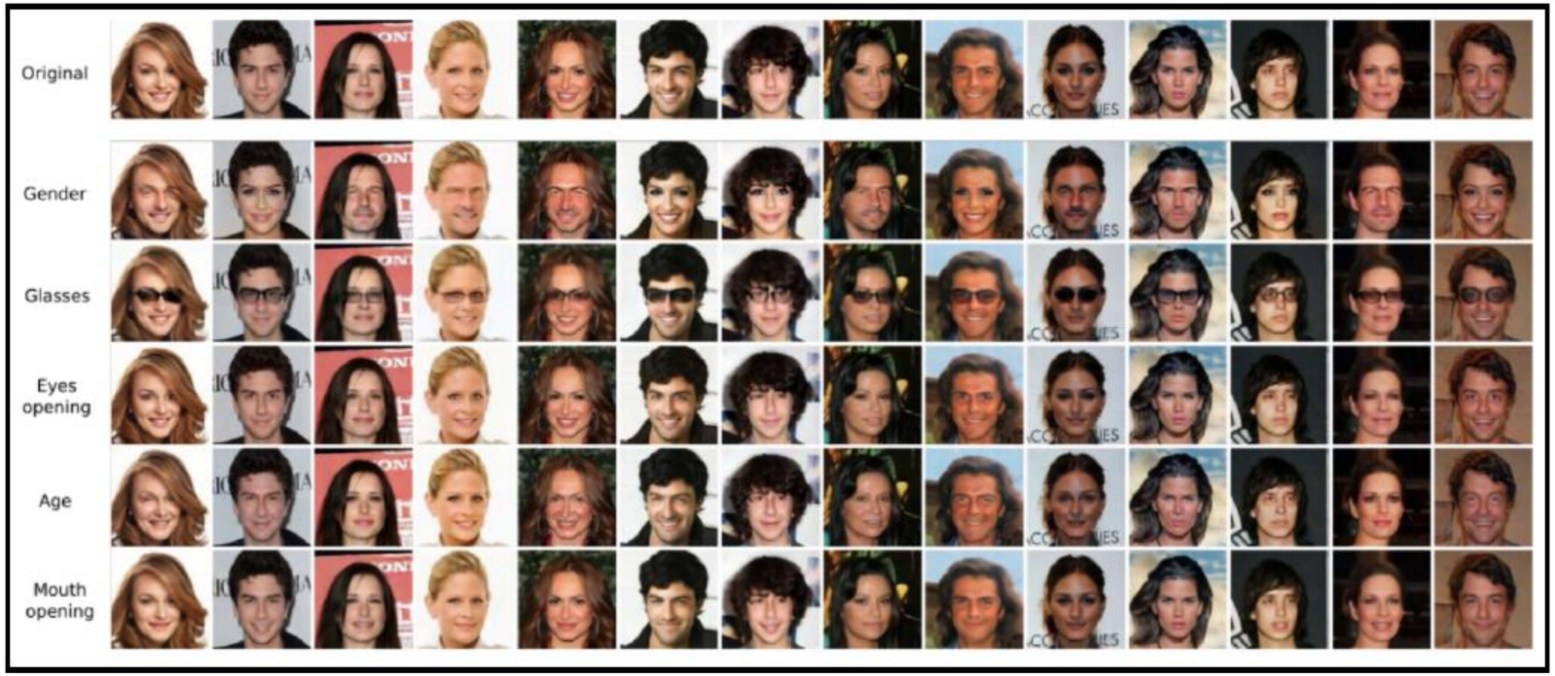
Deep Learning for Computer Vision - Image Captioning


A person riding a motorcycle on a dirt road.
Two dogs play in the grass.
A skateboarder does a trick on a ramp.
A dog is jumping to catch a frisbee.
A group of young people playing a game of frisbee.
Two hockey players are fighting over the puck.
A little girl in a pink hat is blowing bubbles.
A refrigerator filled with lots of food and drinks.
A herd of elephants walking across a dry grass field.
A close up of a cat laying on a couch.
A red motorcycle parked on the side of the road.
A yellow school bus parked in a parking lot.
Describes without errors
Describes with minor errors
Somewhat related to the image
Unrelated to the image

Source: "Deep Learning for Computer Vision", Shanmugamani

Deep Learning for Computer Vision - Image Generation



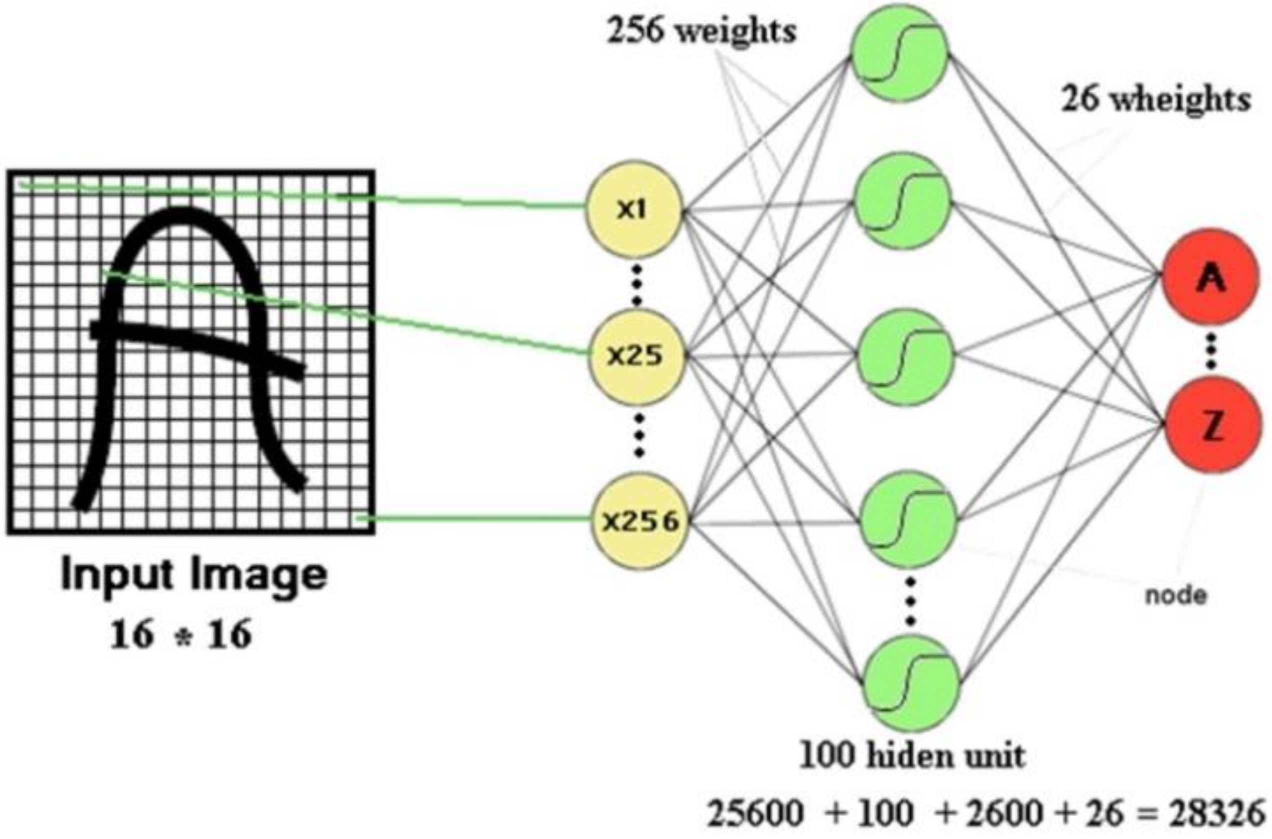


Source: "Deep Learning for Computer Vision", Shanmugamani

Background:
drawbacks of FCN in image recognition

When fully connected networks(FCN) are used for image recognition:

- the number of trainable parameters becomes extremely large

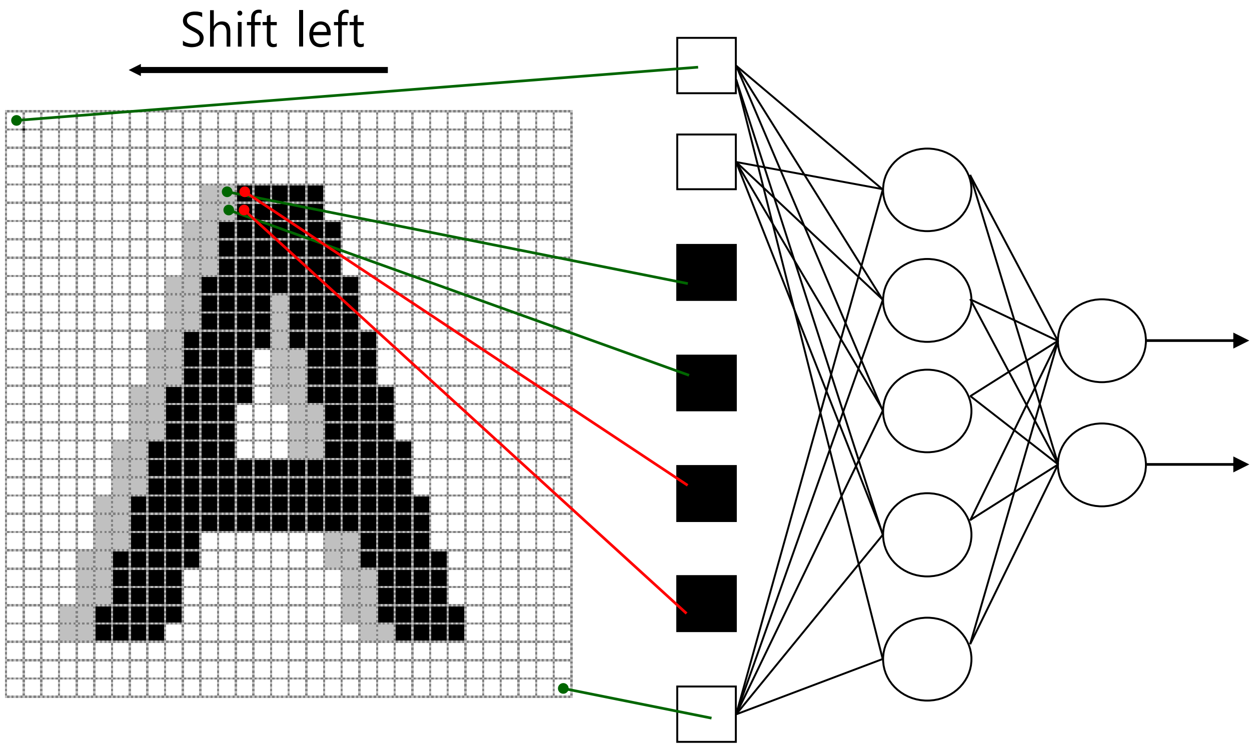


source: Convolutional neural networks by roozgard, abin

Background: drawbacks of FCN in image recognition

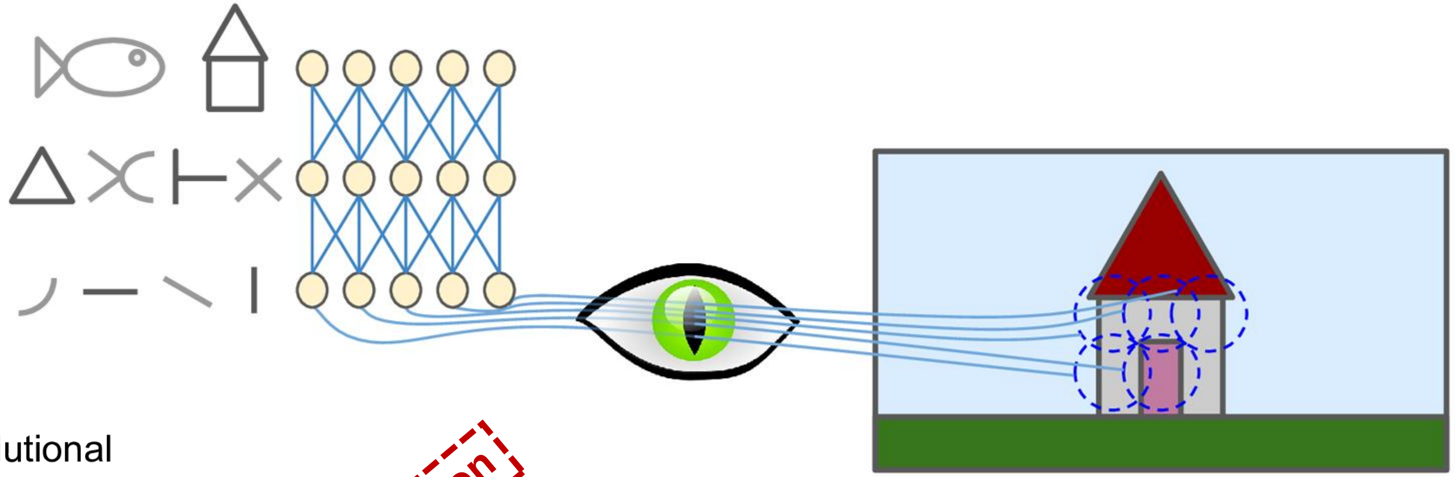
When FCN are used for image recognition:

- Little or no invariance to shifting, scaling, and other forms of distortion
- the spatial structure is completely ignored



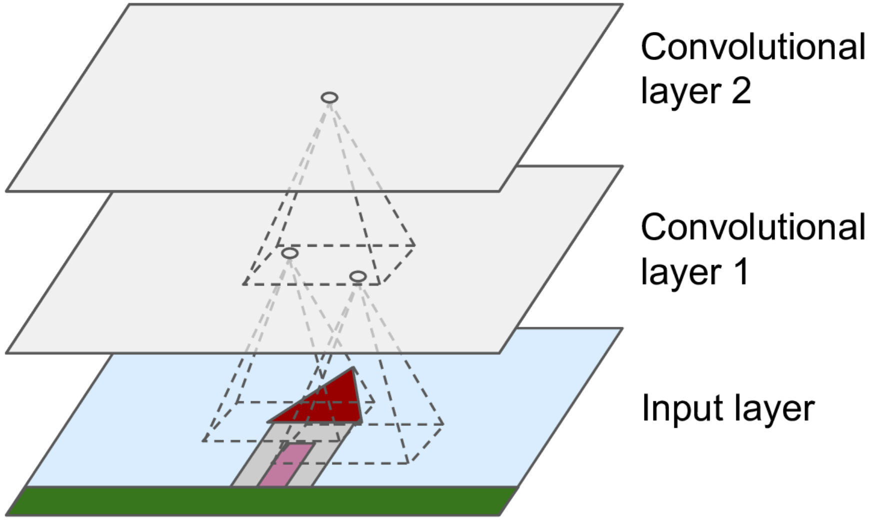
source: Convolutional neural networks by roozgard, abin

Background:
architecture of the visual cortex



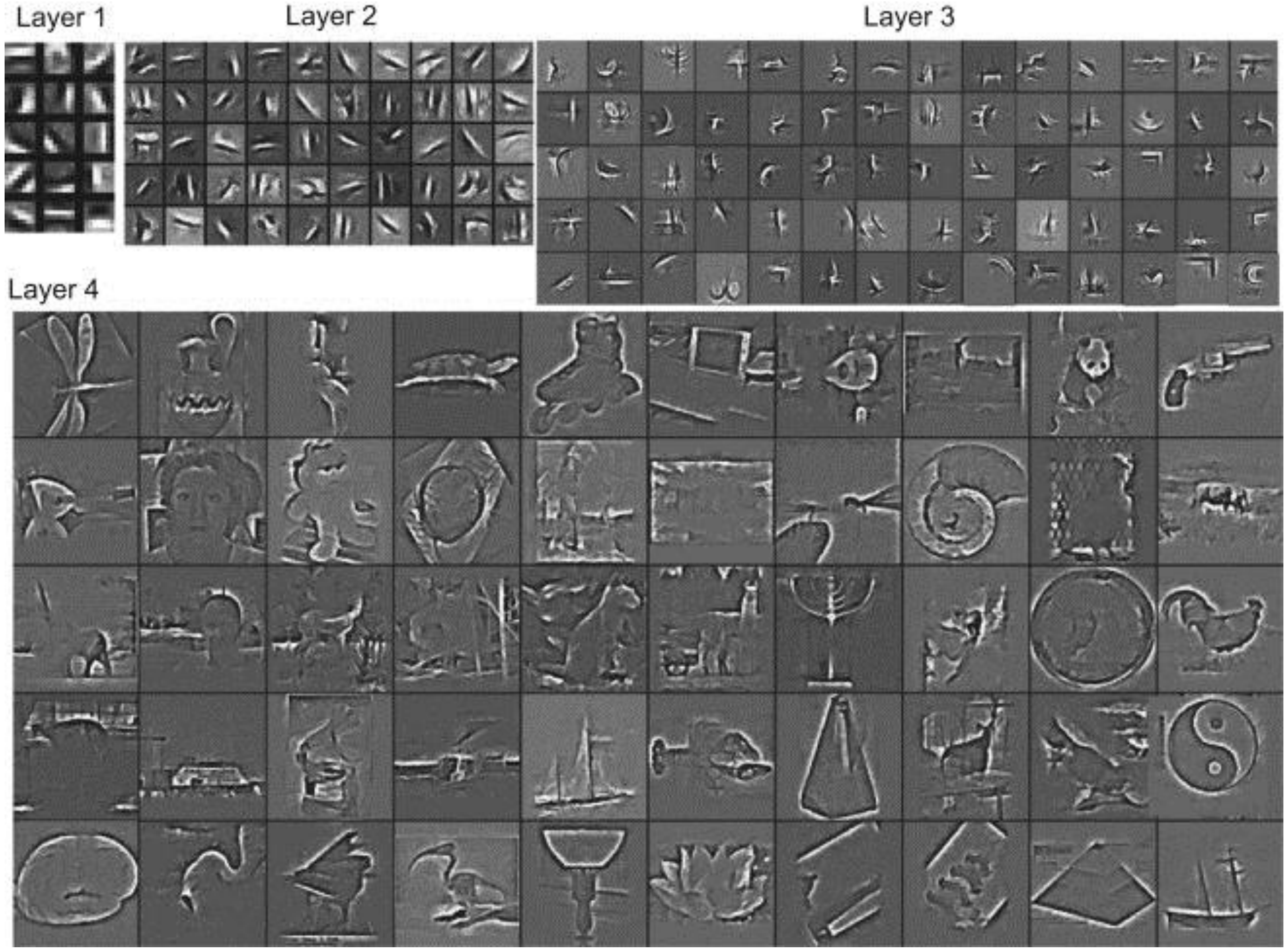
[Local receptive fields in the visual cortex]

