Deep Learning - 2019

Convolutional Neural Networks

Prof. Avishek Anand

What we did last time…

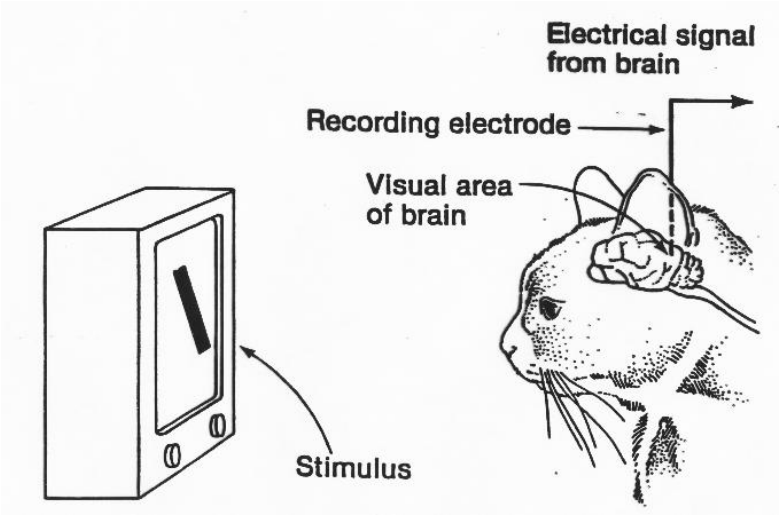
- Perceptrons as simple models for supervised learning
- Fully Connected Networks
  - Linearization of inputs
  - Deep Neural Networks as function compositions
  - Role on activation functions and non-linearity

- Backpropagation as a method for learning model parameters
- Optimization techniques for training Neural Networks (Overview)
- Overview of a key Regularization technique called Dropout
  - Early stopping

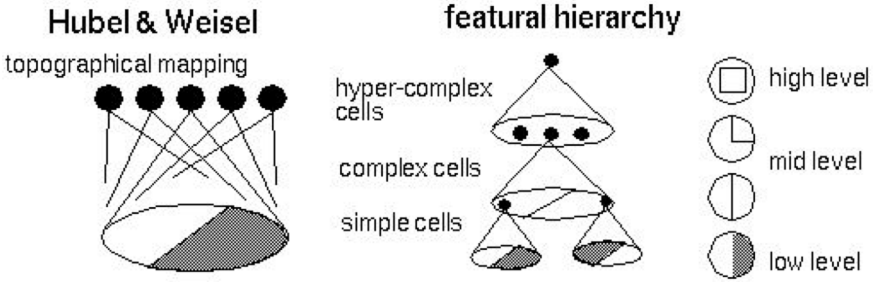
Today's topic is a special kind of NN called Convolutional Neural Networks

Motivations to CNNs

- Biological inspirations
  - Work of Hubel and Wiesel in cat visual cortex (Nobel Prize)
  - Discovery of neurons that are tuned to the movement of bars at fixed orientation and specific location (simple cells)
  - Discovery of neurons that are more location invariant (complex cells)



https://goodpsychology.wordpress.com/2013/03/13/235/



http://cns-alumni.bu.edu/~slehar/webstuff/pcave/hubel.html

[Hubel and Weisel '60-'70]

Motivations to CNNs

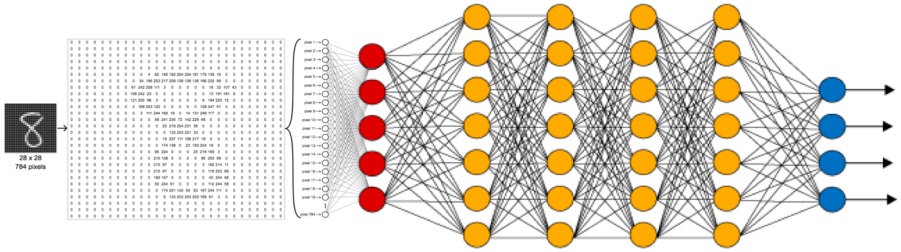
- Statistics of local images
  - Many properties of the image patches are the same independent of their position
  - Intuitively, objects can appear at different locations in the image, and they can also appear at different distance away from the camera

- Hypothesis of hierarchical representations
  - There is a biological evidence for that, e.g., the Hubel & Weisel model
  - hard to be represented by non-hierarchical models (exponential number of parameters) and easily represented by hierarchical models (polynomial number of parameters)

[Ruderman 1994, Hyvarinen et al. 2009]

Motivations to CNNs

- Statistical efficiency argument
  - Modeling grayscale images with shape W x H with a fully connected network with K layers, Dimensionality of D would require
    - W*H*D + K*D*D parameters



- Can we reduce number of parameters to avoid overfitting?
    - This requires to introduce some constraints on the architecture of our neural networks

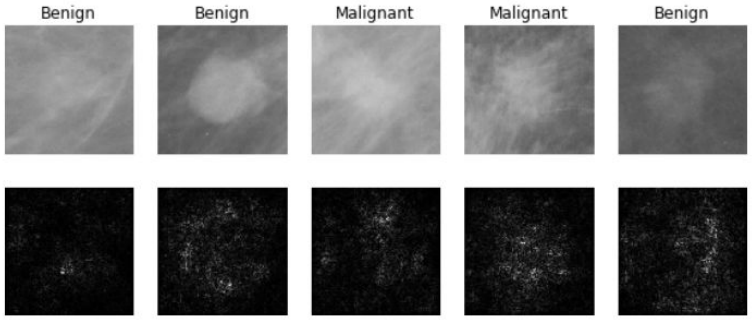
- Can we use locality as a constrain?
    - Fully connected network only "looks at" patches instead of the full image
    - We can "move this" network over the whole image to build the image representation

Computational efficiency argument

- How to speed up computations?

- Instead of having one large tensor (fully connected network) we could have M smaller tensors each doing P*P*K computations (P is the dimension of the patch, we can assume quadratic patches for simplicity)
  - Also better parallelism
  - Smaller memory footprint
  - Better caching
  - Tuning M and P is often easier and better than W and H that requires changes in resolution, e.g., downscaling

CNNs are fundamental in visual systems



[Levy et al. 2016]

Figure copyright Levy et al. 2016. Reproduced with permission.

