Language Technologies Institute

Carnegie Mellon University

Multimodal Machine Learning

Lecture 3.1: Convolutional Neural Networks

Louis-Philippe Morency

* Original version co-developed with Tadas Baltrusaitis

Lecture Objectives

- Convolutional Neural networks
  - Convolution kernels
  - Convolution neural layers
  - Pooling layers
- Convolutional architectures
  - VGGNet and residual networks
  - Class Activation Mapping (CAM)
  - Region-based CNNs
  - Sequential Modeling with convolutional networks

Administrative Stuff

Pre-proposals – Due tomorrow 9/11

- Dataset and research problem
- Input modalities and multimodal challenges
- Initial research ideas
- Teammates and resources

Submit via Gradescope before 11:59pm ET

Upcoming Assignments

- Tuesday 9/10 (today):
  - *Visualizing and Understanding Convolutional Networks*
- Wednesday 9/11 (tomorrow)
  - Pre-proposals
- Thursday 9/12: no reading assignment
  - But presence to the course is expected
- Tuesday 9/17:
  - *Visualizing and Understanding Recurrent Networks*

TA hours

Mondays 3-4:30pm

- Room GHC 5417 or GHC 6121 (to be confirmed)

Instructions for 1-page summaries

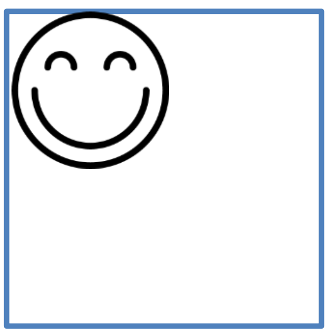
If you plan to be absent for a lecture:

- Please notify us by Piazza (private message)
- You should write a 1-page summary of the paper
- Instructions are on Piazza (resources)
- Submit your summary on Gradescope withing 7 days of the absence day

Convolutional Neural Networks

A Shortcoming of MLP

2 Data Points – detect which head is up!
Easily modeled using one neuron.
What is the best neuron to model this?

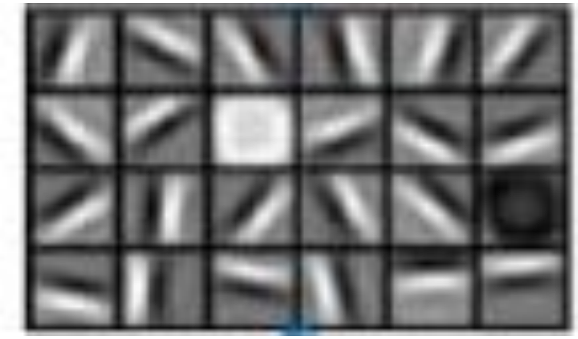


This head may or may not be up – what happened?

Solution: instead of modeling the entire image, model the important region.

Why not just use an MLP for images (1)?

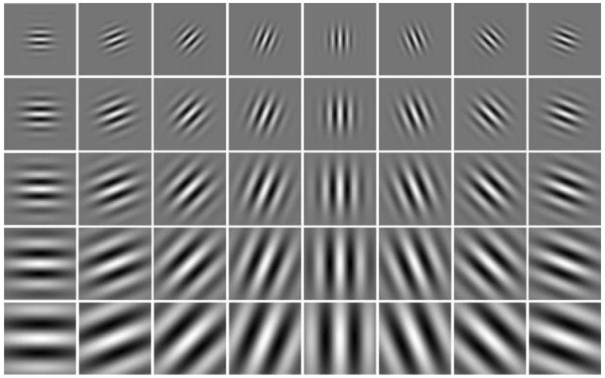
- MLP connects each pixel in an image to each neuron
- Does not exploit redundancy in image structure
  - Detecting edges, blobs
  - Don’t need to treat the top left of image differently from the center



- Too many parameters
  - For a small 200×200 pixel RGB image the first matrix would have $120000 \times n$ parameters for the first layer alone

Why not just use an MLP for images (2)?

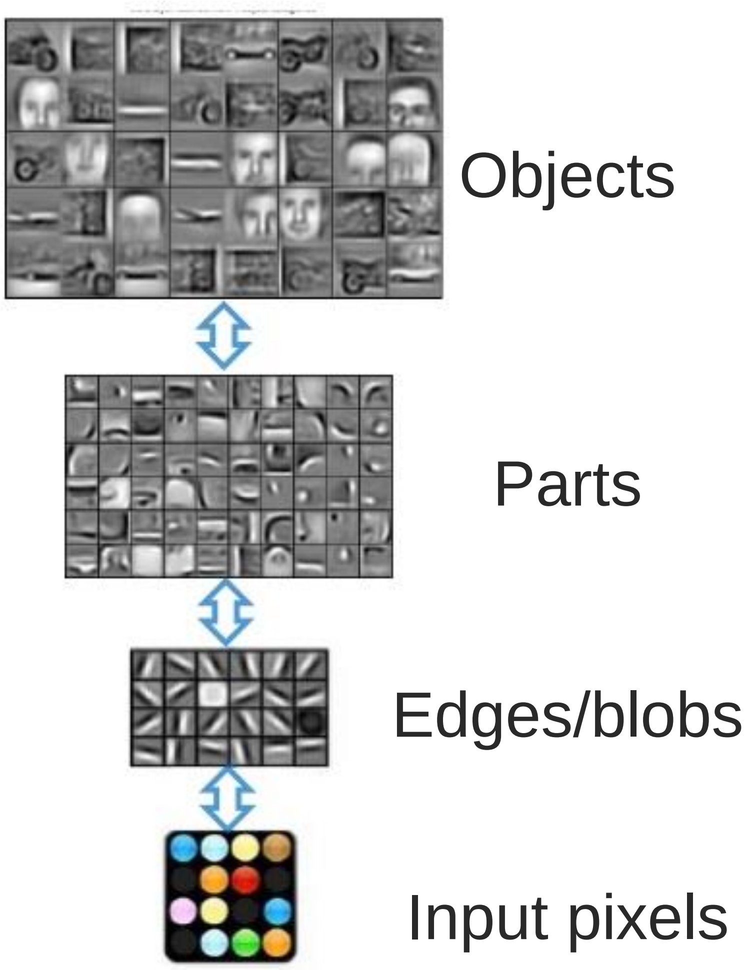
- Human visual system works in a filter fashion
  - First the eyes detect edges and change in light intensity
  - The visual cortex processing performs Gabor like filtering



- MLP does not exploit translation invariance
- MLP does not necessarily encourage visual abstraction

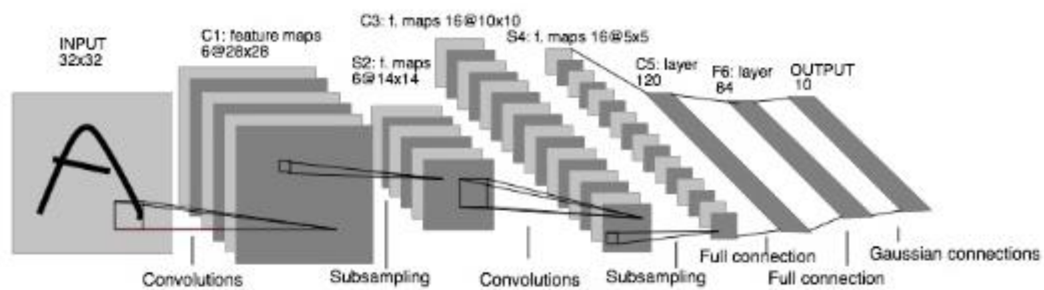
Why use Convolutional Neural Networks

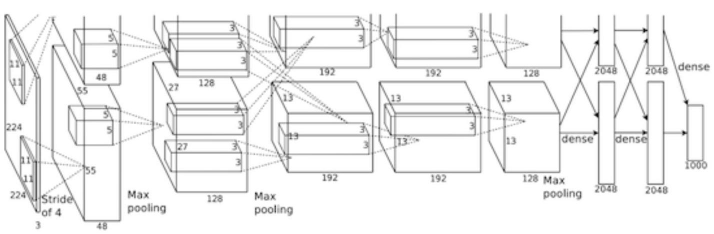
- Using basic Multi Layer Perceptrons does not work well for images
- Intention to build more abstract representation as we go up every layer

