Convolutional Neural Networks

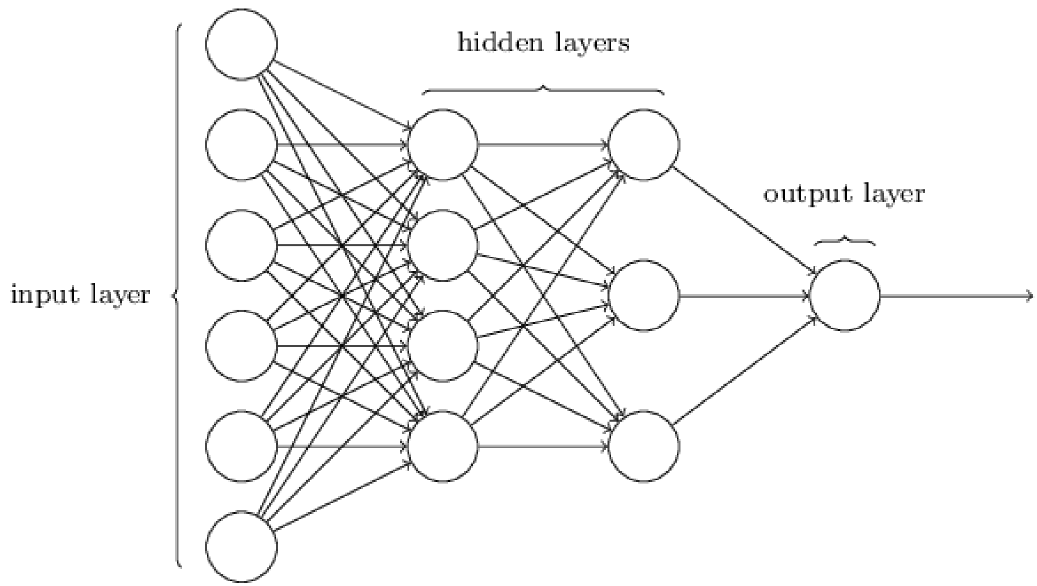
ESM5205 Learning from Big Data | Oct 30, 2019

Seokho Kang

성균관대학교
SUNG KYUN KWAN UNIVERSITY(SKKU)

Feed-forward Neural Networks

- **An FNN is a stack of fully-connected layers.**



how many parameters in a neural network?

what does the input look like?

what if the input is an image of 256x256 pixels

Multidimensional Arrays

- **Scalar: a single number,** $a \in \mathbb{R}$
  - Integers, real numbers, rational numbers, etc.

- **Vector: a 1-D array of numbers,** $\boldsymbol{x} \in \mathbb{R}^n$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- **Matrix: a 2-D array of numbers,** $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

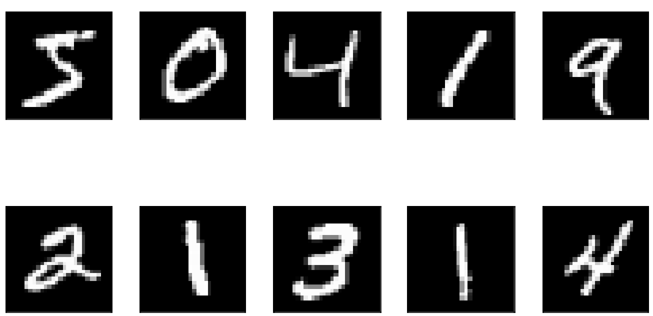
Multidimensional Arrays

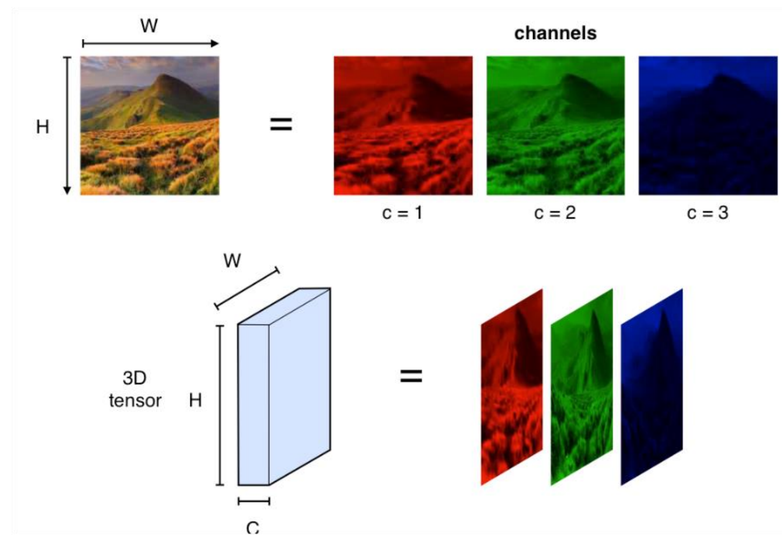
- **Multidimensional Arrays of Numbers**
  - **Scalar → 0-D Array**
  - **Vector → 1-D Array**
  - **Matrix → 2-D Array**
  - **3-D Array, 4-D Array, …**

*a multidimensional array is not necessarily a representation of a tensor

- ***Example:*** a grayscale image is a **2-D Array** / a color image is a **3-D Array**

(or, a **2-D Array** with **3 channels**)





Multidimensional Arrays

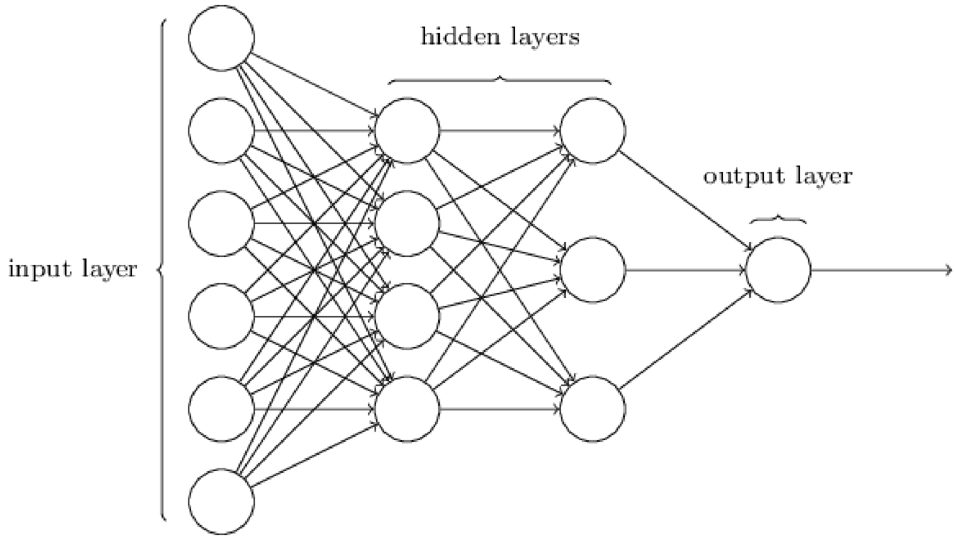
Example

	Single channel	Multi-channel
1-D	Audio waveform: The axis we convolve over corresponds to time. We discretize time and measure the amplitude of the waveform once per time step.	Skeleton animation data: Animations of 3-D computer-rendered characters are generated by altering the pose of a "skeleton" over time. At each point in time, the pose of the character is described by a specification of the angles of each of the joints in the character's skeleton. Each channel in the data we feed to the convolutional model represents the angle about one axis of one joint.
2-D	Audio data that has been preprocessed with a Fourier transform: We can transform the audio waveform into a 2D tensor with different rows corresponding to different frequencies and different columns corresponding to different points in time. Using convolution in the time makes the model equivariant to shifts in time. Using convolution across the frequency axis makes the model equivariant to frequency, so that the same melody played in a different octave produces the same representation but at a different height in the network's output.	Color image data: One channel contains the red pixels, one the green pixels, and one the blue pixels. The convolution kernel moves over both the horizontal and vertical axes of the image, conferring translation equivariance in both directions.
3-D	Volumetric data: A common source of this kind of data is medical imaging technology, such as CT scans.	Color video data: One axis corresponds to time, one to the height of the video frame, and one to the width of the video frame.

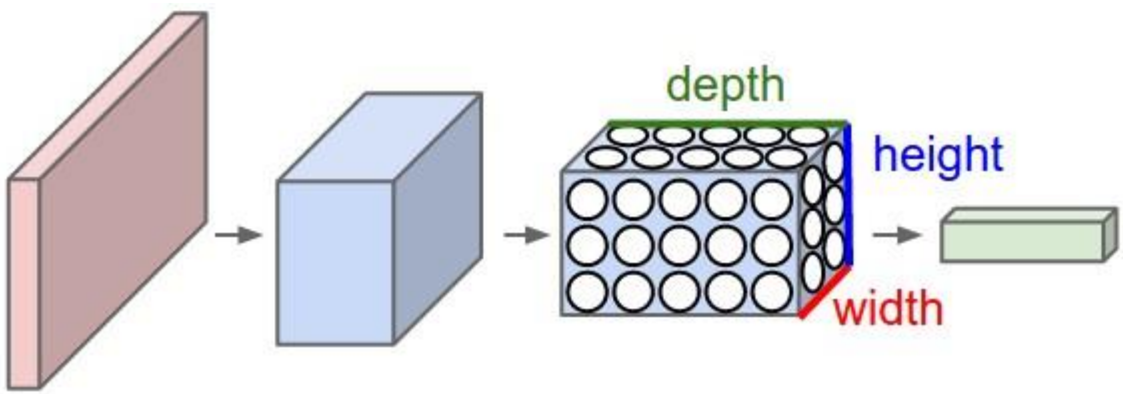
*channels may have no specific ordering (e.g., RGB)

Convolutional Neural Networks

- **Feed-forward Neural Network** (input: vector)

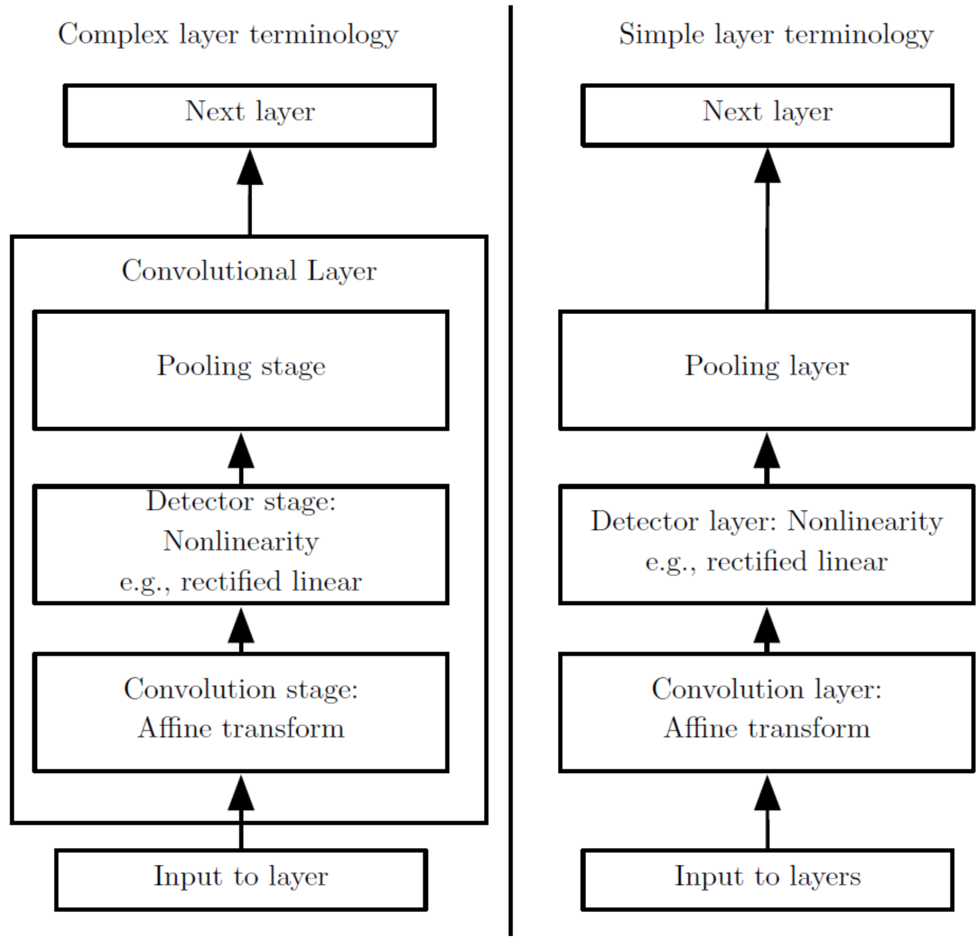


- **Convolutional Neural Network**
  (input: multidimensional array with spatial locality)



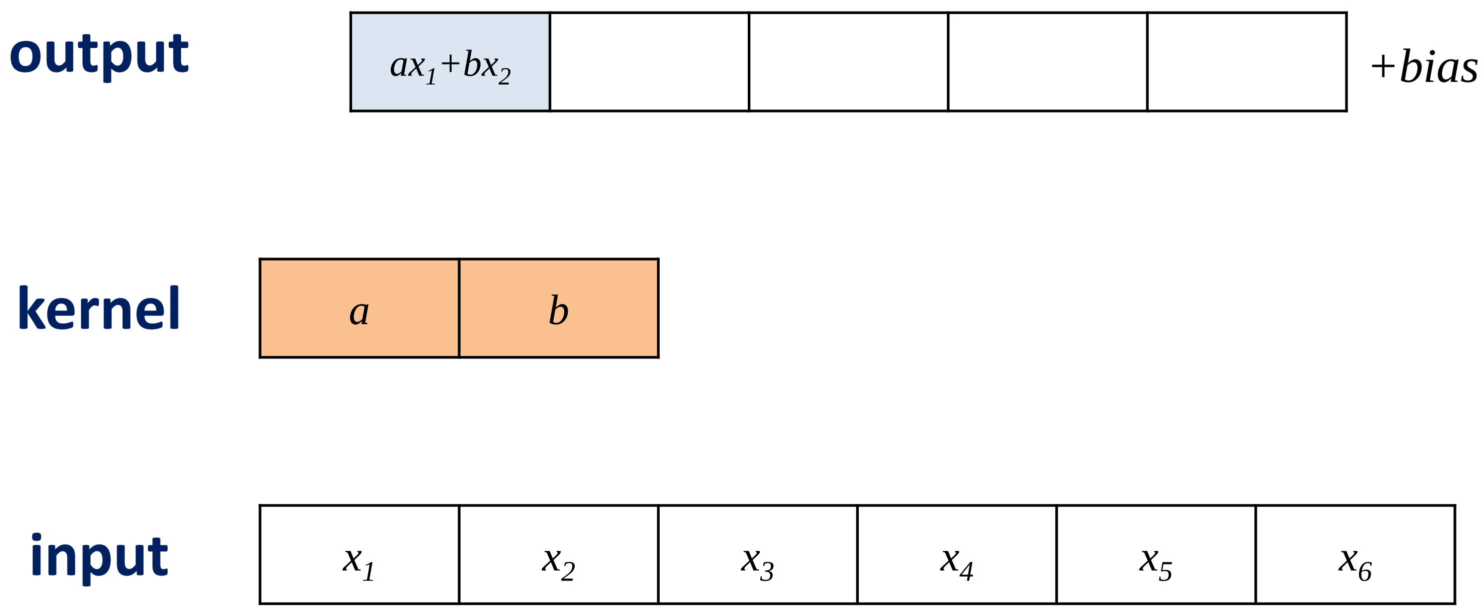
Convolutional Neural Networks

- **Convolutional Neural Networks (CNNs)**
  - **Neural Networks that use convolution in place of general matrix multiplication in at least one of their layers.**
  - **Main Operations:**
    1. Convolution
    2. Detector
    3. Pooling



