Convolution Neural Networks

with

Daniel L. Silver, Ph.D.

Andy McIntyre, Ph.D.

June 24, 2022

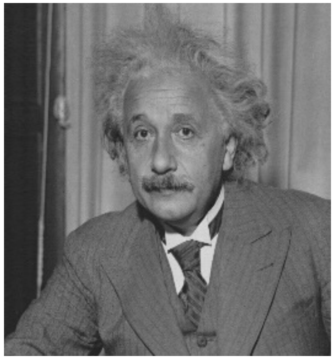
[Slides based on originals by Yann LeCun]

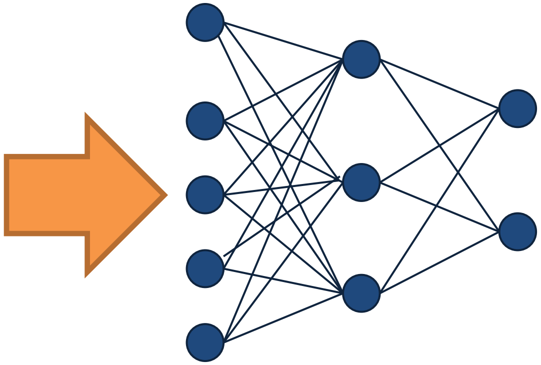
Convolution Neural Networks

- Fundamentally CNNs take advantage of knowledge that we have about the problem's input space
- We know that pixels in an image that are adjacent to each other are related – similar color and brightness
- We can use this background knowledge as a source of inductive bias to help develop better NN models

Receptive Fields

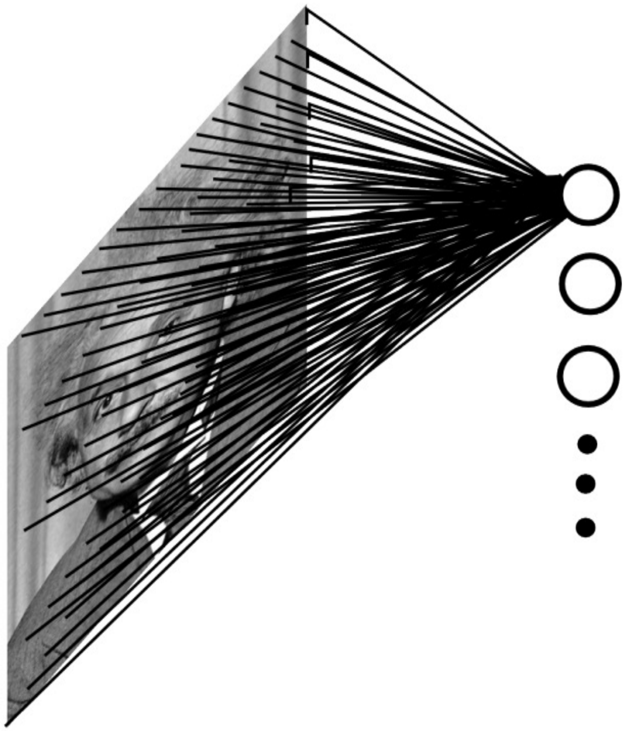
- The **receptive field** of an individual sensory neuron is the particular region of the sensory space (e.g. the retina) in which a stimulus will trigger the firing of that neuron.
  - It is a feature detector (image line orientation, sound frequency)
- How do we design “proper” receptive fields for the input neurons?
- Consider a task with image inputs
  - Receptive fields should provide expressive features from the raw input
  - How would you design the receptive fields for this problem?





Solution 1

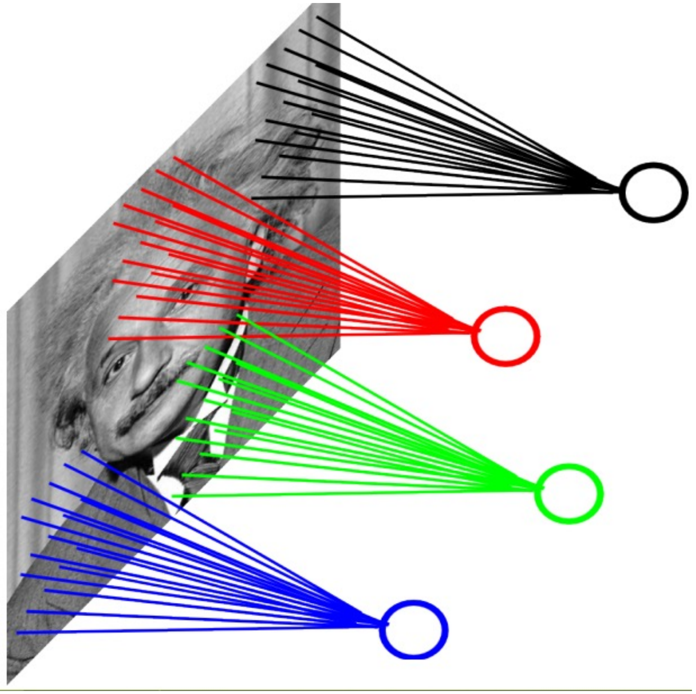
- A **fully connected layer and network**:
  - Example:
    - 100x100 images
    - 1000 units in 1st hidden layer
  - Problems:
    - 10^7 edges!
    - Local spatial correlations lost!
    - Variable sized inputs.