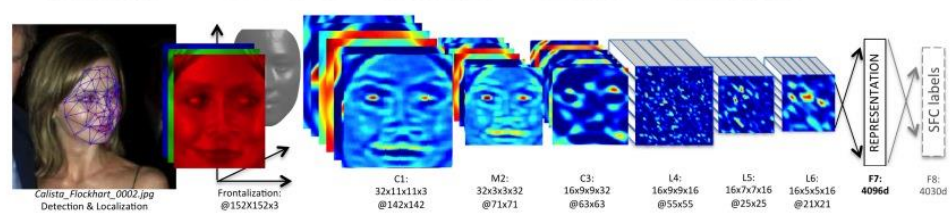


Convolutional Neural Networks

- They are everywhere that uses representation learning with images
- State of the art results – object recognition, face recognition, segmentation, OCR, visual emotion recognition
- Extensively used for multimodal tasks as well

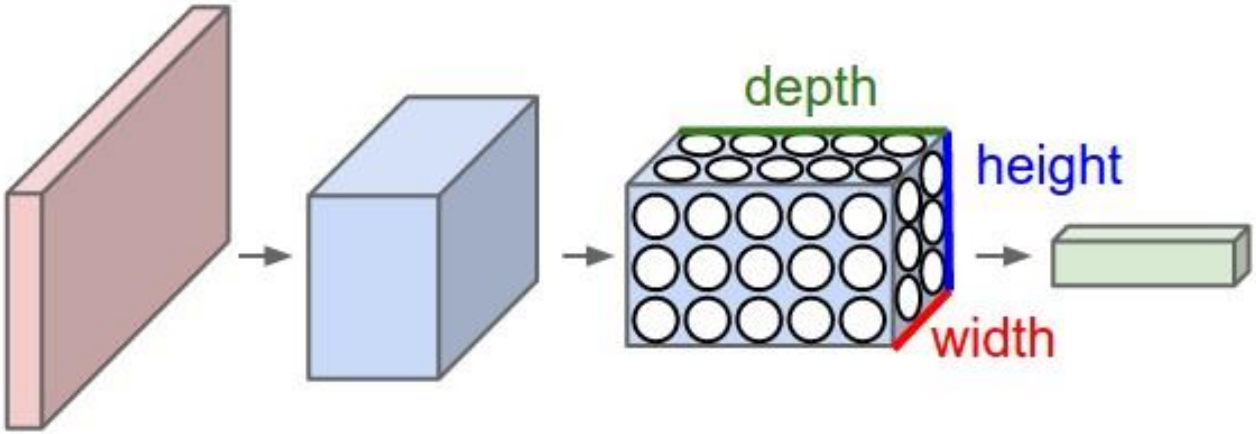






Main differences of CNN from MLP

- Addition of:
  - Convolution layer
  - Pooling layer
- Everything else is the same (loss, score and optimization)
- MLP layer is called Fully Connected layer



Convolution

Convolutional definition

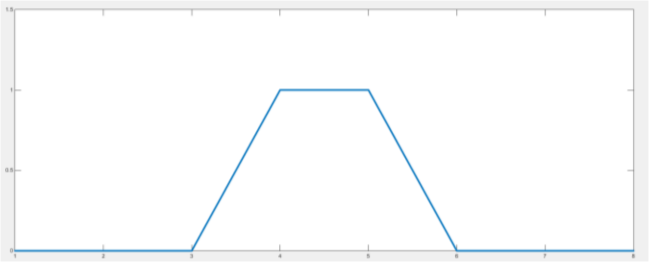
- A basic mathematical operation (that given two functions returns a function)

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

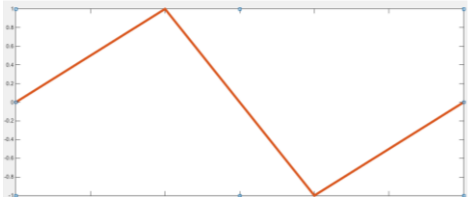
- Have a continuous and discrete versions (we focus on the latter)

Convolution in 1D

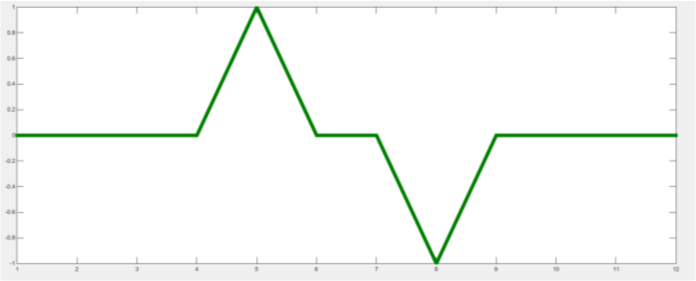
- Example
  - $f = [\dots, 0,1,1,1,0,0, \dots]$



  - $g = [\dots, 0,1,-1,0 \dots]$



- $f * g = [\dots, 0,1,0,0,-1,0,0, \dots]$



$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Convolution in practice

- In CNN we only consider functions with limited domain (not from $-\infty$ to ∞)
- Also only consider fully defined (valid) version
  - We have a signal of length N
  - Kernel of length K
  - Output will be length $\mathrm{N-K+1}$
- $f = [1,2,1]$, $g = [1,-1]$, $f * g = [1,-1]$

Convolution in practice

- If we want output to be different size we can add padding to the signal
  - Just add 0s at the beginning and end
- $f = [0,0,1,2,1,0,0]$, $g = [1, -1]$, $f * g = [0,1,1,-1,-1,0]$

- Also have strided convolution (the filter jumps over pixels or signal)
  - With stride 2
  - $f = [0,0,1,2,1,0,0]$, $g = [1, -1]$, $f * g = [0,1,-1,0]$
  - Why is this a good idea? Where can this fail?