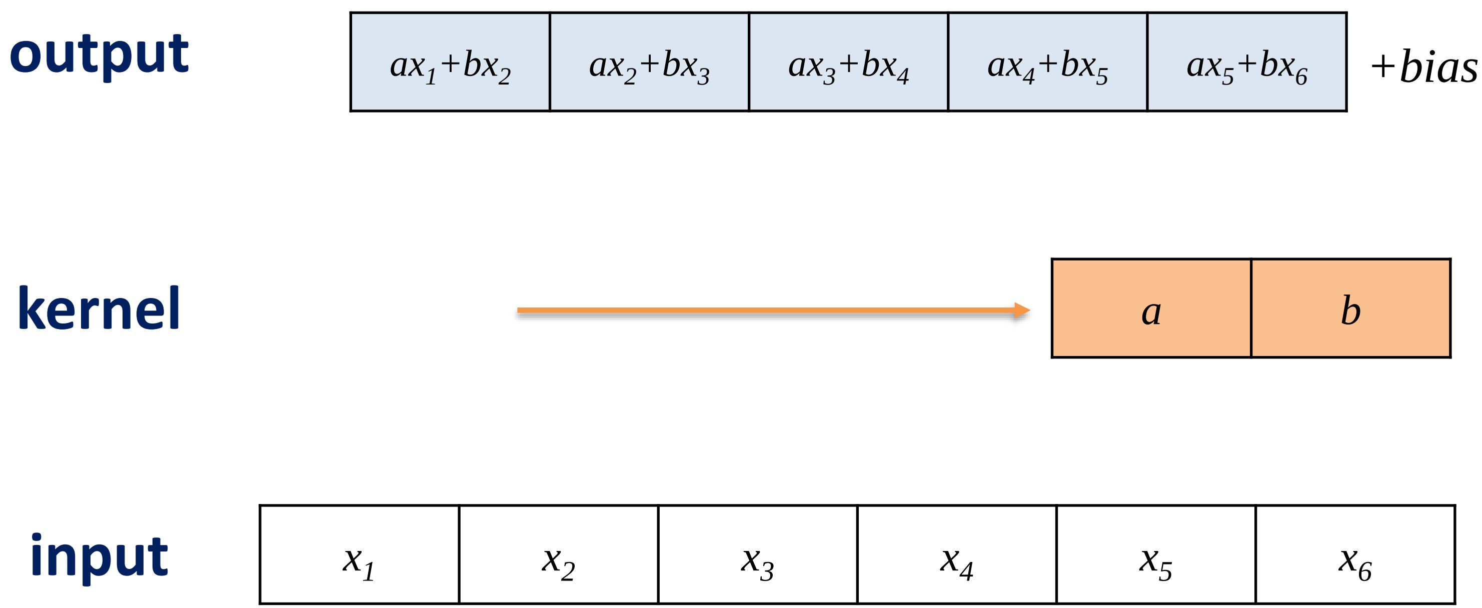
Convolution

- **Convolution with 1-D Array Input**
  - **kernel** = [a,b] ← parameters
    (*a.k.a.* filter)
  - input size m=6, **kernel size** k=2, **stride**(kernel step size) s=1, output size n= (m-k)/s+1=5



Convolution

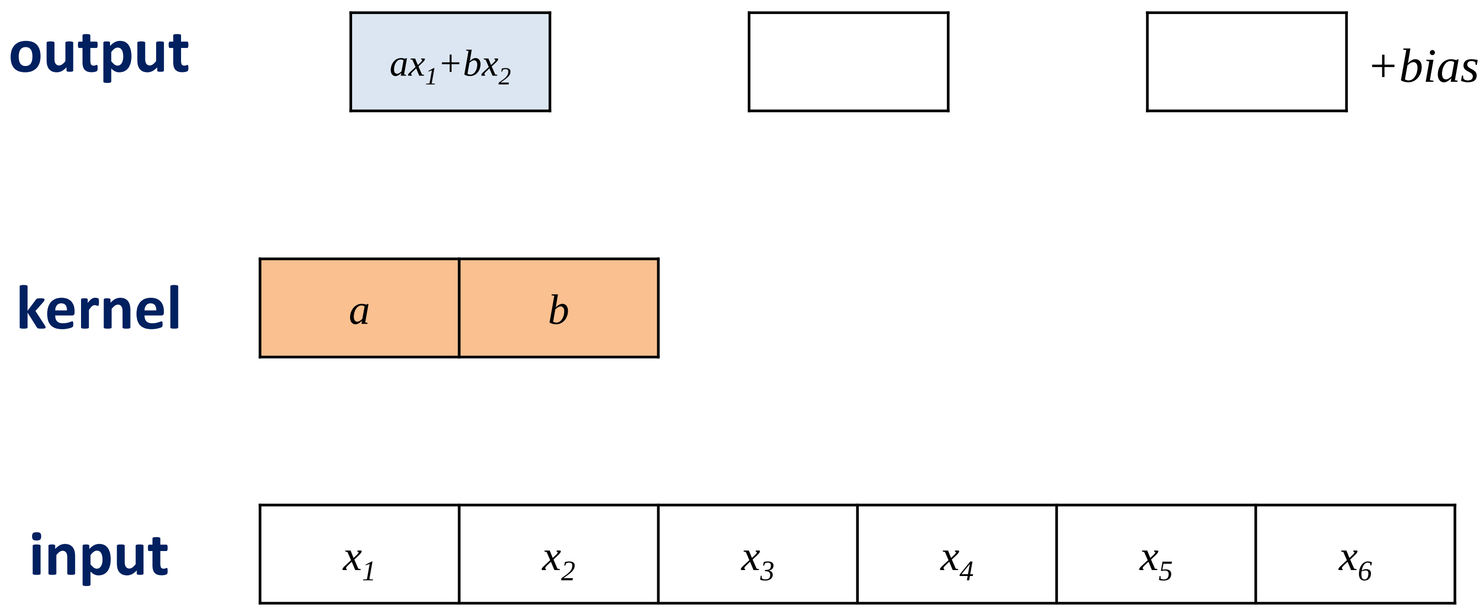
- **Convolution with 1-D Array Input**
  - **kernel** = [a,b] ← parameters
  - input size m=6, kernel size k=2, stride(kernel step size) s=1, output size n= (m-k)/s+1=5



what are parameters?
what are hyperparameters?

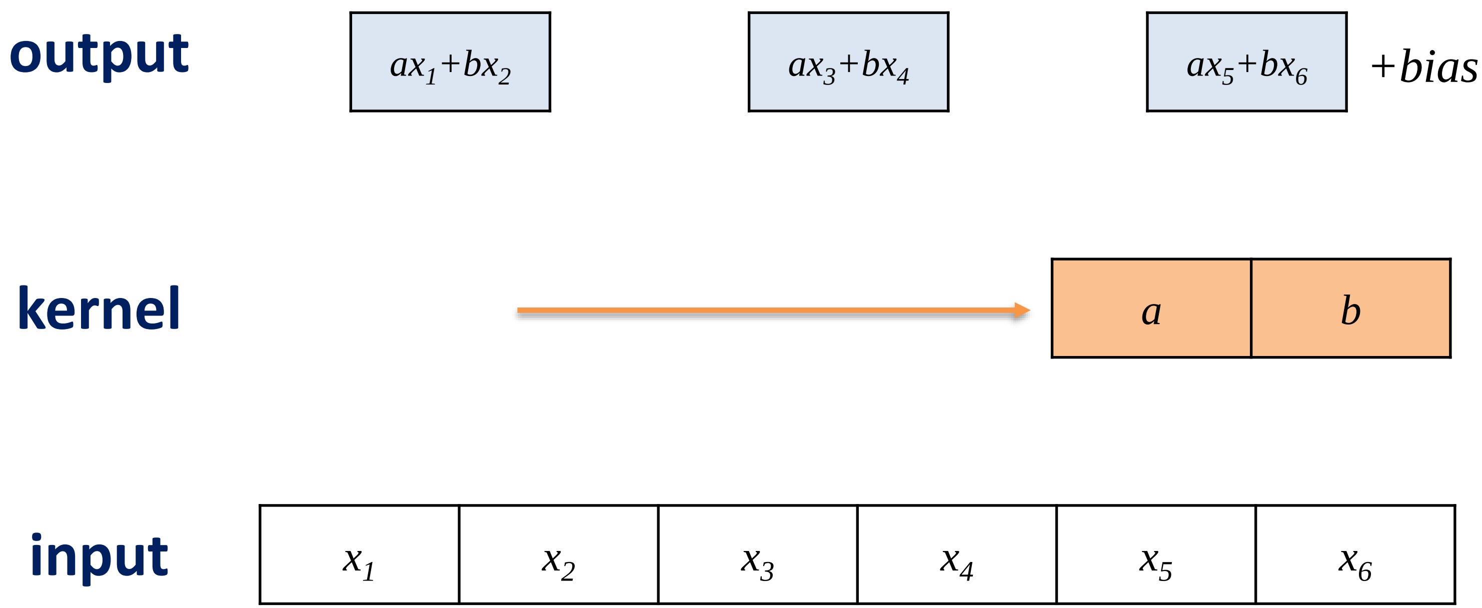
Convolution

- **Convolution with 1-D Array Input**
  - **kernel** = [a,b] ← parameters
  - input size m=6, kernel size k=2, stride(kernel step size) s=2, output size n= (m-k)/s+1=3



Convolution

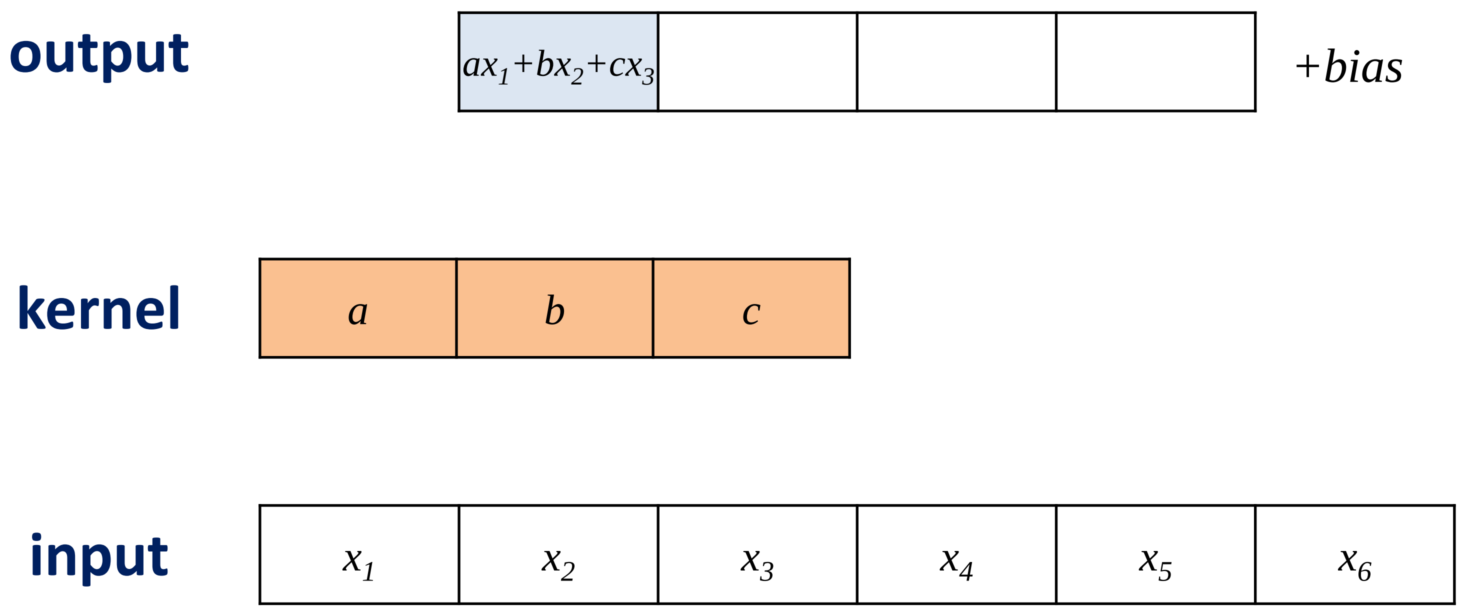
- **Convolution with 1-D Array Input**
  - **kernel** = [a,b] ← parameters
  - input size m=6, kernel size k=2, stride(kernel step size) s=2, output size n= (m-k)/s+1=3



what are parameters?
what are hyperparameters?

Convolution

- **Convolution with 1-D Array Input**
  - **kernel** = [a,b,c] ← parameters
  - input size m=6, kernel size k=3, stride(kernel step size) s=1, output size n= (m-k)/s+1=4



Convolution

- **Convolution with 1-D Array Input**
  - **kernel** = [a,b,c] ← parameters
  - input size m=6, kernel size k=3, stride(kernel step size) s=1, output size n= (m-k)/s+1=4

output

$ax_1+bx_2+cx_3$	$ax_2+bx_3+cx_4$	$ax_3+bx_4+cx_5$	$ax_4+bx_5+cx_6$	$+bias$

kernel

a	b	c

input

x_1	x_2	x_3	x_4	x_5	x_6

what are parameters?
what are hyperparameters?

Convolution

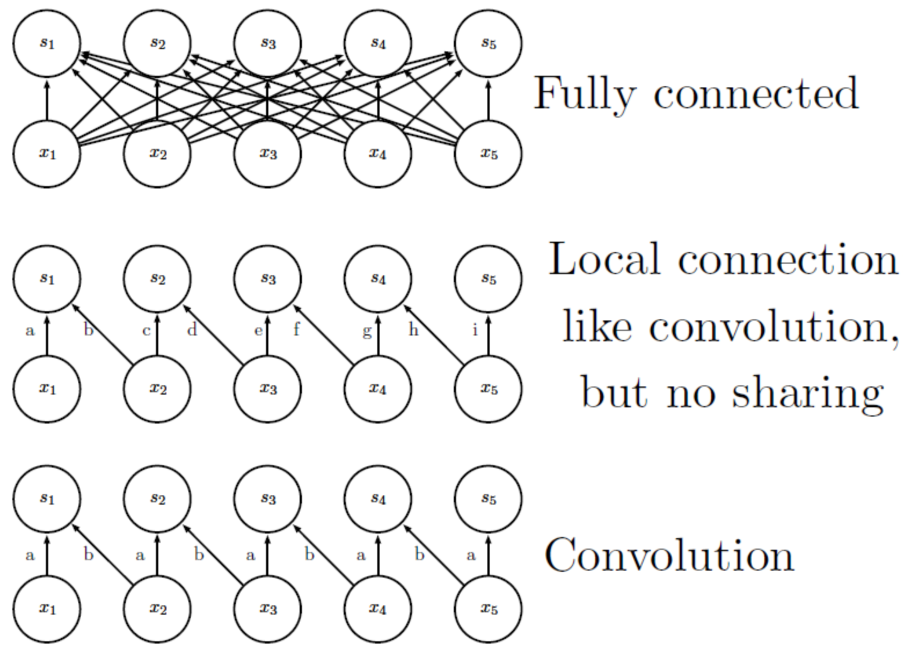
- **Properties**
  - **Sparse Interactions:** Inputs and outputs are not fully connected but have **local connectivity**
  - **Parameter Sharing:** The same kernel is used repeatedly.
  - **Equivariance to transition:** *convolution*(*shift*(input)) = *shift*(*convolution*(input))