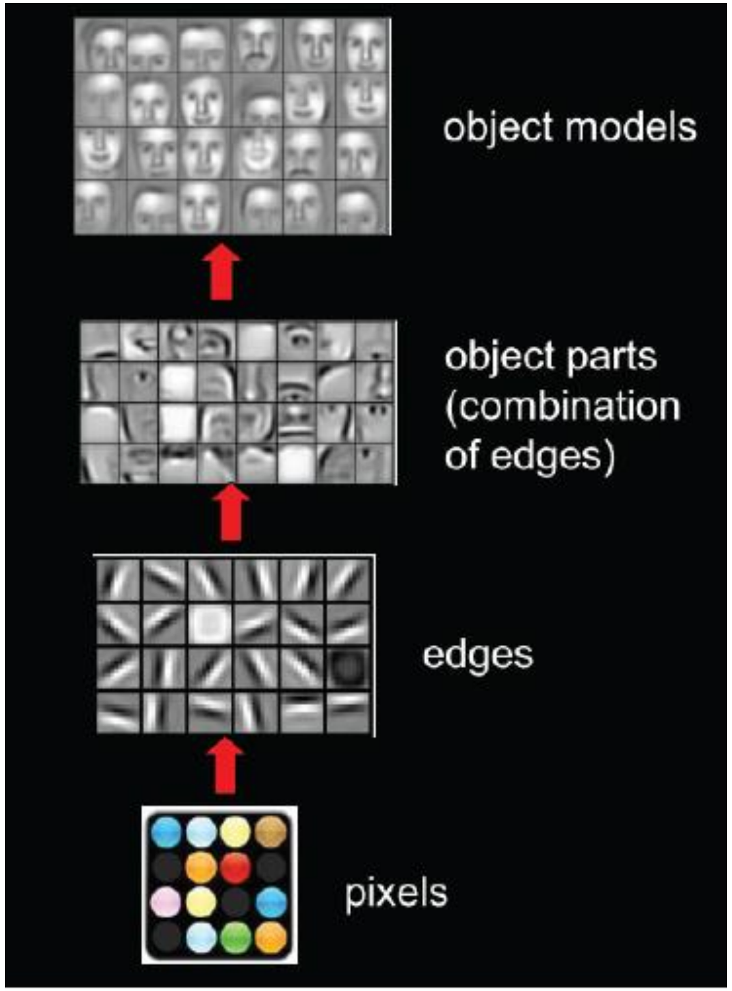
Implementation



[CNN layers with rectangular local receptive fields]

Background:
architecture of the visual cortex

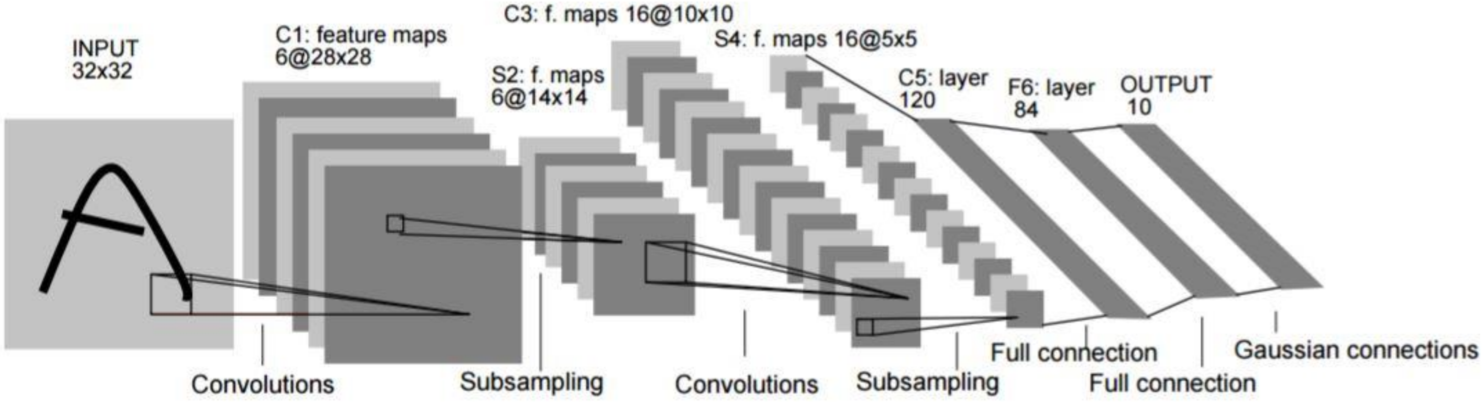




source: Detection guided deconvolutional network for hierarchical feature learning from [Liu, et al., 2015]

Convolutional neural networks

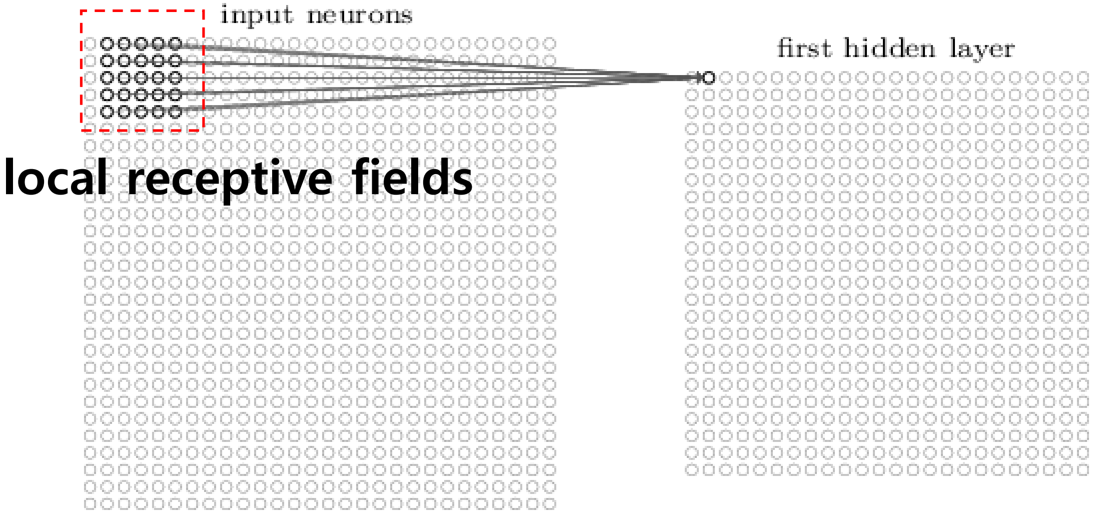
- "Gradient-based learning applied to document recognition", by Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, 1998.
- **CNN**s (or **convnet**s) use a special architecture which is particularly well-adapted to classify images.
    - an architecture which tries to take advantage of the **spatial** structure
    - layers have neurons arranged in 3 dimensions: **width**, **height**, and **depth**(or **channel**).
- Three basic ideas:
    - local receptive fields
    - shared weights
    - pooling



LeNet-5

Local receptive fields

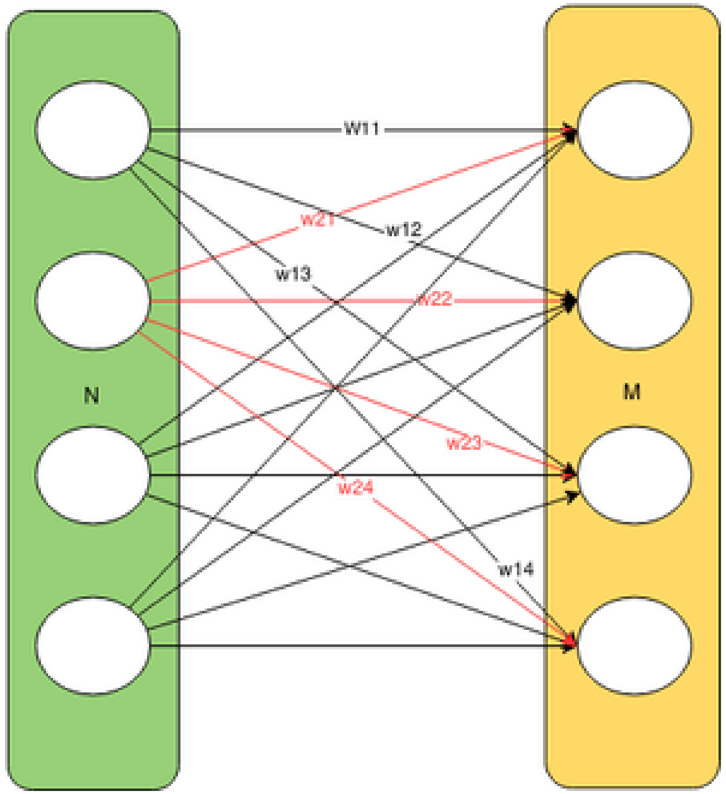
- Each neuron in the first hidden layer will be connected to a small region of the input neurons, say, for example, a 5×5 region, corresponding to 25 input pixels
- That region in the input image is called the **local receptive field** for the hidden neuron
- We then slide the local receptive field across the entire input image. If we have a 28×28 input image, and 5×5 local receptive fields, then there will be 24×24 neurons in the hidden layer



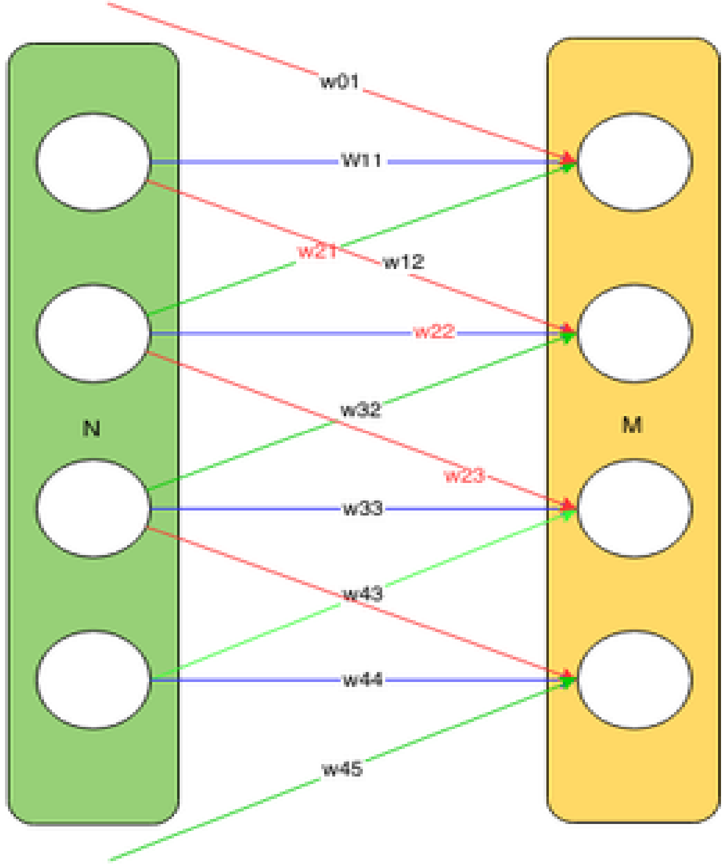
Shared weights and biases

- The same weights and bias are used for each of the 24×24 hidden neurons
  - All the neurons in the first hidden layer detect exactly the same feature
  - CNNs are well adapted to the *translation invariance* of images
- They greatly reduce the number of parameters involved in a CNN.
  - 25=5×5 shared weights, plus a shared bias.
  - A fully connected first layer, with 784=28×28 input neurons, and 30 hidden neurons. That's a total of 784×30 weights, plus an extra 30 biases, for a total of 23,550 parameters.

Shared weights and biases



FCN

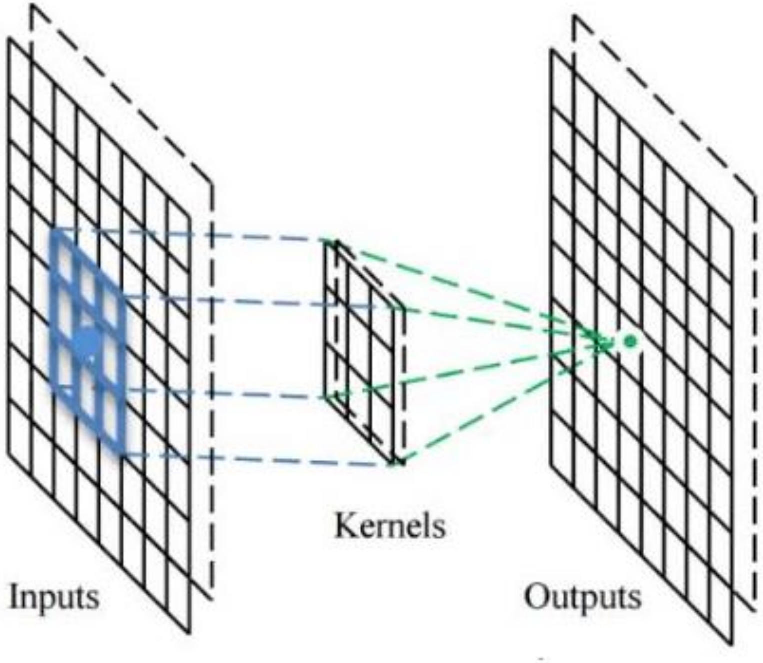


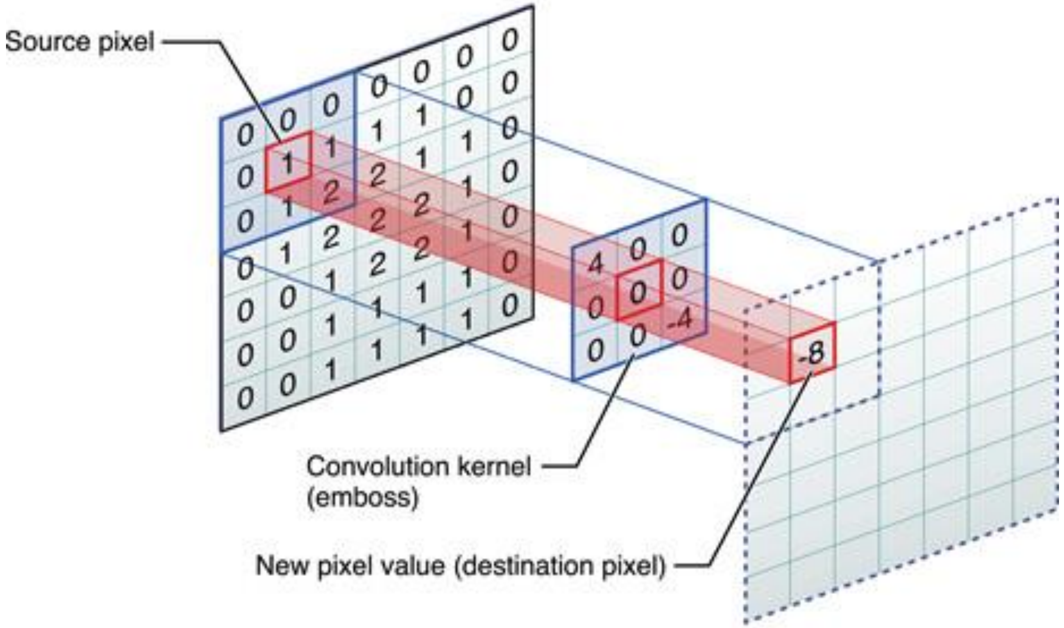
CNN

source: https://pennlio.wordpress.com/2014/04/11/fully-connected-locally-connected-and-shared-weights-layer-in-neural-networks/

Filters

- A neuron's weights can be represented as a small image the size of the receptive field.
- The image is called a **filter** or **kernel**.