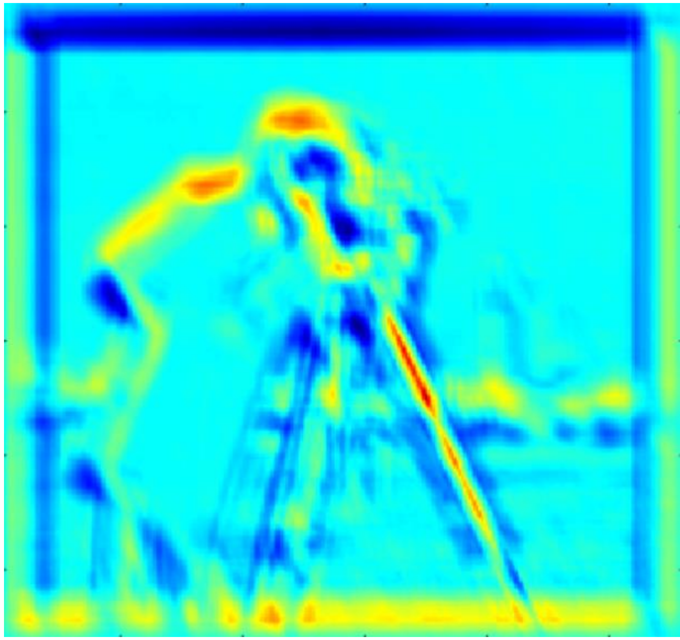
Convolution in 2D

- Example of image and a kernel

*

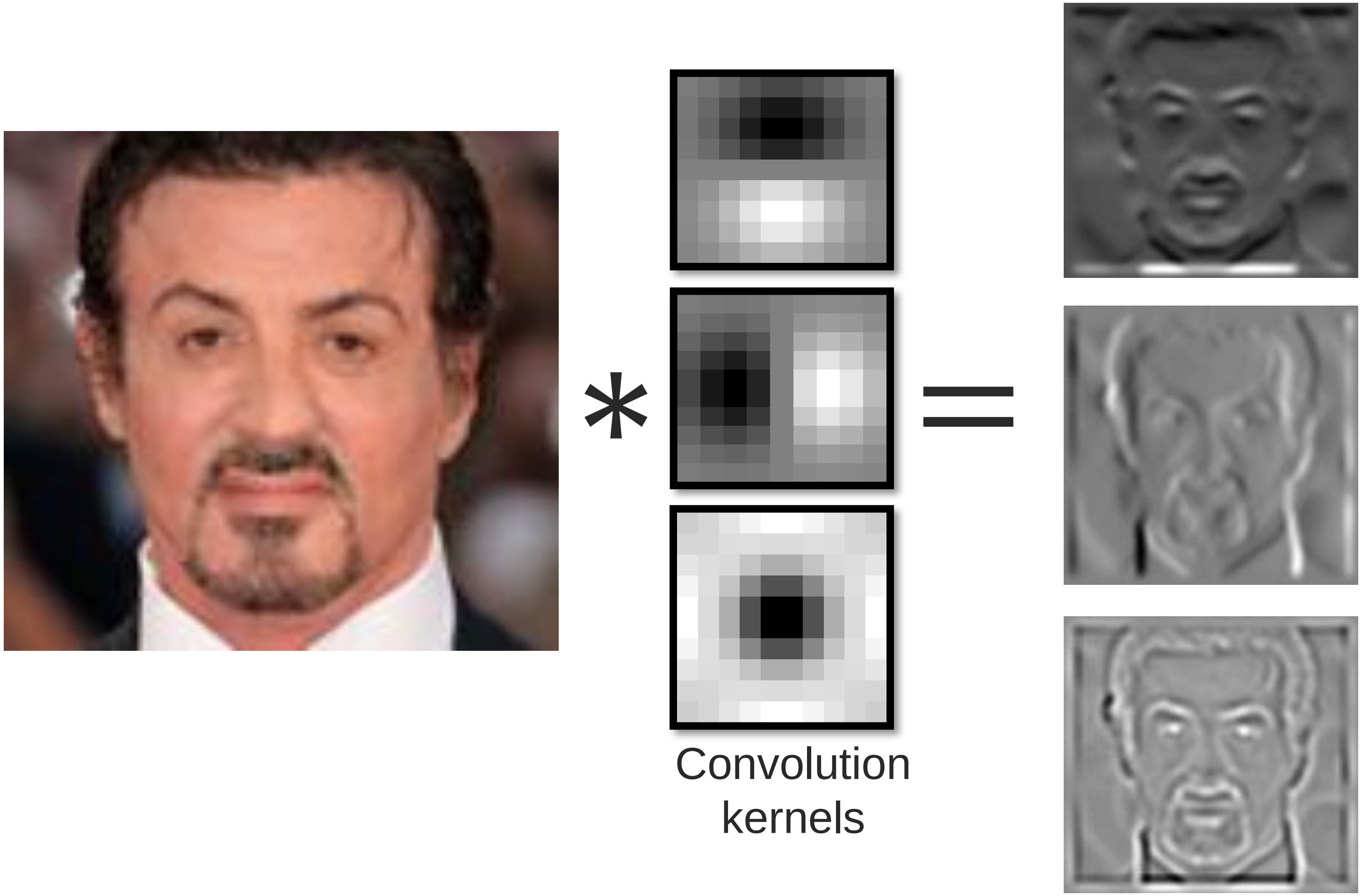
Convolution kernel

=



Response map

Convolution in 2D



Convolution kernels

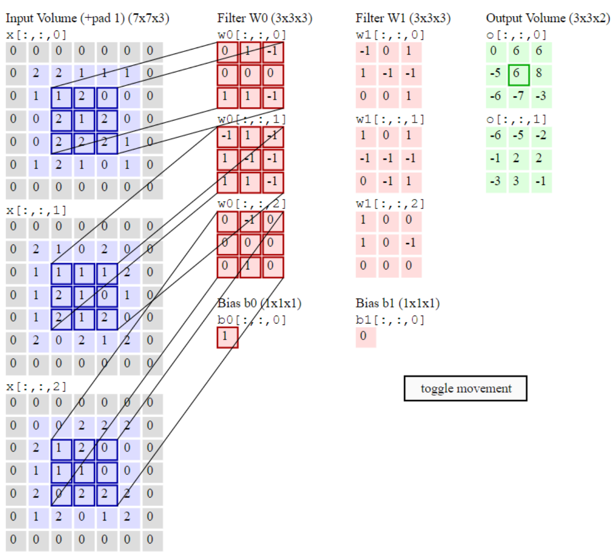
Response maps

Convolution intuition

- Correlation/correspondence between two signals
  - Template matching
- Why are we interested in convolution
  - Allows to extract structure from signal or image
  - A very efficient operation on signals and images

Sample CNN convolution

- Great animated visualization of 2D convolution
- http://cs231n.github.io/convolutional-networks/



Convolutional Neural Layer

Fully connected layer

- Weighted sum followed by an activation function

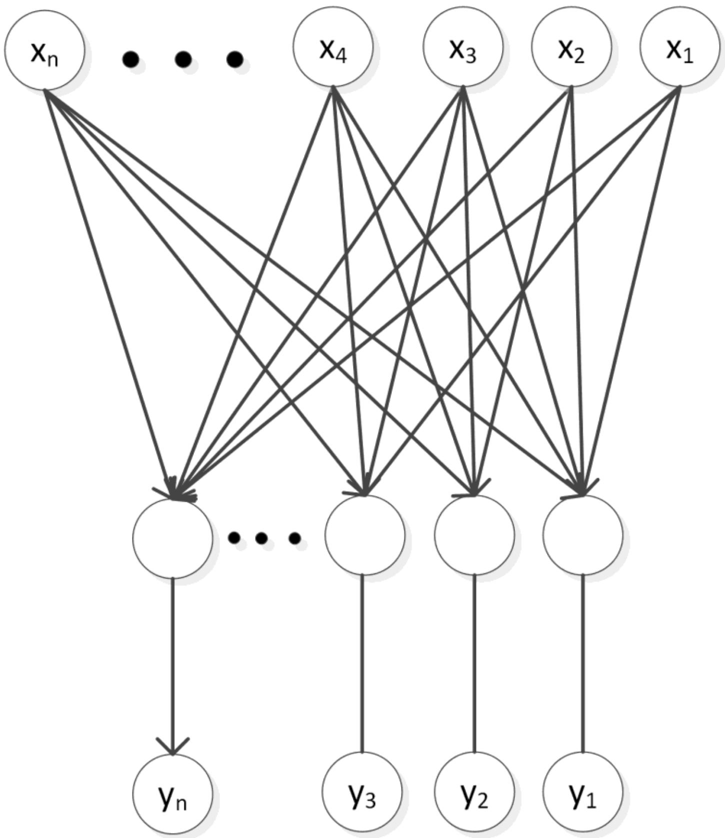
Input

Weighted sum
$Wx + b$

Activation function

Output

$y = f(Wx + b)$

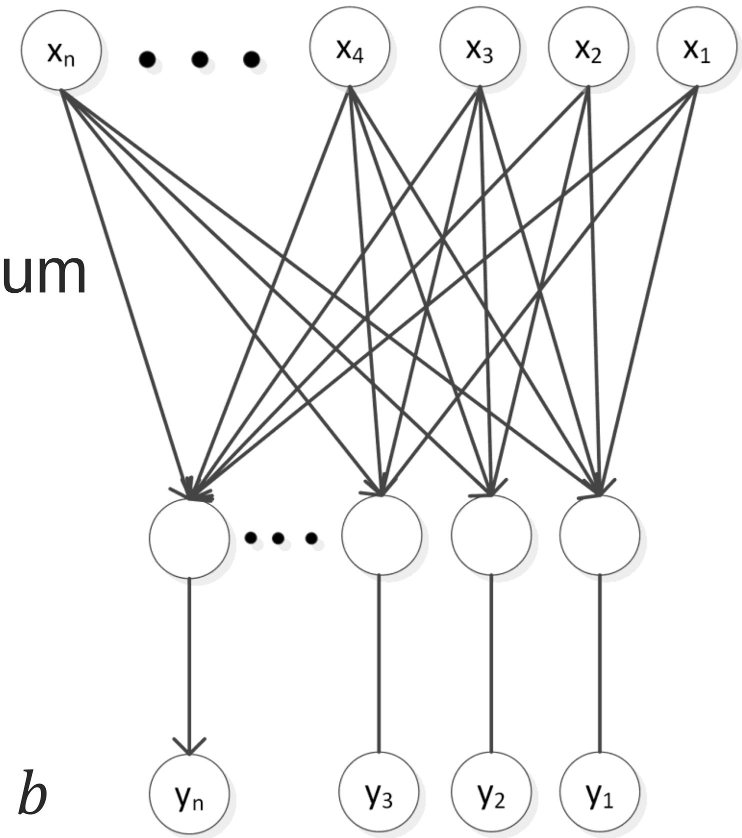


Convolution as MLP (1)