***Example*: If there are *m* inputs and *n* outputs,**

Fully-connected layer: *(m+1)* x *n* parameters

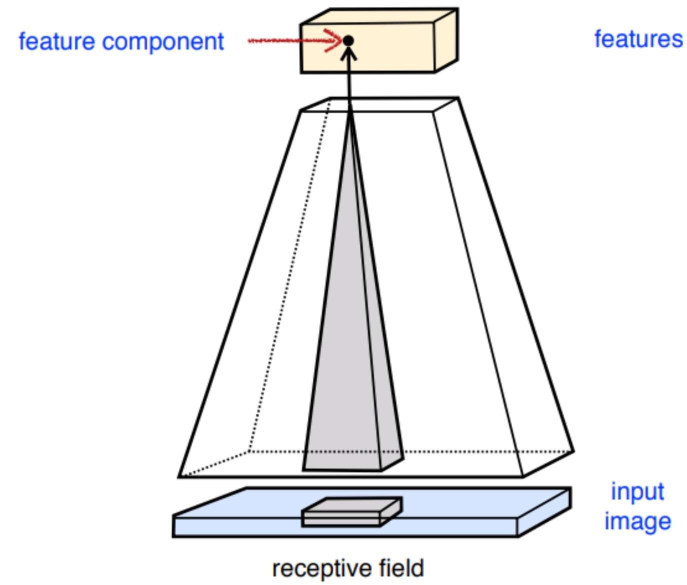
Convolution: *k+1* parameters (k: kernel size)

→ *reduced memory usage, computation*



Convolution

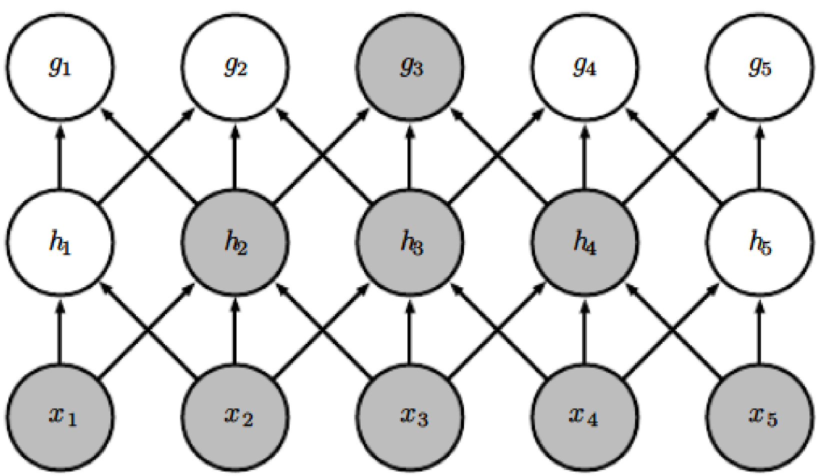
- **Receptive field: Spatial locality**
  - Each element of the output (feature map) processes only for its receptive field (a local region of the input)
  - higher kernel size k → larger receptive field
  - Higher-level layers → larger receptive field



Example: a simple CNN with k=3

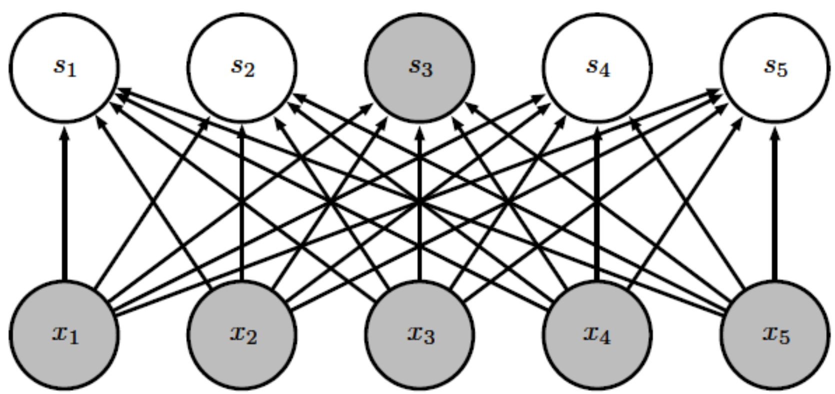
Receptive field of h_3 in the input layer = {x2,x3,x4}

Receptive field of g_3 in the input layer = {x1,x2,x3,x4,x5}



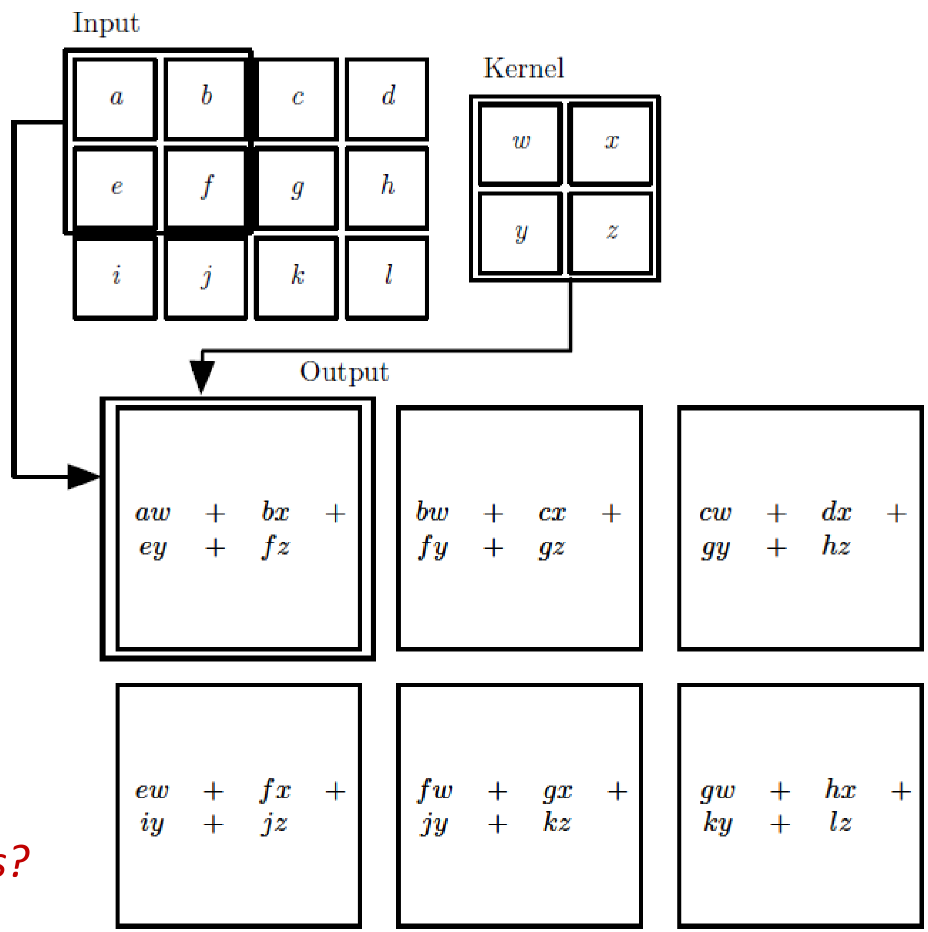
Example: a simple FNN (fully connected)

Receptive field of a hidden unit?



Convolution

- **Convolution with 2-D Array Input**
  - input size **m**=(4,3), kernel size **k**=(2,2),
    stride(kernel step size) s=1, output size **n**=(**m**-**k**)/s+1=(3,2)

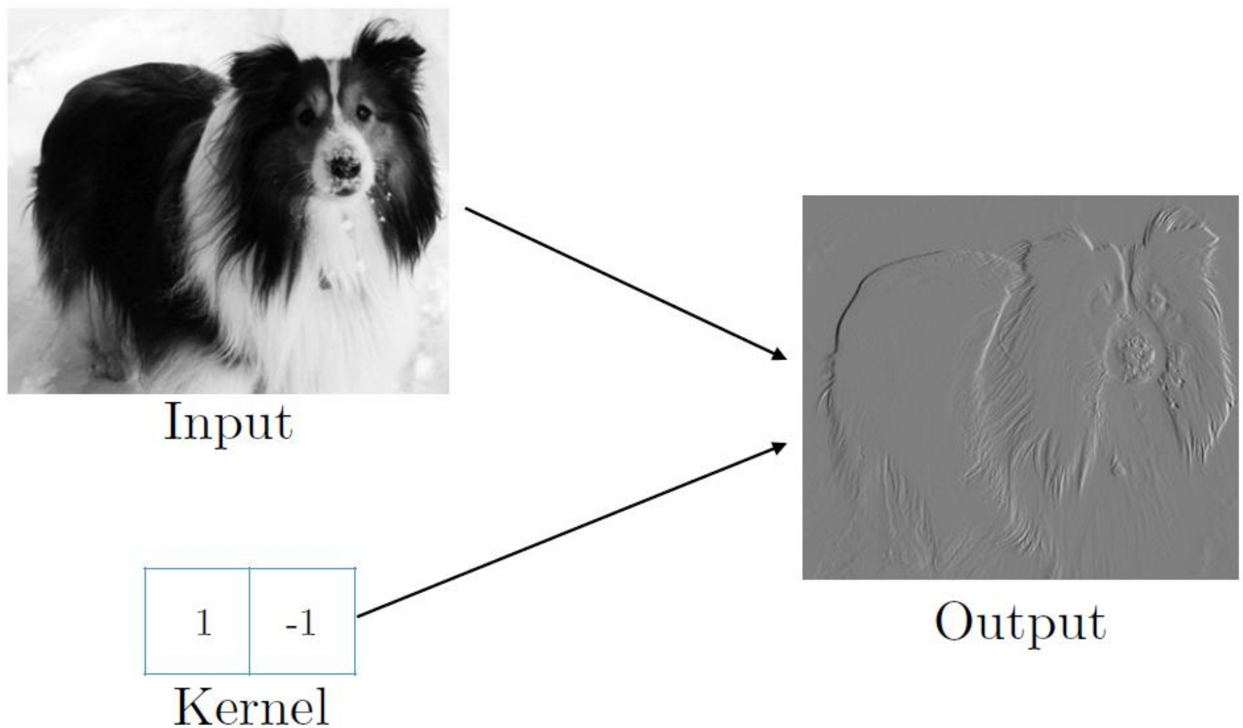


+*bias*

what are parameters?
what are hyperparameters?

Convolution

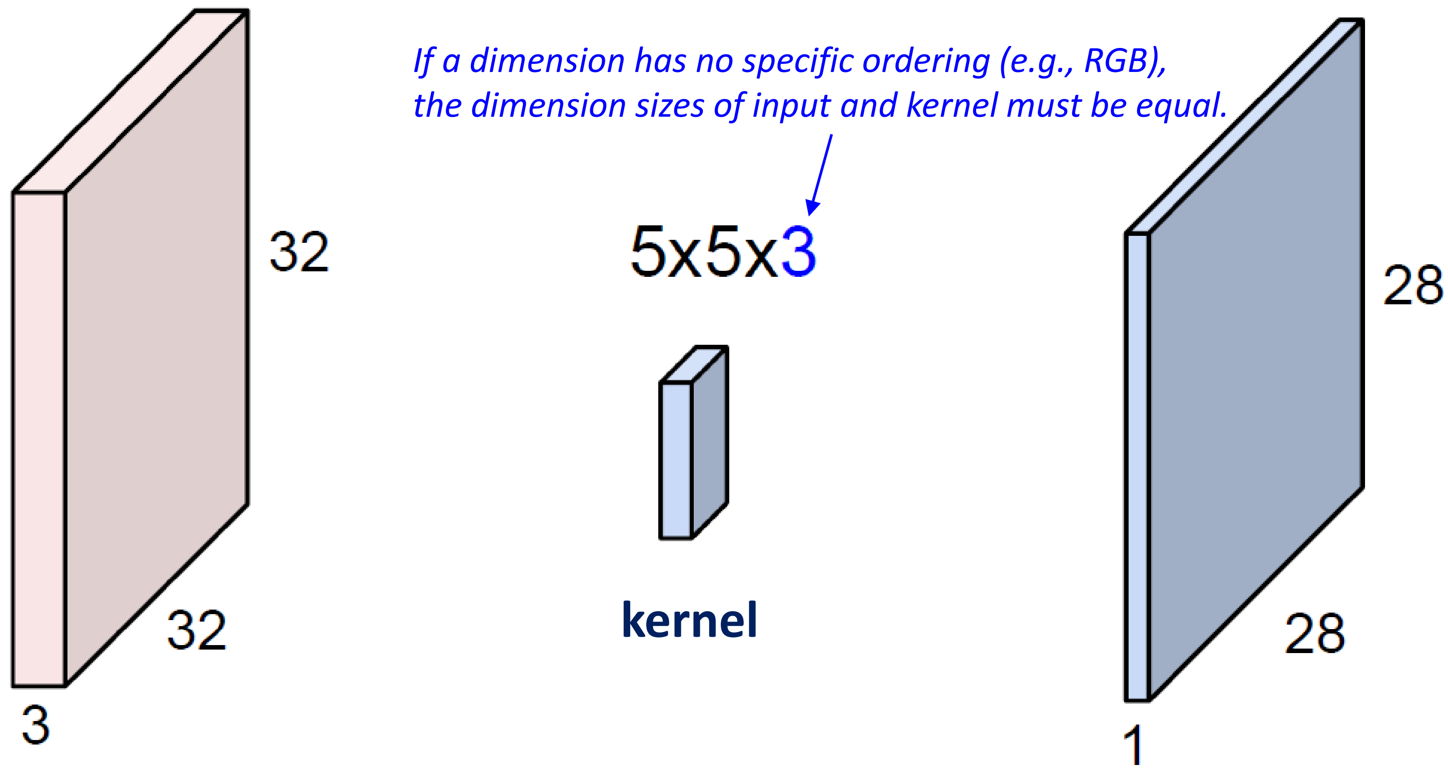
- ***Example*: Edge Detection by Convolution with 2-D Array Input**
  - input size **m**=(320,280), kernel size **k**=(2,1),
    stride(kernel step size) s=1, output size **n**=(**m**-**k**)/s+1=(319,280)



Convolution

what are parameters?
what are hyperparameters?

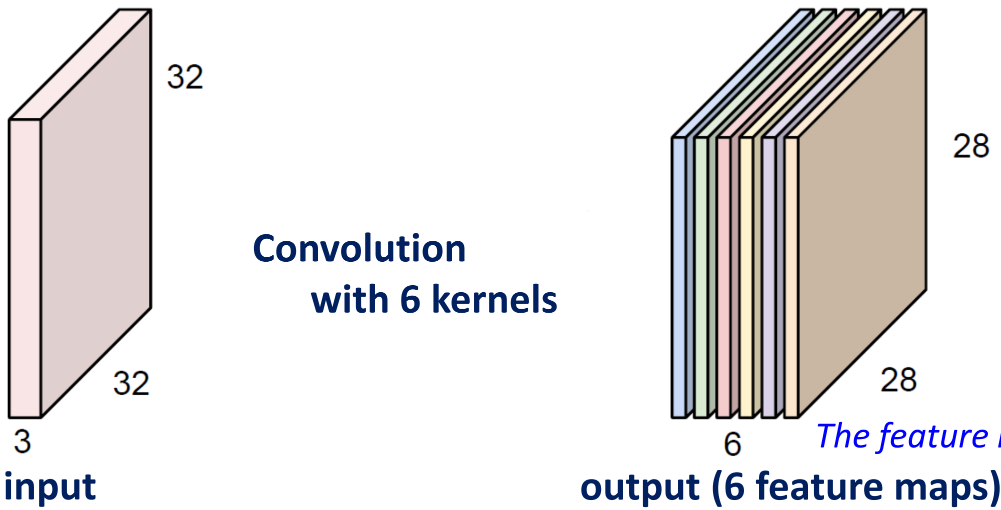
- **Convolution with 3-D Array Input**
  - input size **m**=(32,32,3), kernel size **k**=(5,5,3), stride(kernel step size) s=1, output size **n**=(**m**-**k**)/s+1=(28,28,1)



input **output (feature map)**

Convolution

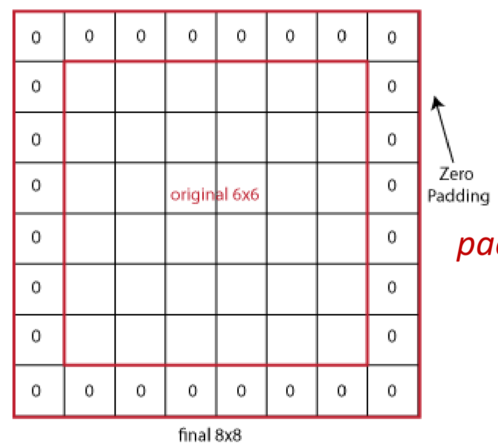
- **Multiple Kernels**
  - input size **m**=(32,32,3), kernel size **k**=(5,5,3), number of kernels=6, stride(kernel step size) s=1, output size **n**=(**m**-**k**)/s+1=(28,28,1), number of feature maps=6



The feature maps have no specific ordering.