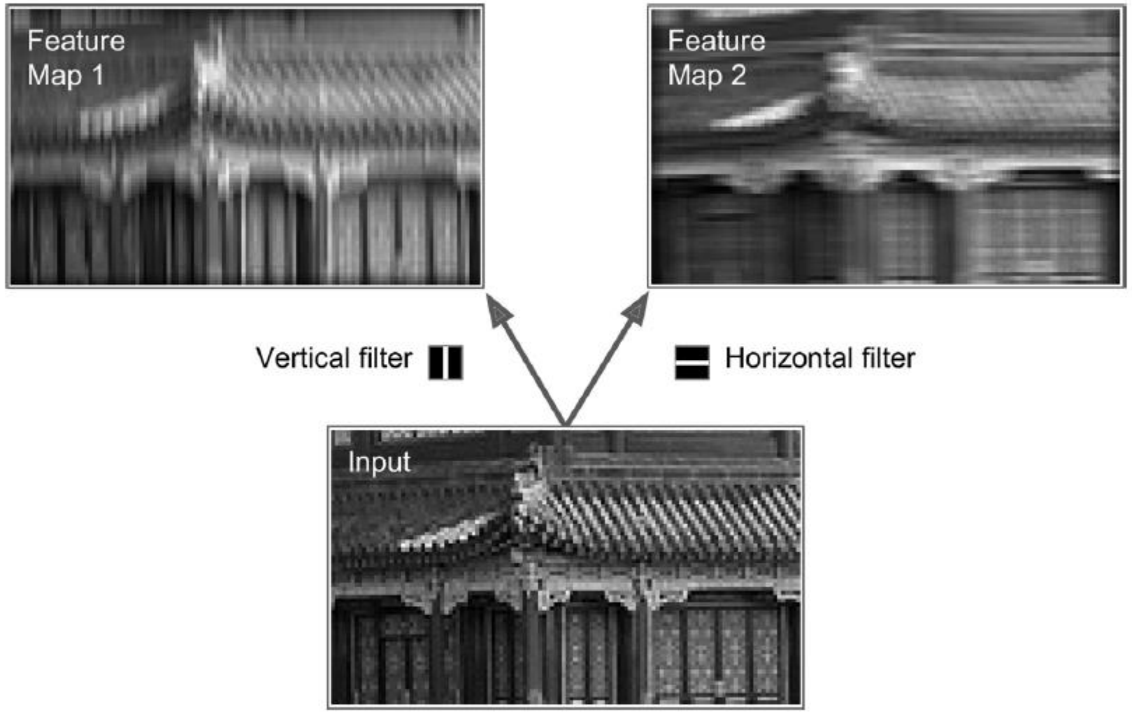




source: https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/Art/kernel_convolution.jpg

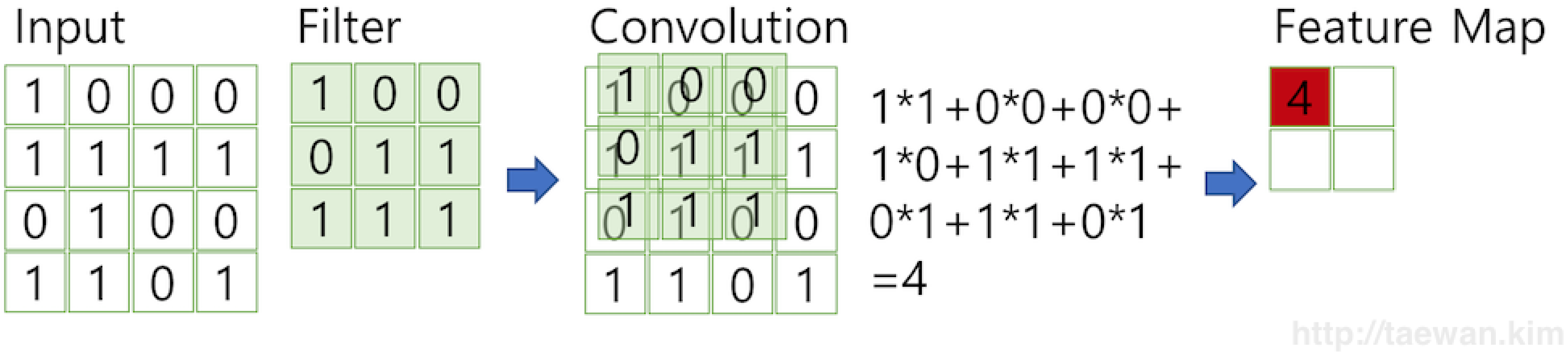
Filters

- The right figure shows two possible filters
  - The first one is represented as a black square with a *vertical white line* in the middle; neurons using these weights will ignore everything in their receptive field except for the central vertical line
  - The second filter is a black square with a *horizontal white line* in the middle.