- Remove activation

Input

Weighted sum
$Wx + b$



$$y = Wx + b$$

Kernel

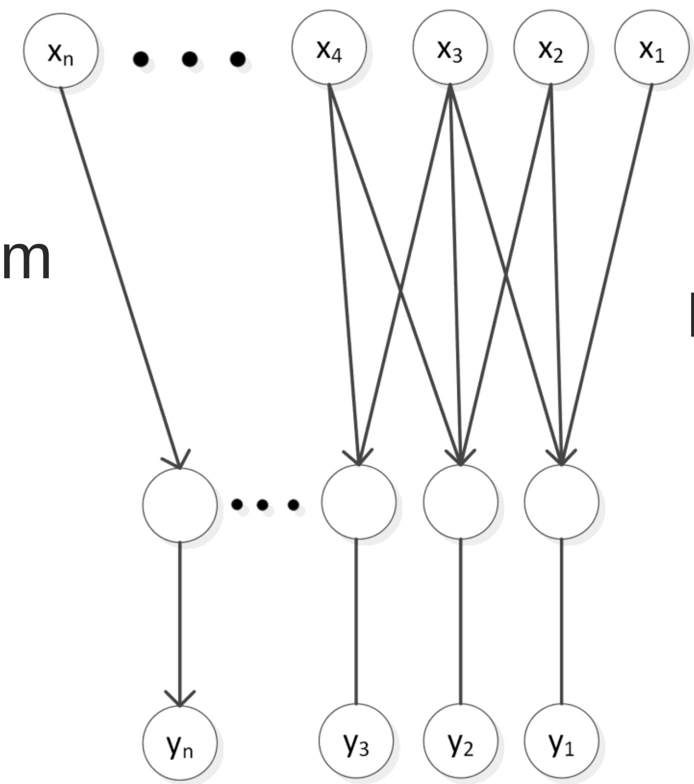
$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$

Convolution as MLP (2)

- Remove redundant links making the matrix W sparse (optionally remove the bias term)

Input

Weighted sum
Wx



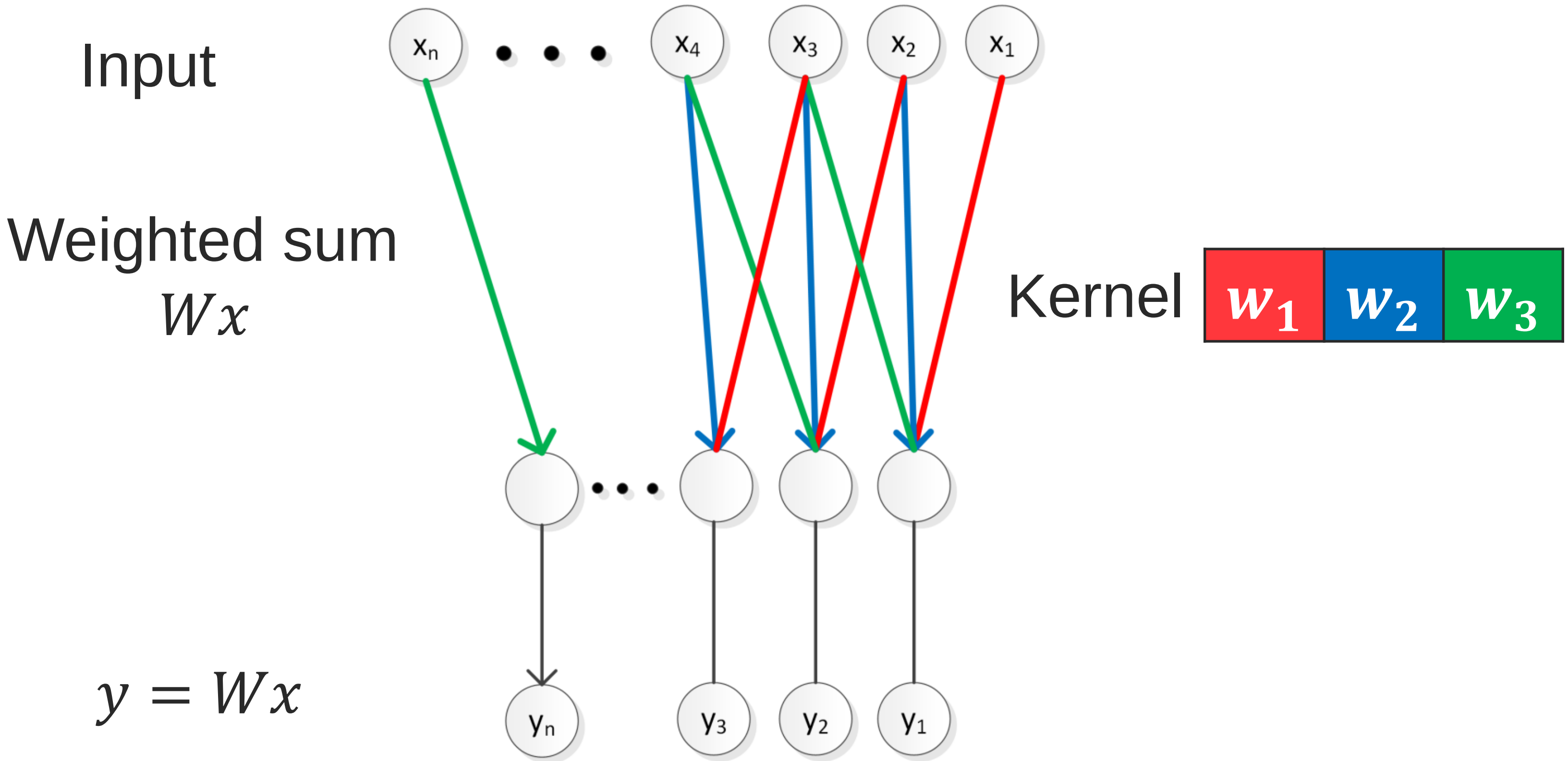
Kernel

$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$

$$y = Wx$$

Convolution as MLP (3)

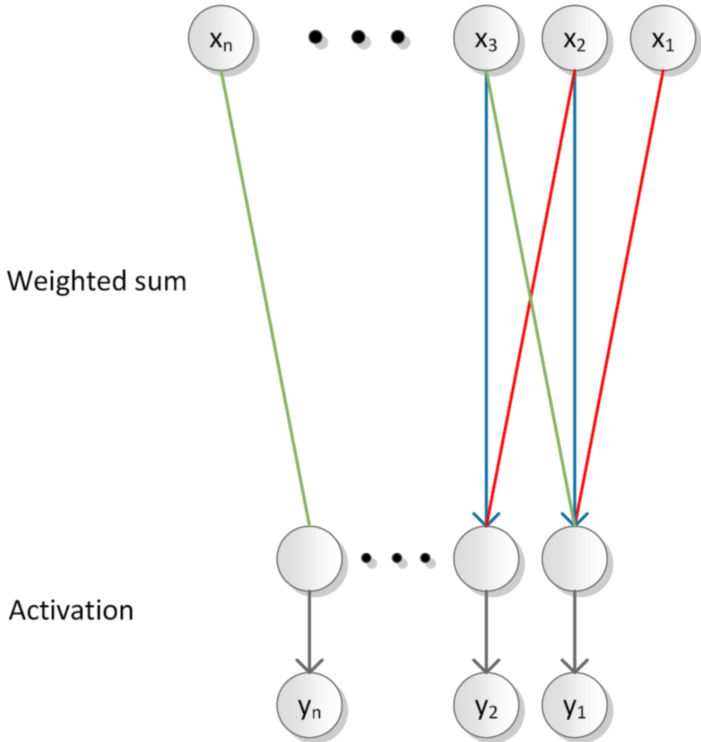
- We can also share the weights in matrix W not to do redundant computation



How do we do convolution in MLP recap

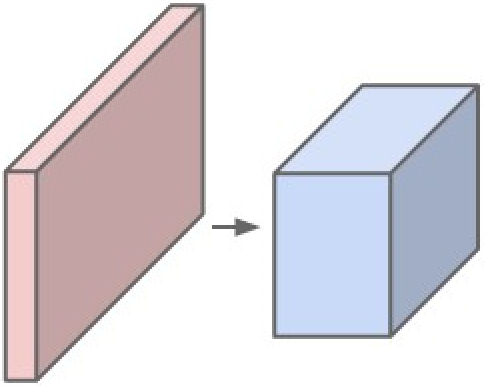
- Not a fully connected layer anymore
- Shared weights
    - Same colour indicates same (shared) weight

$$W = \begin{pmatrix} w_1 & w_2 & w_3 & & 0 & 0 & 0 \\ 0 & w_1 & w_2 & \cdots & 0 & 0 & 0 \\ 0 & 0 & w_1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & w_3 & 0 & 0 \\ 0 & 0 & 0 & \cdots & w_2 & w_3 & 0 \\ 0 & 0 & 0 & & w_1 & w_2 & w_3 \end{pmatrix}$$