- **Zero Paddings of the Input**
  - pad the input with zeros around the border.
  - *e.g.*, if we want ***output volume=input volume***

    kernel size=k, stride=1, padding size=(k-1)/2

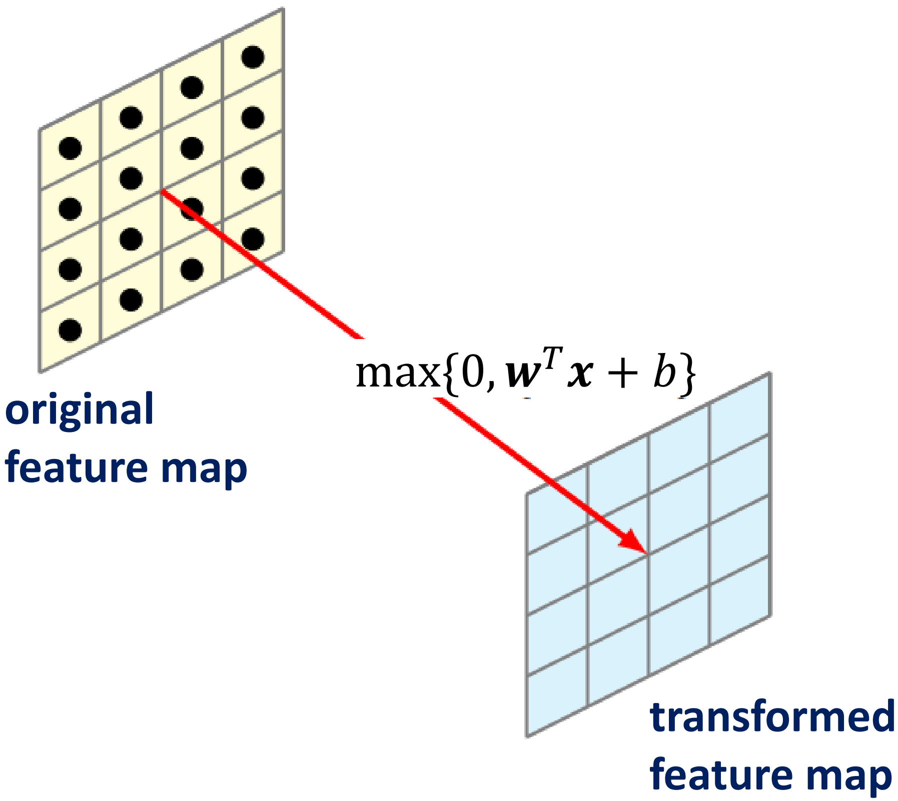


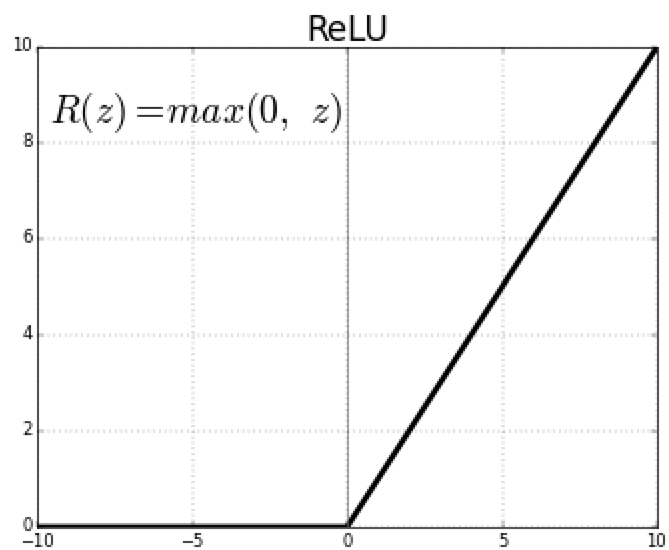
padding size=1

Detector

- **Element-wise non-linearity to obtain a transformed feature map**
  - Each feature map is run through a non-linear activation function
  - "ReLU" is a popular choice. (or, its variants)

additional parameters?



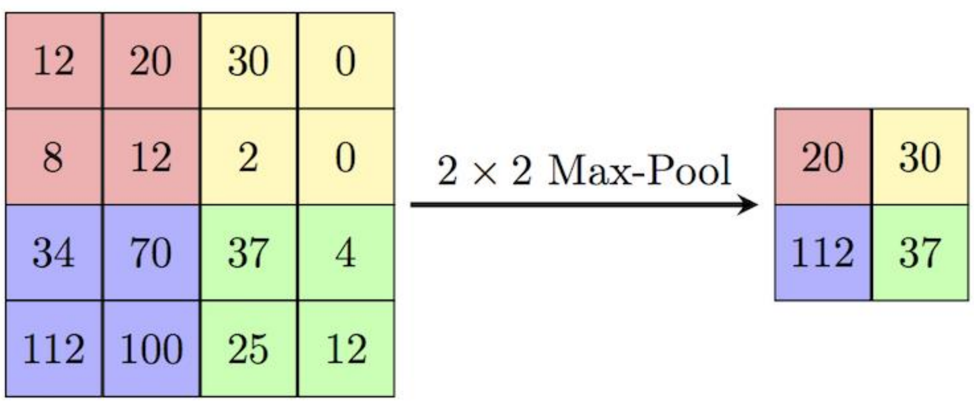


Pooling

- **Summarization of each “transformed feature map”**
  - makes the representations smaller and more manageable (downsampling)
  - reduces the computational burden on the next layer
  - helps to make the representation *slightly* invariant to small translations of the input.
  - can handle inputs of varying size
  - **Various strategies:** max pooling, average pooling, …

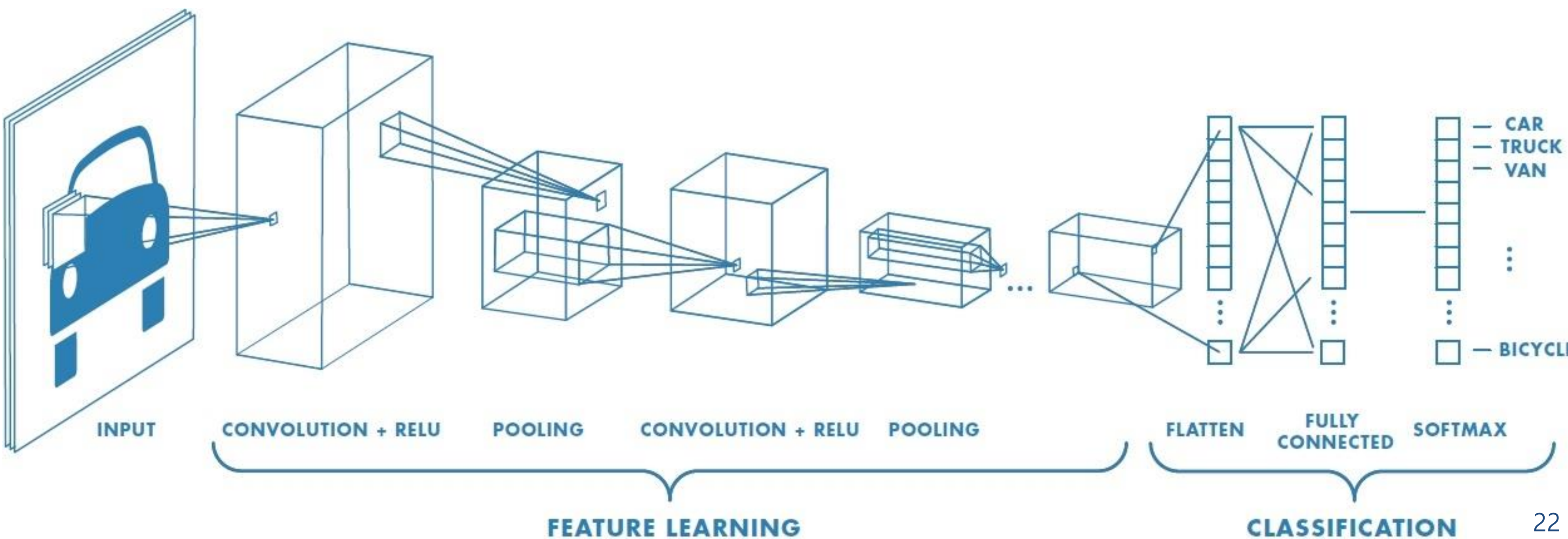
additional parameters?

Example: 2x2 max pooling with stride (step size) 2



Convolutional Neural Networks

- **A CNN is a stack of convolutional layers (convolution, detector, and pooling) and fully-connected layers**
  - **Recent Trends:**
    1. Deeper architectures
    2. Only convolutional layers
    3. Smaller kernels



Convolutional Neural Networks

- Given a training dataset $D = \{(\boldsymbol{X}_1, \boldsymbol{y}_1), (\boldsymbol{X}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{X}_n, \boldsymbol{y}_n)\}$ such that $\boldsymbol{X}_i$ is the *i*-th input array (often 2-D or 3-D) and $\boldsymbol{y}_i$ is the corresponding label of the output variable.

- The model: $\widehat{\boldsymbol{y}} = f(\boldsymbol{X})$
- The cost function (to be minimized)

$$J = \frac{1}{n} \sum_{(\boldsymbol{X}_i, \boldsymbol{y}_i) \in D} L(\boldsymbol{y}_i, \widehat{\boldsymbol{y}}_i)$$

- For training, any gradient-based optimization algorithms can be used.

Convolutional Neural Networks

Example: Empirical results (Goodfellow et al., 2014)

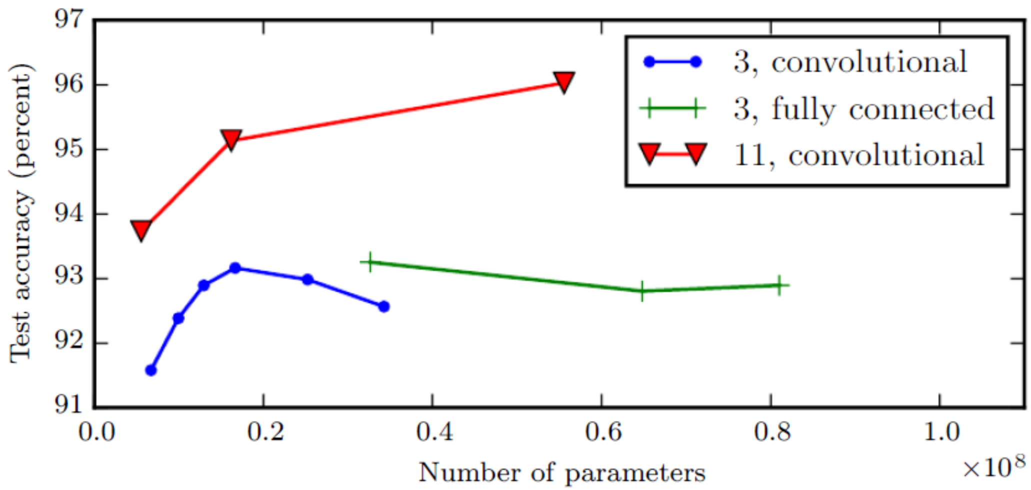
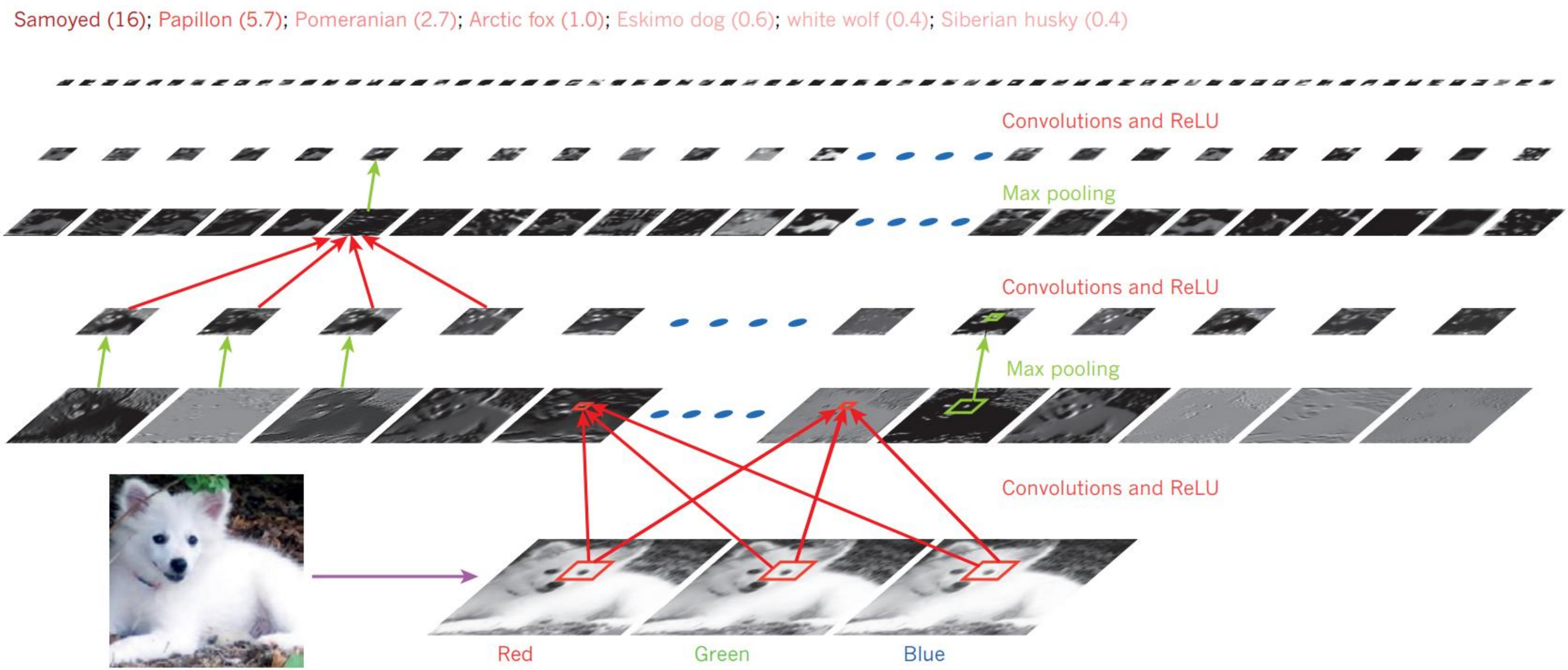


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from Goodfellow *et al.* (2014d) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance. The legend indicates the depth of network used to make each curve and whether the curve represents variation in the size of the convolutional or the fully connected layers. We observe that shallow models in this context overfit at around 20 million parameters while deep ones can benefit from having over 60 million. This suggests that using a deep model expresses a useful preference over the space of functions the model can learn. Specifically, it expresses a belief that the function should consist of many simpler functions composed together. This could result either in learning a representation that is composed in turn of simpler representations (e.g., corners defined in terms of edges) or in learning a program with sequentially dependent steps (e.g., first locate a set of objects, then segment them from each other, then recognize them).