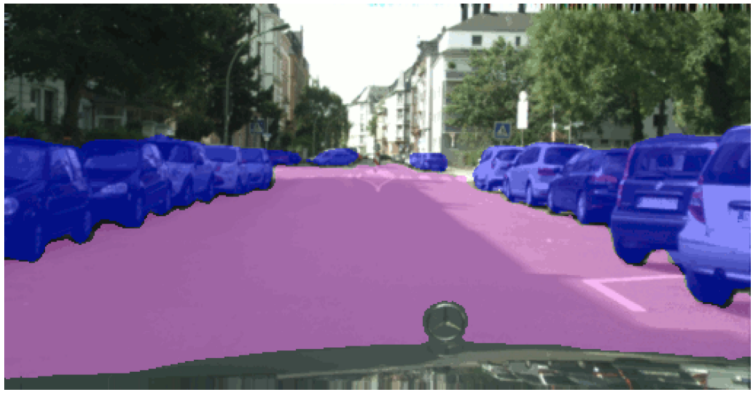
Cancer detection

A man riding a wave on top of a surfboard

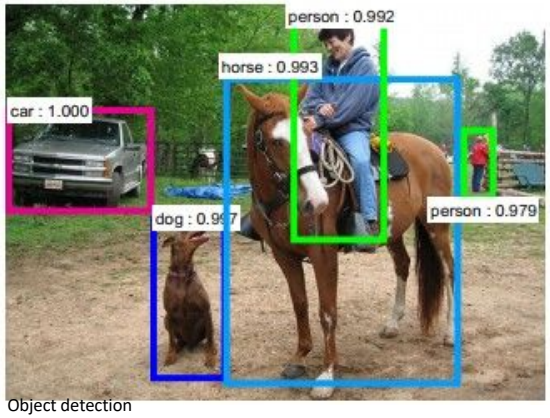
Image Captioning

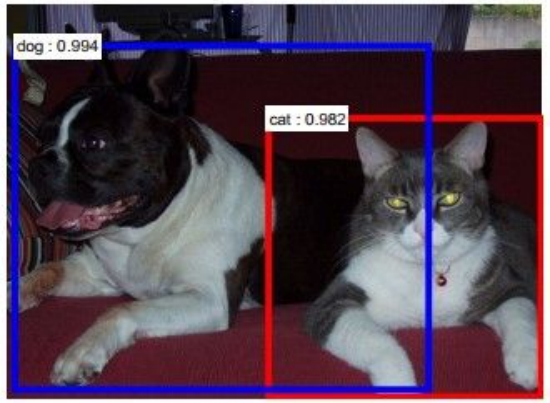
A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard



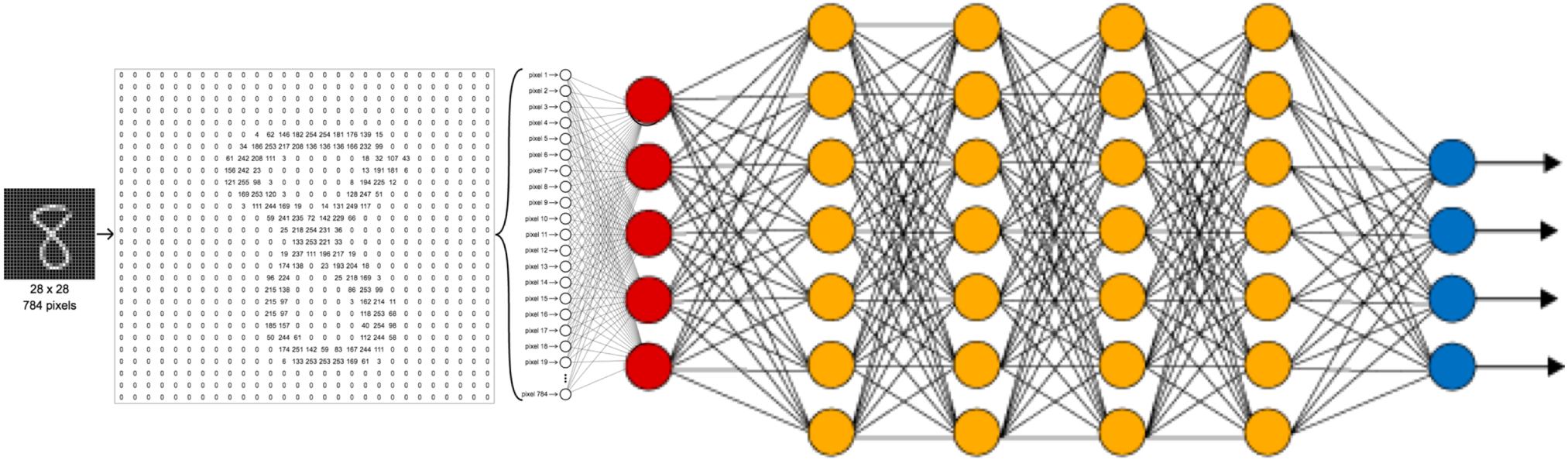
Semantic Segmentation (can be used in Self-driving cars)