Slide Credit: Marc'Aurelio Ranzato

Solution 2

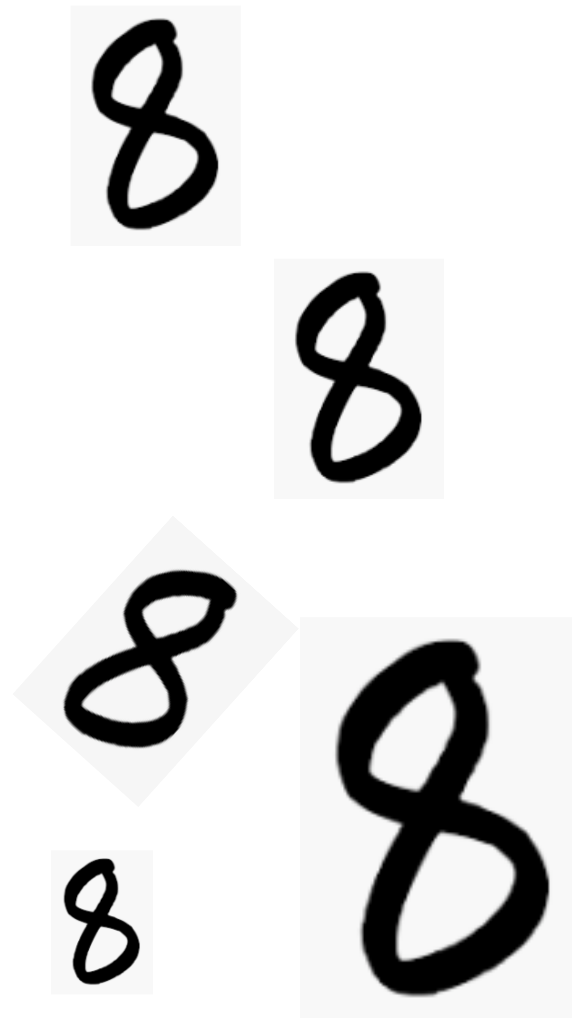
- A **locally connected layer**:
  - Example:
    - 100x100 images
    - 1000 units in the input
    - Filter size: 10x10
  - Local correlations preserved!
  - Problems:
    - 10^5 edges
    - This parameterization is good when input image is registered (e.g., face recognition).
    - Variable sized inputs, again.



Slide Credit: Marc'Aurelio Ranzato

Variation of Objects In Images

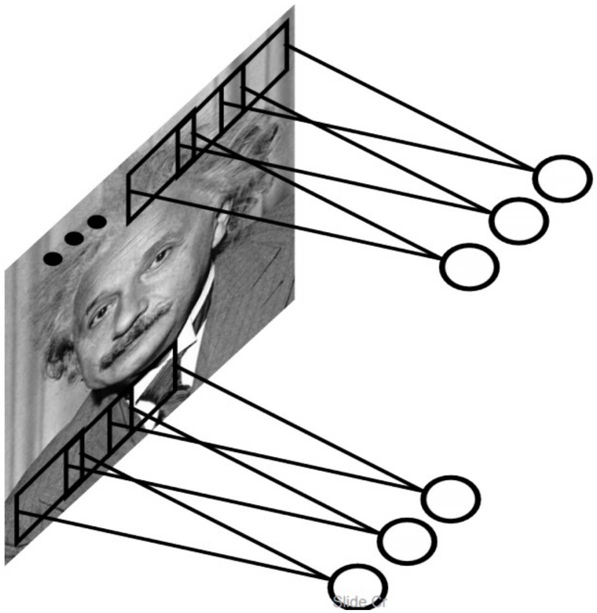
- Fixed objects can vary in terms of:
  - Translation (location)
  - Rotation
  - Scale
- Handwritten digits can vary due to nuances of pen stroke



Better Solution - Convolution

- **A solution:**
  - **Filters** to capture different patterns in the input space.
    - **Share** parameters across different locations or rotations
    - **Convolutions** with learned filters
  - Filters will be **learned** during training.
  - The issue of variable-sized inputs will be resolved with a **pooling** layer.

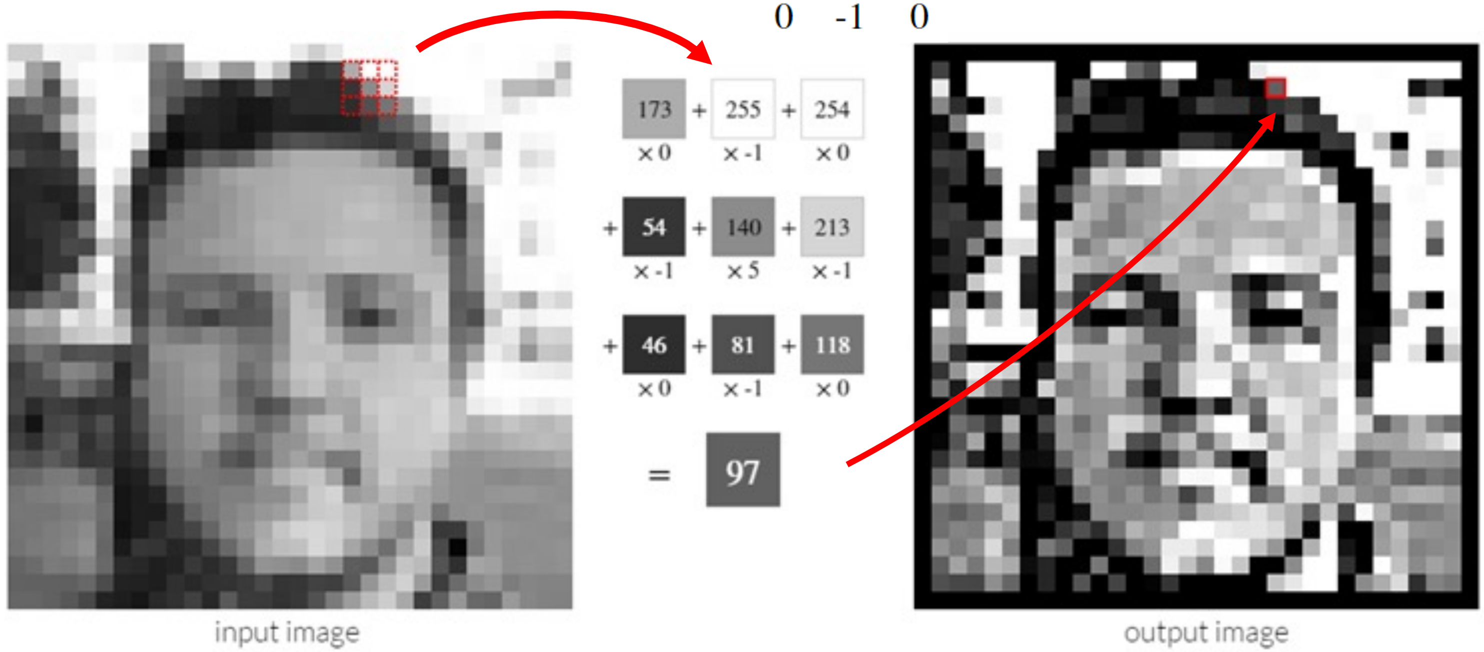
So what is a convolution?