More on convolution

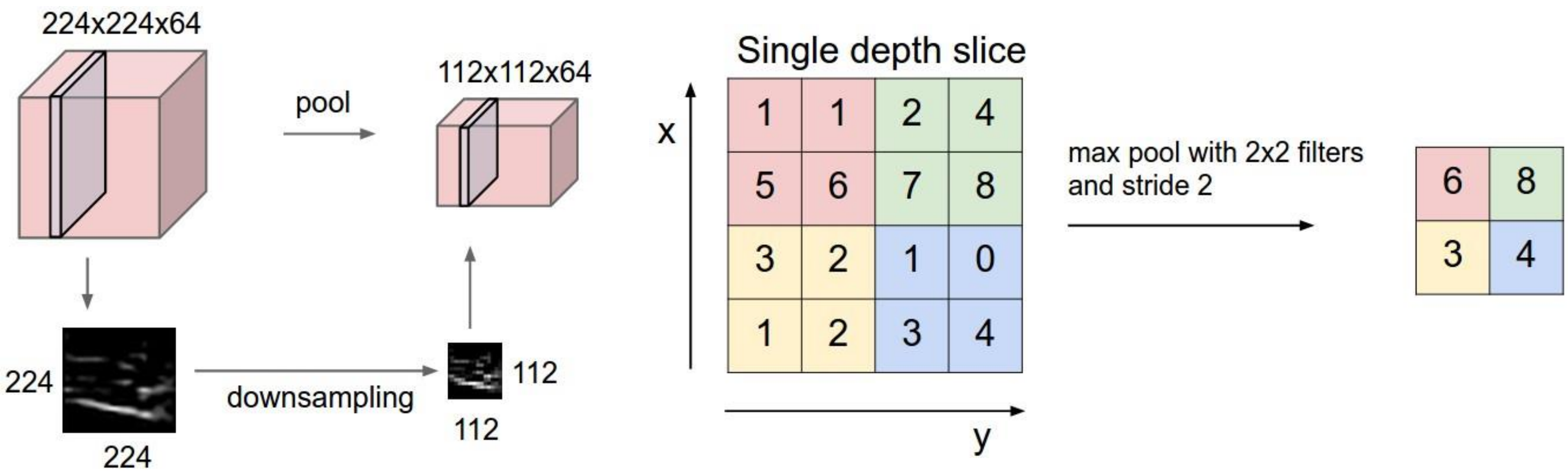
- Can expand this to 2D (or even 3D!)
  - Just need to make sure to link the right pixel with the right weight
- Can expand to multi-channel 2D
  - For RGB images
- Can expand to multiple kernels/filters
  - Output is not a single image anymore, but a **volume** (sometimes called a feature map)
  - Can be represented as a tensor (a 3D matrix)
- Usually also include a bias term and an activation



Pooling layer

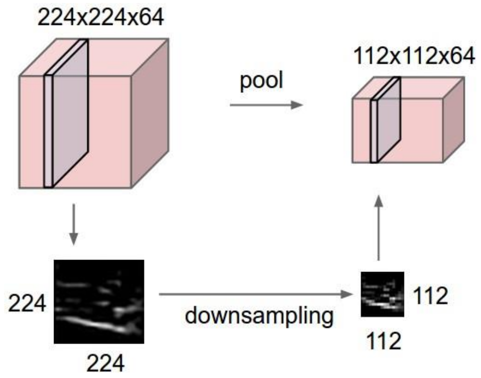
Pooling layer

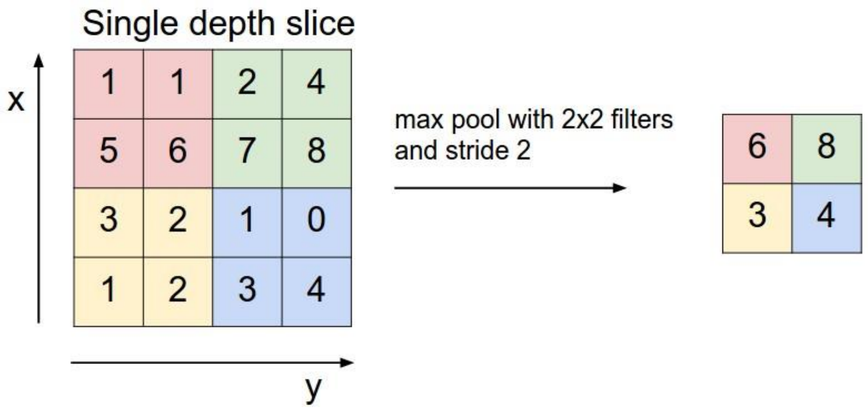
- Image subsampling



Pooling layer motivation

- Used for sub-sampling
    - Allows summarization of response
- Helps with translational invariance
- Have filter size and stride (hyperparameters)