Object detection

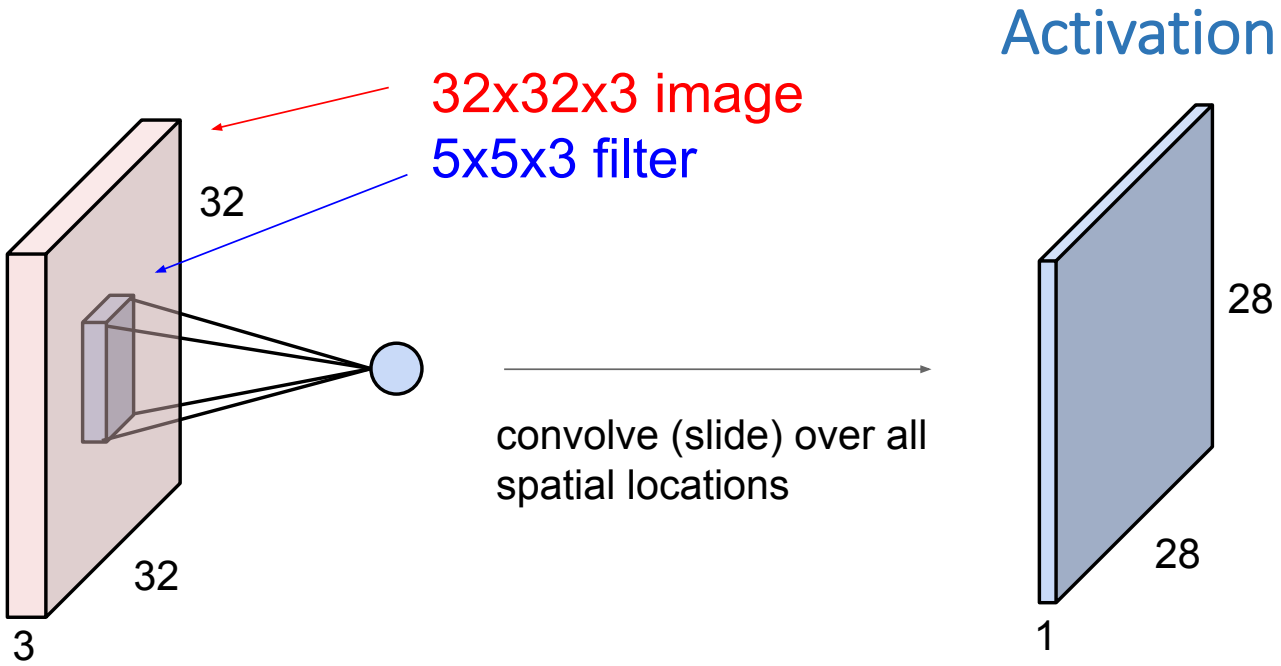
Fully Connected NN



*The entire image input is **linearized** into an array of length 784*

No locality preservation

Convolution Layer

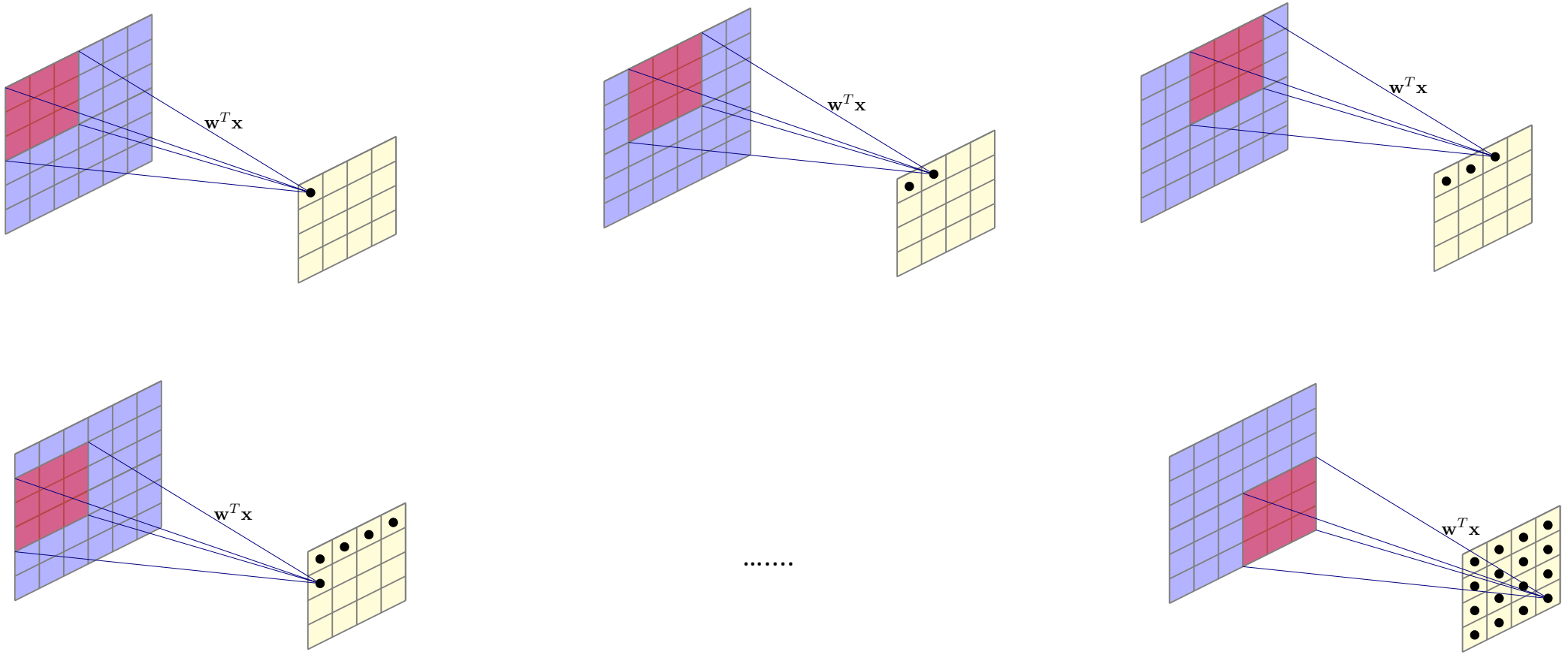


Convolve the filter/kernel with the image by sliding the filter w

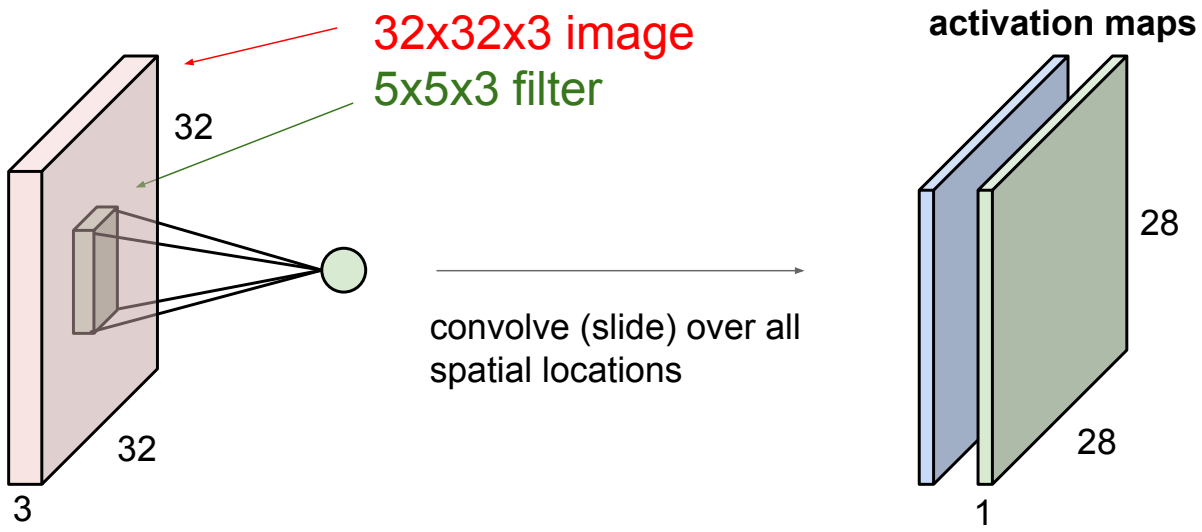
For each configuration, compute the dot product $w^T x + b$

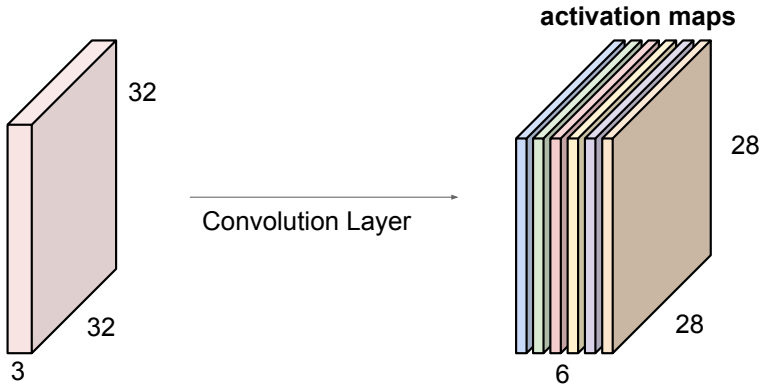
Why the dimensions are 28x28x1 ?

Convolution Layer



Multiple filters





Multiple filters can be used over the same input image

6 separate activation maps

Why the dimensions are 28x28x1 ?