Slide Credit: Marc'Aurelio Ranzato

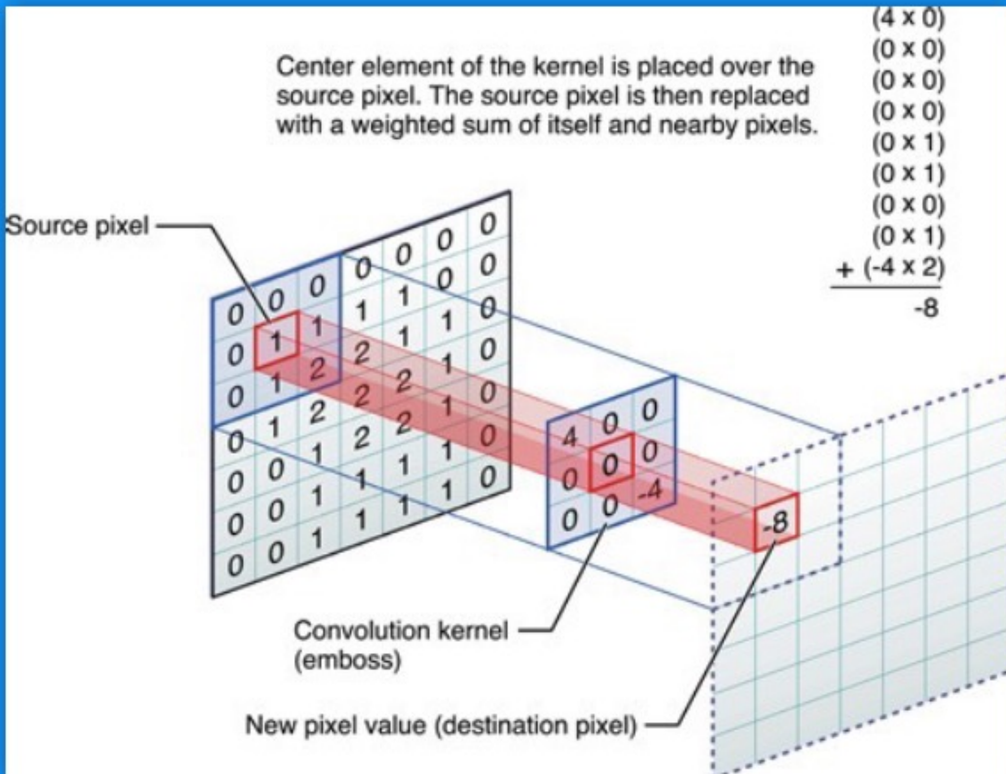
Convolution Operator

- Convolution in two dimension:
  - Takes two functions and produces another function
  - 2D: Take one matrix and slide it over the other matrix
  - Example: Sharpen kernel: $\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$



Try other kernels: http://setosa.io/ev/image-kernels/

Convolution Operator



Example of Blurring Filter

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

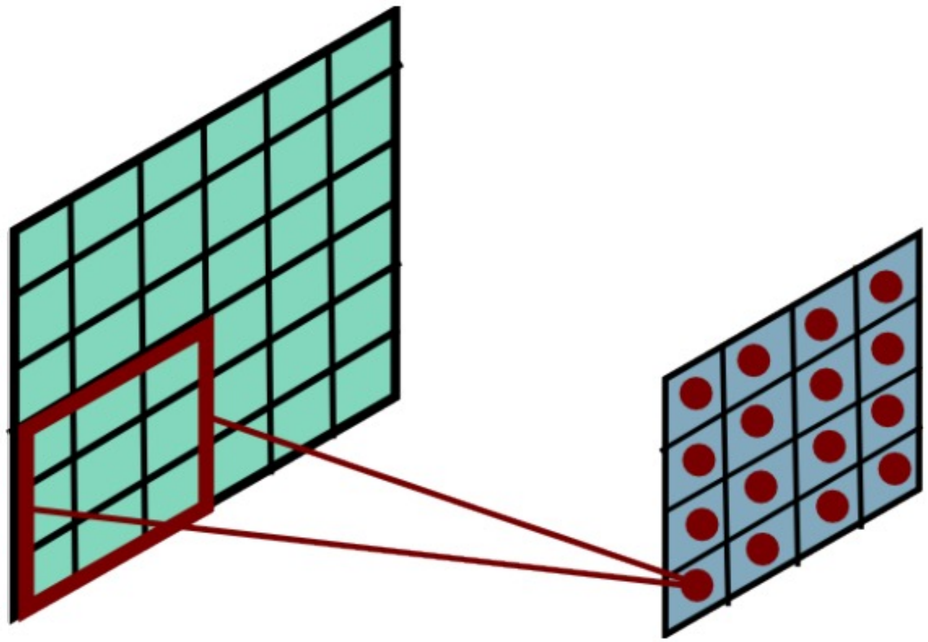
Example of Edge Detector

	0	1	0	
	1	-4	1	
	0	1	0	

Convolution Operator

- Convolution in two dimension:
  - The same idea: flip one matrix and slide it on the other matrix