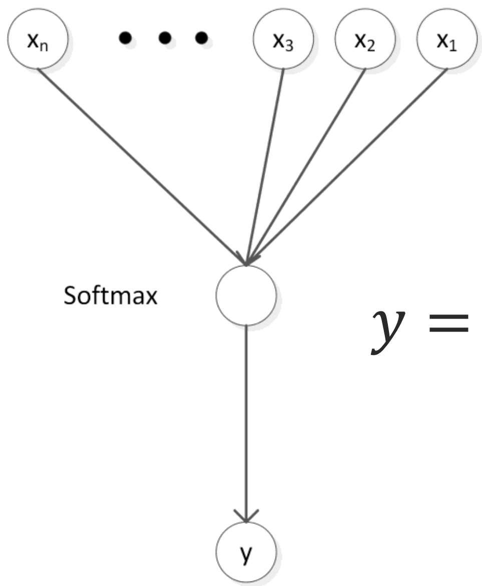




Pooling layer gradient

1. Record during forward pass which pixel was picked and use the same in backward pass

2. Pick the maximum value from input using a smooth and differentiable approximation

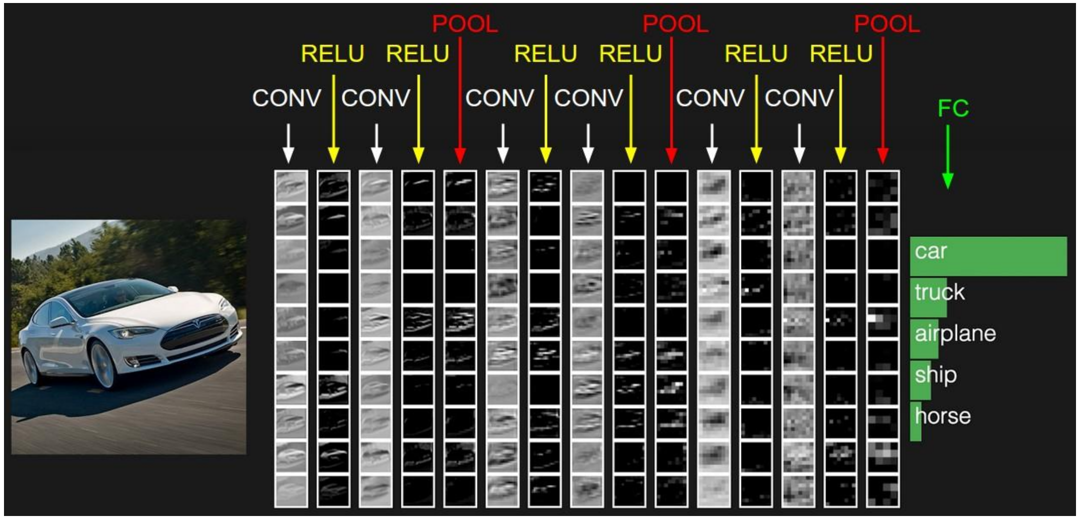


$$y = \frac{\sum_{i=1}^{n} x_i e^{\alpha x_i}}{\sum_{i=1}^{n} e^{\alpha x_i}}$$

VGGNet and Residual Networks

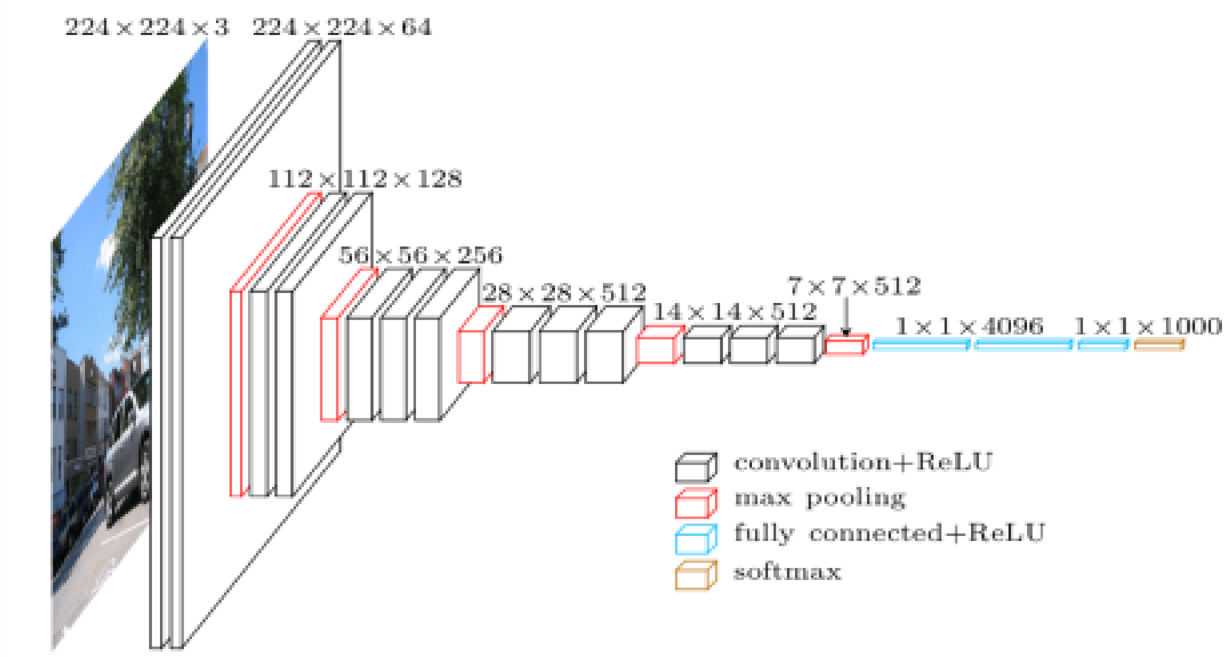
Common architectures

- Start with a convolutional layer follow by non-linear activation and pooling
- Repeat this several times
- Follow with a fully connected (MLP) layer

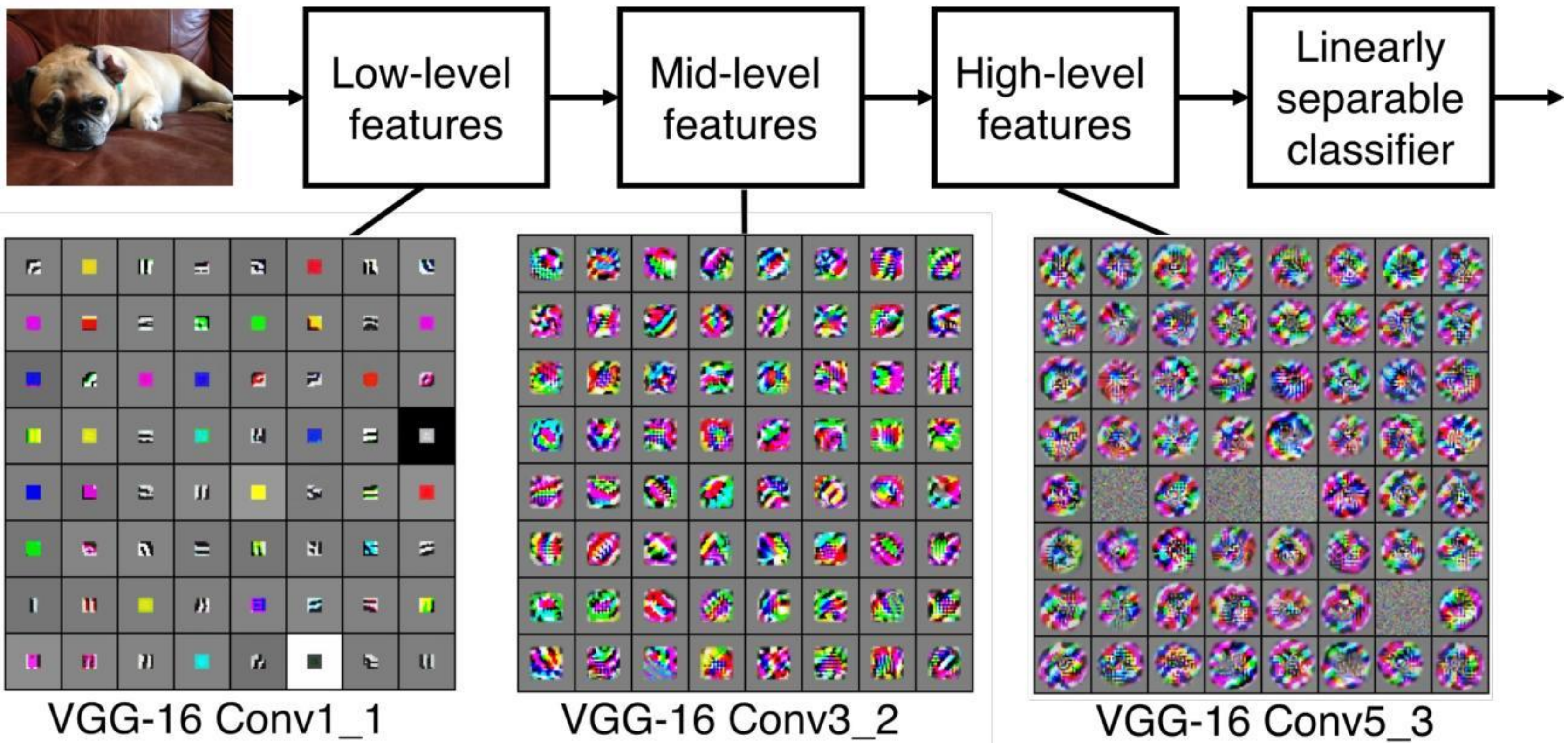


VGGNet model

- Used for object classification task
  - 1000 way classification task – pick one
  - 138 million params