Convolutions with filters

Original

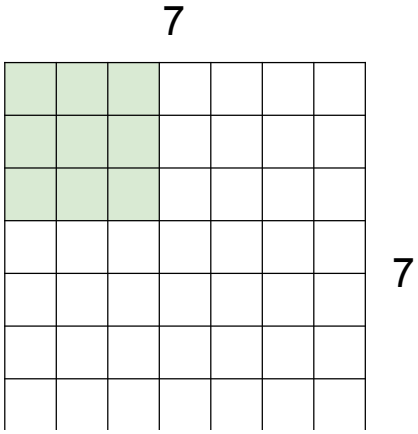
Sharpen

Edge Detect

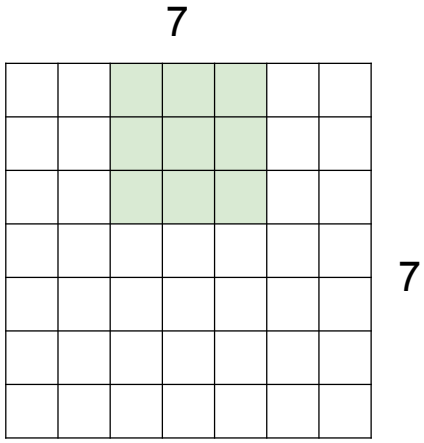
Strong Edge Detect

- We can hand-craft filters and use convolutions to process images
  - Apply a set of weights – a filter – to extract local features
  - Use multiple filters to extract different features
  - Spatially share parameters of each filter
- The idea of convolving image with filters in CNNs is the same, but we want to learn the filters from data, and hence specific for the task

Dimensions



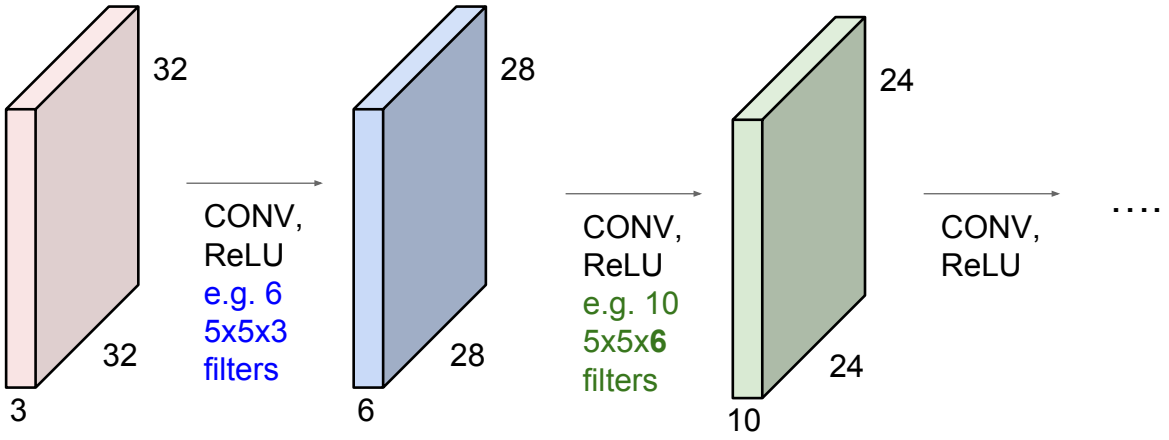
7x7 input with a *filter 3x3* with *stride 1.*
What is the output size ?



7x7 input with a *filter 3x3* with *stride 2.*
What is the output size ?

(N-F) / Stride + 1

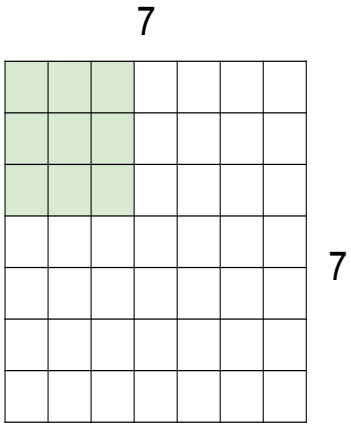
ConvNet



A convnet is a sequence of convolution layers interspersed with activation functions (ReLU)

Convolution of two signals

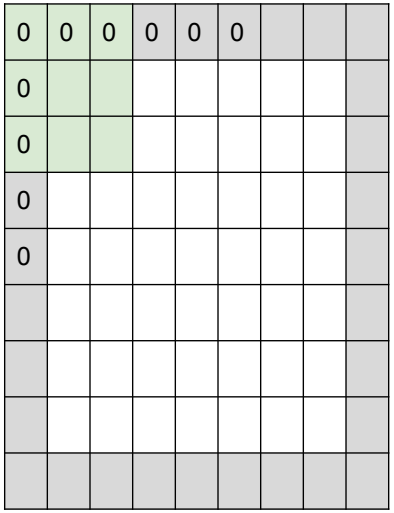
Zero Padding



7x7 input with a *filter 3x3* with *stride 3.*

What is the output size ?

(N-F) / Stride + 1



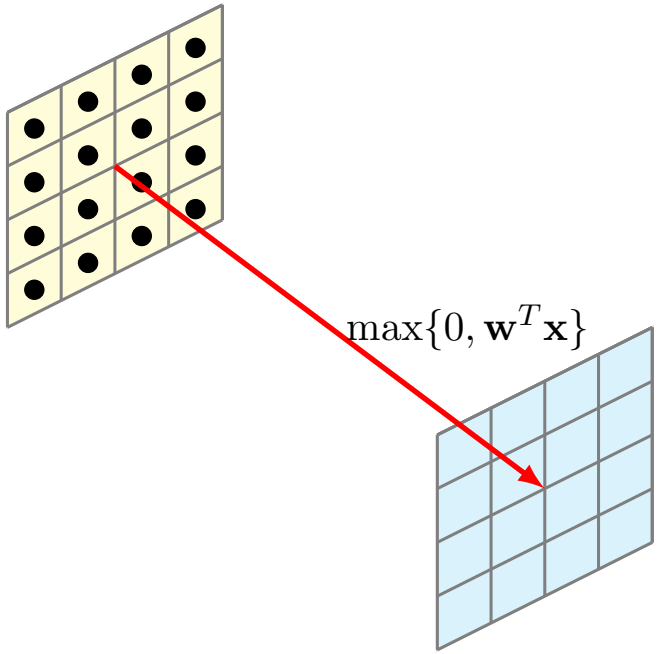
7x7 input + 0 padding with a filter 3x3 with stride 1.

What is the output size ?