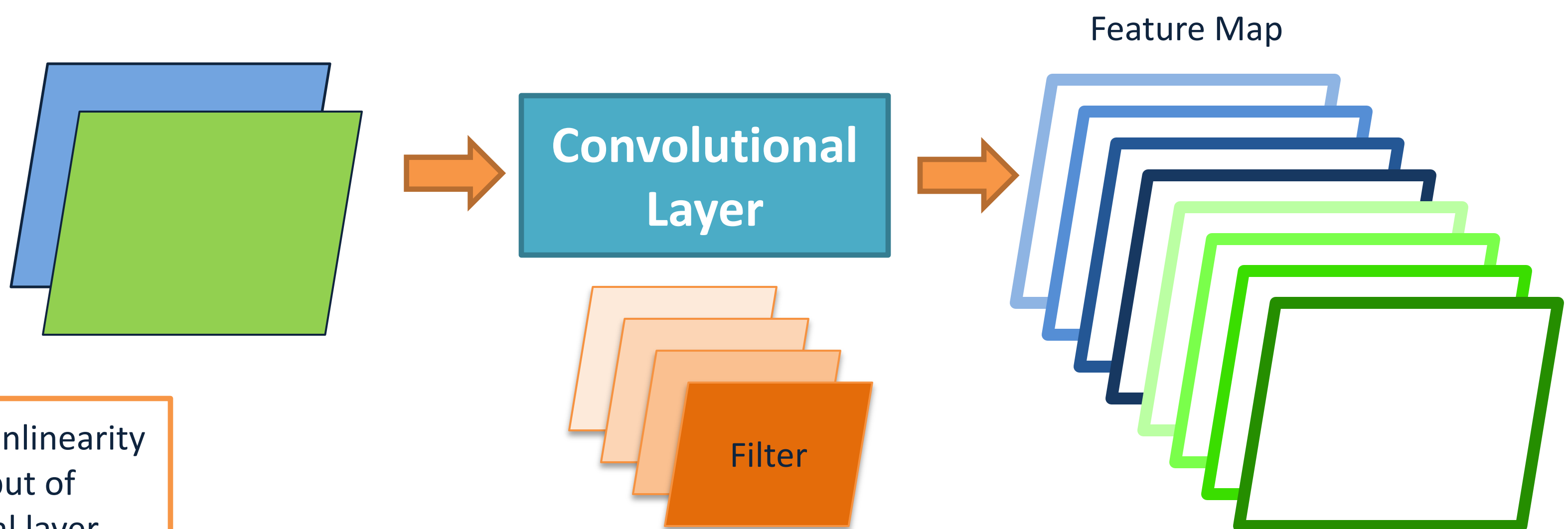
6 x 6 x 1 image



Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer

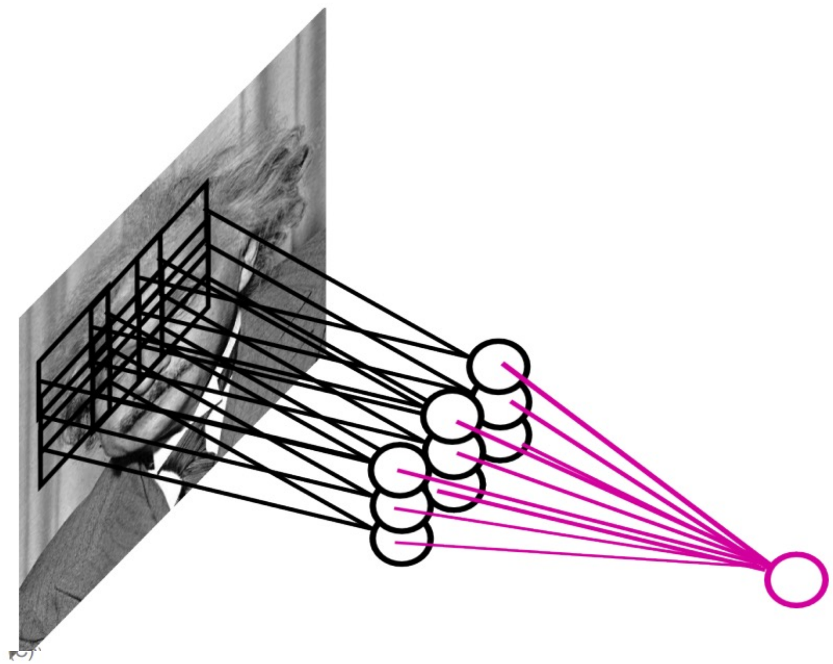
- The convolution of the **input (vector/matrix)** with weights **(vector/matrix)** results in a **response vector/matrix**.
- We can have **multiple (weight-based) filters** in each convolutional layer, each producing an output **feature map**
- If it is an intermediate layer, it can have **multiple inputs**



One can add nonlinearity at the output of convolutional layer

Pooling Layer

- How to handle variable sized inputs?
  - A layer which reduces inputs of different size, to a fixed size.
  - **Pooling (also called subsampling)**