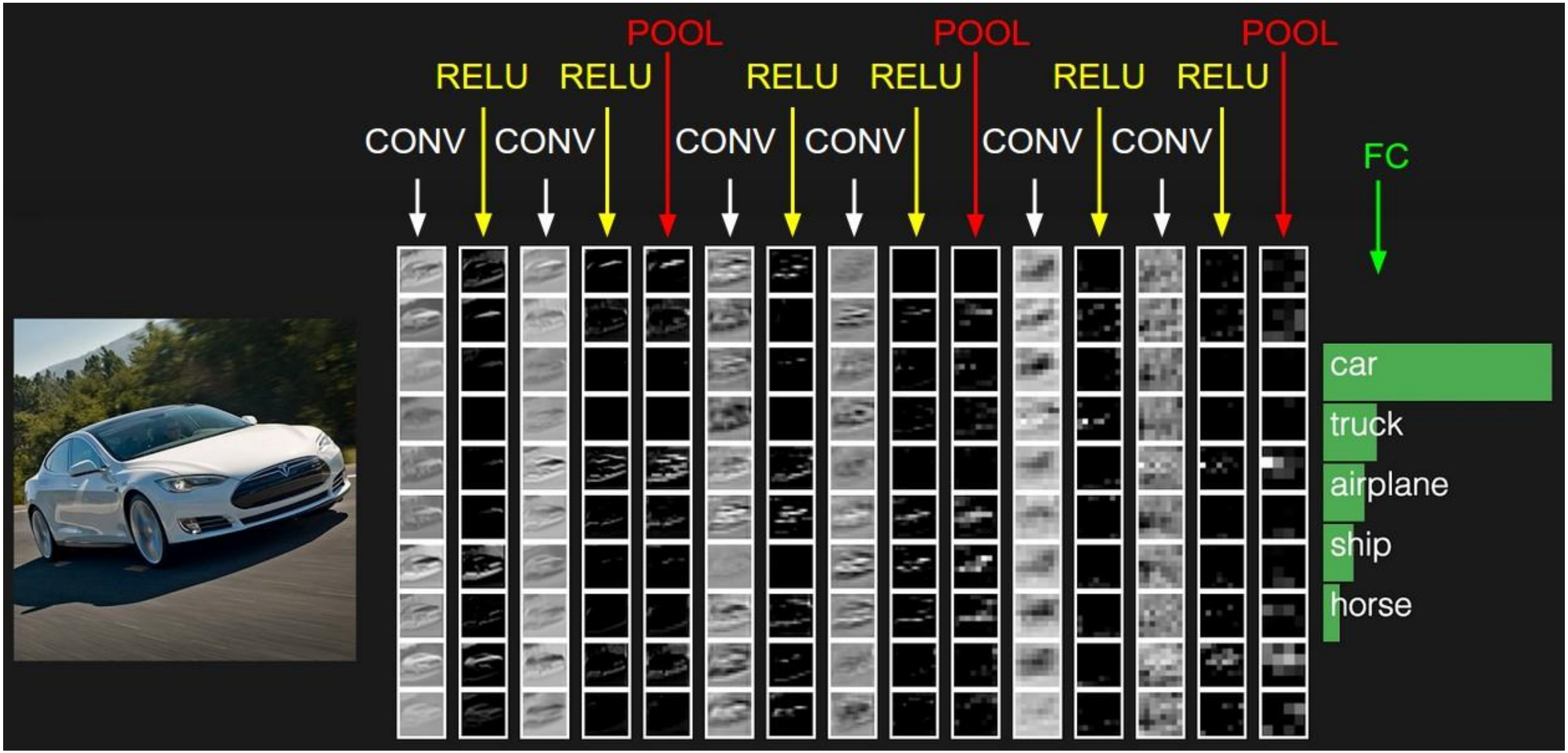
Convolutional Neural Networks

Example: Illustration (LeCun et al., 2015)



Convolutional Neural Networks

Example: Illustration (http://cs231n.github.io/convolutional-networks/)



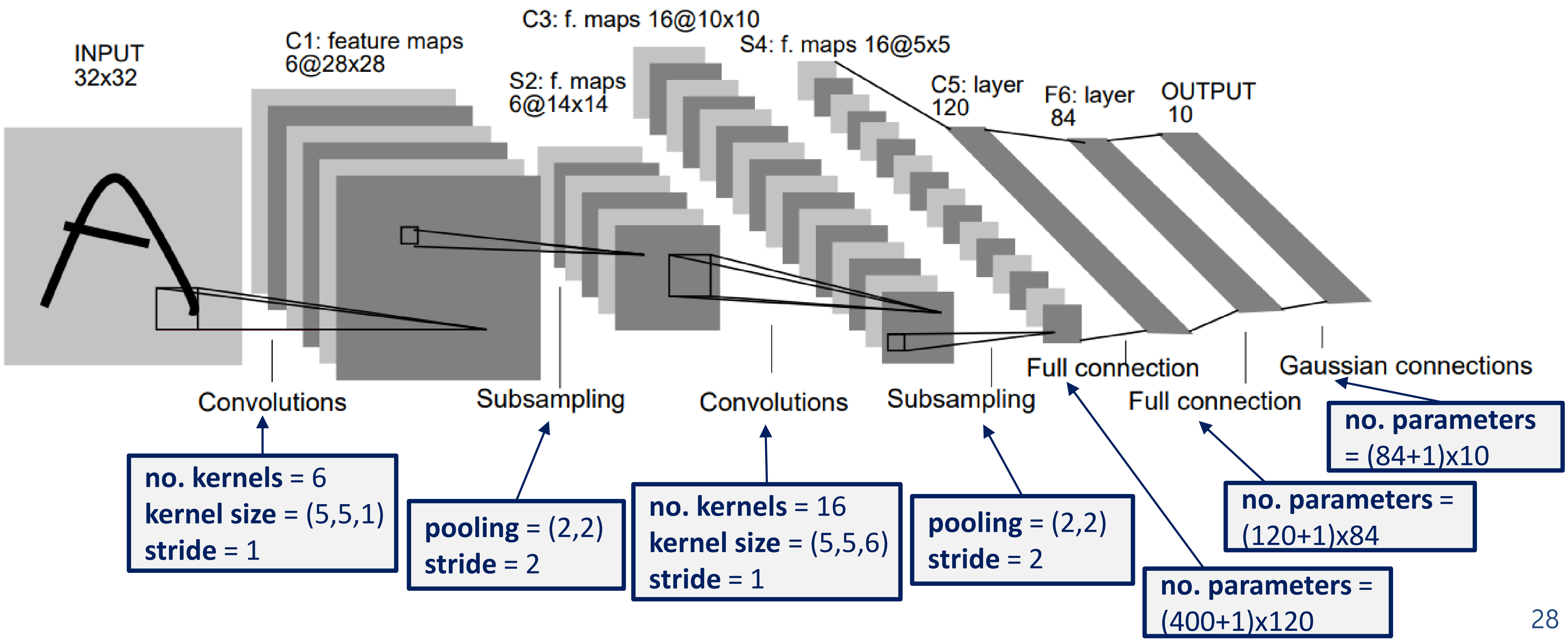
Popular CNN Architectures

- **LeNet-5**
- **AlexNet**
- **VGGNet**
- **GoogLeNet (InceptionNet)**
- **ResNet**
- **DenseNet**

LeNet-5

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

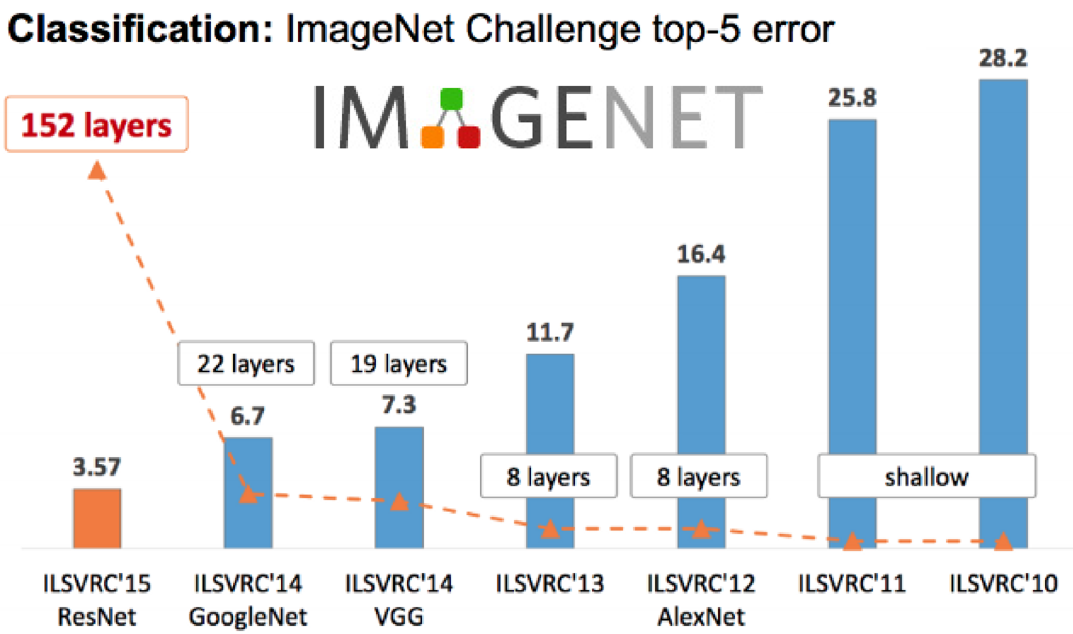
- **MNIST dataset (2D images [32x32])**
- **5 Layers (2 Convolutional Layers (average pooling) + 3 Fully Connected Layers)**
- **No. Parameters = 60k**

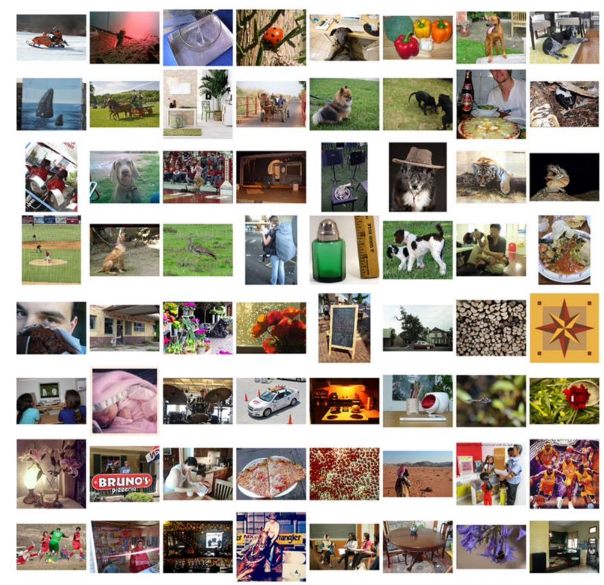


ILSVRC

- **ILSVRC (ImageNet Large Scale Visual Recognition Challenge)**
  : Make 5 guesses about the image label!

http://www.image-net.org/challenges/LSVRC/



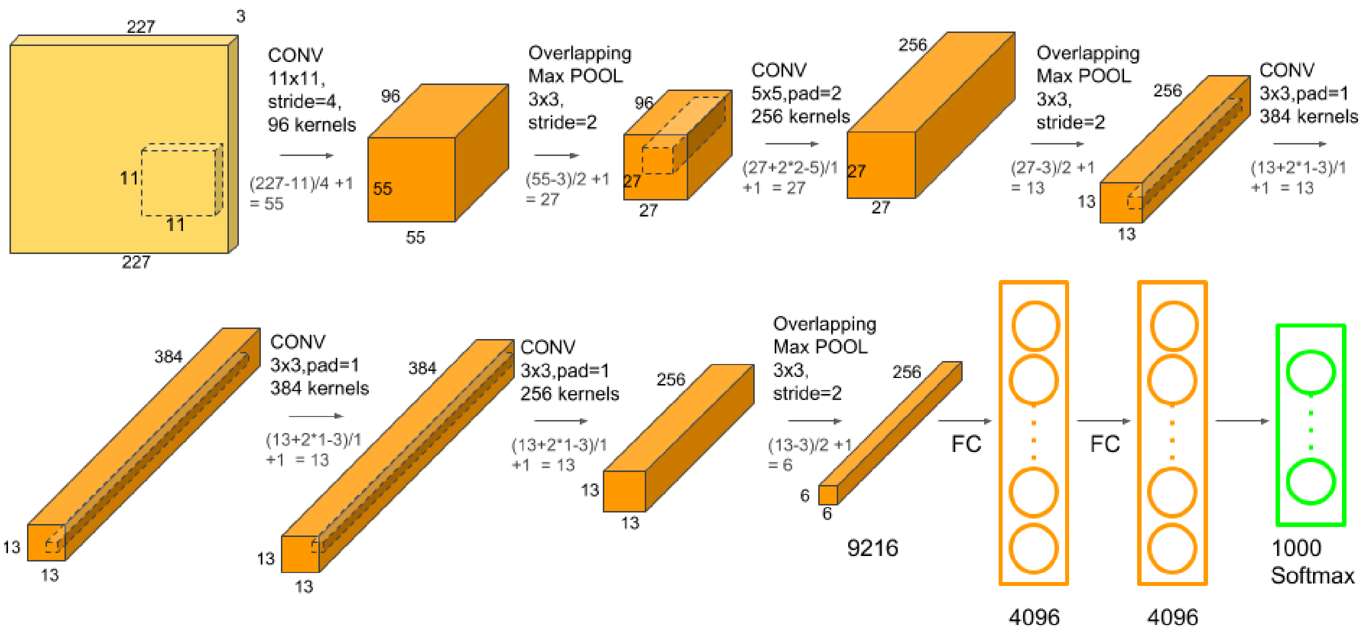


Year	Train images (per class)	Val images (per class)	Test images (per class)
Image classification annotations (1000 object classes)			
ILSVRC2010	1,261,406 (668–3047)	50,000 (50)	150,000 (150)
ILSVRC2011	1,229,413 (384–1300)	50,000 (50)	100,000 (100)
ILSVRC2012-14	1,281,167 (732–1300)	50,000 (50)	100,000 (100)

AlexNet

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097-1105).