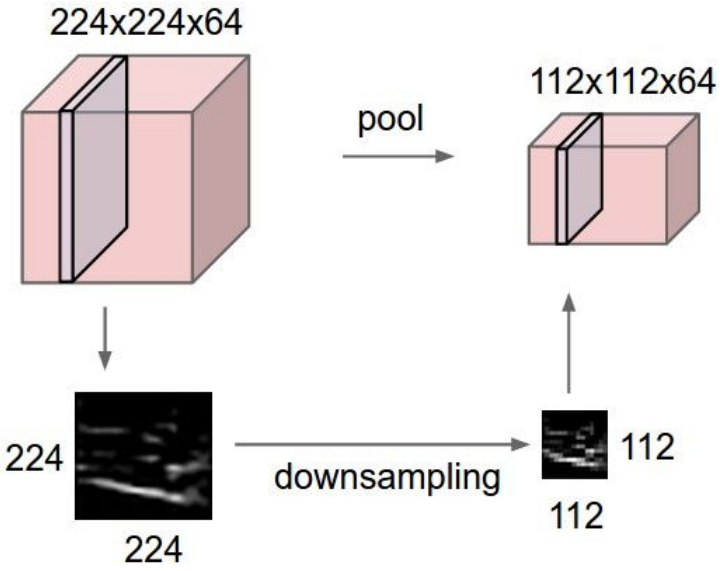
Common design choice: *CONV layers with stride 1 or 2, filters of size F and zero-padding =* (F-1)/2

Non-Linearity

- After obtaining feature map, apply an elementwise non-linearity to obtain a transformed feature map (same size).
- Often RELU (max{0,x}) is often used due to some nice properties
  - speed, gradient vanishing, sparsity, scale invariance



Pooling Layer



- Makes representations more manageable
- Depth doesn't change
- Spatial dimensions change
- Increases invariance ("Complex cells" in the Hubel & Wiesel model), but loses information

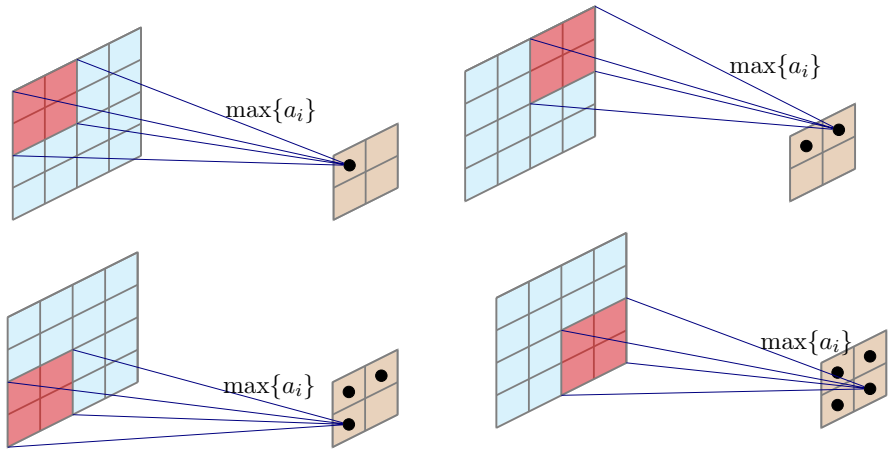
Max Pooling

Single depth slice

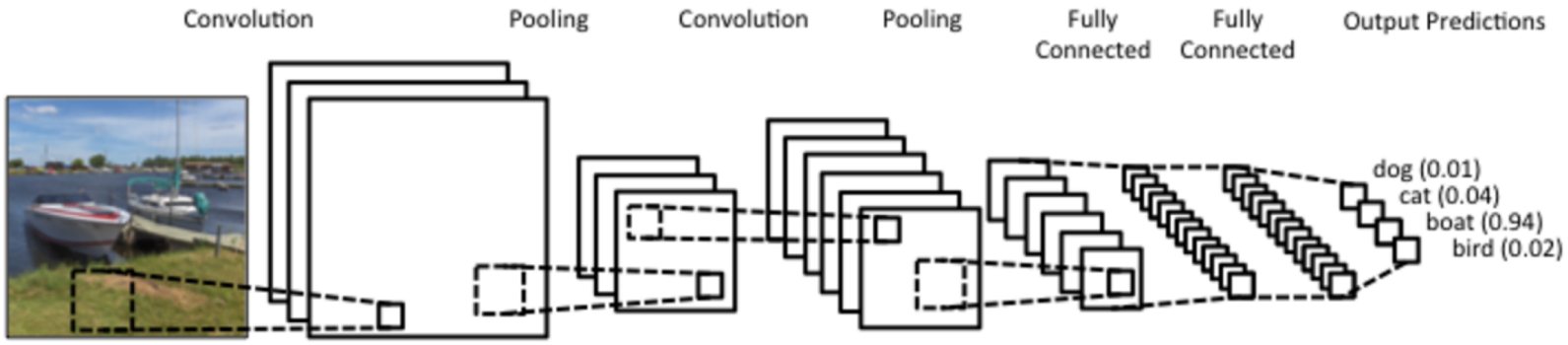
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters and stride 2

6	8
3	4



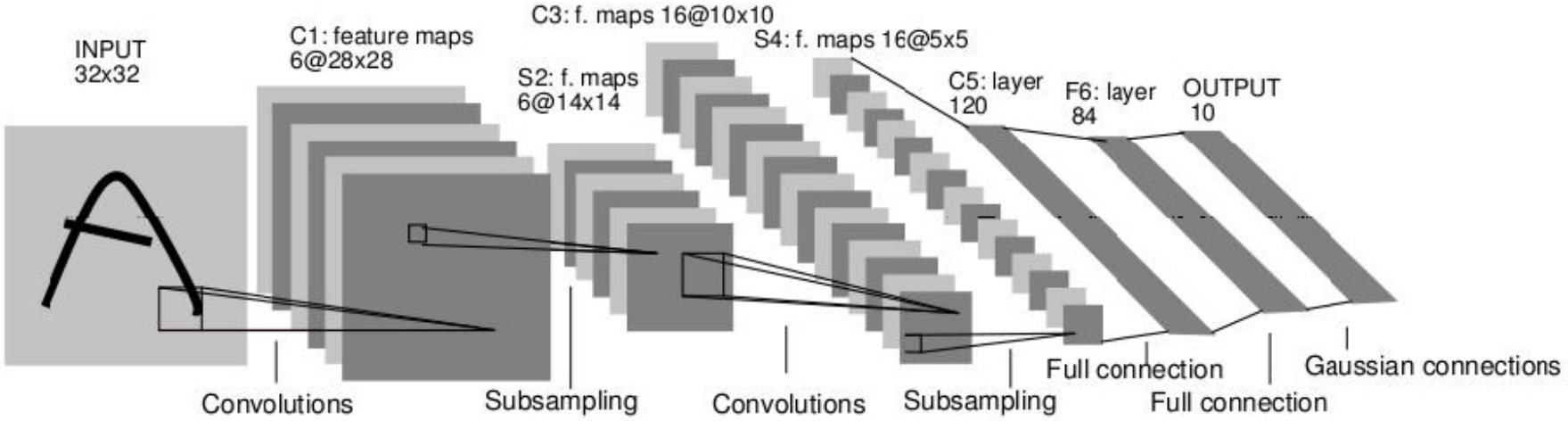
CNN Pipeline



- Convolution: Apply filters with learned weights to generate feature maps
- Non-linearity: Often ReLU
- Pooling: Downsampling operation on each feature map
- Fully Connected Layers: at the end for classification

- Train model with image data and learn weights of filters in the convolution layers

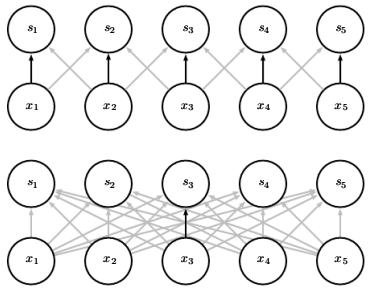
LeNet 1989 (Homework)



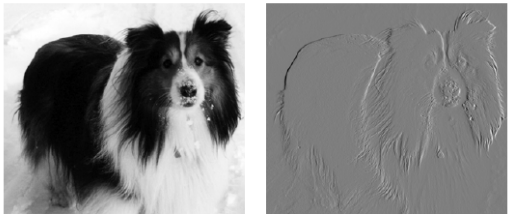
- Filters are of size 5 × 5, stride 1 Pooling is 2 × 2, with stride 2.
- How many parameters?