Slide Credit: Marc'Aurelio Ranzato

Pooling Layer

- How to handle variable sized inputs?
  - A layer which reduces inputs of different size, to a fixed size.
  - **Pooling (also called subsampling)**
  - Different variations
    - Max pooling

$$h_i[n] = \max_{i \in N(n)} \tilde{h}[i]$$

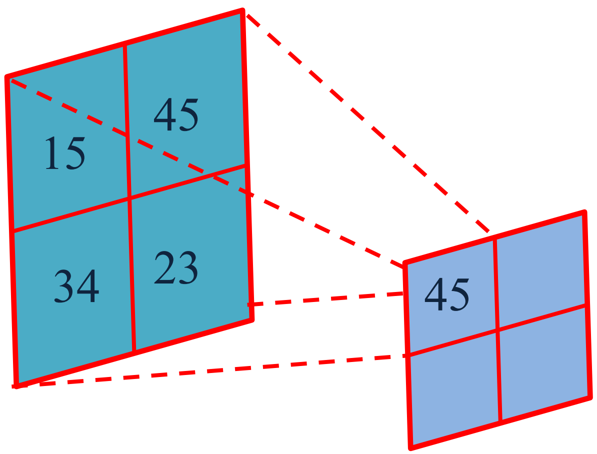
    - Average pooling

$$h_i[n] = \frac{1}{n} \sum_{i \in N(n)} \tilde{h}[i]$$

    - L2-pooling

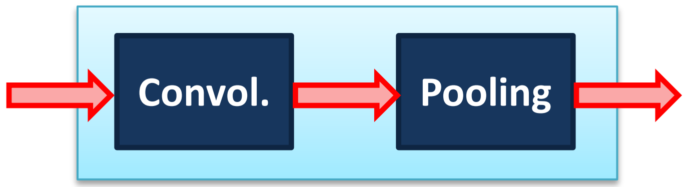
$$h_i[n] = \frac{1}{n} \sqrt{\sum_{i \in N(n)} \tilde{h}^2[i]}$$

    - etc

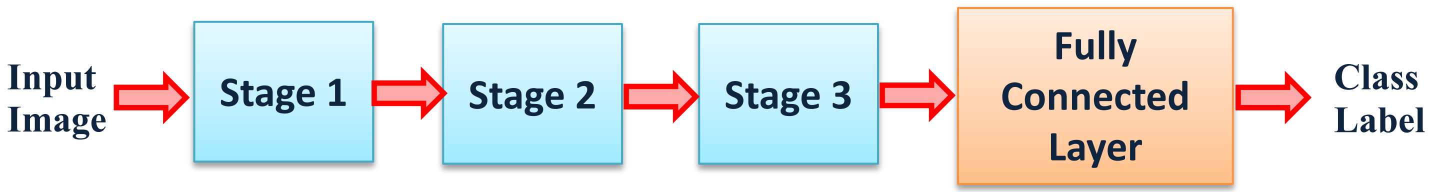


Convolutional Nets

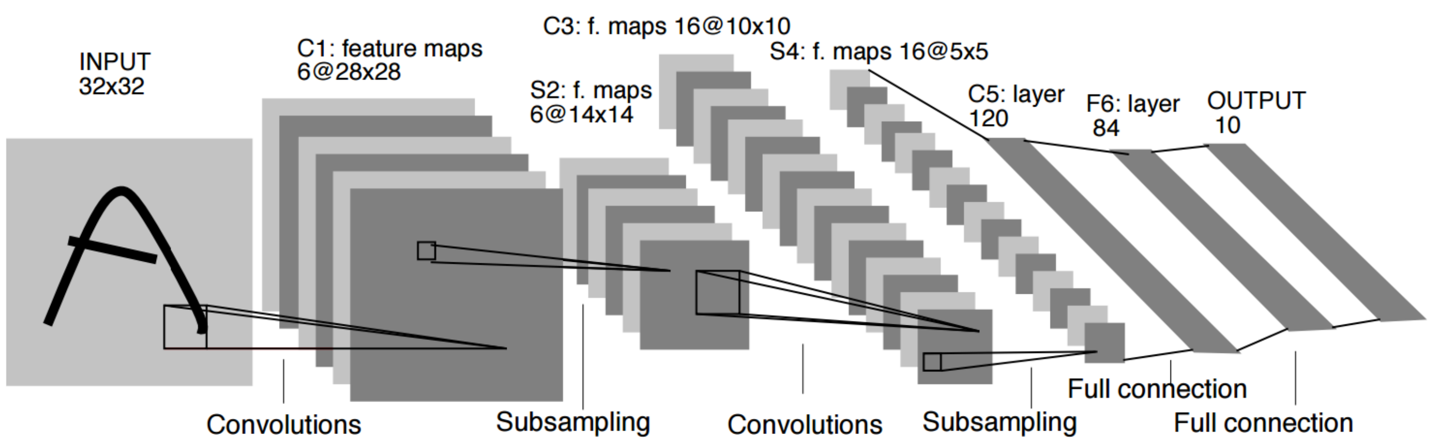
- One stage structure:



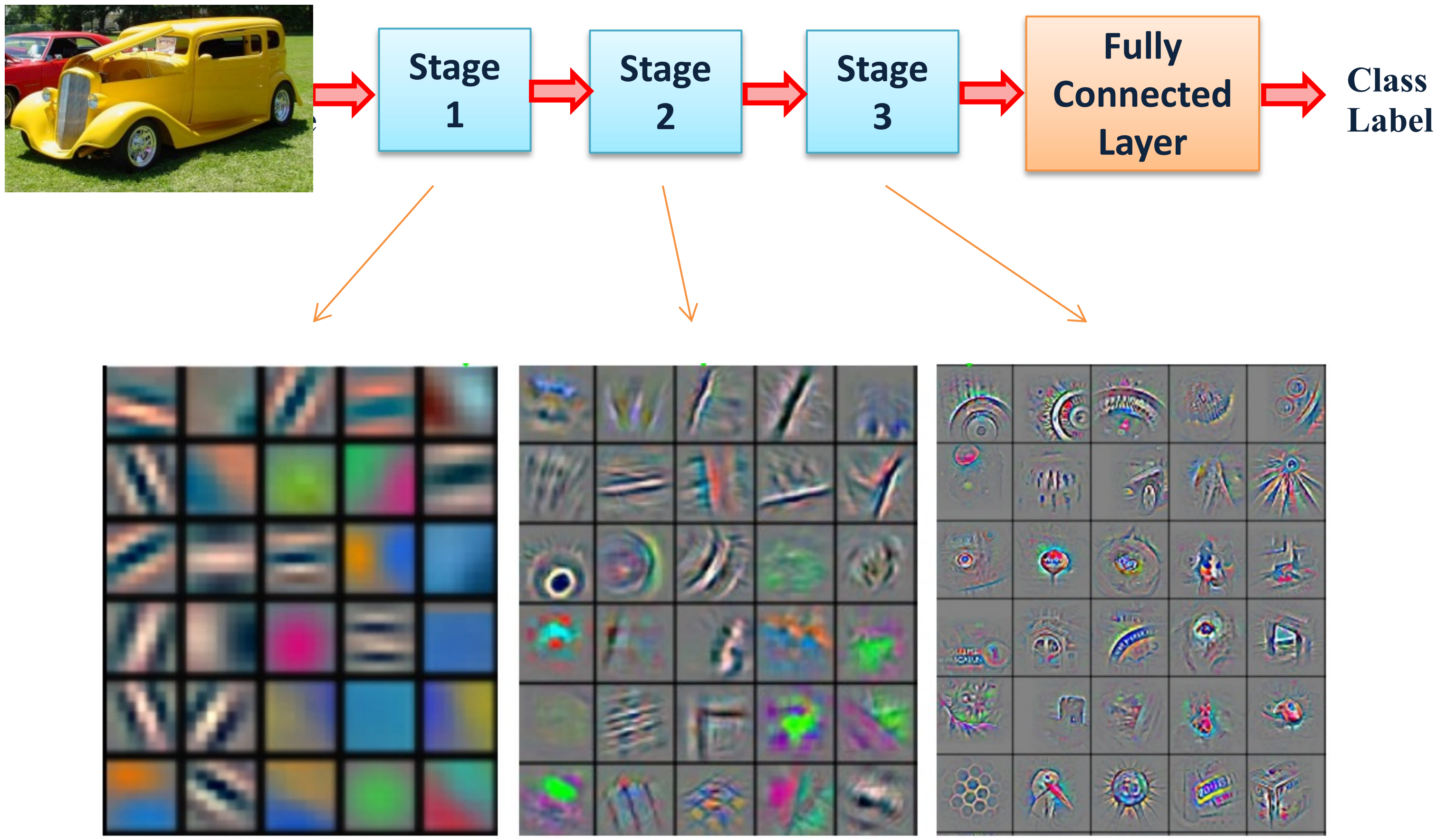
- Whole system:



An example system (LeNet):



Convolutional Nets

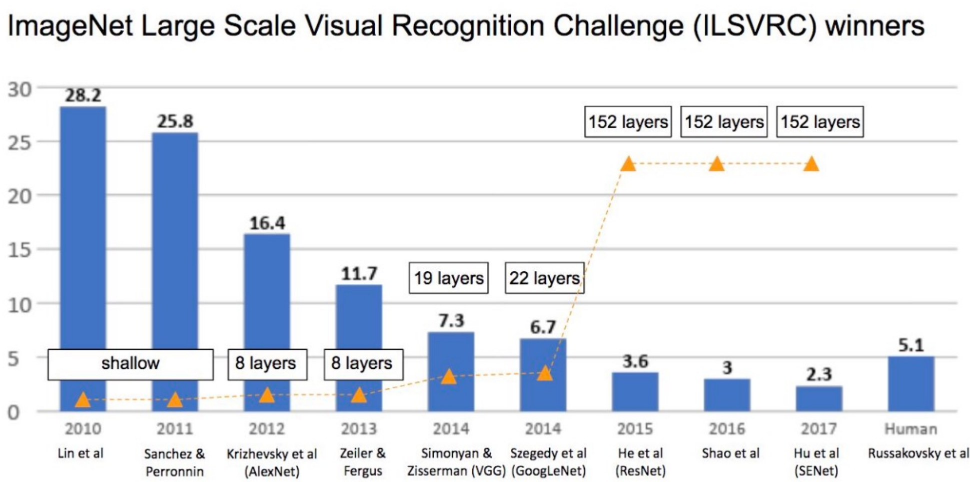


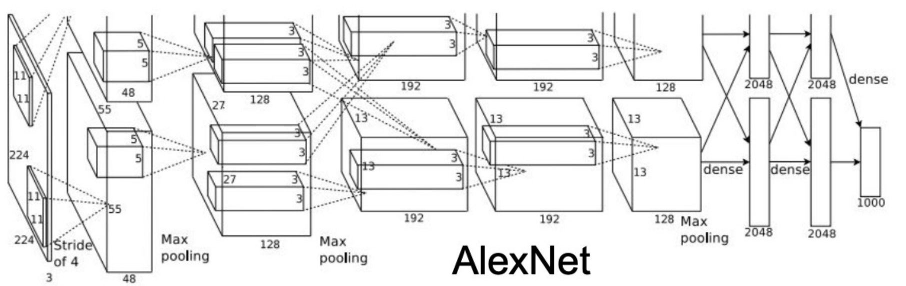
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