Ex) HxW Convolution

- 4x4 image
- 3x3 filter
- stride = 1



Input

1	0	0	0
1	1	1	1
0	1	0	0
1	1	0	1

Filter

1	0	0
0	1	1
1	1	1

Convolution

1*1+0*0+0*0+
1*0+1*1+1*1+
0*1+1*1+0*1
=4

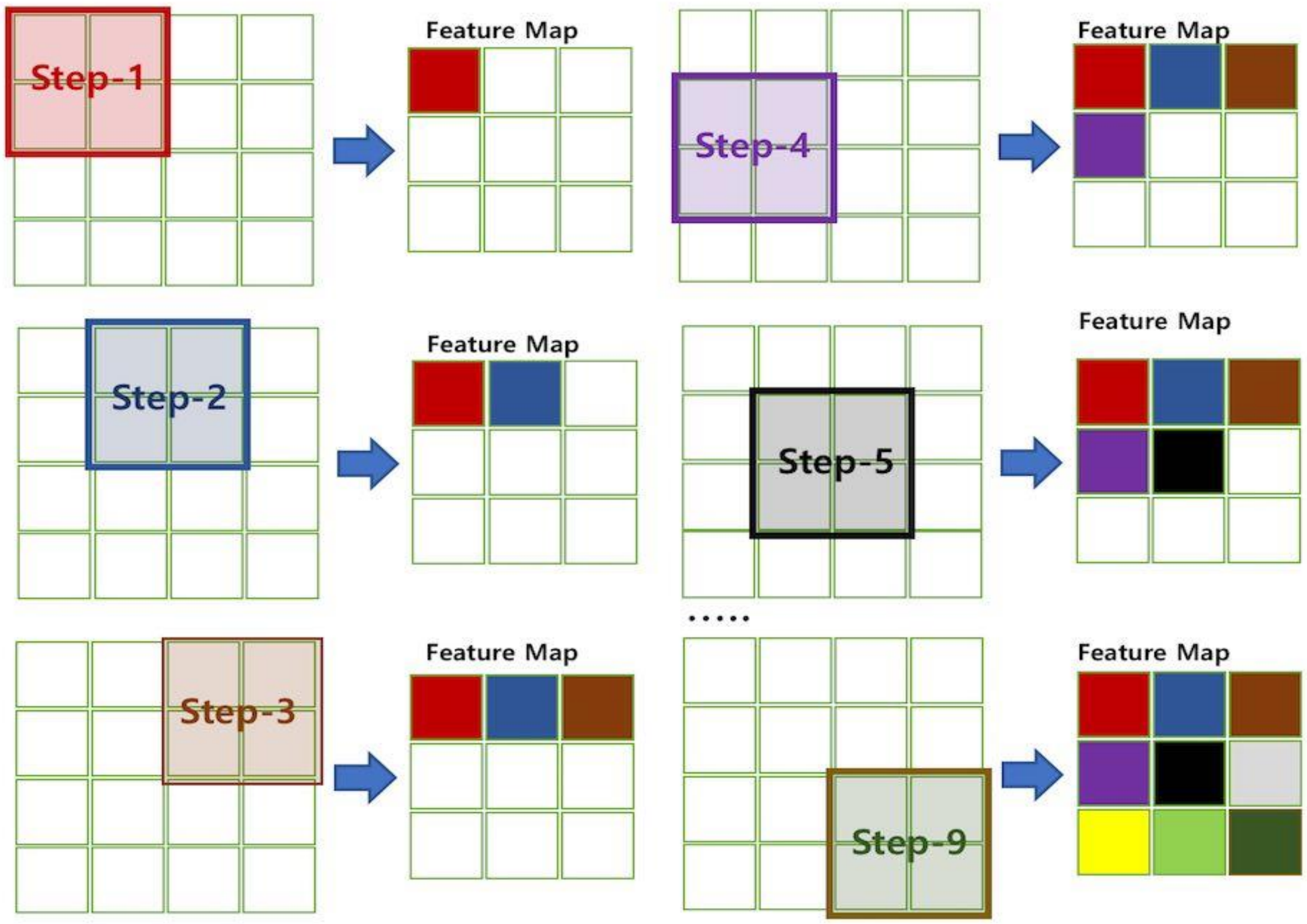
Feature Map

4	

source: http://taewan.kim/post/cnn/

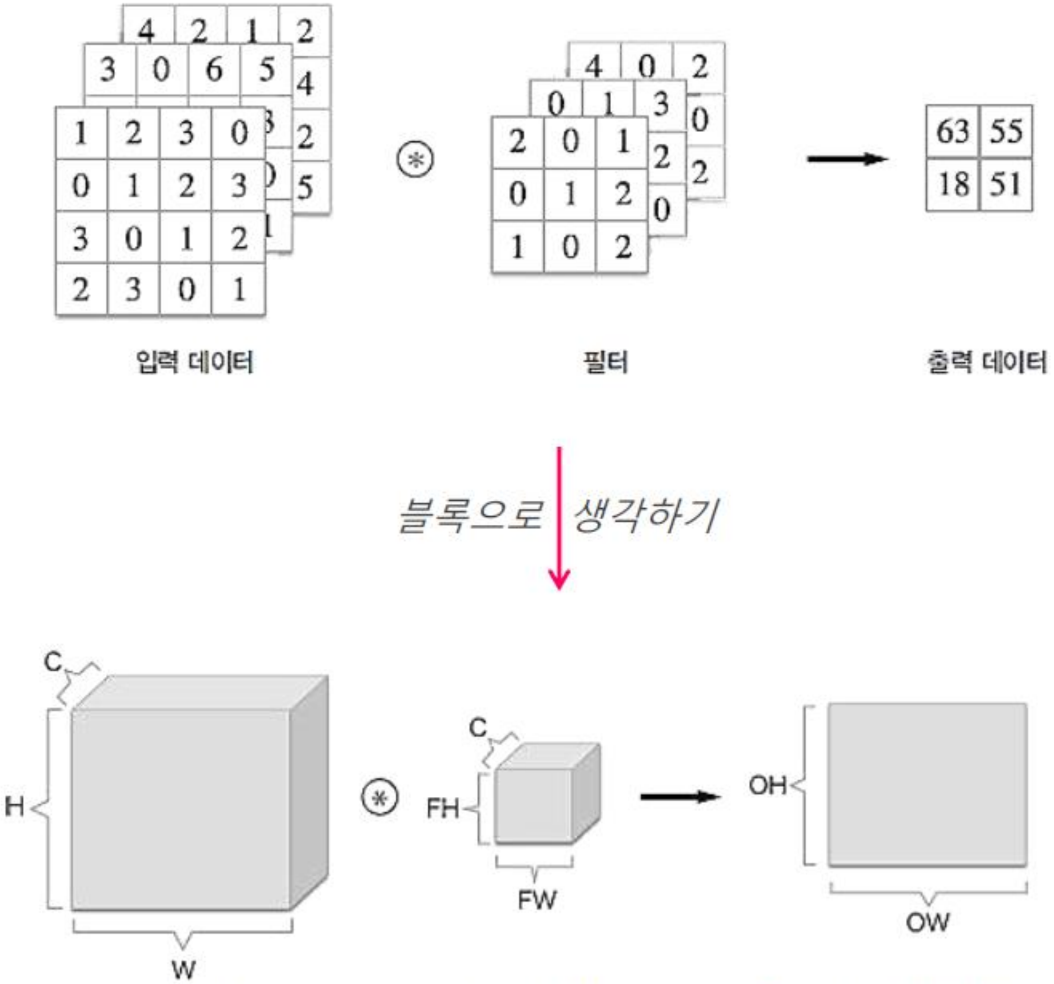
Ex) HxW Convolution

- 4x4 image
- 2x2 filter
- stride = 1

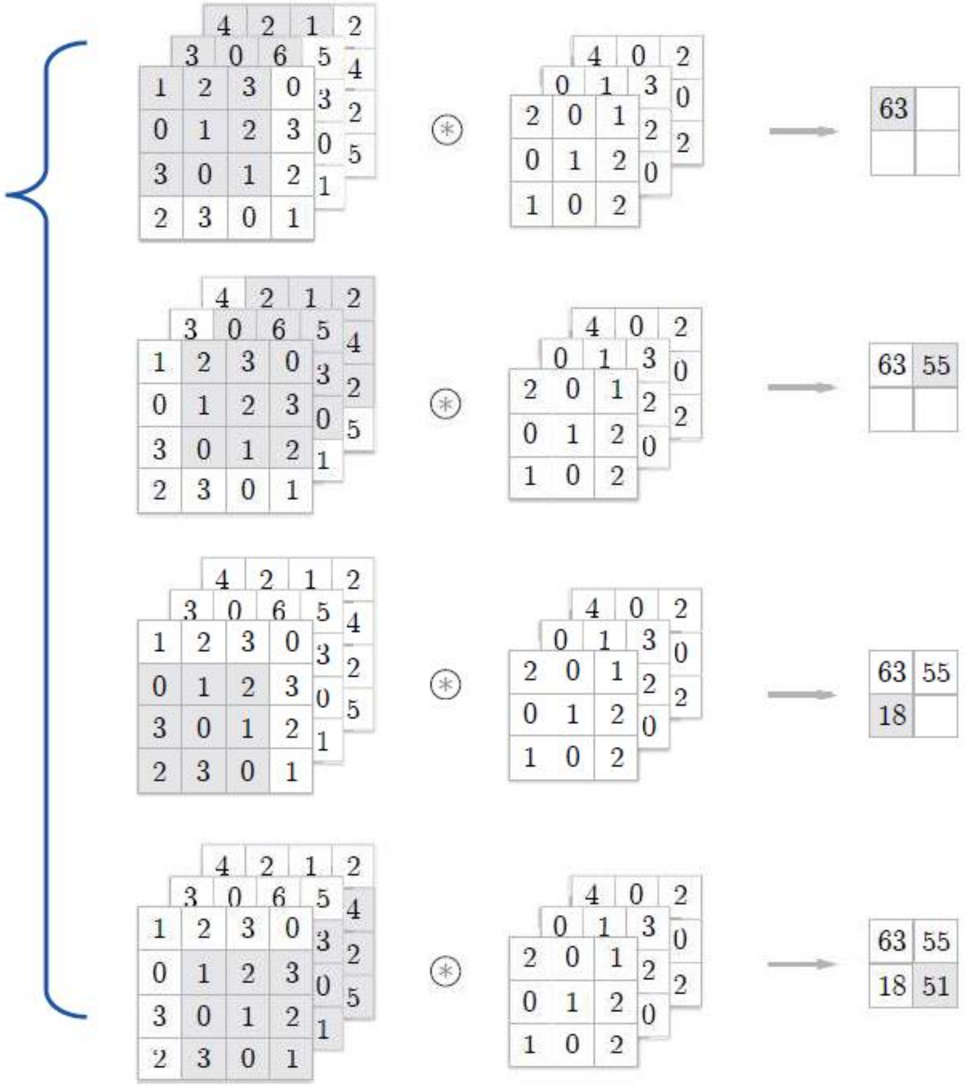


source: http://taewan.kim/post/cnn/

Ex) HxWxC Convolution

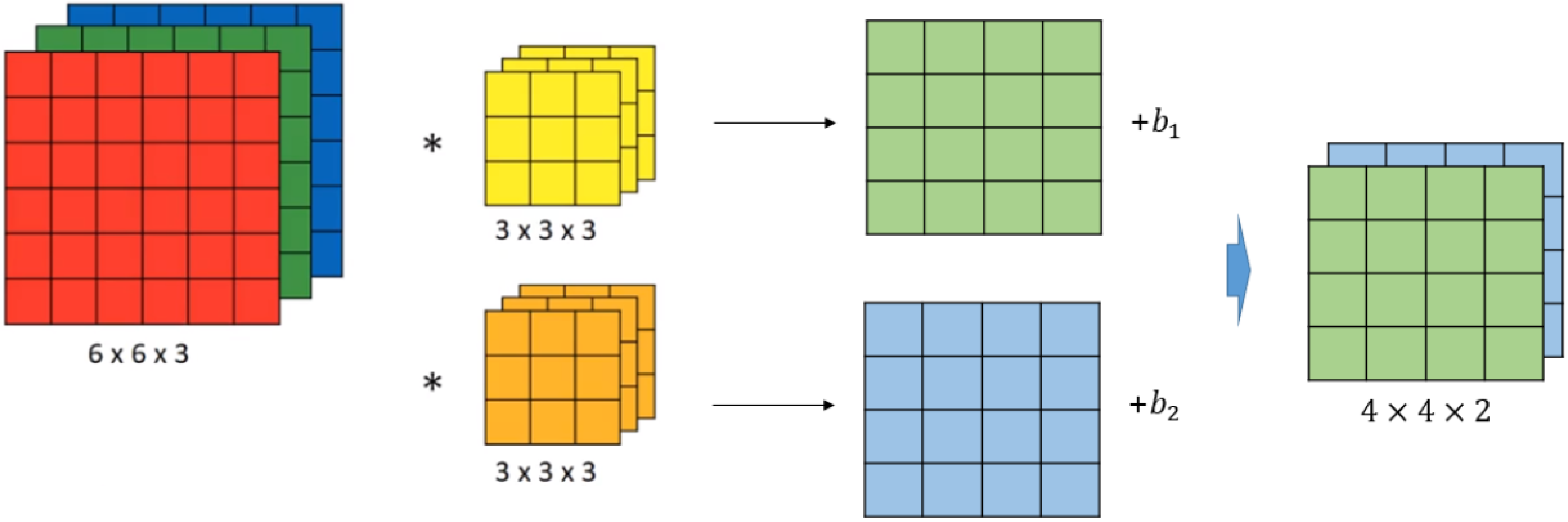


(C, H, W) ⊛ (C, FH, FW) → $(1, OH, OW)$



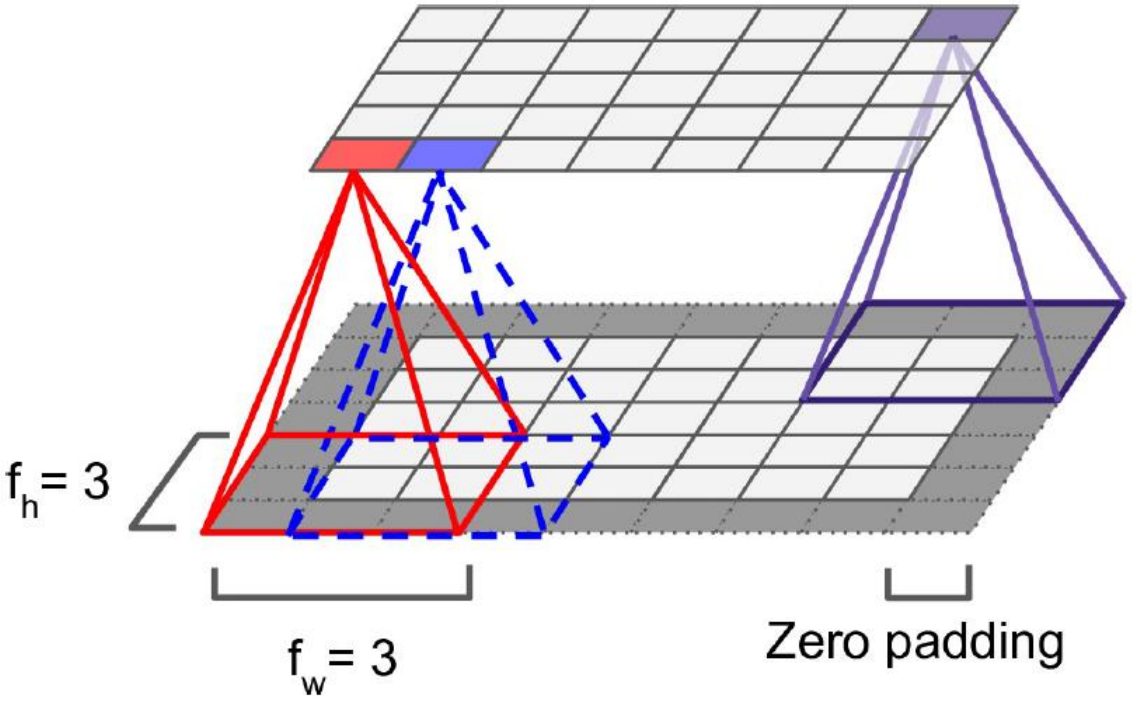
source: "Deep Learning from Scratch", 2017.

Ex) WxHxCxN Convolution



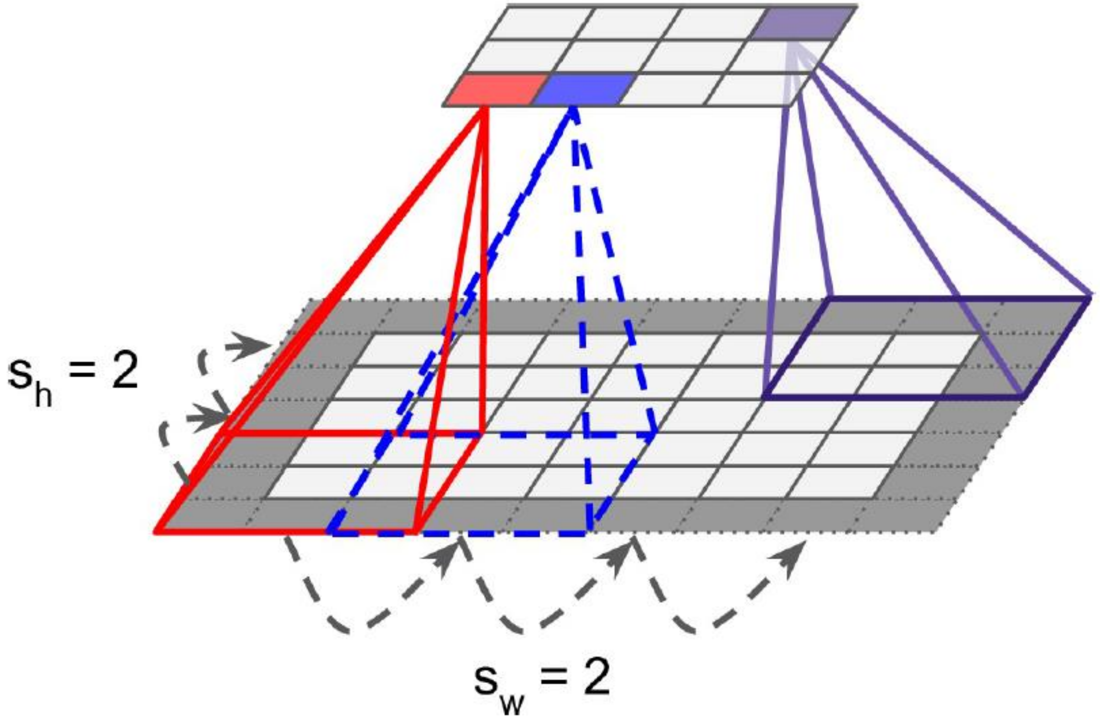
Padding

- In order for a layer to have the same height and width as the previous layer, it is common to add zeros around the inputs
- This is called **zero padding**

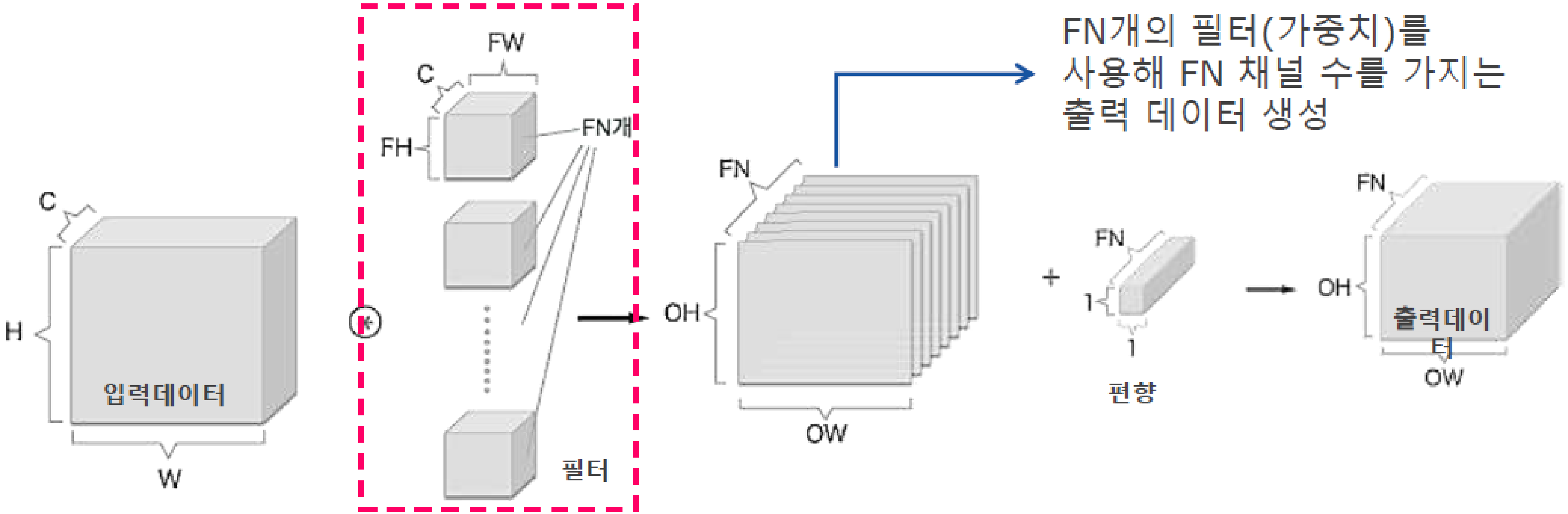


Stride

- It is also possible to connect a large input layer to a much smaller layer by spacing out the receptive fields.
- The distance between two consecutive receptive fields is called the **stride**.
- In the diagram, a 5 × 7 input layer (plus zero padding) is connected to a 3 × 4 layer, using 3 × 3 receptive fields and a stride of 2



Convolutional layer: summary



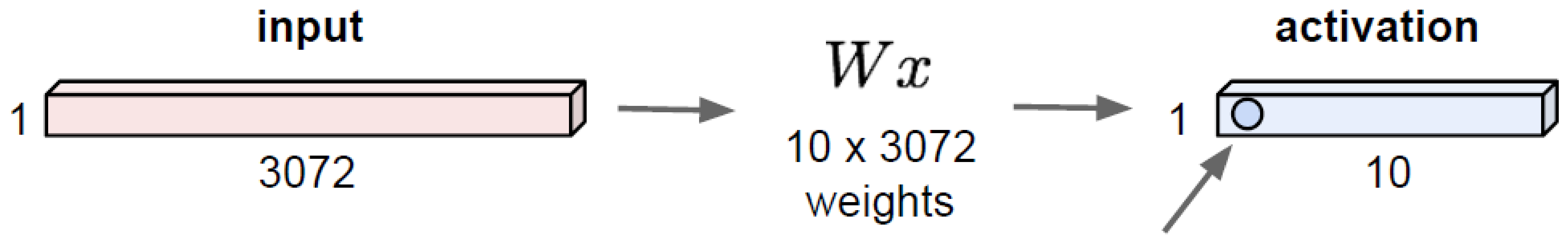
FN개의 필터(가중치)를
사용해 FN 채널 수를 가지는
출력 데이터 생성

$$(C,H,W) \circledast (FN,C,FH,FW) \longrightarrow (FN,OH,OW) + (FN,1,1) \longrightarrow (FN,OH,OW)$$

source: "Deep Learning from Scratch", 2017.

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



input

1

3072

Wx

10 x 3072
weights

activation

1

10

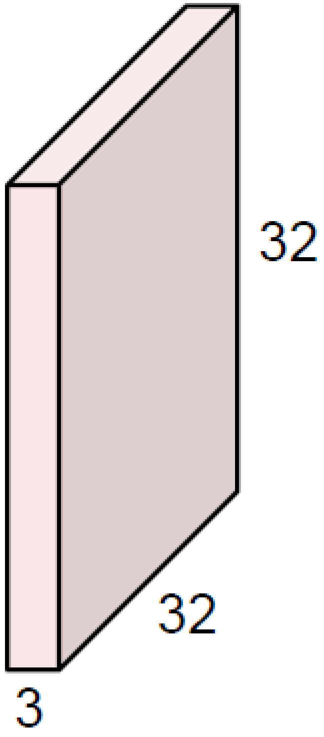
1 number:
the result of taking a dot product
between a row of W and the input
(a 3072-dimensional dot product)

[Source] CS231n: Convolutional Neural Networks for Visual Recognition

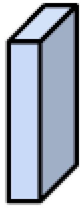
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



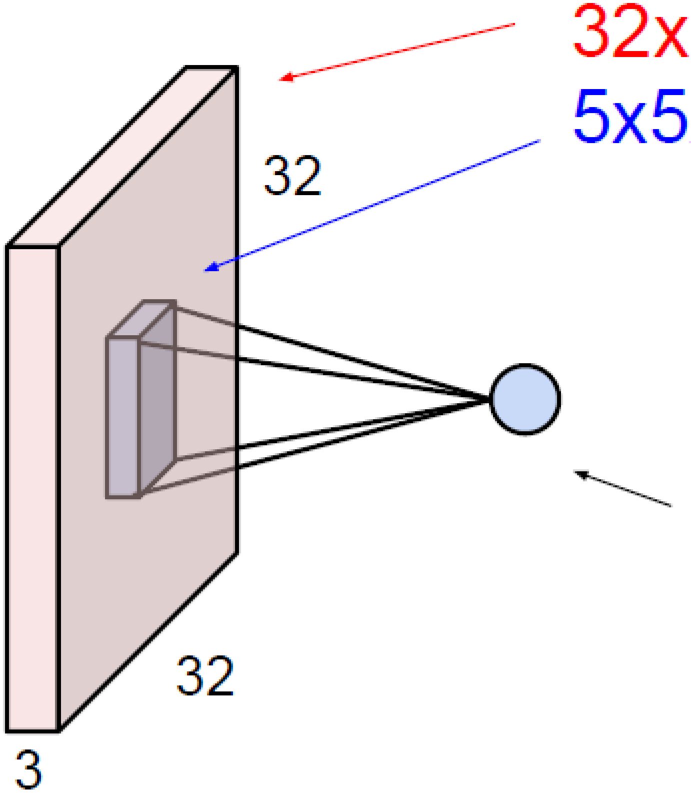
5x5x3 filter



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

[Source] CS231n: Convolutional Neural Networks for Visual Recognition

Convolution Layer



32x32x3 image

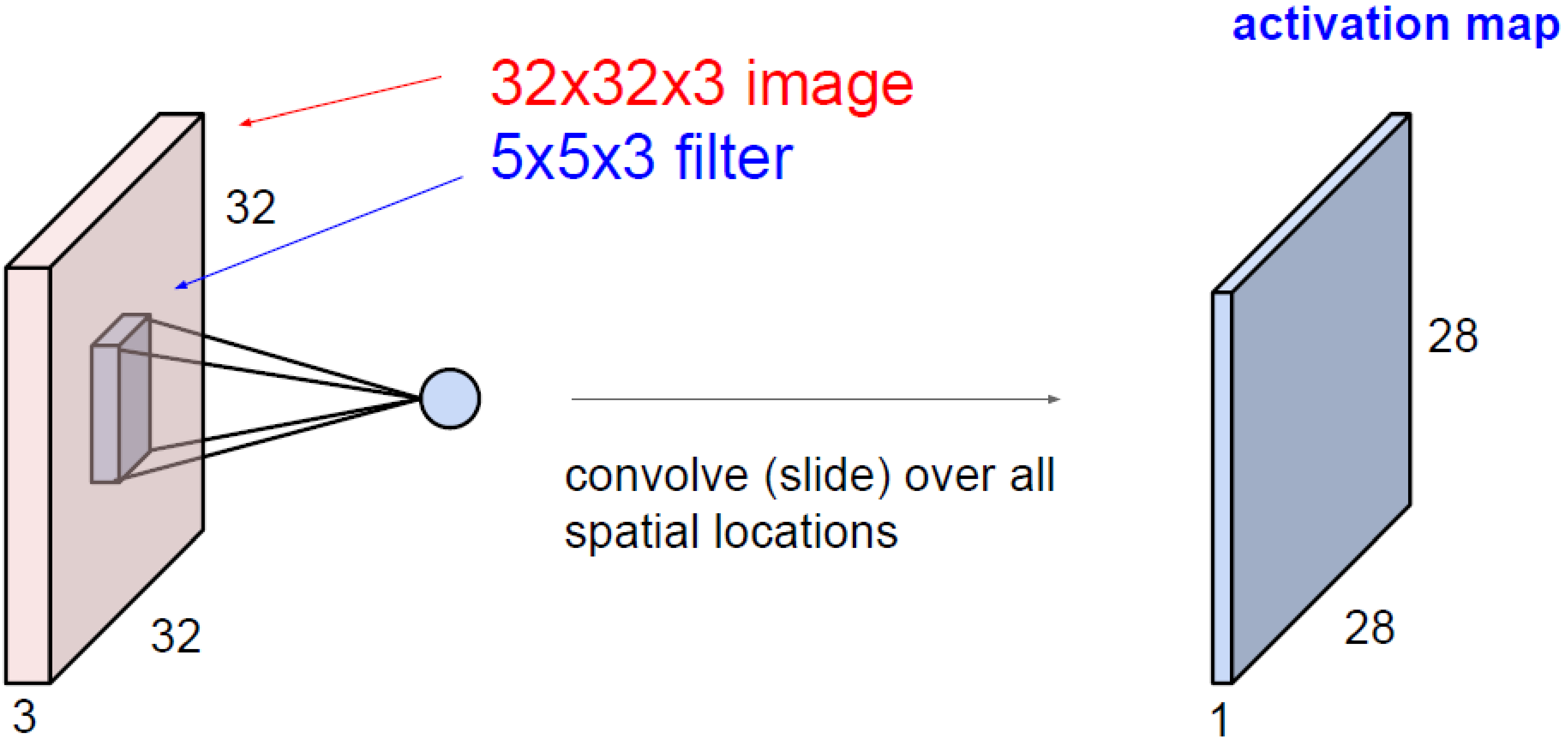
5x5x3 filter w

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)

$$w^T x + b$$

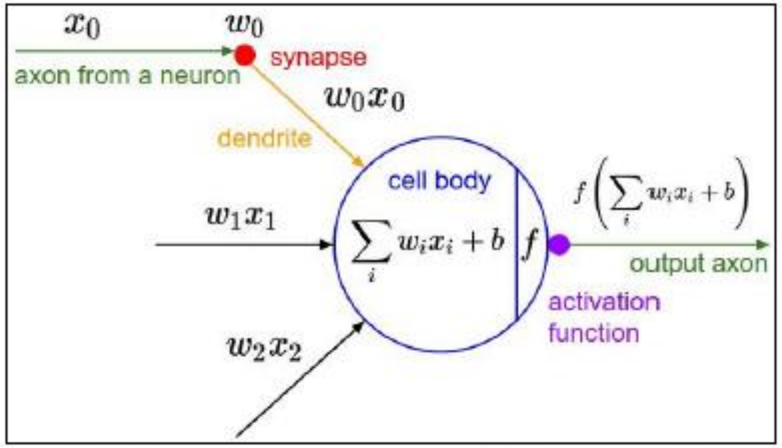
[Source] CS231n: Convolutional Neural Networks for Visual Recognition