Why Convolution ? Parameter Sharing

- **Traditional NN:** Each weight/parameter used only once (multiplied by one element of the input and then never revisited)



- Parameter sharing (tied weights) refers to using the same parameter for more than one function in a model
  - Rather than learning a separate set of parameters for every location, we learn only one set



Filter of size 2 (250k vs 8 Bi. Params)

- Sparse connections due to small filter size
  - Can detect small, meaningful features such as edges with small kernels
  - Store fewer parameters -- O(m × n) versus O(k × n)
  - Low memory requirements, Faster training and inference

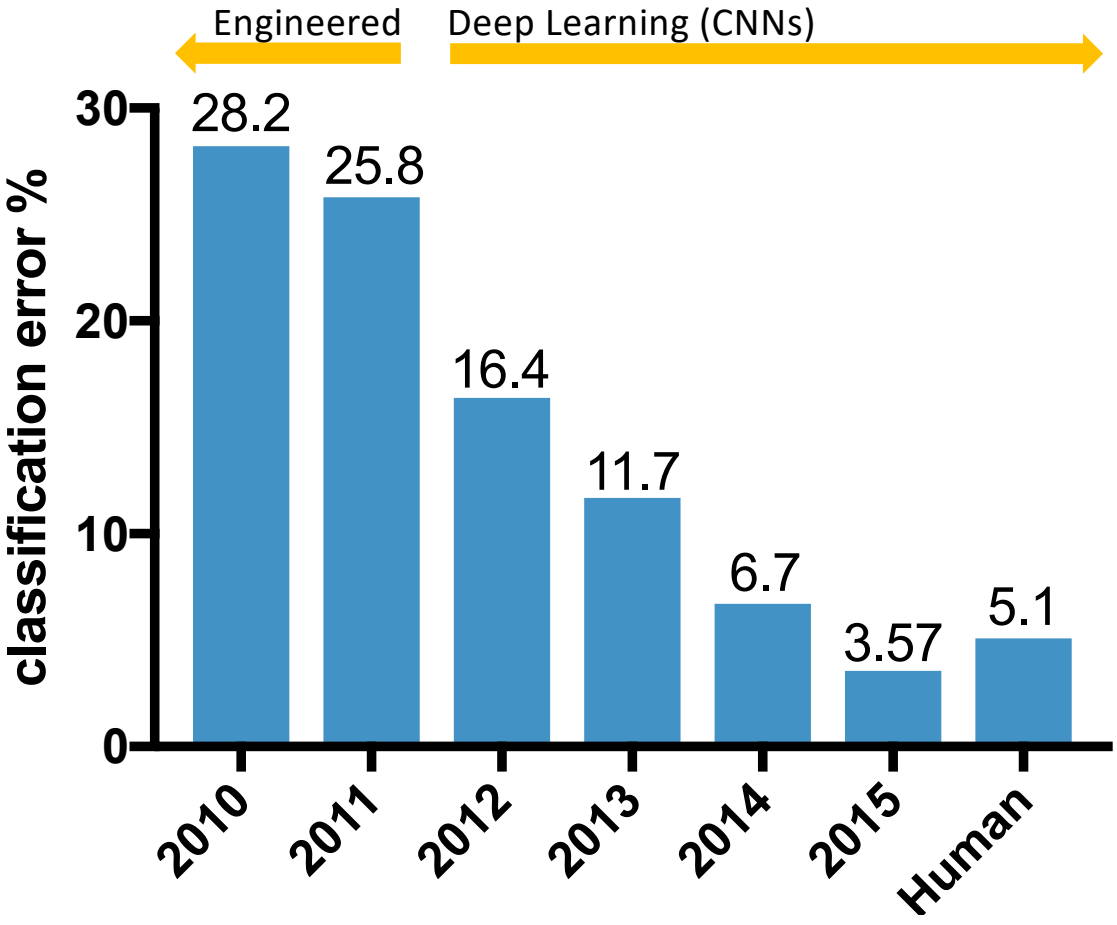
Why Convolution ? Equivariance

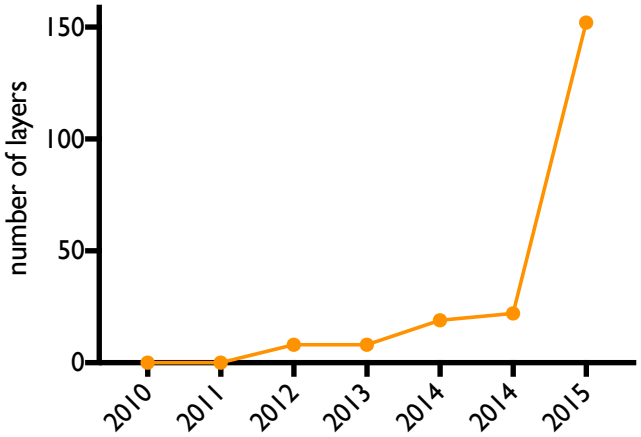
- Equivariance: $f(T(x)) = T(f(x))$
- CNNs equivariance to translation: Unshifted representation for the object same as the representation for the object after shifting
  - The form of parameter sharing used by CNNs causes each layer to be equivariant to translation
  - property is useful when we know some local function is useful everywhere (e.g. edge detectors)
    - There is some controversy regarding the shift equivariance of the most modern networks; e.g. when striding=2 is used
- Not naturally equivariant to scale or rotation of an image

ImageNet

- Dataset: With over 14 Million images across 21,841 categories
- **Classification task:** Produce a list of object categories present in image. (1000 categories)
- "Top 5 error": rate at which the model does not output correct label in top 5 predictions

Progress using CNNs

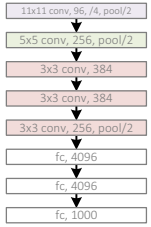




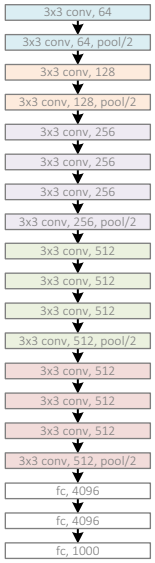
- Large datasets help
- Over parameterized deep layers help