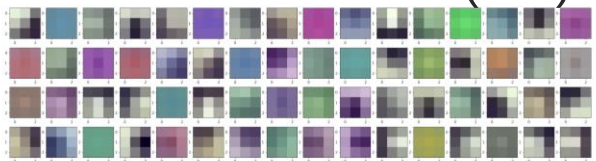


VGGNet Convolution Kernels

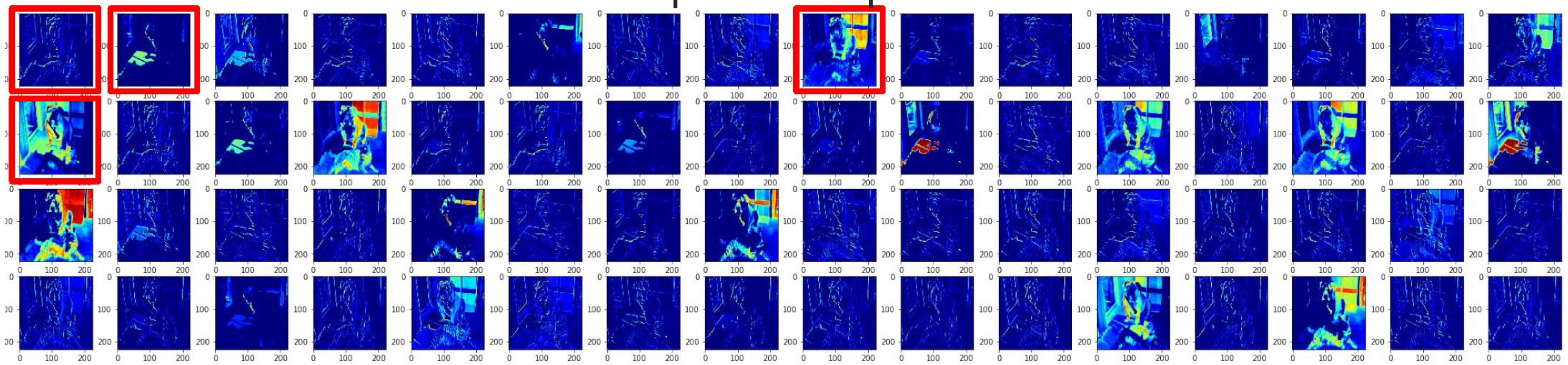


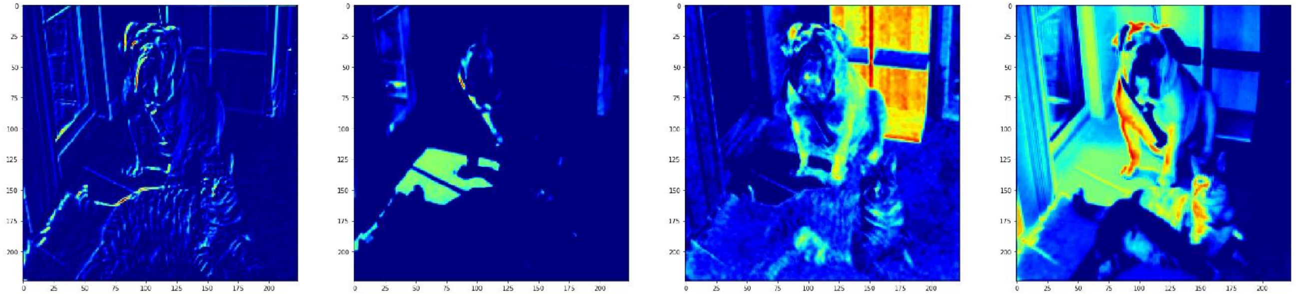
VGGNet Response Maps (aka Activation Maps)

Convolution kernels (3x3)



Response Maps





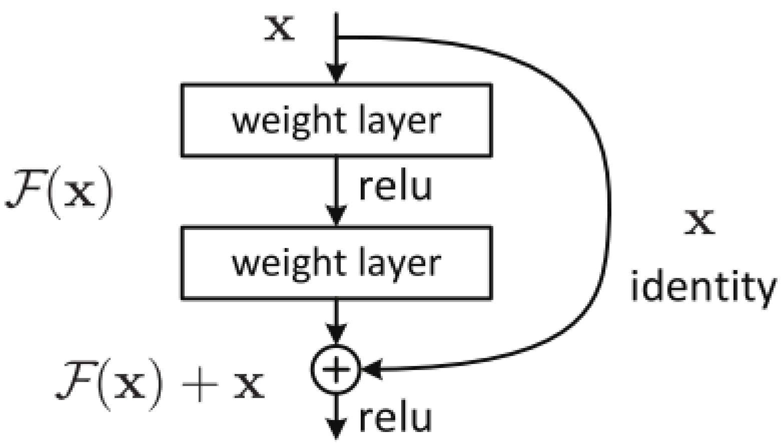
Other architectures

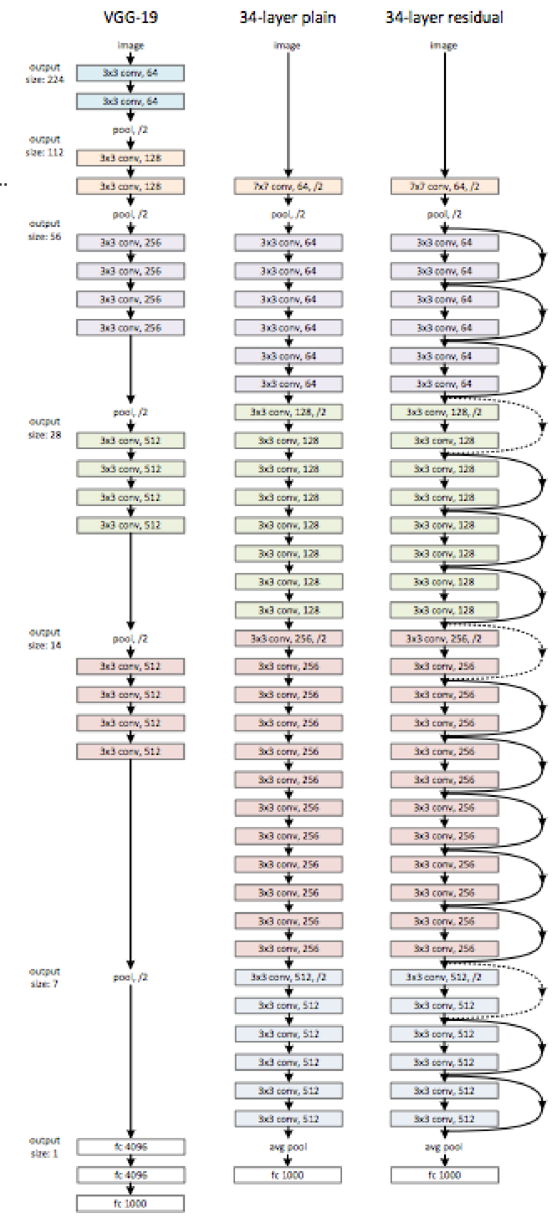
- LeNet – an early 5 layer architecture for handwritten digit recognition
- DeepFace – Facebook’s face recognition CNN
- VGGFace – For face recognition (from VGG folks)
- AlexNet – Object Recognition

- **Already trained models for object recognition can be found online**

Residual Networks

- Adding residual connections



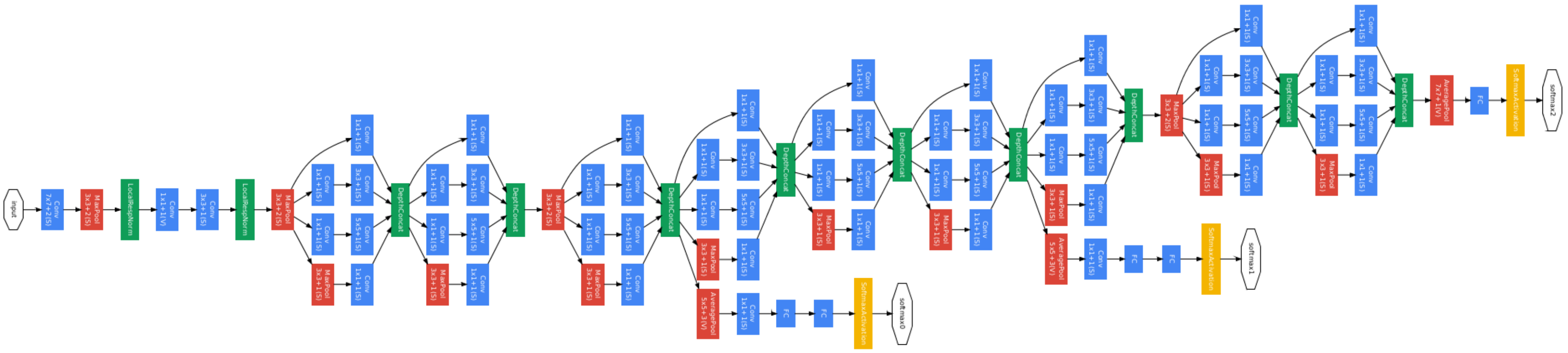


ResNet (He et al., 2015)

- Up to 152 layers!

Googlenet

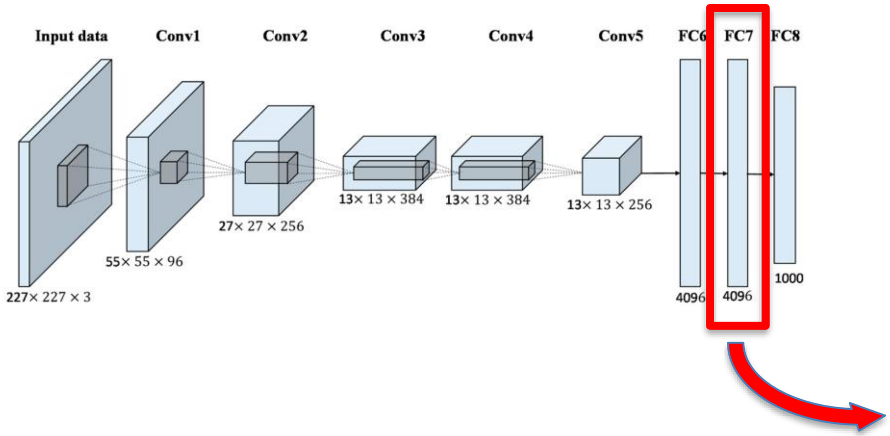
- Using residual blocks
  - Loss function in different layers of the network