Convolution Layer

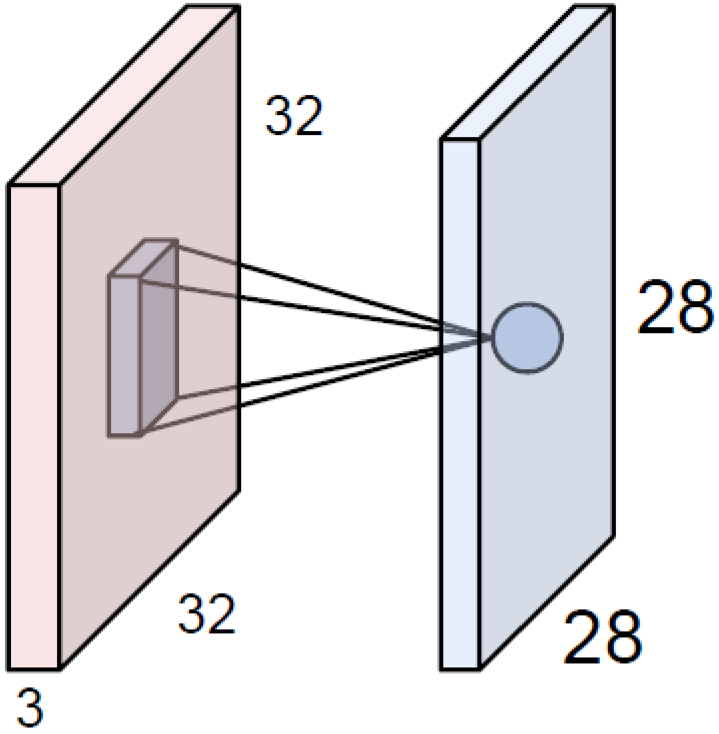


[Source] CS231n: Convolutional Neural Networks for Visual Recognition

The brain/neuron view of CONV Layer



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



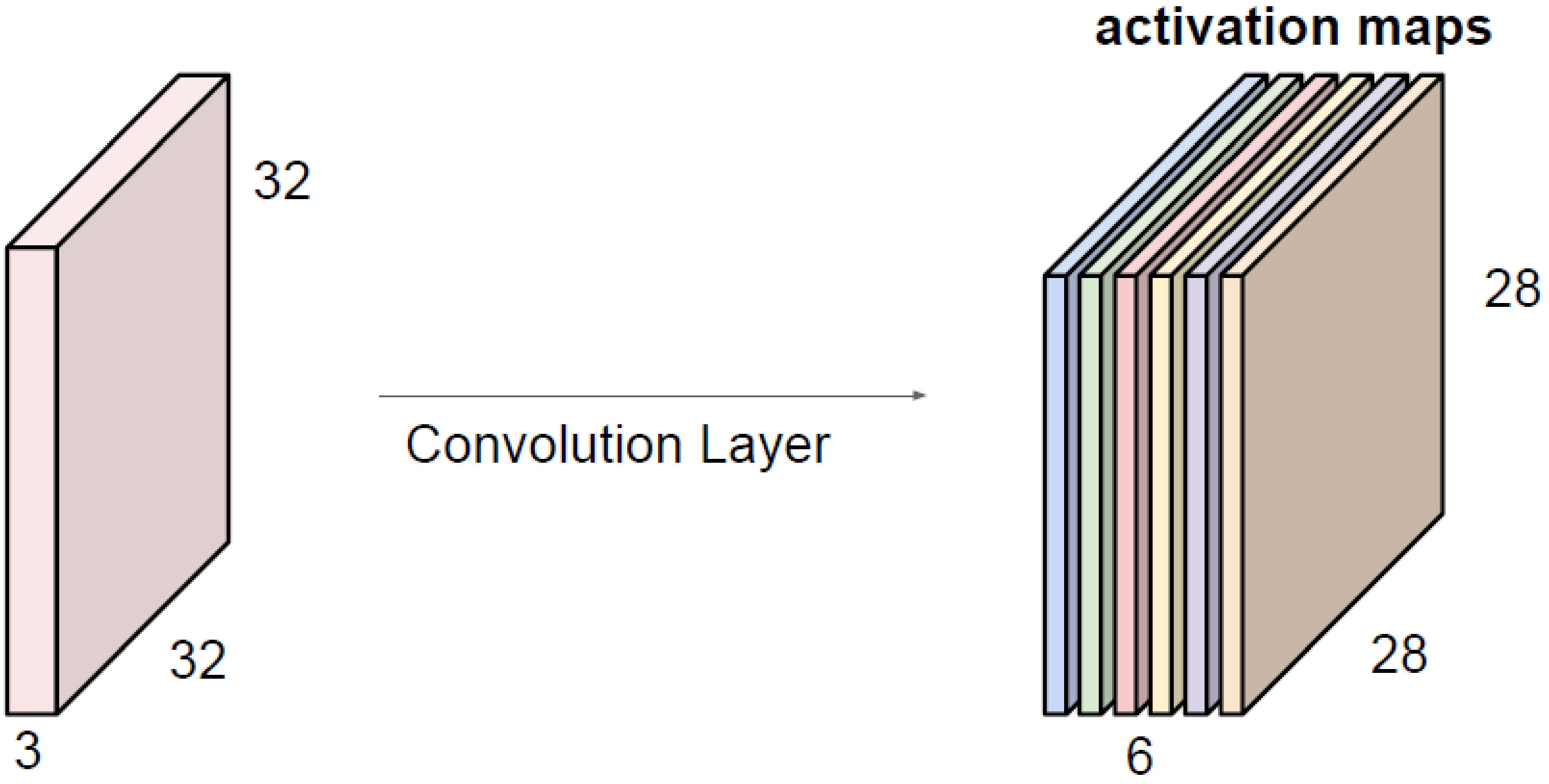
An activation map is a 28x28 sheet of neuron outputs:

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"

[Source] CS231n: Convolutional Neural Networks for Visual Recognition

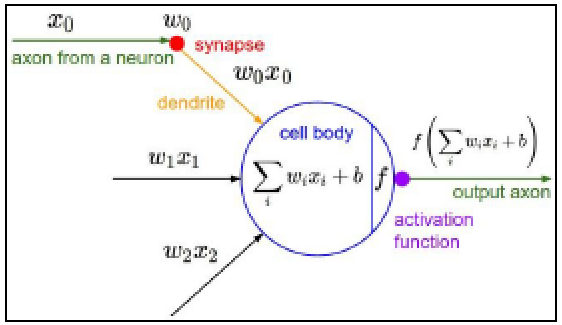
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

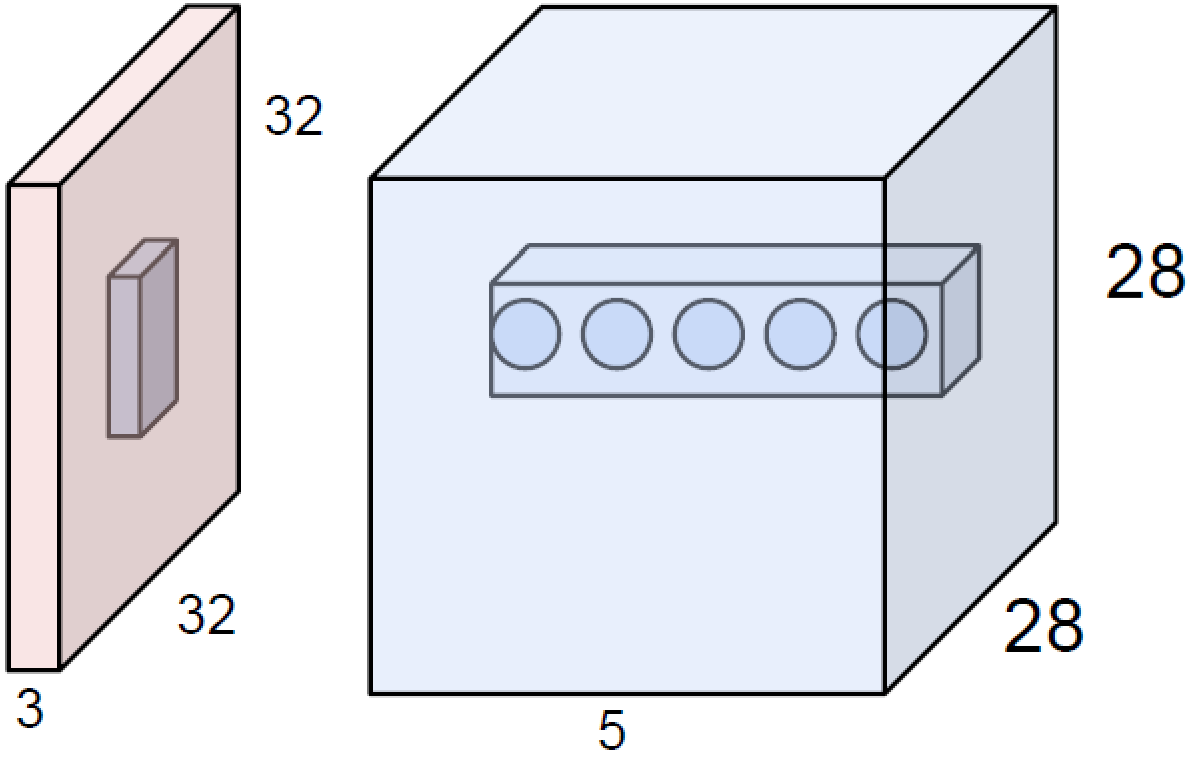


We stack these up to get a "new image" of size 28x28x6!

[Source] CS231n: Convolutional Neural Networks for Visual Recognition

The brain/neuron view of CONV Layer



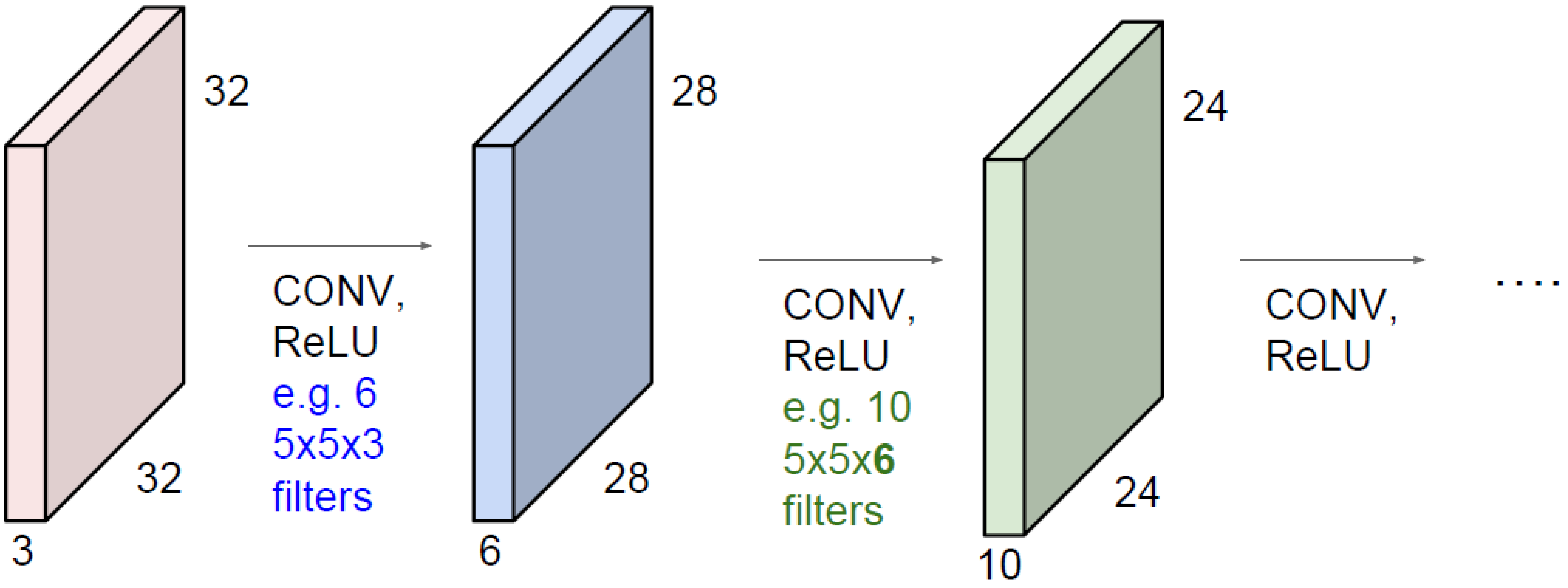


E.g. with 5 filters,
CONV layer consists of
neurons arranged in a 3D grid
(28x28x5)

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

[Source] CS231n: Convolutional Neural Networks for Visual Recognition

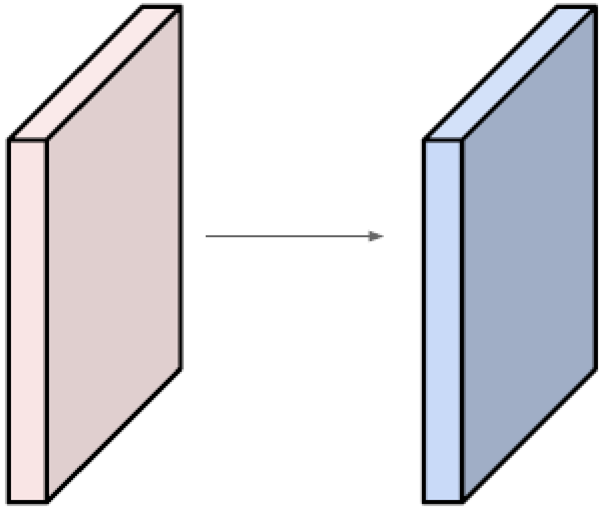
ConvNet is a sequence of Convolution Layers, interspersed with activation functions



[Source] CS231n: Convolutional Neural Networks for Visual Recognition

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

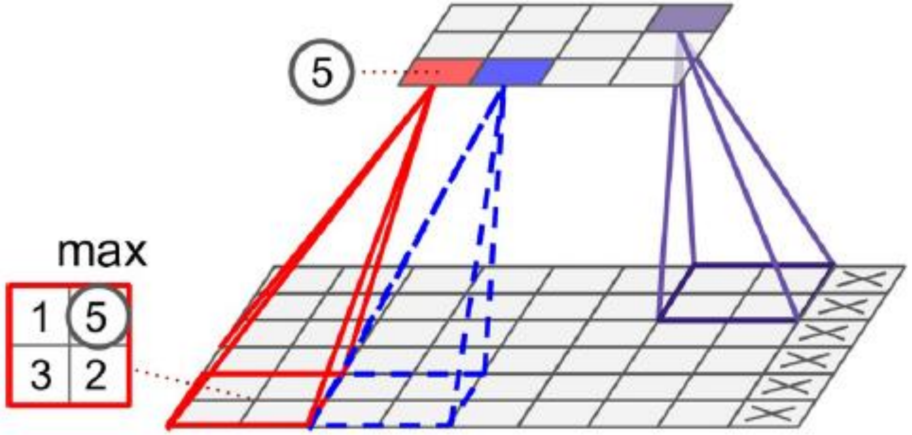
Number of parameters in this layer?