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

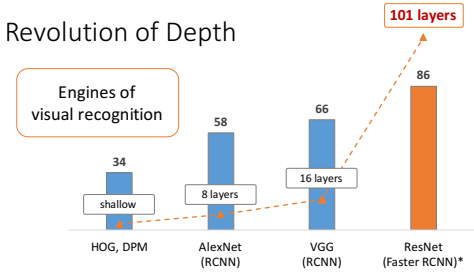


VGG, 19 layers
(ILSVRC 2014)



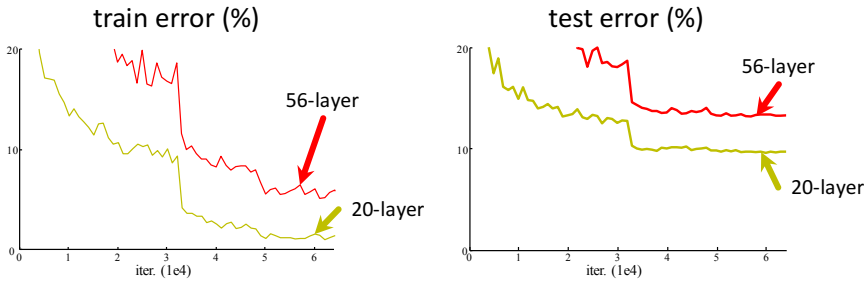
GoogleNet, 22 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)



But is learning as simple as stacking more and more layers ?

CIFAR-10



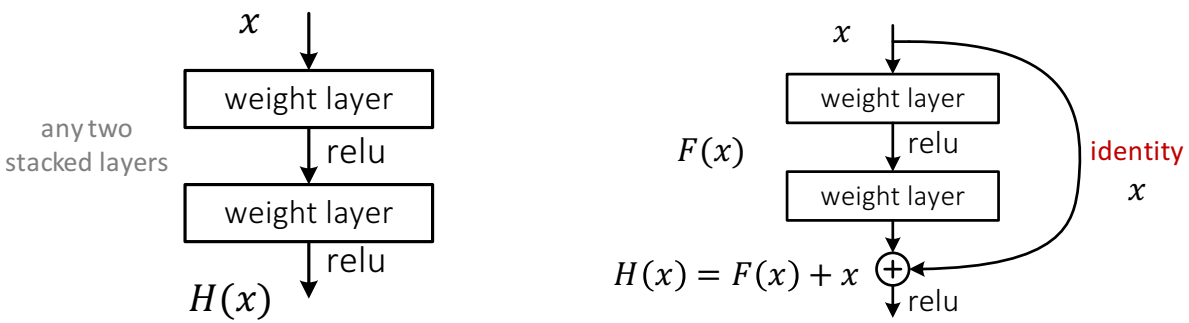
What are the problems in training Deep Nets ?

- Vanishing and Exploding gradients
  - The gradients coming from the deeper layers have to go through continuous matrix multiplications because of the the chain rule, and as they approach the earlier layers,
  - **Vanishing gradients:** if they have small values (<1), they shrink exponentially until they vanish and make it impossible for the model to learn
  - **Exploding gradients:** if they have large values (>1) they get larger and eventually blow up
- How can we control for this ?
  - Better initialization (next lecture)
  - Better normalization (next lecture)
  - Gating mechanism (lecture on RNNs)
  - Residual Connections

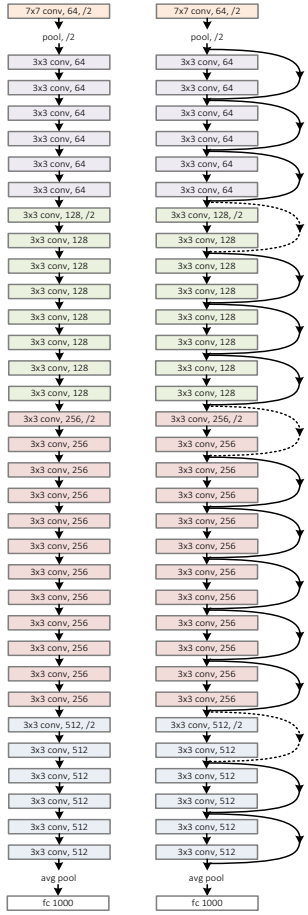
https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet-7c81c017b848

[He et al 2015]

Residual Connections



Standard Deep Conv Net



ResNet

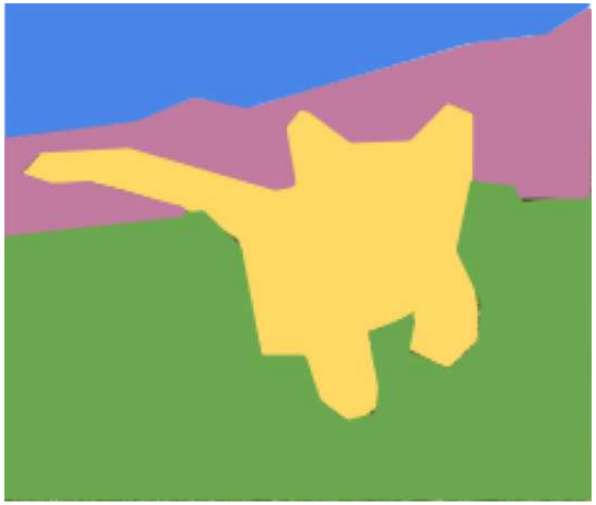
- **Residual network:** directly copy the input matrix to the second transformation output and sum the output in final ReLU with the residual operation
- Very deep and successful architecture

Intuition:

- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

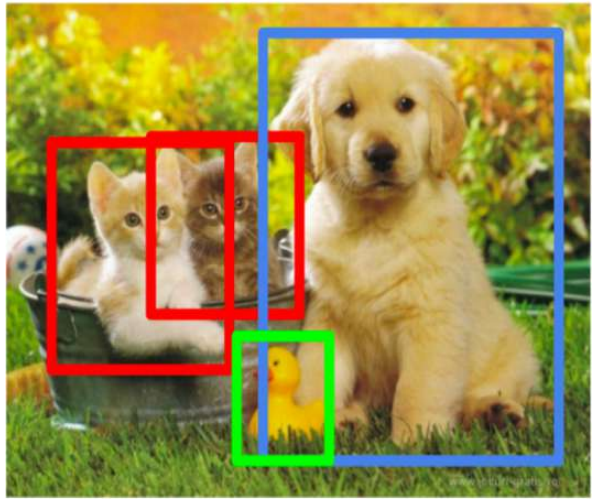
Beyond Classification

Semantic Segmentation



CAT

Object Detection



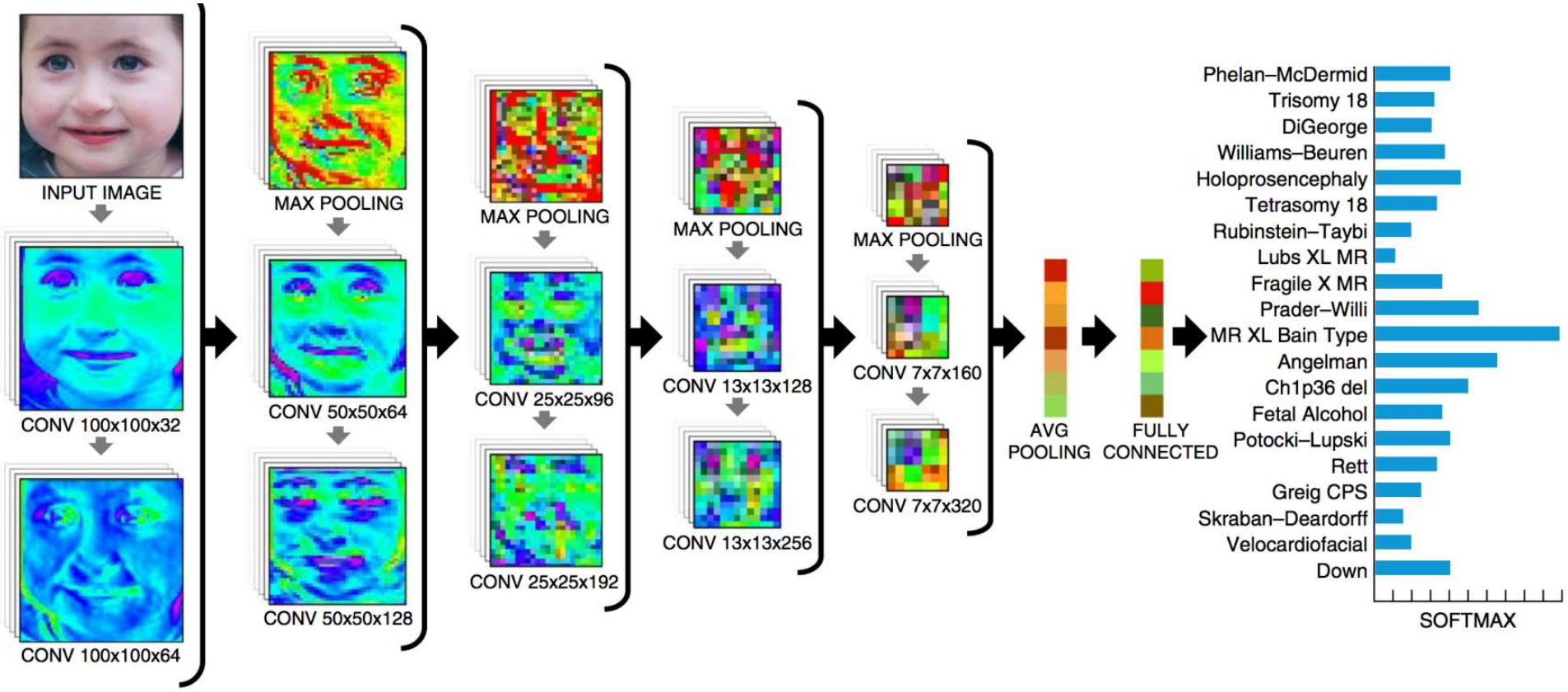
CAT, DOG, DUCK

Image Captioning

The cat is in the grass.

Healthcare

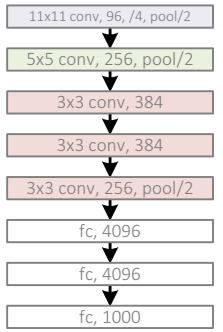
Identifying facial phenotypes of genetic disorders using DL

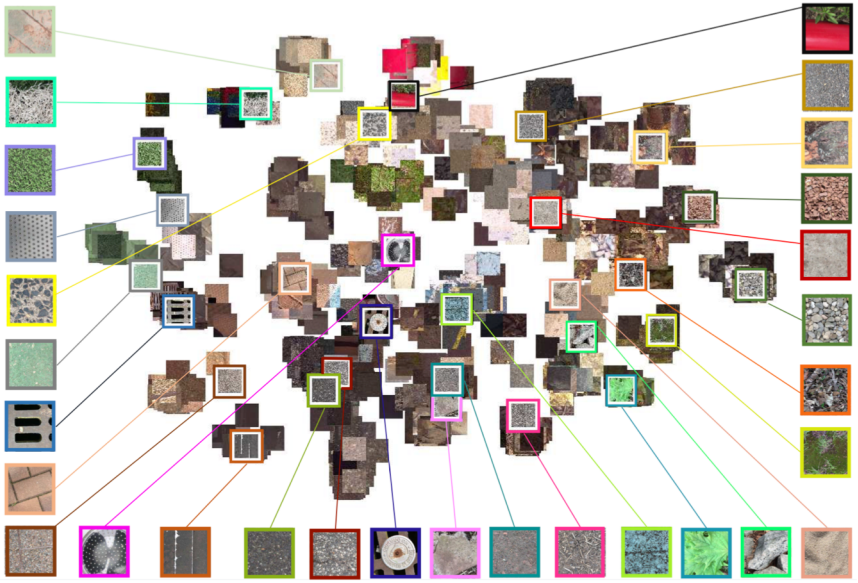


DL as representation Learning

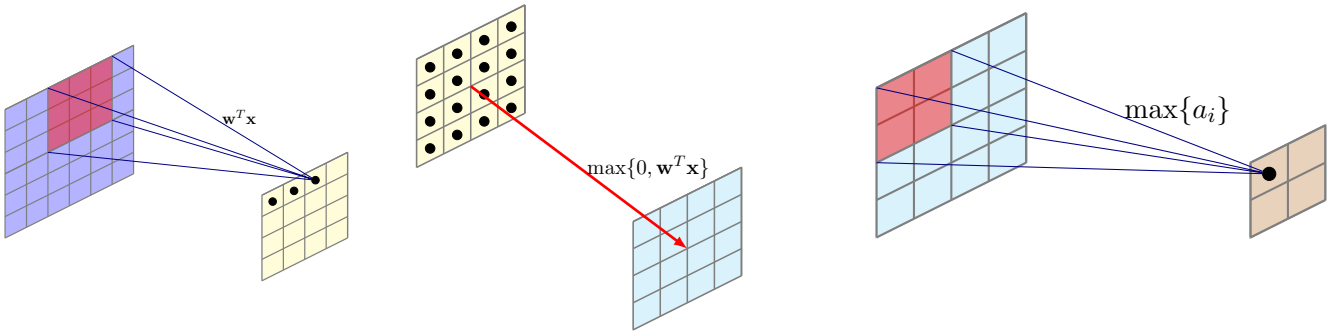
- You can use the representations learned from models learnt for image classification in many other tasks involving images
- People share their learnt representations like
  - VGGNET, RESNET, …
- You can use these representations (features of an image) for other supervised tasks
  - Visual question answering
  - Face detection
  - Image localization
- Not on for images – Text, Images, Graphs, …

AlexNet, 8 layers
(ILSVRC 2012)





Conclusions



- Convolution networks exploit local structure in input
- Have large statistical as well as computational efficiency
- ConvNets have been at the forefront of the DL revolution
- Deeper Networks help but one has to account for gradient-based problems

- Representations learnt from ConvNets can be used for multiple tasks

References

- Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”
- Vanishing and exploding gradients: http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf
- Dive into Deep Learning: https://en.d2l.ai/d2l-en.pdf