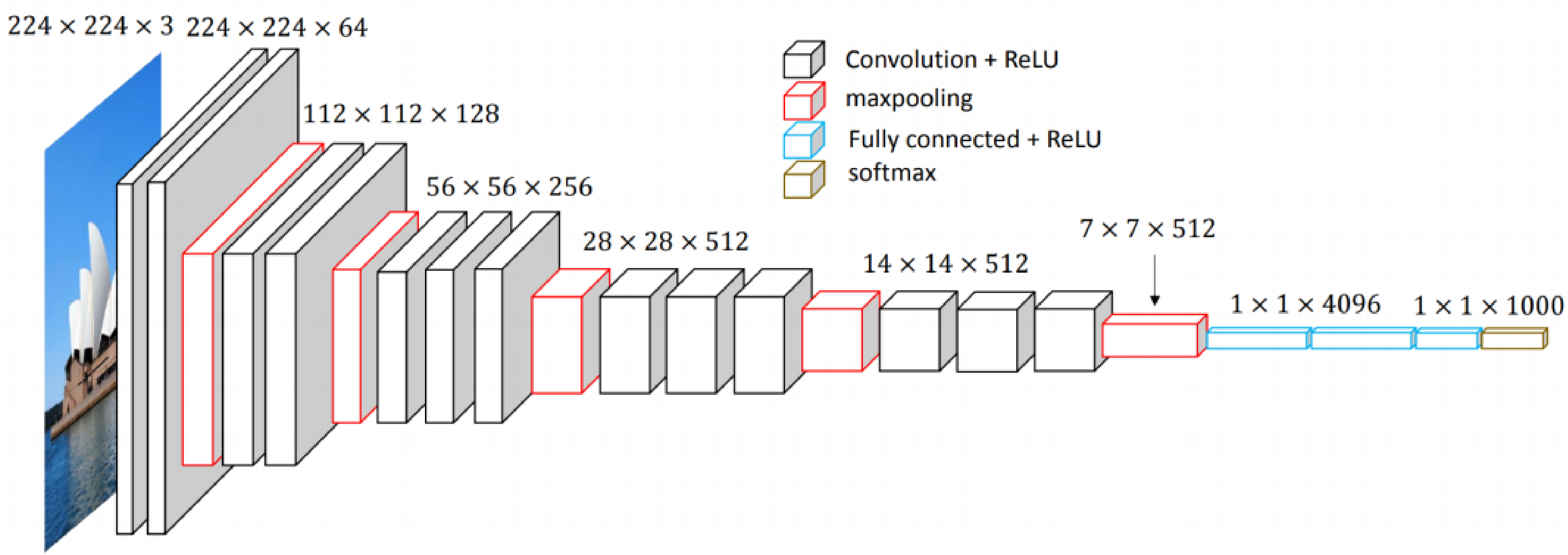
- **ReLU activation function**
- **Data Augmentation (left-right flip, random crops of 227x227 from 256x256)**
- **8 Layers (5 Convolutional Layers + 3 Fully-connected Layers)**
- **62M parameters**



VGGNet

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

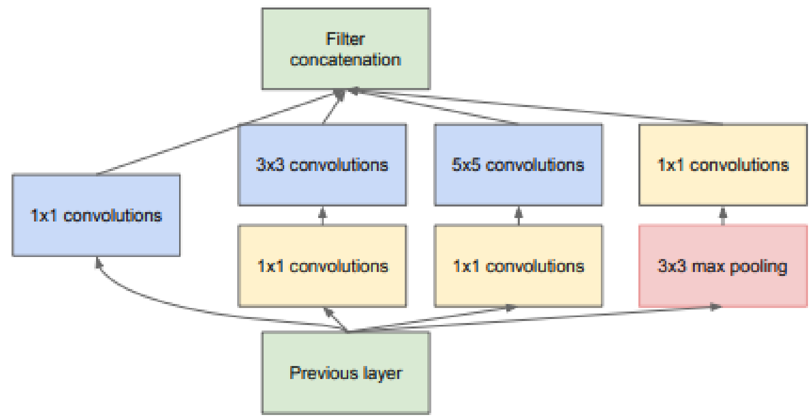
- **Deeper architecture with smaller convolution kernels (3x3)**
- **19 Layers**
- **138M Parameters**

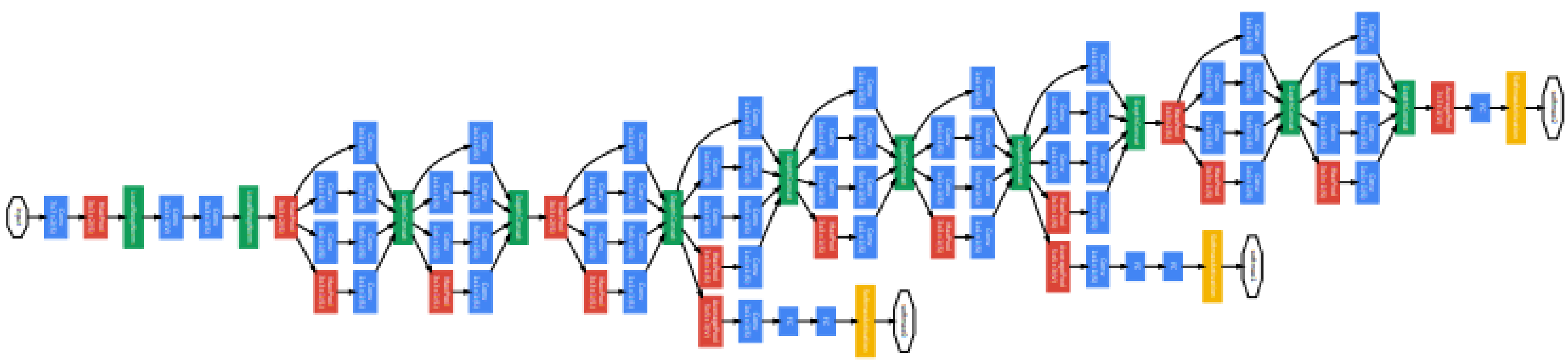


GoogLeNet (InceptionNet)

Szegedy, C., et al. (2015). Going deeper with convolutions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-9).

- **"Inception module":** parallel paths with different receptive field sizes
- **1x1 convolution:** reducing the number of feature maps
- **No fully connected layers**
- **22 layers**
- **4M parameters**

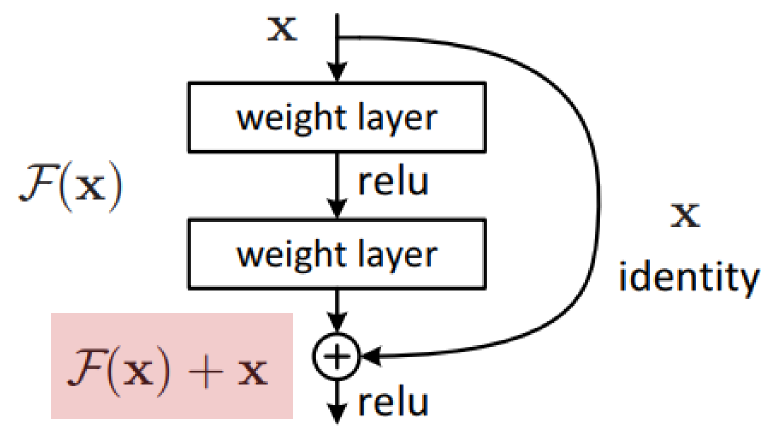




ResNet

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778).

- **“Skip Connections”:** feature reuse, alleviated vanishing gradient
  element-wise addition of outputs from two different layers
- **Much deeper architecture**
- **152 Layers**
- **60M parameters**

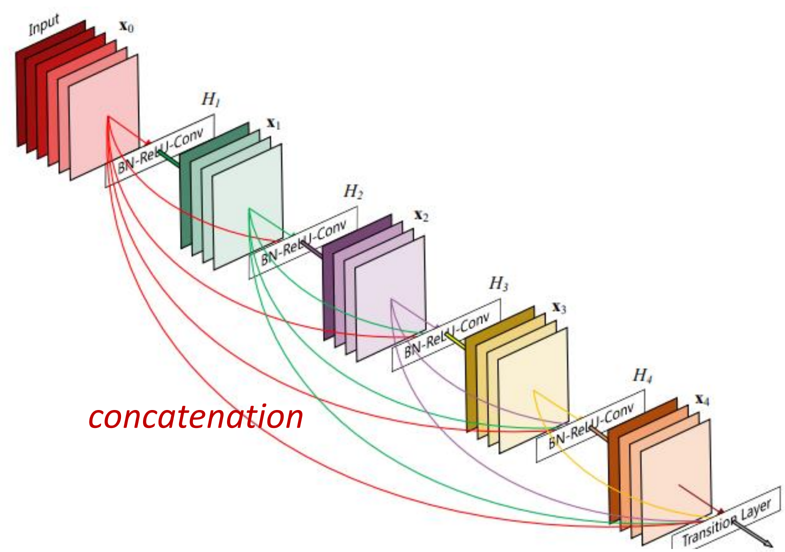


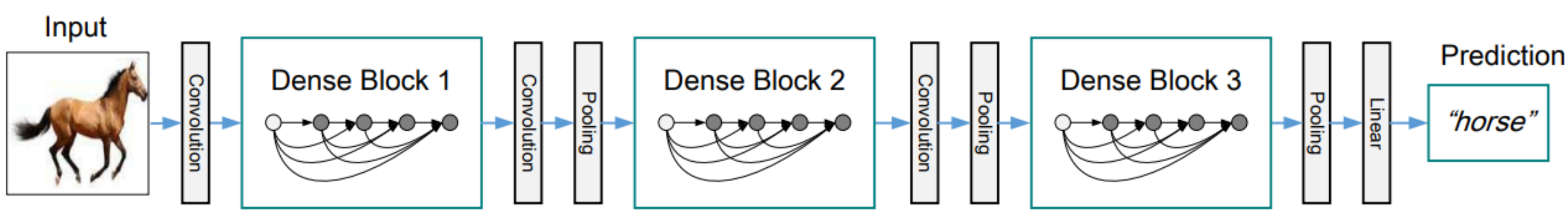
element-wise addition

DenseNet

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4700-4708).

- **"Dense Connections":** feature reuse, alleviated vanishing gradient
  concatenation of outputs from previous layers
- **Deeper and deeper…**
- **201 Layers**
- **20M parameters**





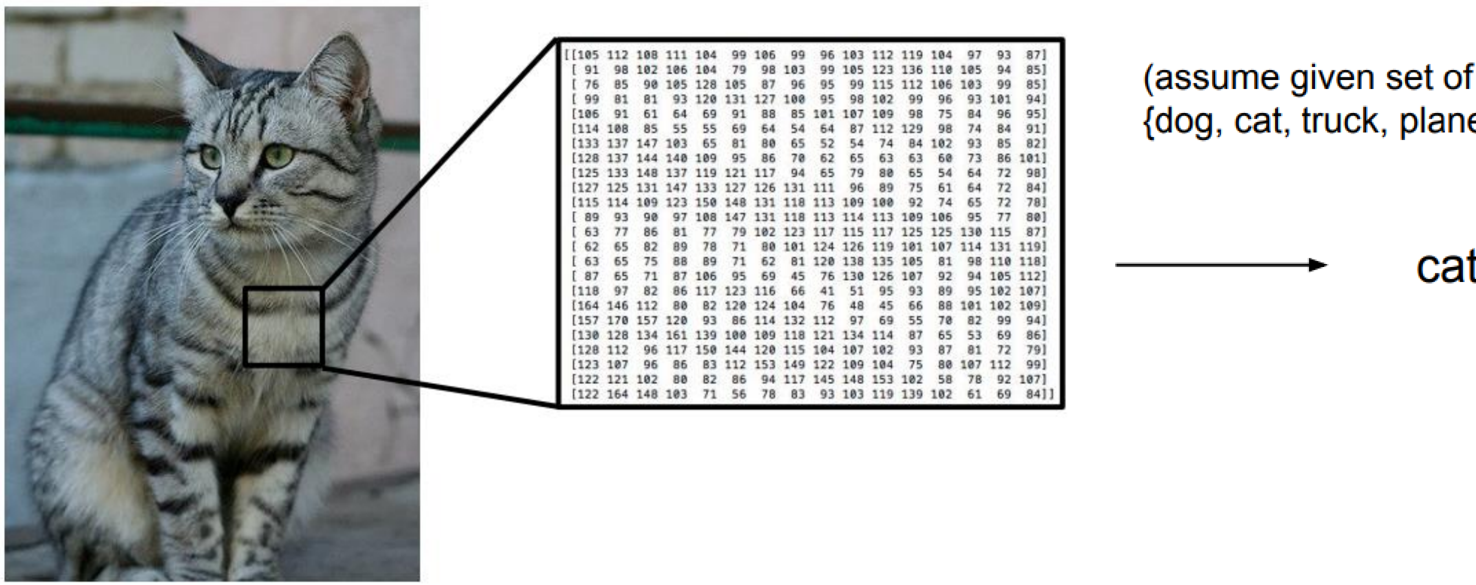
CNN for Image Classification

- **Further Readings**
  - Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1492-1500).
  - Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of AAAI Conference on Artificial Intelligence*.
  - Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7132-7141).
  - Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8697-8710).
  - Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of International Conference on Machine Learning* (pp. 6105-6114).
  - …

What’s Next?