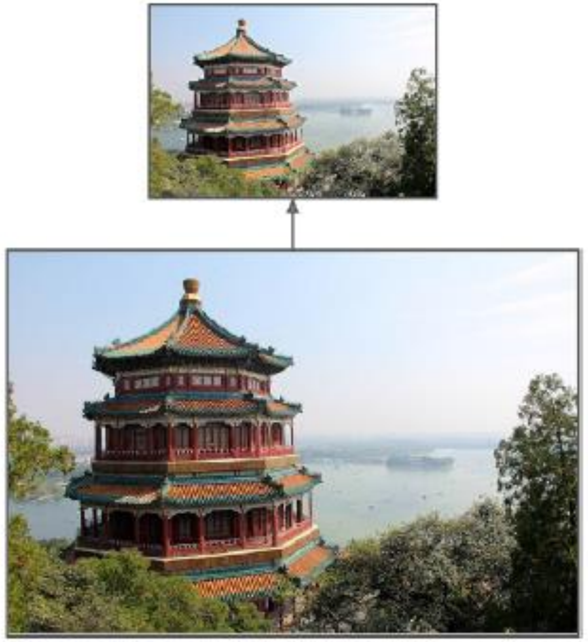


[Source] CS231n: Convolutional Neural Networks for Visual Recognition

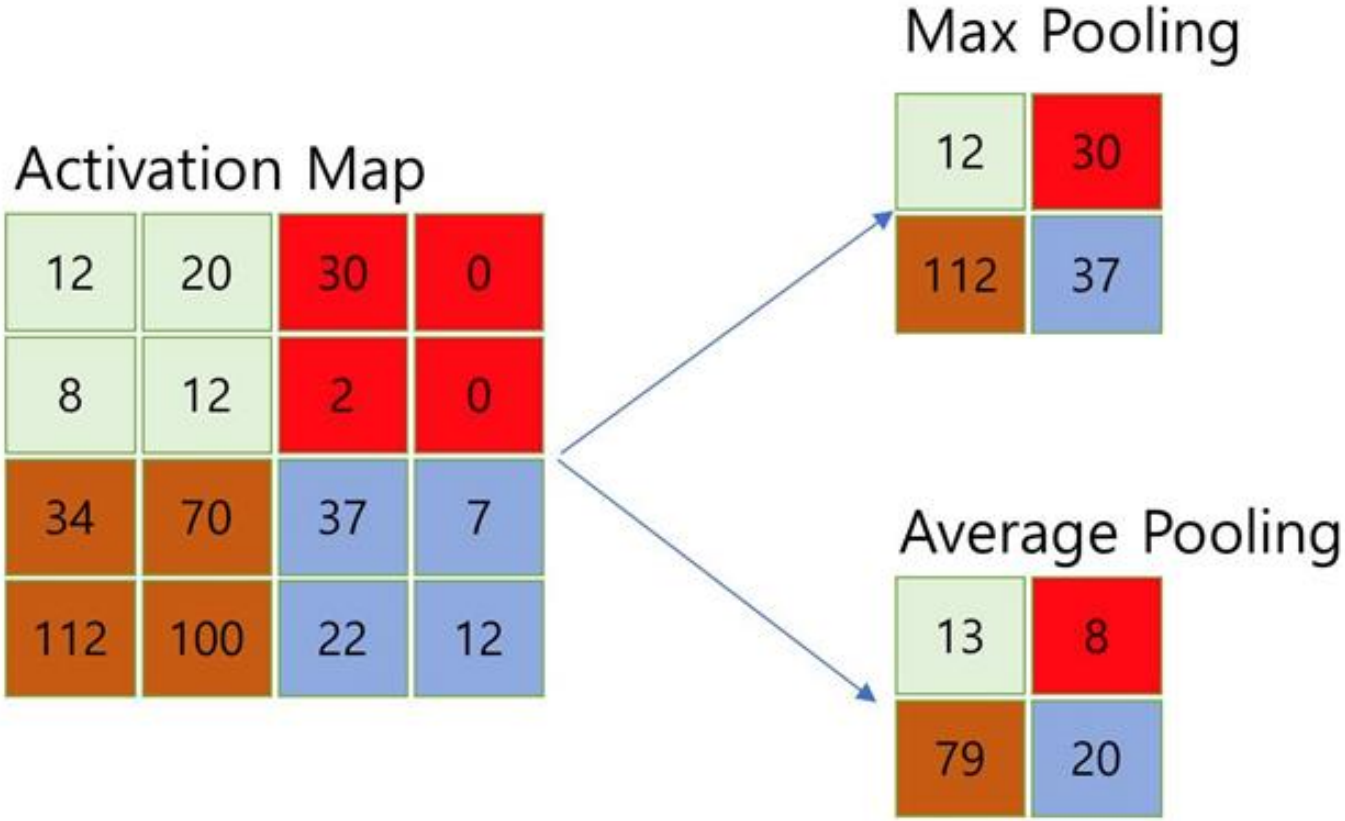
Pooling layer

- *Subsampling* (i.e., shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters (thereby limiting the risk of overfitting).
- Reducing the input image size also makes the neural network tolerate a little bit of image shift (*location invariance*).
- You define its size, the stride, and the padding type. However, a pooling neuron has no weights





Ex) Pooling



Activation Map

12	20	30	0
8	12	2	0
34	70	37	7
112	100	22	12

Max Pooling

12	30
112	37

Average Pooling

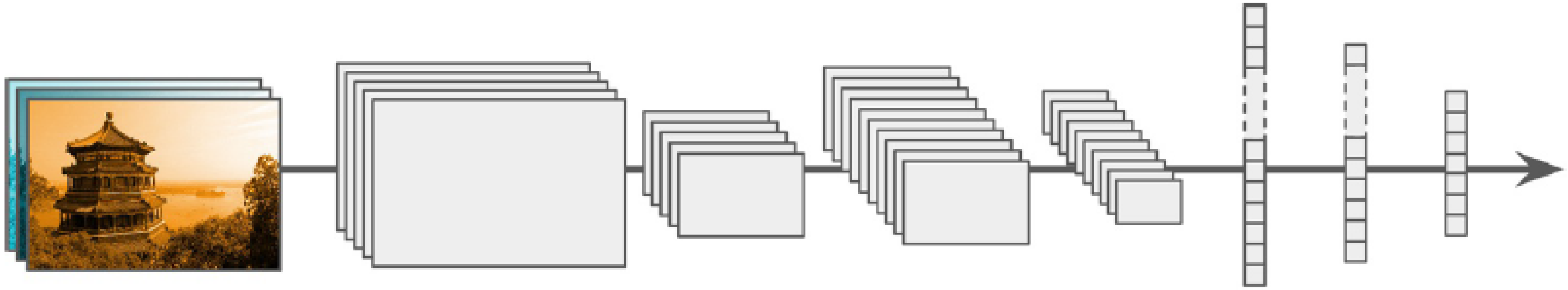
13	8
79	20

source: http://taewan.kim/post/cnn/

Putting it all together

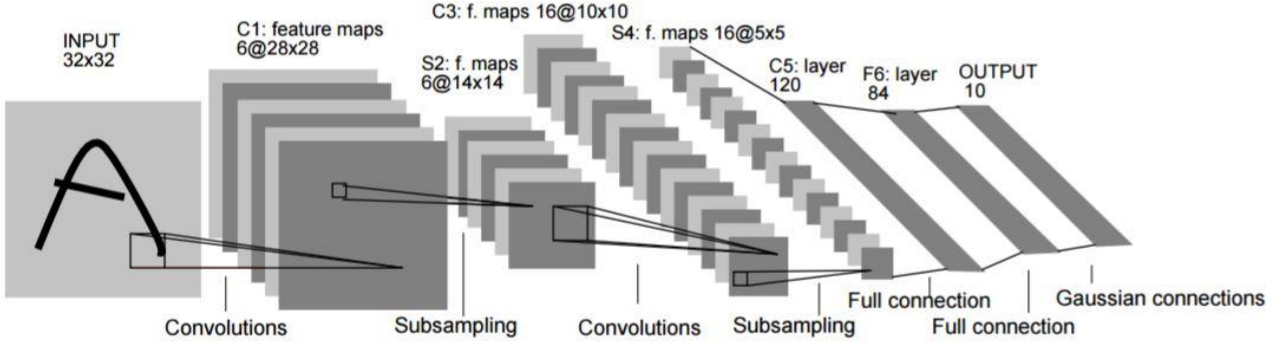
ConvNets stack CONV, POOL, FC layers

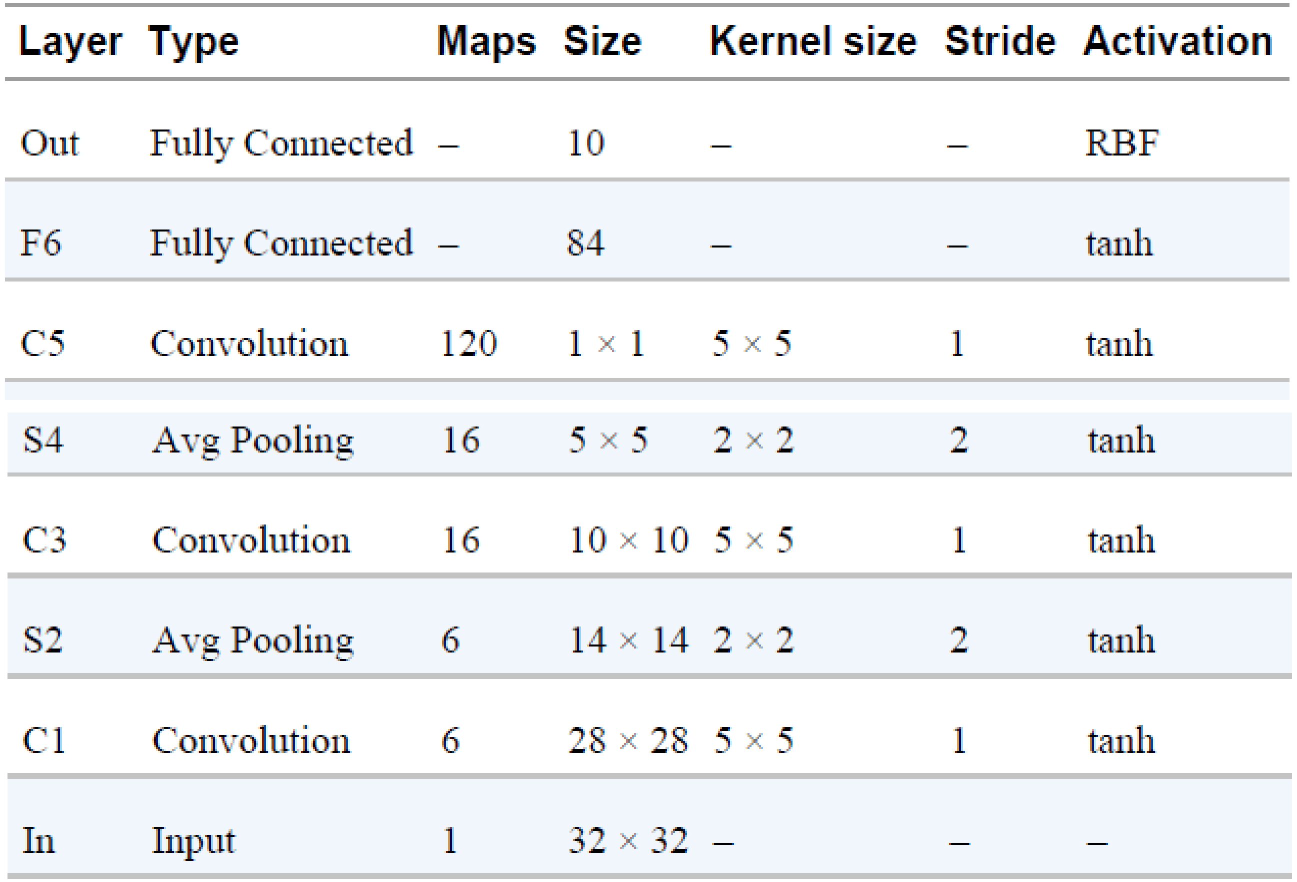
- Typical CNN architectures stack a few convolutional layers, then a pooling layer, then another few convolutional layers (+ReLU), then another pooling layer, and so on.
  - The image gets smaller and smaller as it progresses through the network, but it also typically gets deeper and deeper
- At the top of the stack, a regular feedforward neural network is added, composed of a few fully connected layers (+ReLUs), and the final layer outputs the prediction (e.g., Softmax layer)



LeNet-5

- It was created by Yann LeCun in 1998 and widely used for handwritten digit recognition (MNIST).
- architecture is [CONV-POOL-CONV-POOL-FC-FC]

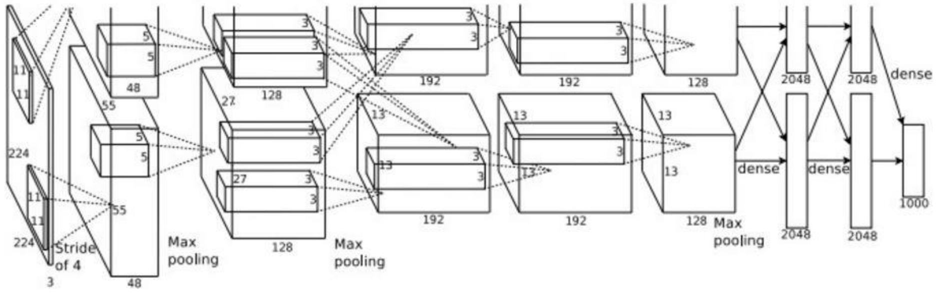


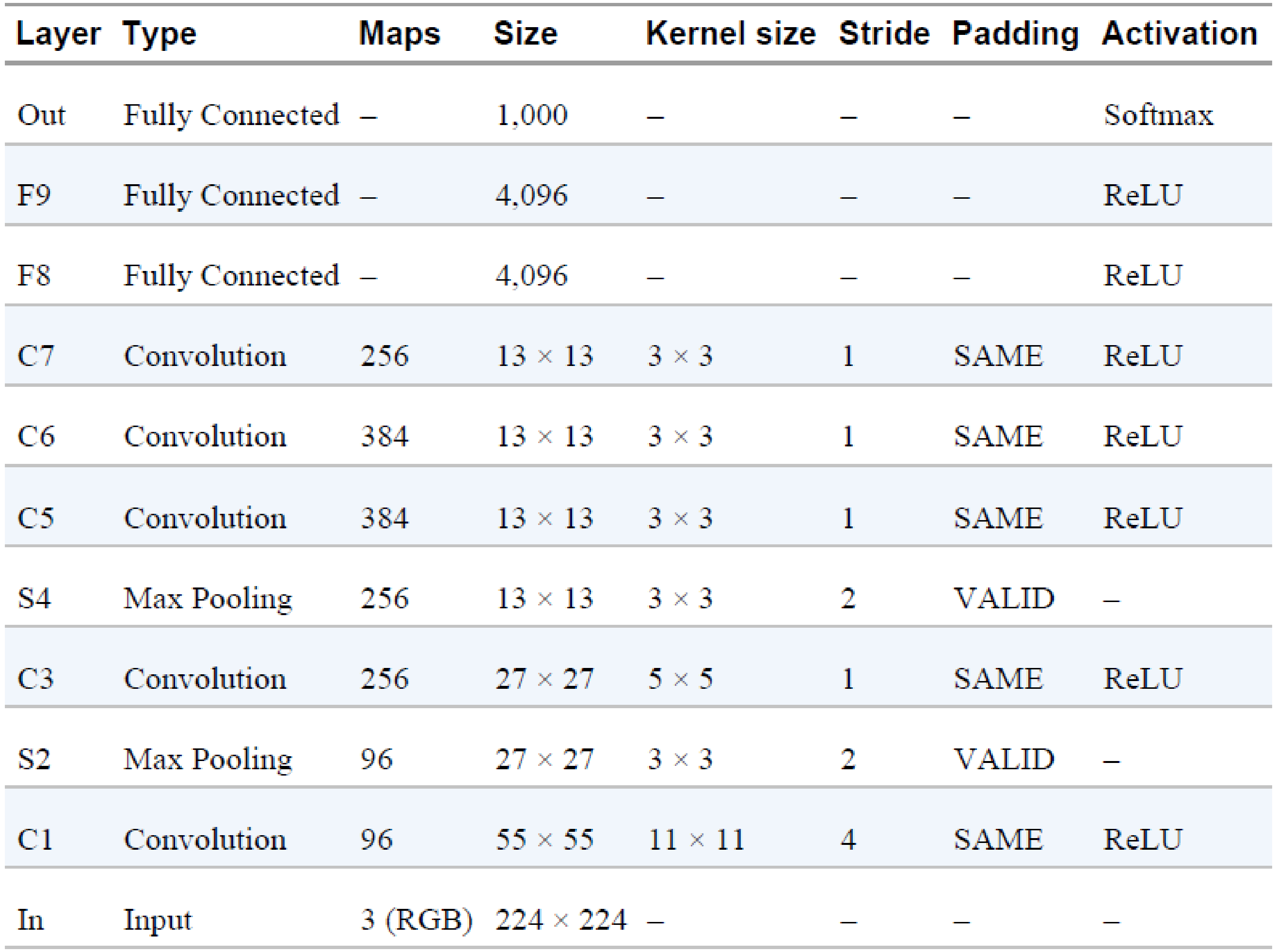


Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg Pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg Pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–

AlexNet

- It was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton.
- It is quite similar to LeNet-5, only much larger and deeper, and it was the first to stack convolutional layers directly on top of each other
- Other characteristics:
  - first use of ReLU
  - heavy data augmentation
  - dropout 0.5