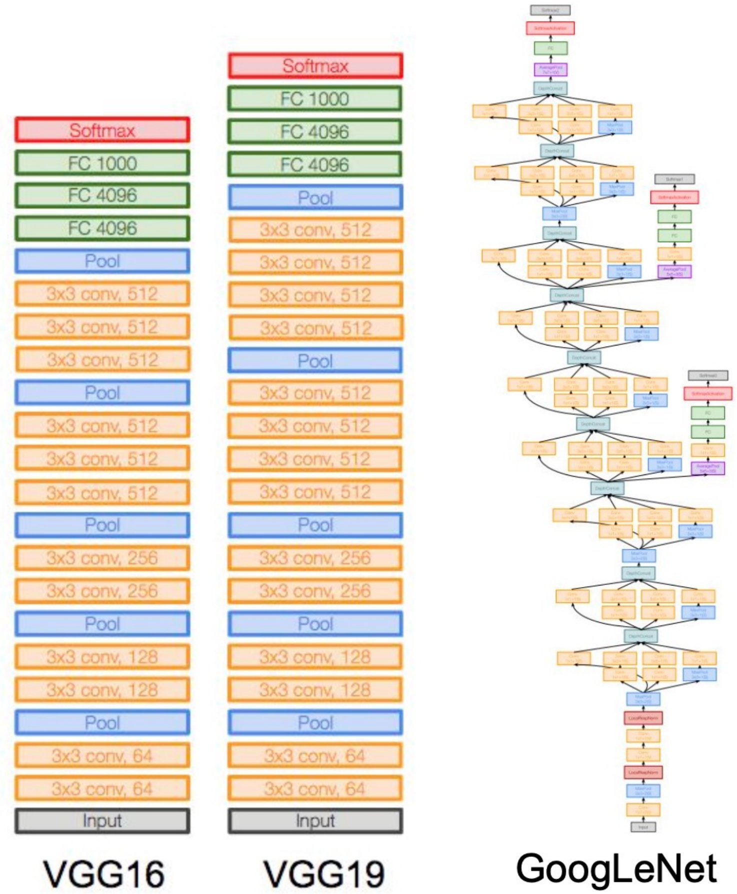
An example system :

Convolutional Nets

Advances in CNNs



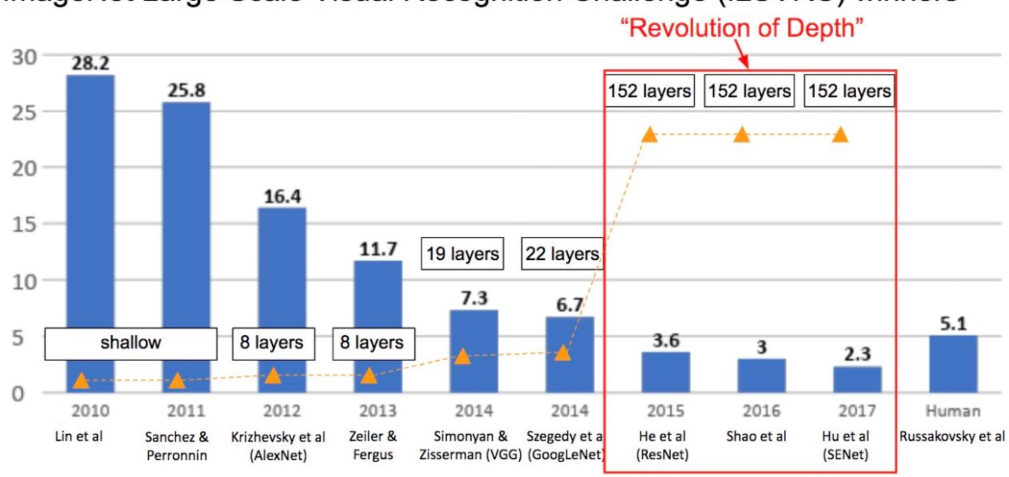


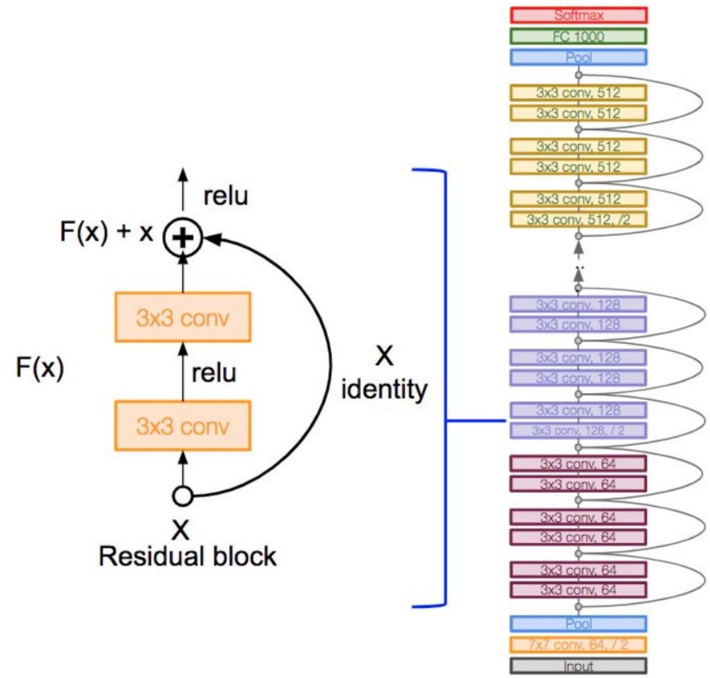


[Source: Fei-Fei Li, Ranjay Krisna, Danfei Xu / Stanford]