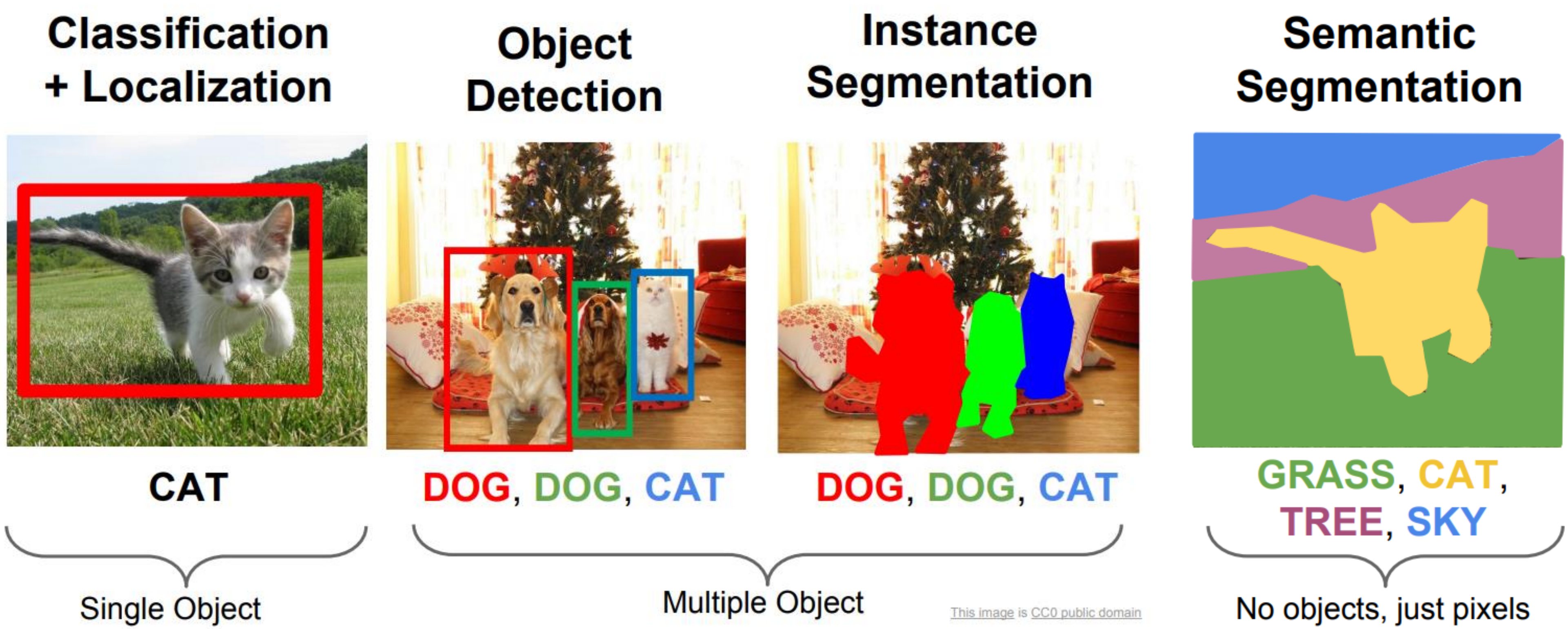
Computer Vision Tasks

Image Classification Task



(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

Computer Vision Tasks



Computer Vision Tasks

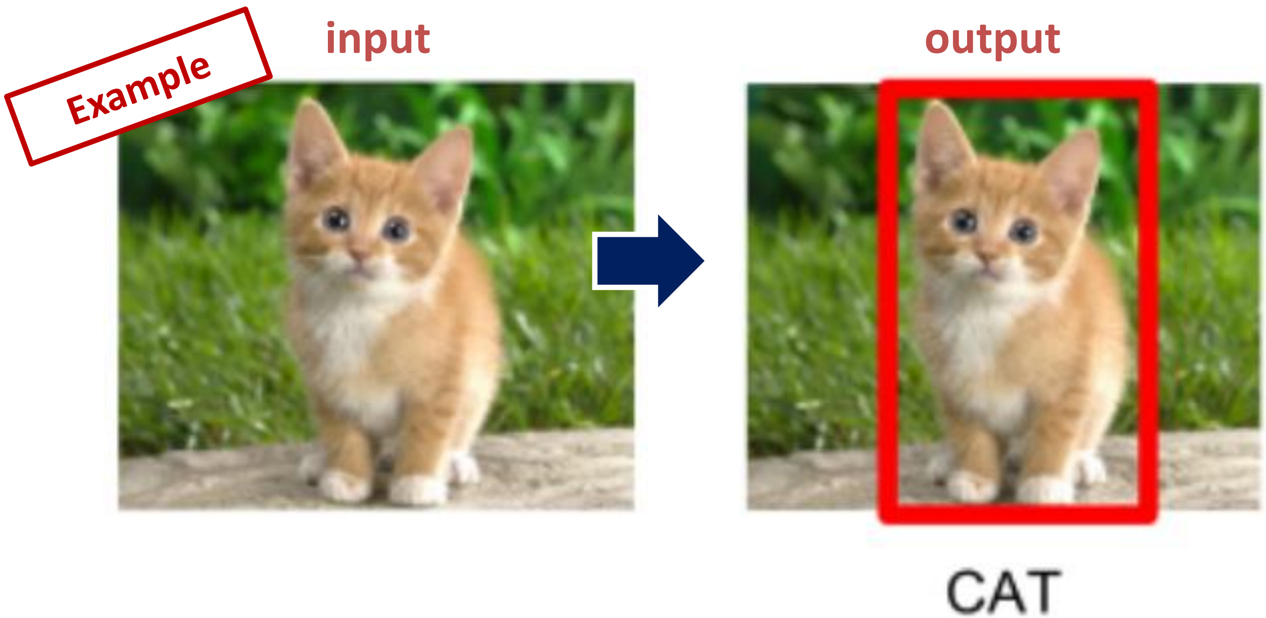
- **Pascal VOC (Visual Object Classes) Challenges**
  - http://host.robots.ox.ac.uk:8080/pascal/VOC/

- **COCO Challenges**
  - http://cocodataset.org/

Localization/Detection

- **Training Dataset**
  - **Input (X):** image
  - **Output (Y):** object category, object box location (x,y,w,h)

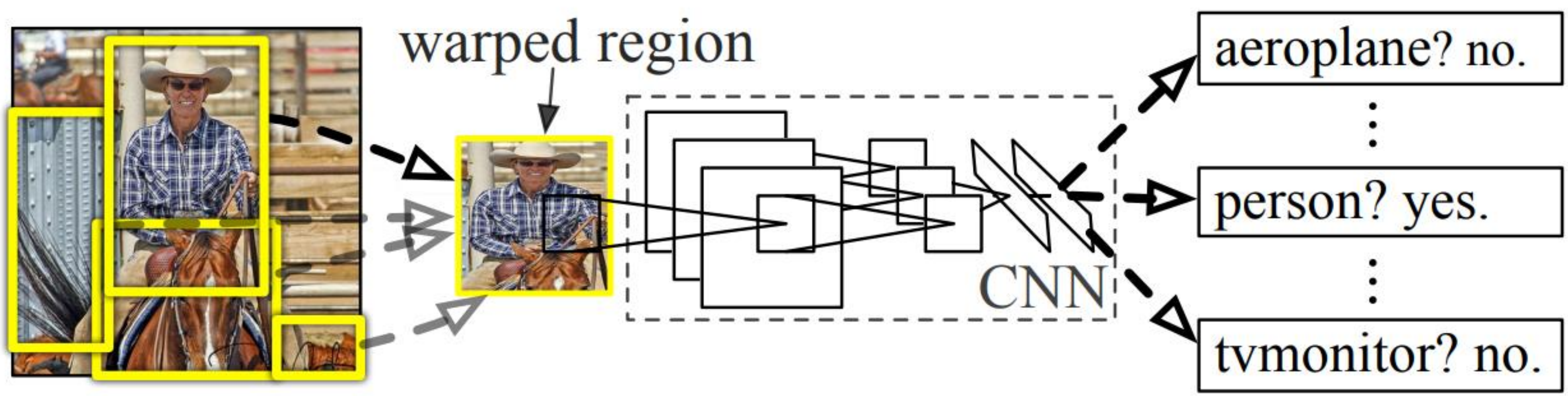
- **Single object → Localization**
- **Multiple objects → Detection**



Localization/Detection

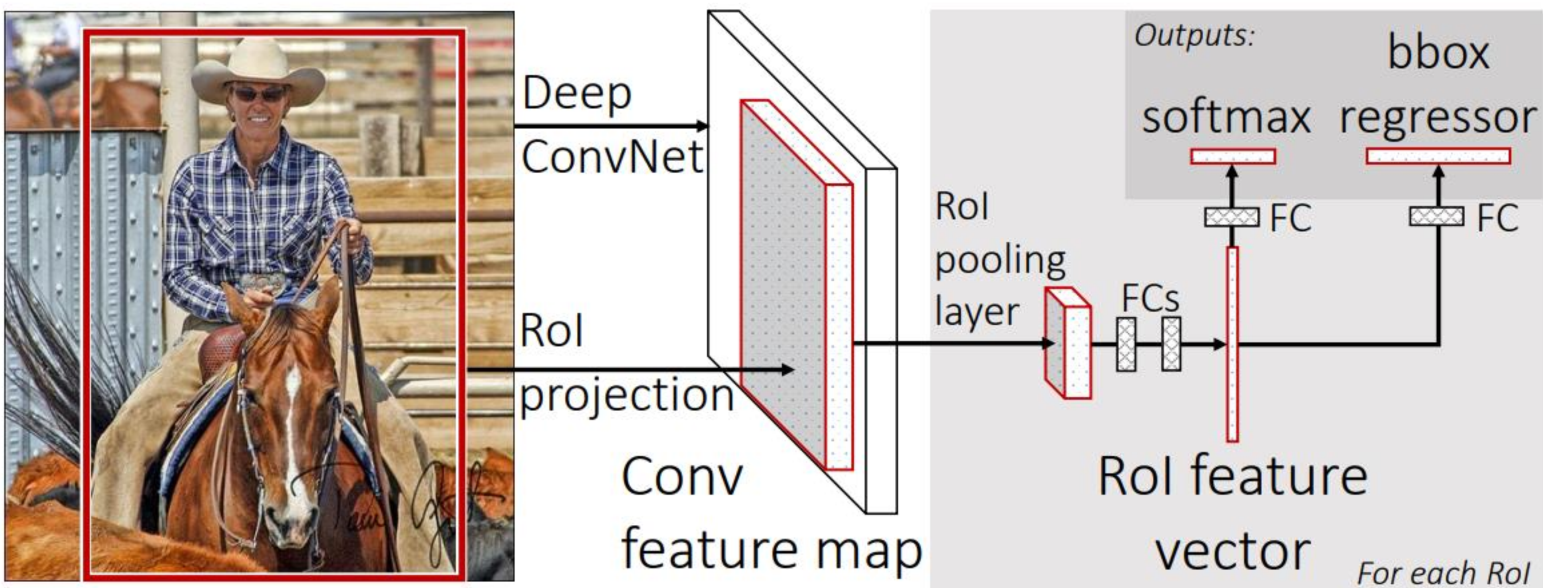
- **Region-based Convolutional Neural Network (R-CNN)**
  - Selective search to identify ~2k region proposals
  - Feature extraction from each (resized) region using a CNN (modified AlexNet)
  - Classification of the region with a support vector machine → the category of the object
  - Adjustment of the region (bounding box) with a linear regression model → offsets of the box coordinates for the object (*Δx,Δy,Δw,Δh*)

issues in training & test?



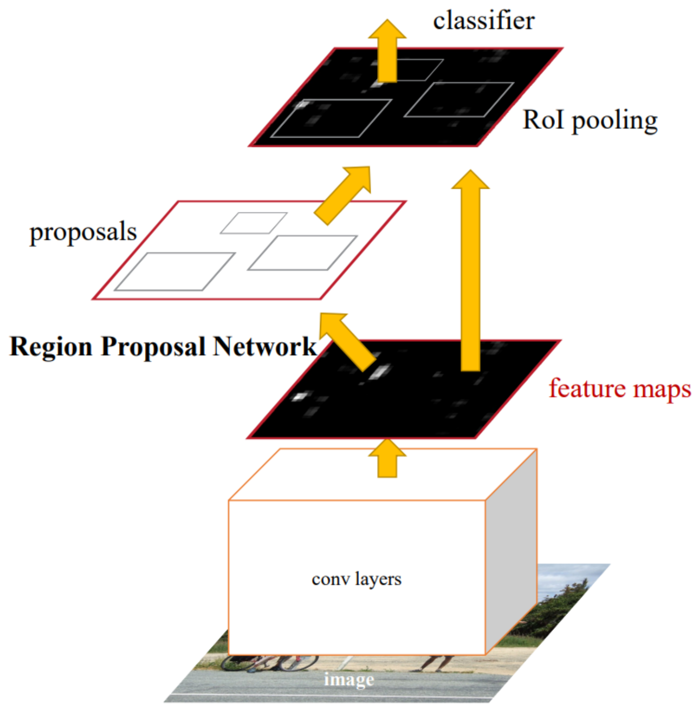
Localization/Detection

- **Fast R-CNN**
  - The concept is similar to the R-CNN
  - Main Differences from the R-CNN (which make it faster)
    - Feature extraction from the entire image (not region proposals) using a CNN
    - “Region of Interest (RoI) Pooling” to reshape each region into fixed size
    - Implementation as a single joint neural network – *multi-task learning*



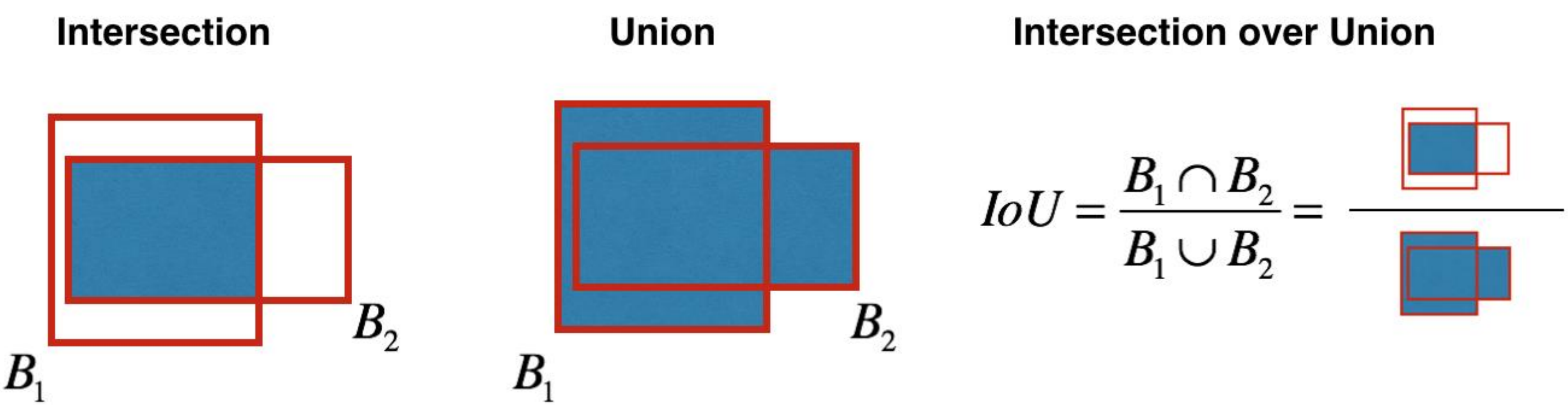
Localization/Detection

- **Faster R-CNN**
  - Much faster than R-CNN and Fast R-CNN
  - Main Differences from the Fast R-CNN (which make it faster)
    - No need for selective search
    - Prediction of the region proposals within the network – *multi-task learning*
    - End-to-end training



Localization/Detection

- **Evaluation (Performance Metric)**
  - **Intersection over Union** (IoU) between the prediction (B_1) and the ground-truth (B_2)

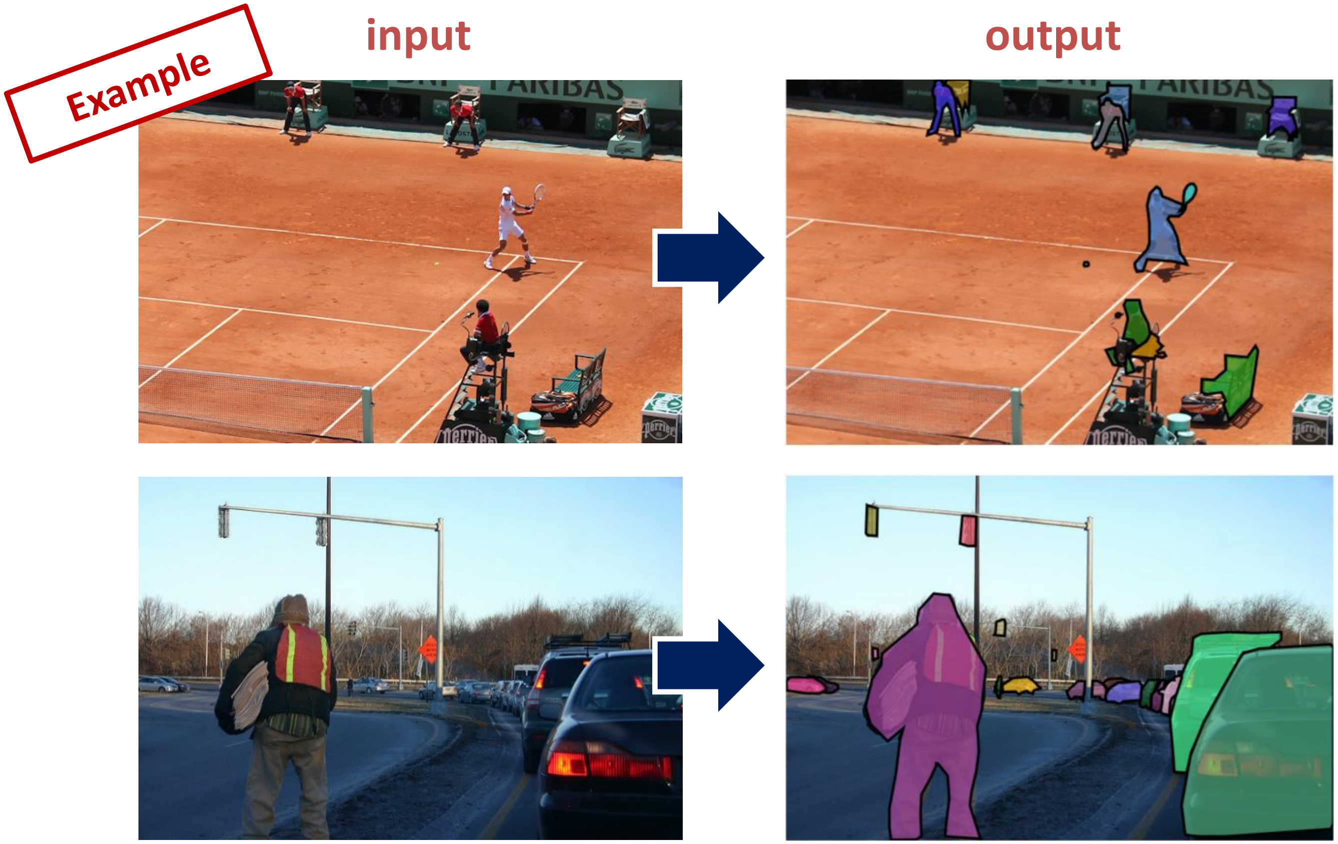


Localization/Detection

- **Further Readings**
  - Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once (YOLO): Unified, real-time object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779-788).
  - Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In *Proceedings of European Conference on Computer Vision* (pp. 21-37).
  - Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2921-2929). → *training without using object location information*
  - …