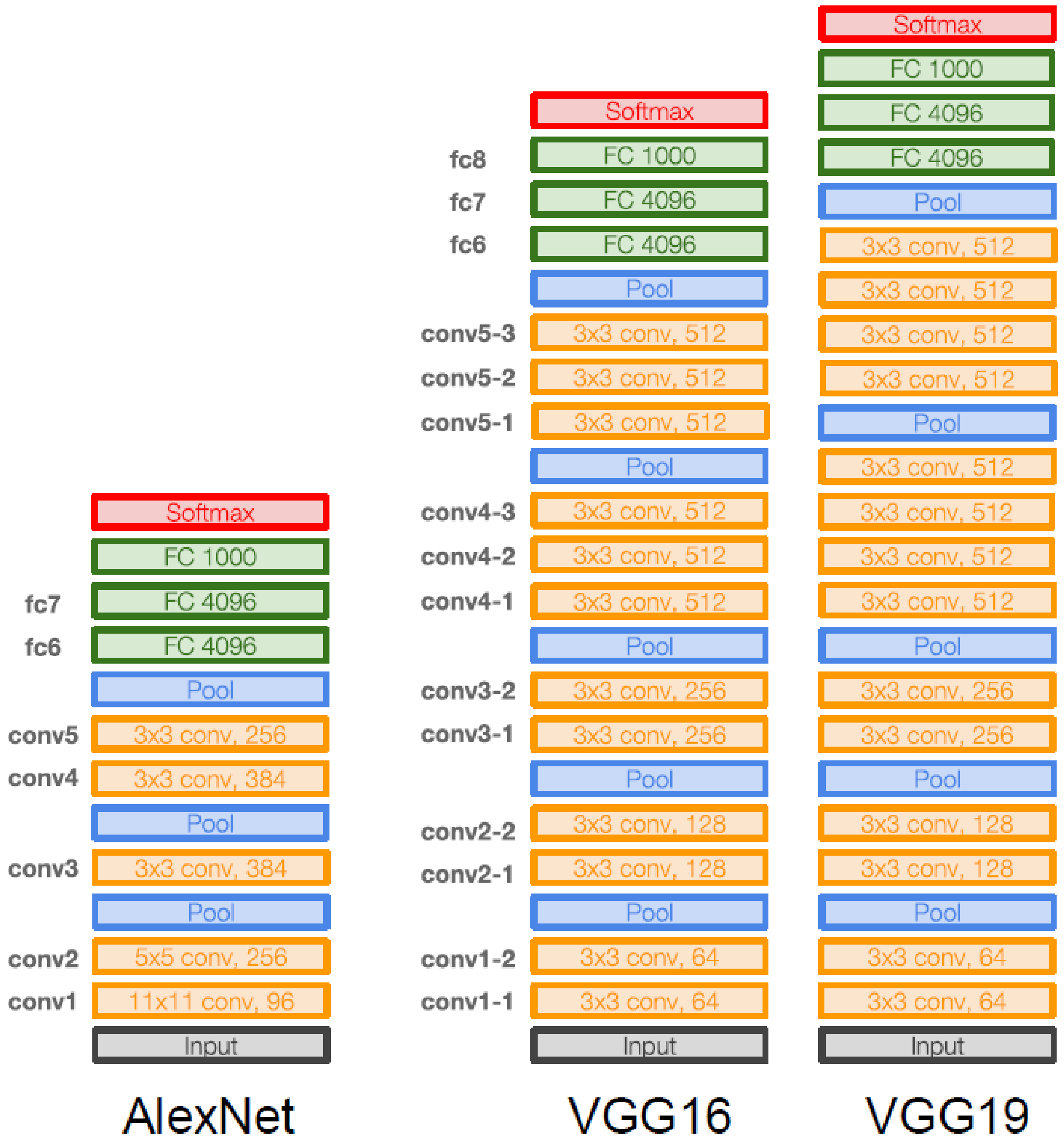




Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13 × 13	3 × 3	1	SAME	ReLU
C6	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
C5	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
S4	Max Pooling	256	13 × 13	3 × 3	2	VALID	–
C3	Convolution	256	27 × 27	5 × 5	1	SAME	ReLU
S2	Max Pooling	96	27 × 27	3 × 3	2	VALID	–
C1	Convolution	96	55 × 55	11 × 11	4	SAME	ReLU
In	Input	3 (RGB)	224 × 224	–	–	–	–

VGGNet

- ILSVRC'14 2nd in classification
- Small filters, deeper networks
- 8 layers (AlexNet)
  -> 16 - 19 layers
- Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2
- Use ensembles for best results



Softmax
FC 1000
fc7 FC 4096
fc6 FC 4096
Pool
conv5 3x3 conv, 256
conv4 3x3 conv, 384
Pool
conv3 3x3 conv, 384
Pool
conv2 5x5 conv, 256
conv1 11x11 conv, 96
Input

AlexNet

Softmax
fc8 FC 1000
fc7 FC 4096
fc6 FC 4096
Pool
conv5-3 3x3 conv, 512
conv5-2 3x3 conv, 512
conv5-1 3x3 conv, 512
Pool
conv4-3 3x3 conv, 512
conv4-2 3x3 conv, 512
conv4-1 3x3 conv, 512
Pool
conv3-2 3x3 conv, 256
conv3-1 3x3 conv, 256
Pool
conv2-2 3x3 conv, 128
conv2-1 3x3 conv, 128
Pool
conv1-2 3x3 conv, 64
conv1-1 3x3 conv, 64
Input

VGG16

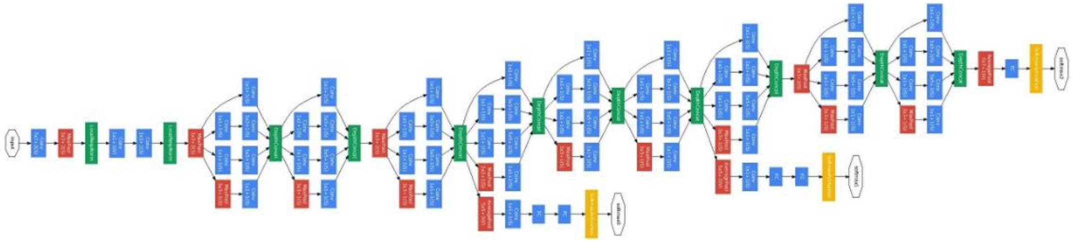
Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input

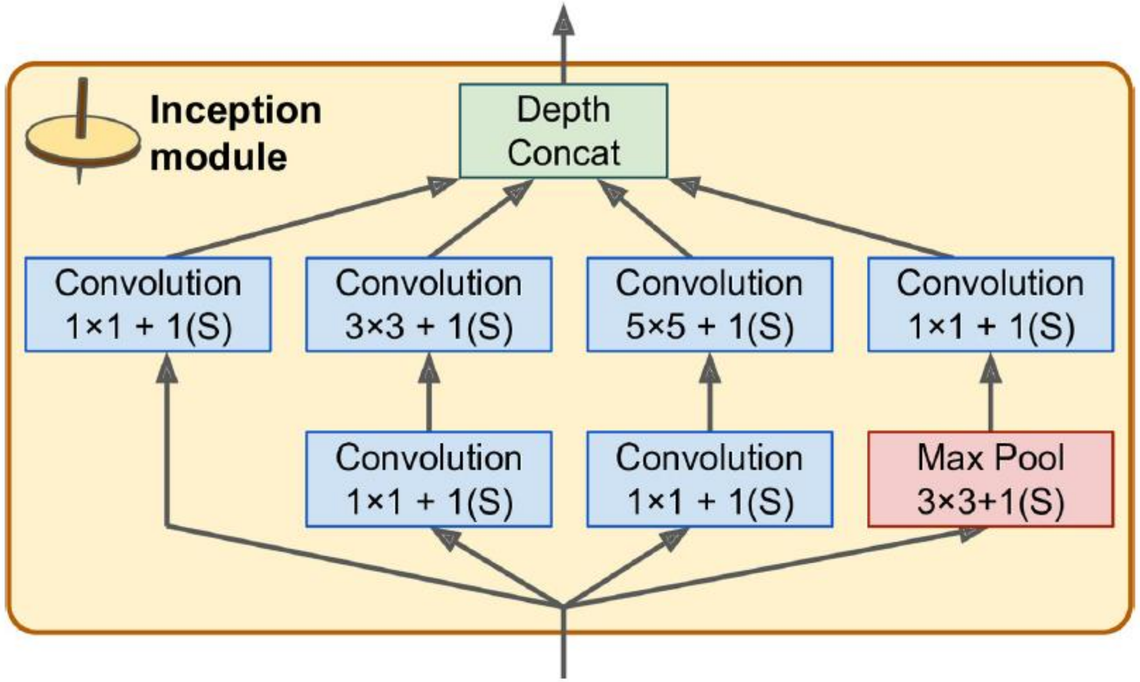
VGG19

source: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture09.pdf

GoogLeNet

- ILSVRC'14 classification winner (6.7% top 5 error)
- Deeper networks, with computational efficiency
- 22 layers with efficient "Inception" module, and without FC layers
- *Inception modules*
  - design a good local network topology (network within a network) and then stack these modules on top of each other
  - apply parallel filter operations on the input from previous layer, and concatenate all filter outputs together depth-wise
  - use 1x1 filters to reduce feature depth (bottleneck layers)
  - 10 times fewer parameters than AlexNet





Inception module

Depth Concat

Convolution 1×1 + 1(S)

Convolution 3×3 + 1(S)

Convolution 5×5 + 1(S)

Convolution 1×1 + 1(S)

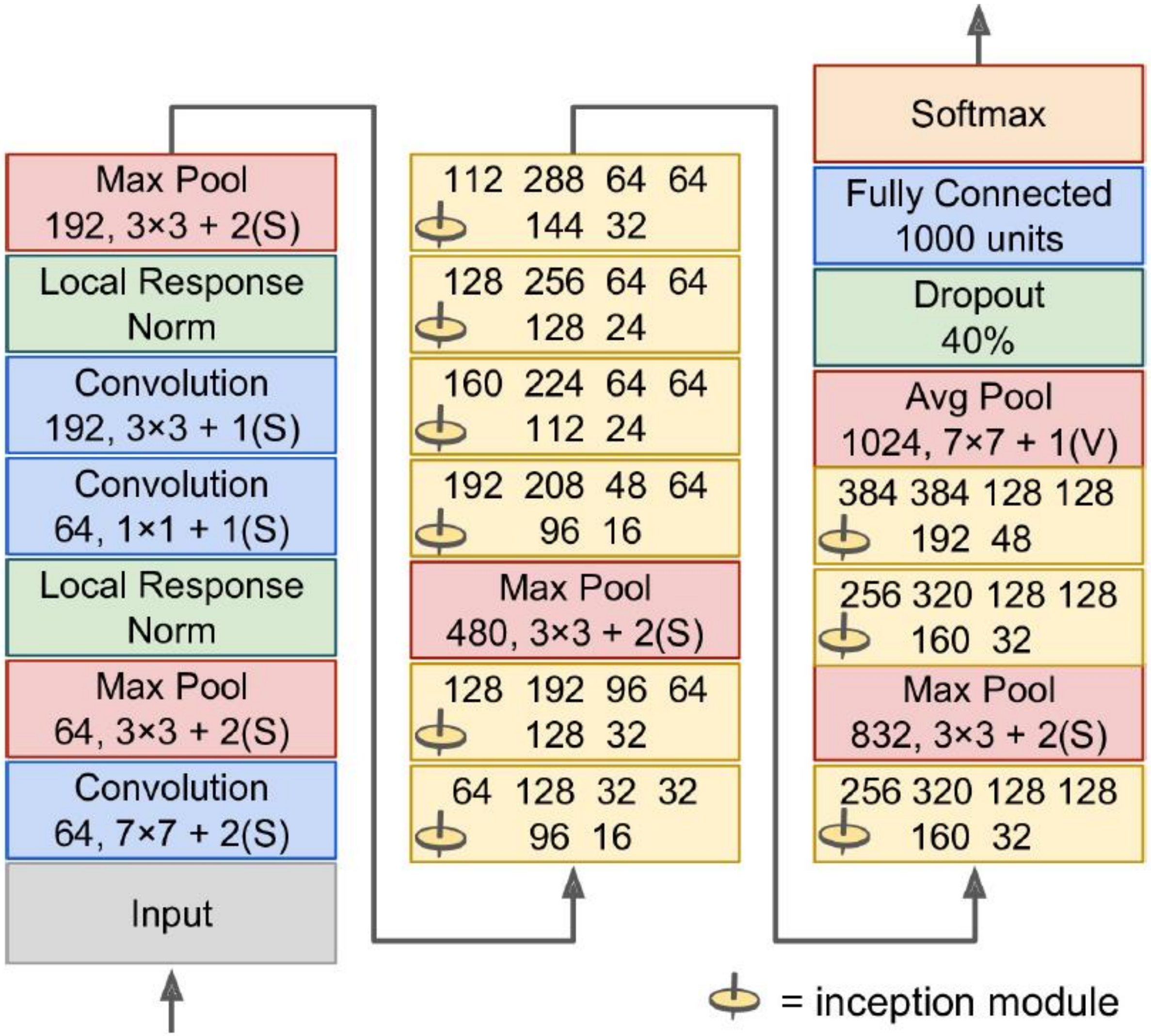
Convolution 1×1 + 1(S)

Convolution 1×1 + 1(S)

Max Pool 3×3+1(S)

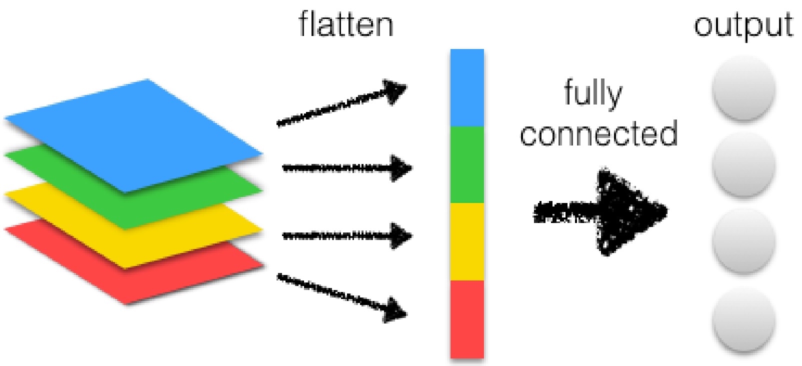
GoogLeNet

- GoogLeNet has 9 inception modules
- Full architecture: Conv-Poop-2x Conv-Pool ⇨ Stacked Inception Modules ⇨ Classifier output (removed expensive FC layers)
- All the convolutional layers use the ReLU activation function.
- *Global average pooling*
  - After generating one feature map for each corresponding class labels in the last conv layer, the average of each feature map is fed directly into the softmax layer.
  - This makes it unnecessary to have several fully connected layers at the top of the CNN (Most of the parameters in CNN models belong to the fully connected layers!)

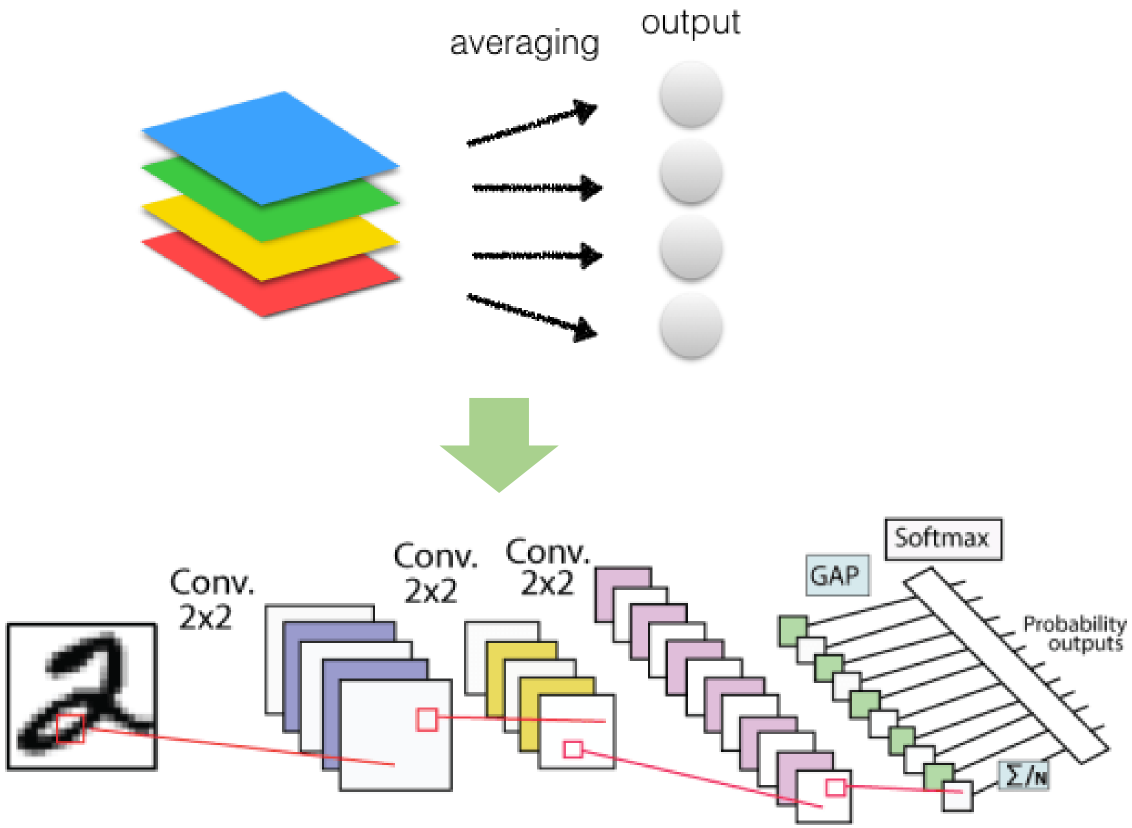


FC Layer vs. GAP

Fully Connected Layer



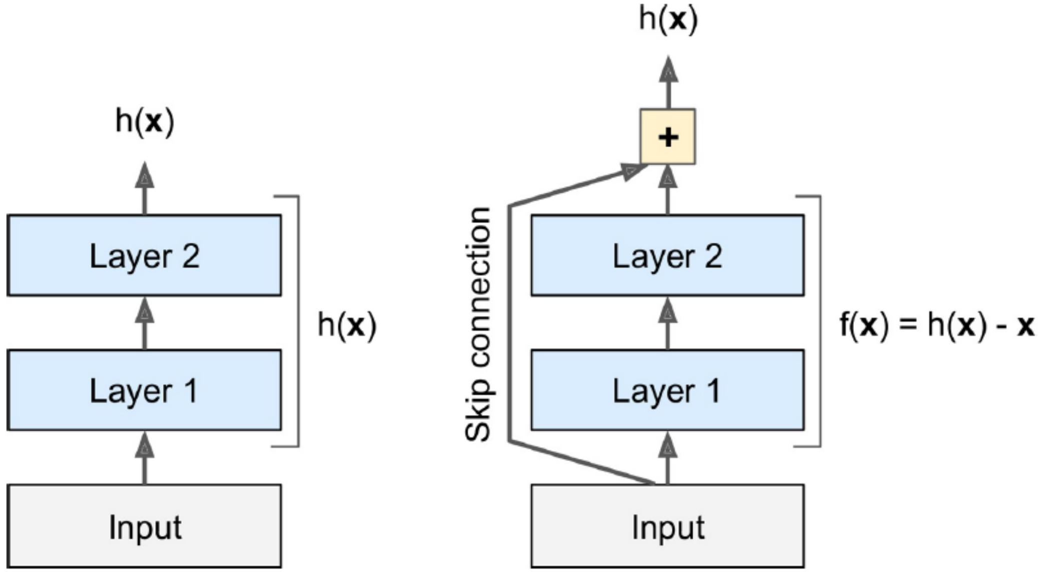
Global Average Pooling



Source: https://elix-tech.github.io/ja/2017/02/06/gan.html
Source: http://principlesofdeeplearning.com/index.php/a-tutorial-on-global-average-pooling/