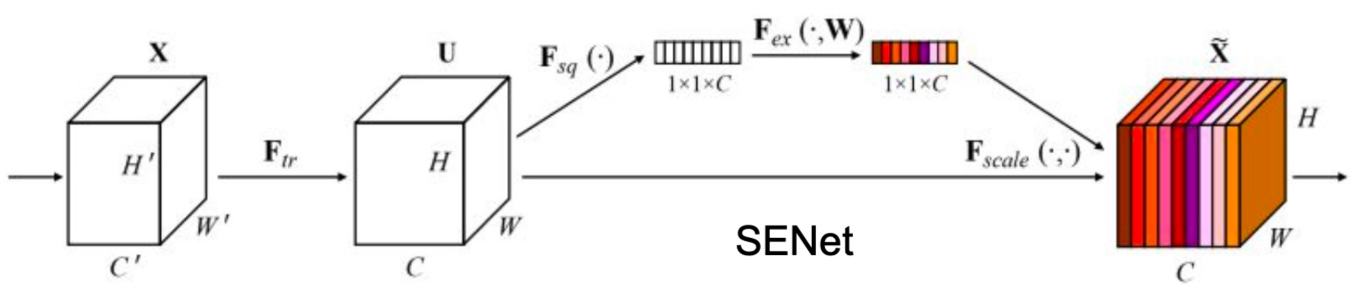
Advances in CNNs

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners





ResNet



[Source: Fei-Fei Li, Ranjay Krisna, Danfei Xu / Stanford]

Practical Tips

- Before large scale experiments, test on a small subset of the data and check the error should go to zero.
  - Overfitting on small training
- Visualize features (feature maps need to be uncorrelated) and have high variance
- Bad training: many hidden units ignore the input and/or exhibit strong correlations.

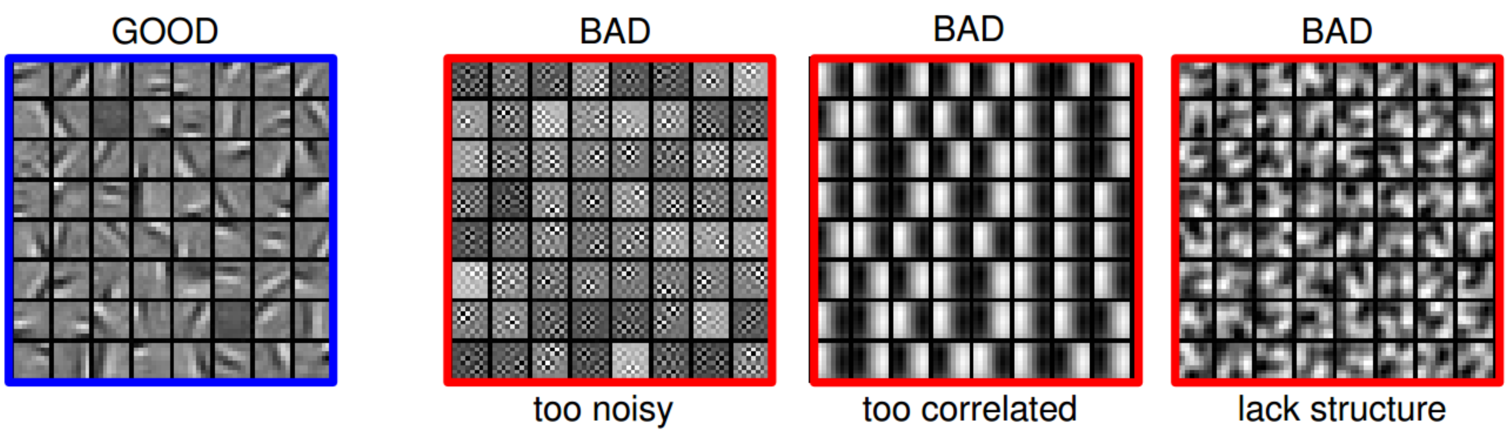
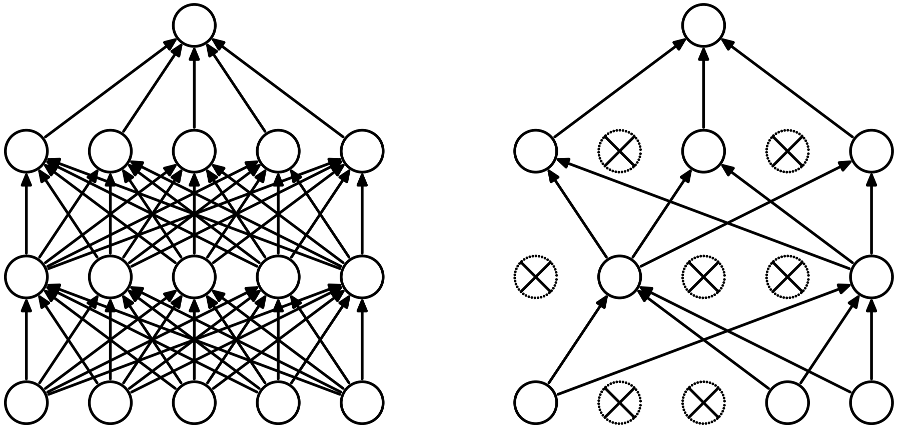


Figure Credit: Marc'Aurelio Ranzato

Dropout typically useful



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

TUTORIAL 4

- Develop and train a CNN network using Keras and Tensorflow (keras_mnist_cnn_val.ipynb)

References

- Demo: https://www.cs.cmu.edu/~aharley/vis/conv/
- Further Reading:
  - https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
  - https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/