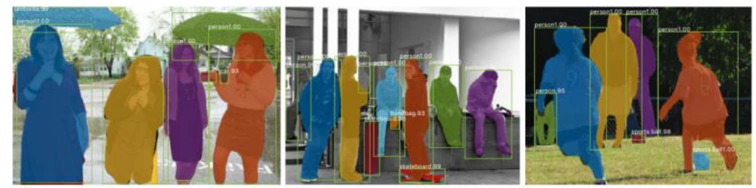
Instance Segmentation

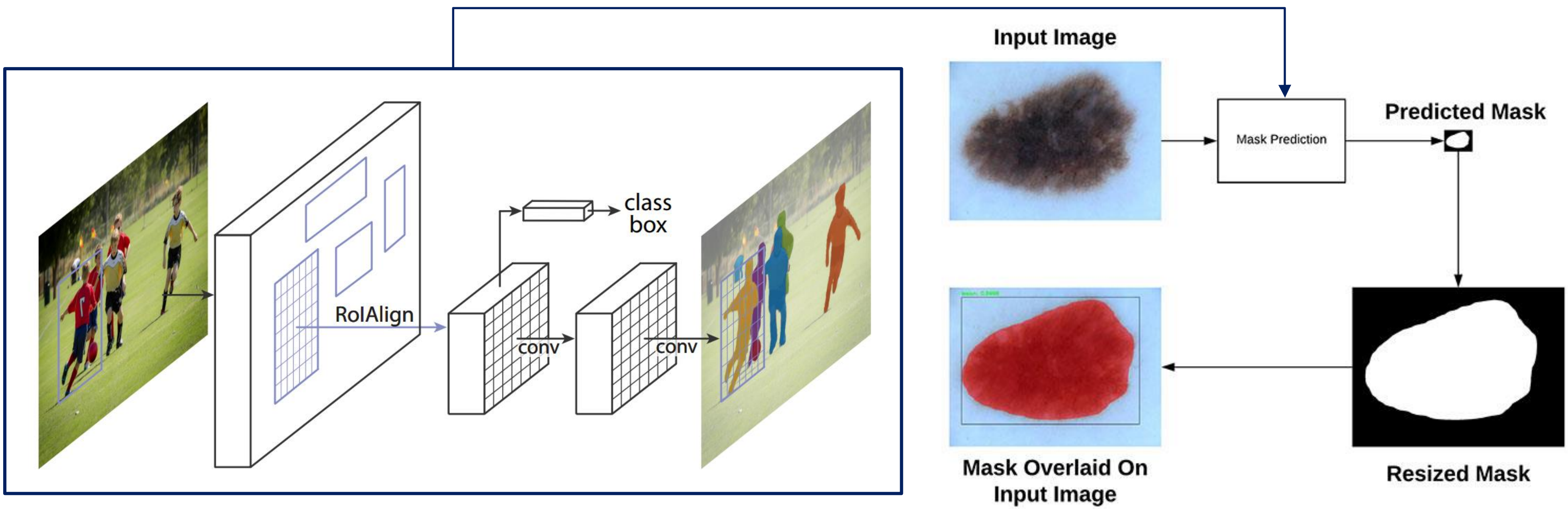
- **Training Dataset for Instance Segmentation** (*Pixel-Wise Classification*)
  - **Input (X):** image, **output (Y):** object category, object pixels



Instance Segmentation

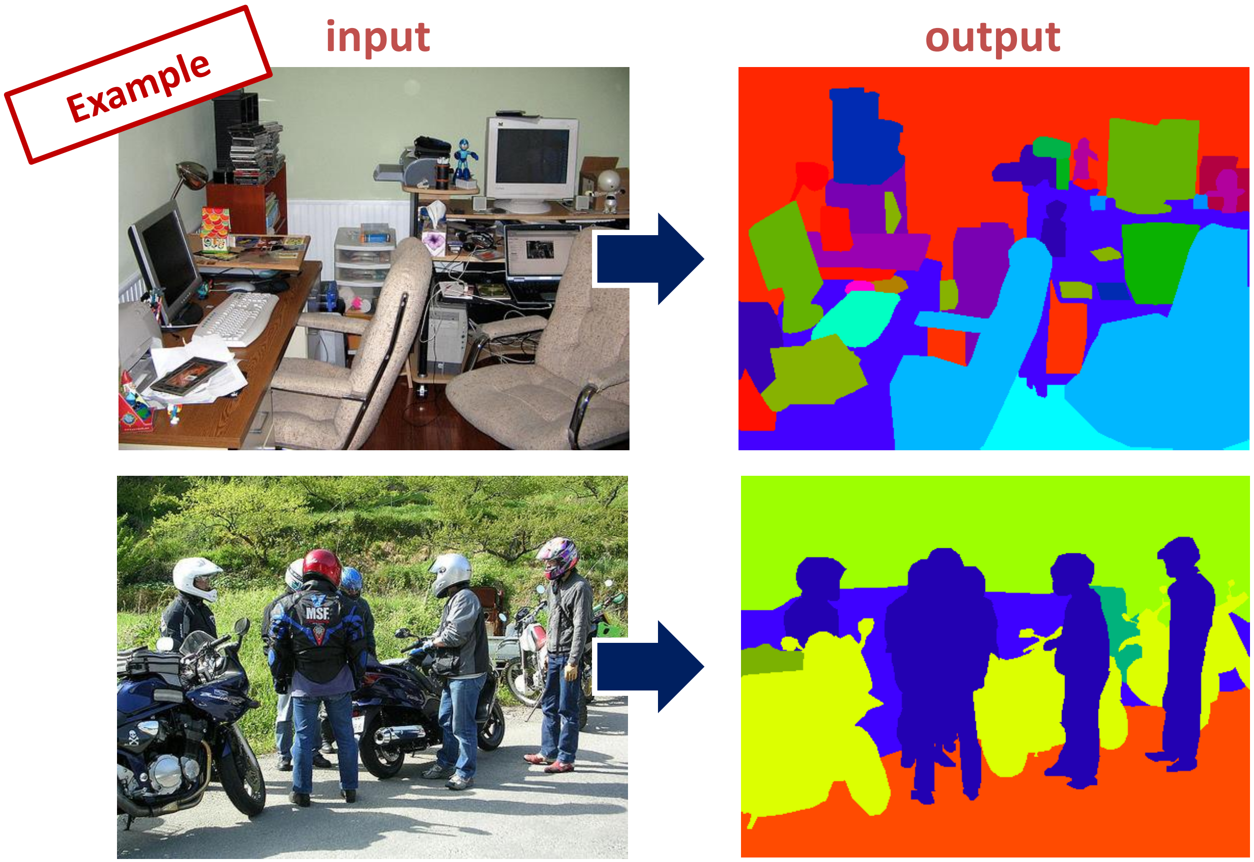
- **Mask R-CNN**
  - An extension of Faster R-CNN
  - Performs three tasks: the model predicts (1) the object category, (2) bounding box for the object, and (3) pixel-wise mask for the object





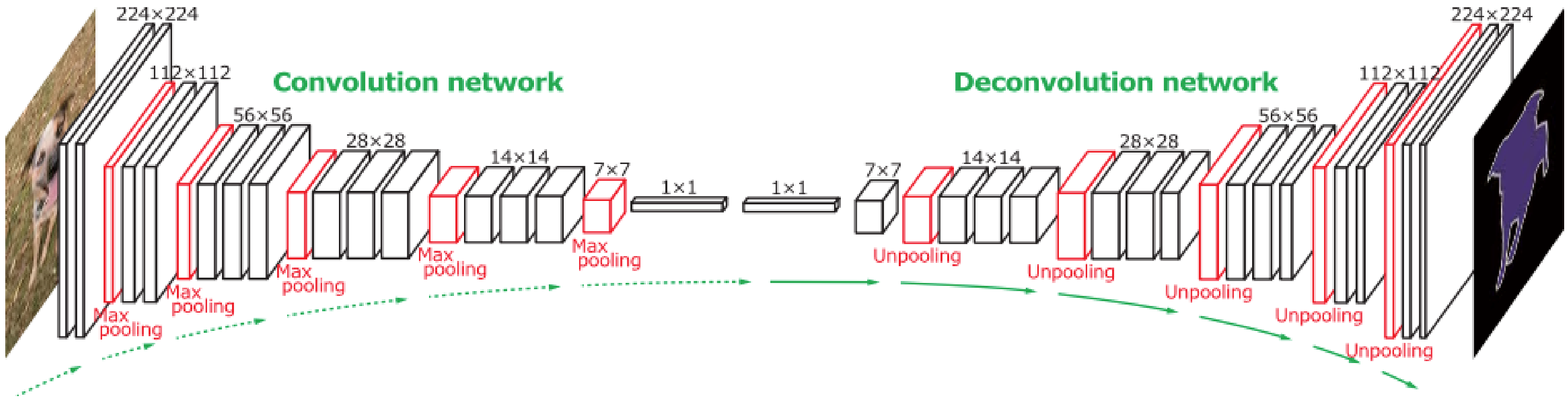
Semantic Segmentation

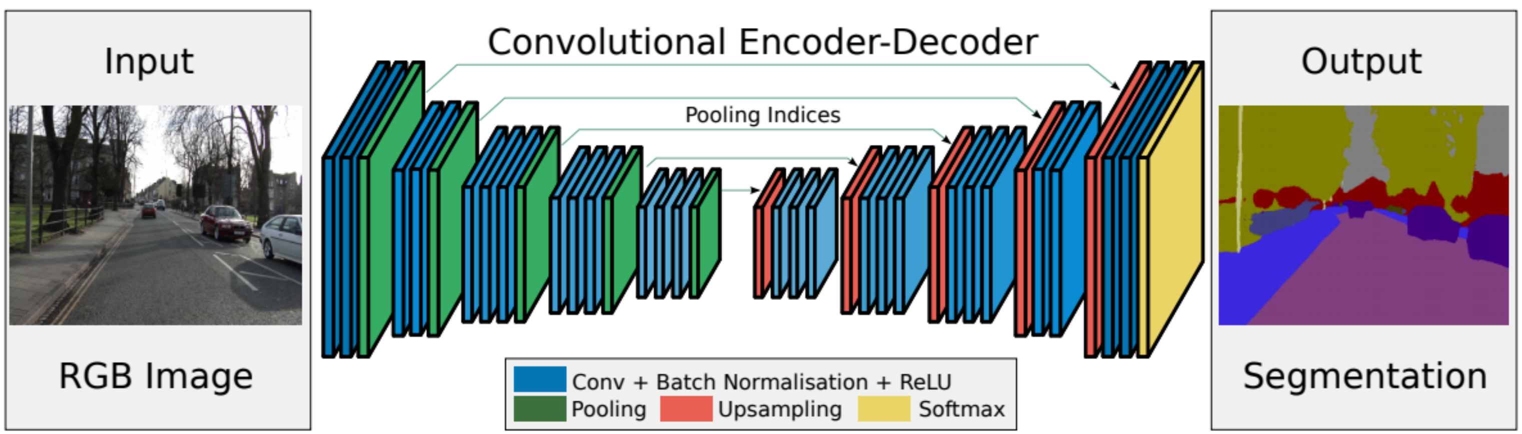
- **Training dataset for Semantic Segmentation** (*Pixel-Wise Classification*)
  - **Input (X):** Image, **output (Y):** pixel-level categories
  - **input width and height = output width and height (different no. channels)**



Instance/Semantic Segmentation

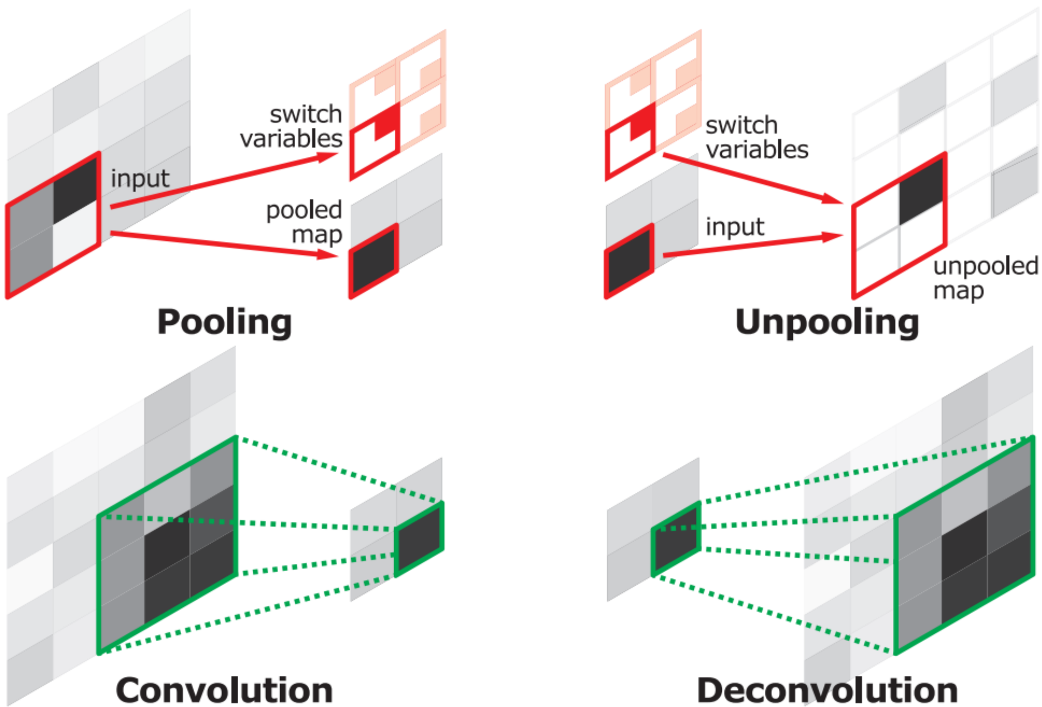
- **Encoder-Decoder Architecture**
  - **Encoder:** Convolutional Neural Network (for downsampling)
  - **Decoder:** **Deconvolutional Neural Network** (for upsampling)





Instance/Semantic Segmentation

- **Deconvolutional network: Convolutional network run in reverse**
  - **Pooling → Unpooling** (reconstruct feature map, enlarged but sparse)
  - **Convolution → Deconvolution*** (densify sparse feature map, enlarged and dense)



* ***Deconvolution*** is a bad name.
Better alternatives are
transpose convolution,
upconvolution,
fractionally strided convolution,
backward strided convolution, …

Instance/Semantic Segmentation

- **Further Readings**
  - Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3431-3440).
  - Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of IEEE International Conference on Computer Vision* (pp. 1520-1528).
  - Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 234-241).
  - Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12), 2481-2495.
  - Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2881-2890).
  - Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 834-848.
  - …