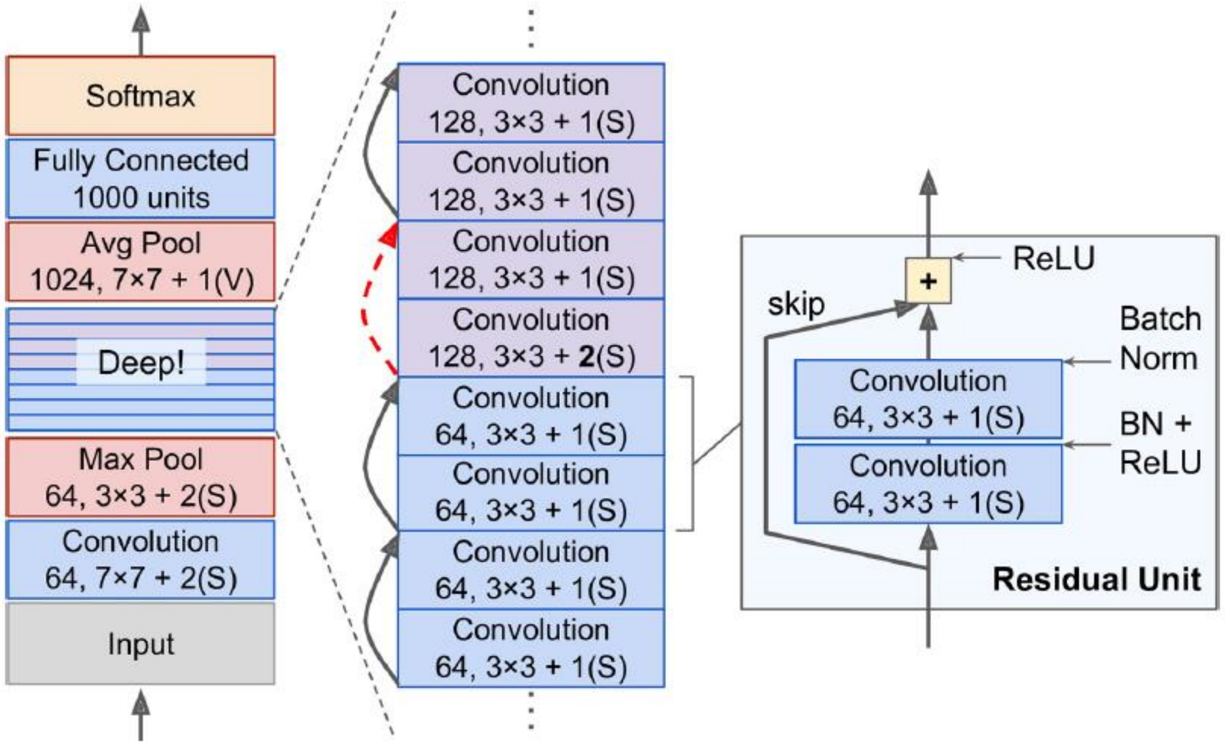
ResNet

- ILSVRC'15 classification winner (3.57% top 5 error)
- Very deep networks using residual connections: 152-layer model
- *residual learning*
  - When training a neural network, the goal is to make it model a target function *h*(**x**).
  - If you add the input **x** to the output of the network (i.e., you add a skip connection), then the network will be forced to model *f*(**x**) = *h*(**x**) – **x** rather than *h*(**x**).



ResNet

- It starts and ends exactly like GoogLeNet (except without a dropout layer), and in between is just a very deep stack of simple residual units.
- Each residual unit is composed of two convolutional layers, with Batch Normalization (BN) and ReLU activation, using 3 × 3 kernels and preserving spatial dimensions (stride 1, SAME padding)



Softmax
Fully Connected 1000 units
Avg Pool 1024, 7×7 + 1(V)
Deep!
Max Pool 64, 3×3 + 2(S)
Convolution 64, 7×7 + 2(S)
Input
Convolution 128, 3×3 + 1(S)
Convolution 128, 3×3 + 1(S)
Convolution 128, 3×3 + 1(S)
Convolution 128, 3×3 + 2(S)
Convolution 64, 3×3 + 1(S)
Convolution 64, 3×3 + 1(S)
Convolution 64, 3×3 + 1(S)
Convolution 64, 3×3 + 1(S)
ReLU
+
skip
Batch Norm
Convolution 64, 3×3 + 1(S)
BN + ReLU
Convolution 64, 3×3 + 1(S)
Residual Unit

Summary: CNN architectures

- Layer Patterns
  - The most common form:

    INPUT ⇨ [[CONV ⇨ RELU]*N ⇨ POOL?]*M ⇨ [FC ⇨ RELU]*K ⇨ SOFTMAX

    *where the * indicates repetition, and the POOL? indicates an optional pooling layer. Moreover, N is usually up to ~5, M is large, 0 <= K <= 2.*

  - Trend towards smaller filters and deeper architectures

    stacking CONV layers with tiny filters as opposed to having one CONV layer with big filters allows us to express more powerful features of the input, and with fewer parameters.

  - Trend towards getting rid of POOL/FC layers (just CONV)
  - use whatever works best on ImageNet

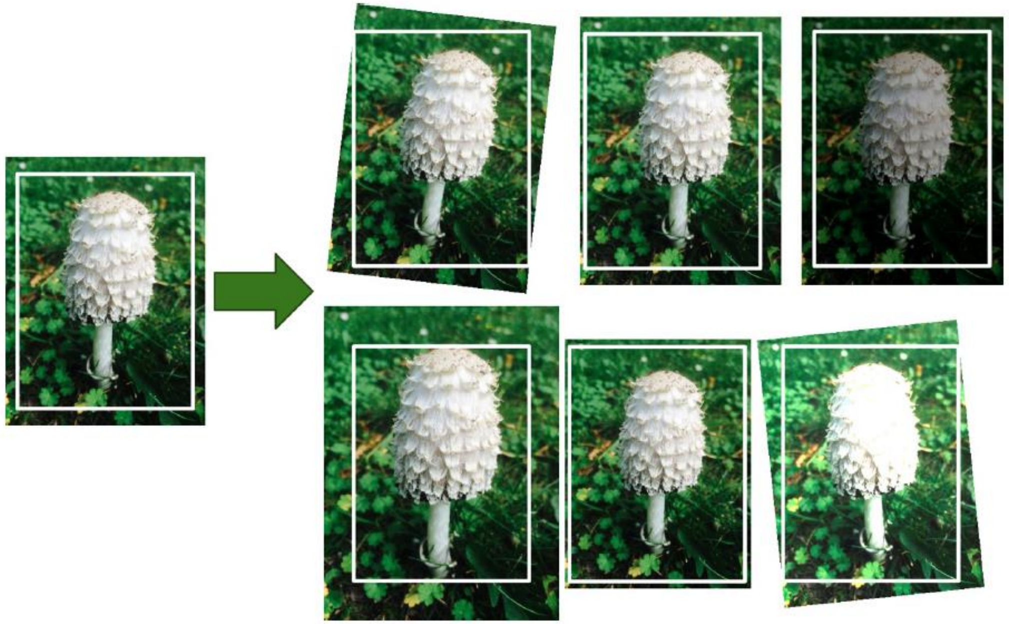
    Instead of rolling your own architecture for a problem, you should look at whatever architecture currently works best on ImageNet, download a pretrained model and fine-tune it on your data.

Summary: CNN architectures

- Layer Sizing Patterns
  - **Input layer** should be divisible by 2 many times. Common numbers include 32 (e.g. CIFAR-10), 64, 96 (e.g. STL-10), or 224 (e.g. common ImageNet ConvNets), 384, and 512.
  - **Convolution layers** should be using small filters (e.g. 3x3 or at most 5x5), using a stride of S=1, and crucially, padding the input volume with zeros
  - The most common setting of **pooling layers** is to use max-pooling with 2x2 receptive fields (i.e. F=2), and with a stride of 2 (i.e. S=2).

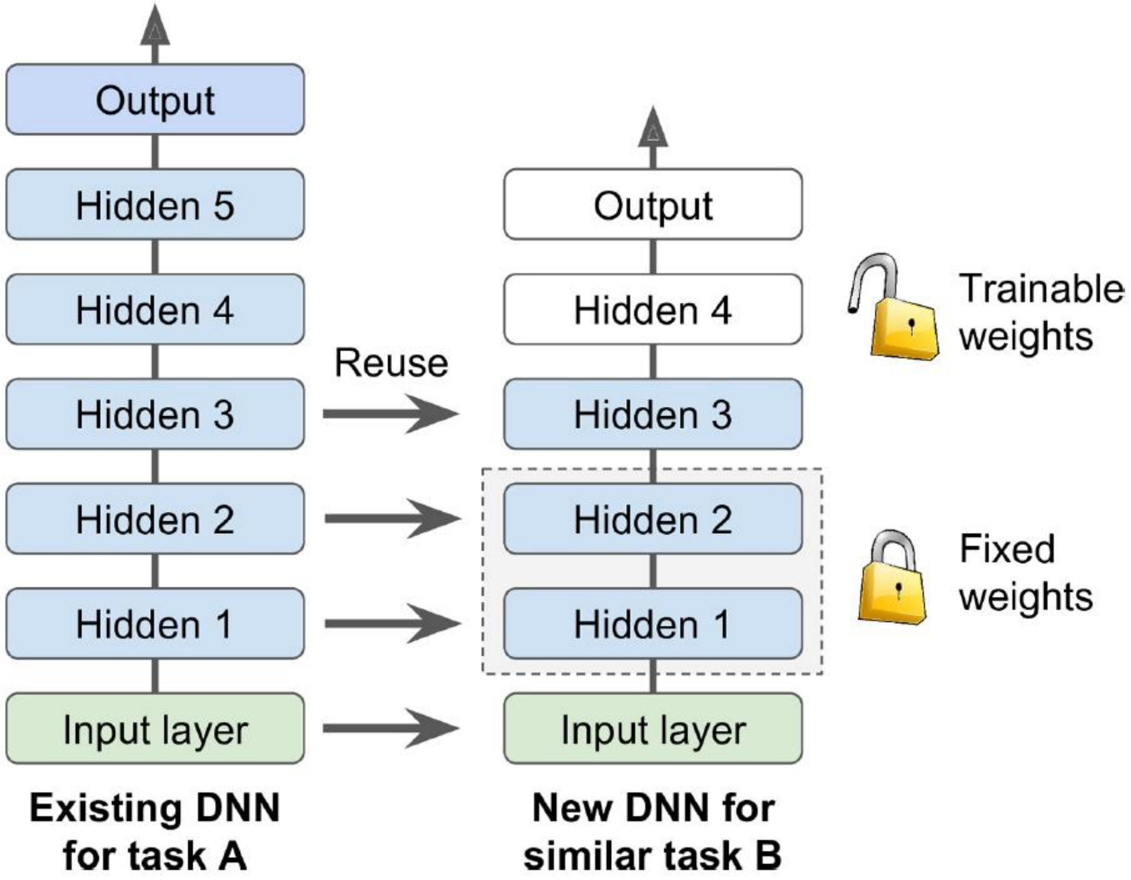
Data augmentation

- It consists of generating new training instances from existing ones, artificially boosting the size of the training set
  - The trick is to generate realistic training instances; ideally, a human should not be able to tell which instances were generated and which ones were not.
  - The modifications you apply should be learnable (white noise is not).
  - For example, if your model is meant to classify pictures of mushrooms, you can slightly shift, rotate, and resize every picture in the training set by various amounts and add the resulting pictures to the training set
  - If you want the model to be more tolerant to lighting conditions, you can similarly generate many images with various contrasts
- Keras offers several image manipulation operations



Reusing pretrained layers

- *transfer learning*
  - to find an existing neural network that accomplishes a similar task to the one you are trying to tackle, then just reuse the lower layers of this network
  - There are two ways: feature extraction and fine-tuning

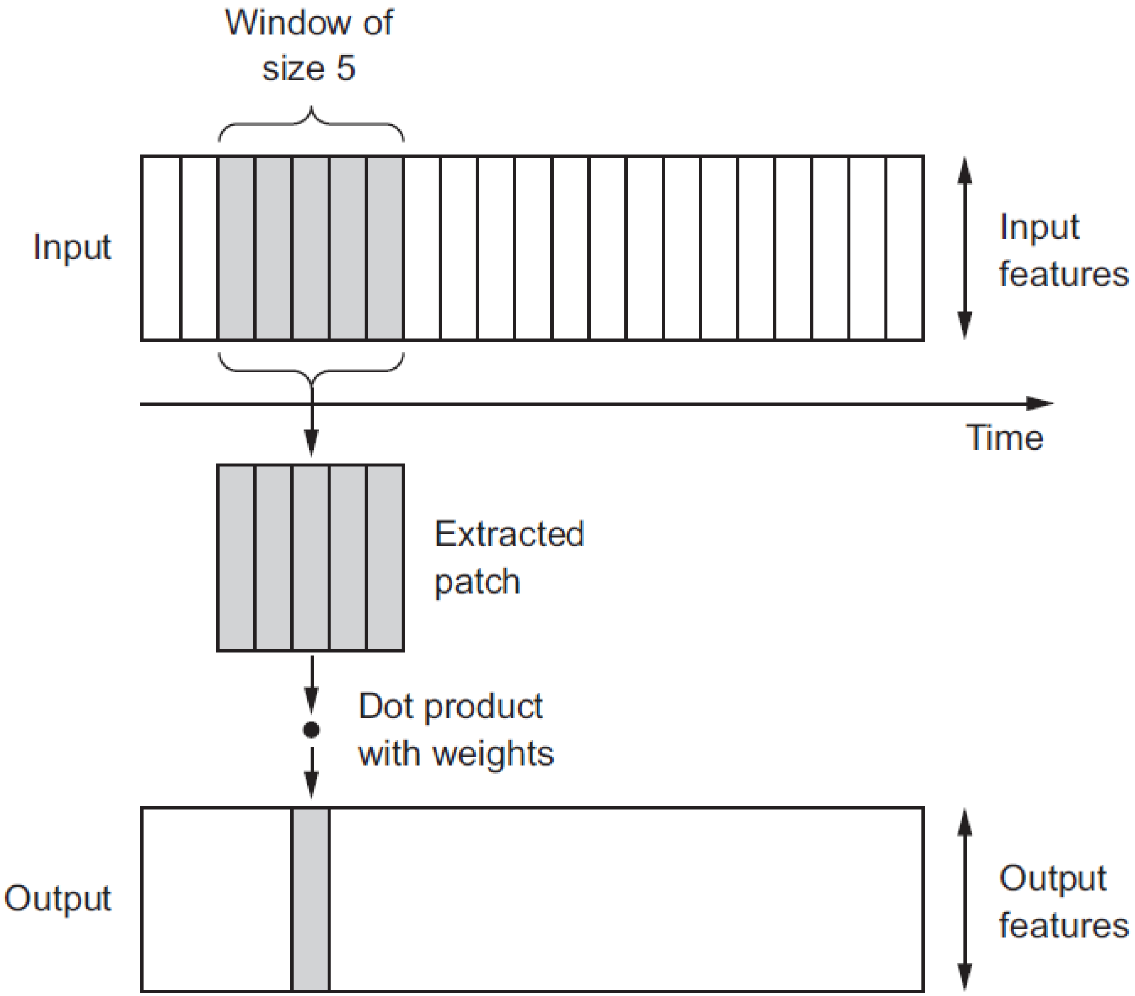


CNN structure with Keras

- Input shape
  - (image_height, image_width, image_channels)

- Layers
  - keras.layers.Conv2D(filters=32, kernel_size=(3,3), strides=(1,1), padding="valid", activation="relu", input_shape=(28,28,3))
  - keras.layers.MaxPooling2D(pool_size=(2,2))
  - keras.layers.Flatten()
  - keras.layers.GlobalMaxPooling2D()

1D convolution for sequence data



- keras.layers.Embedding(input_dim=10000, output_dim=128, input_length=21)
- keras.layers.Conv1D(filters=32, kernel_size=5, strides=1, activation="relu")
- keras.layers.MaxPooling1D(pool_size=3)
- keras.layers.GlobalMaxPooling1D()

References

- Hands-On Machine Learning with Scikit-Learn & TensorFlow,
  Aurelien Geron, O'REILLY, 2017.
- Deep Learning with Python,
  Francois Chollet, MANNING, 2017.
- CS231n: Convolutional Neural Networks for Visual Recognition,
  Lecture 5 & 9
- Convolutional neural networks,
  Abin – Roozgard
- Deep Learning from Scratch,
  Hanbit Publishing, 2017