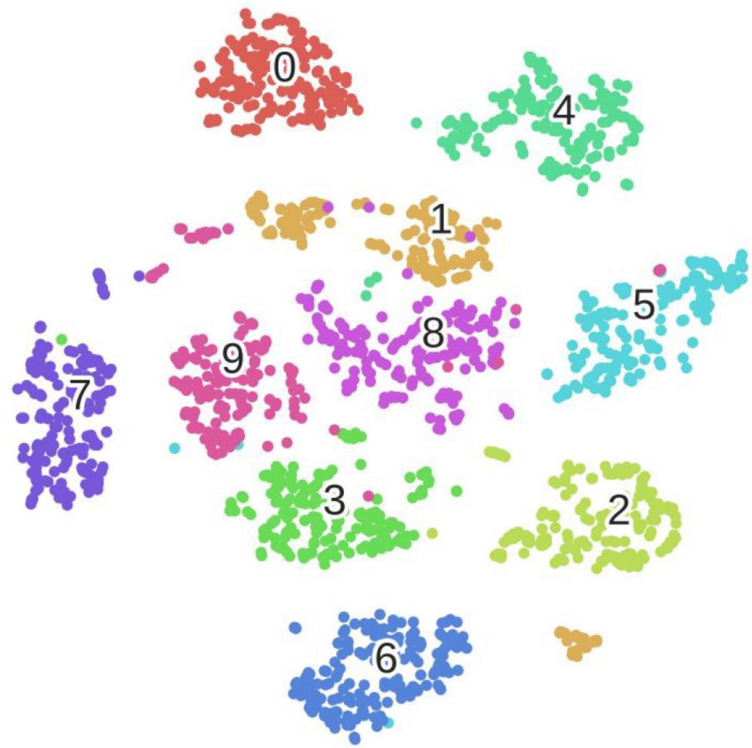


Visualizing CNNs

Visualizing the Last CNN Layer: t-sne

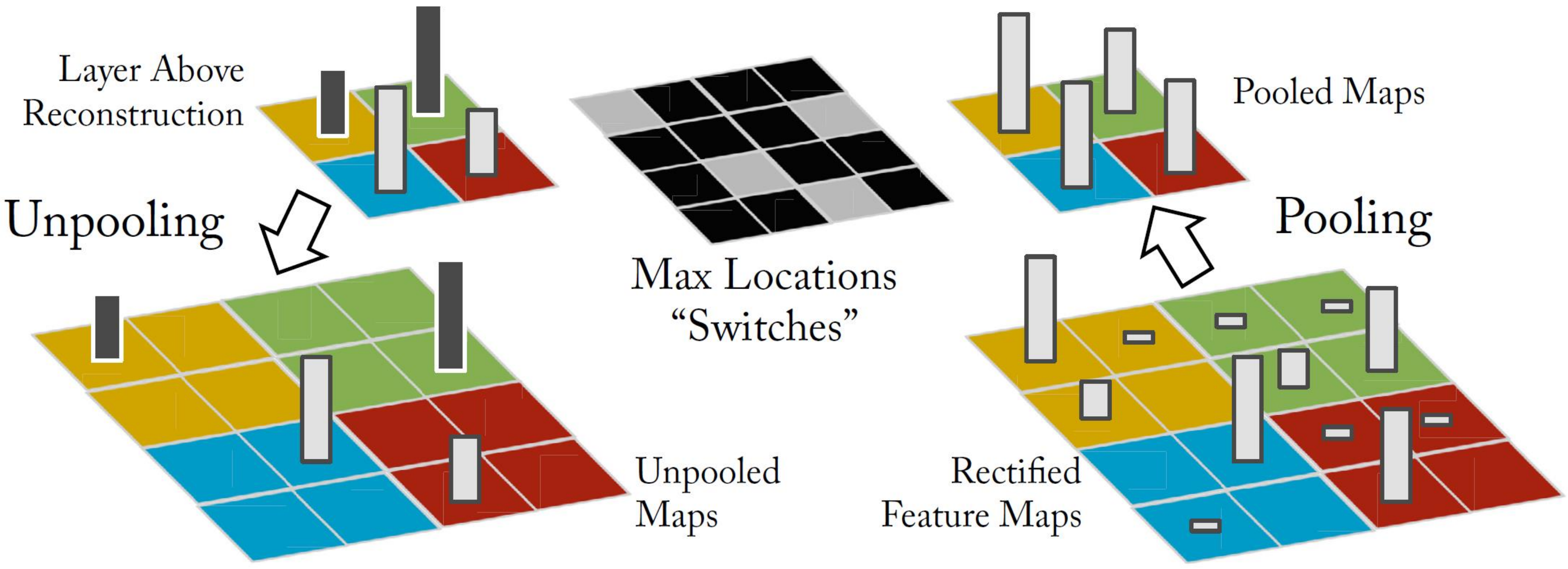
Alex Net



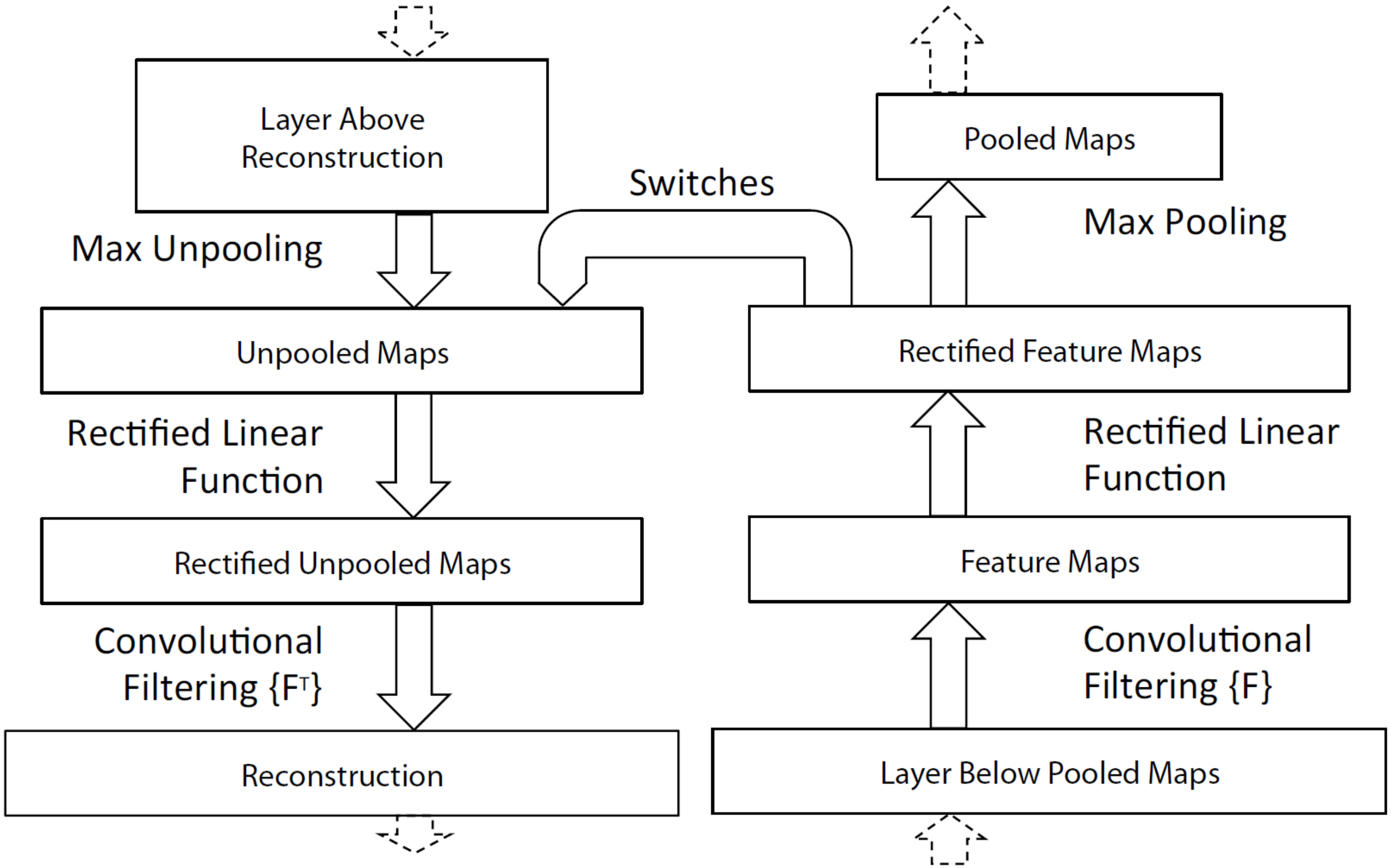


Embed high dimensional data points (i.e. feature codes) so that pairwise distances are conserved in local neighborhoods.

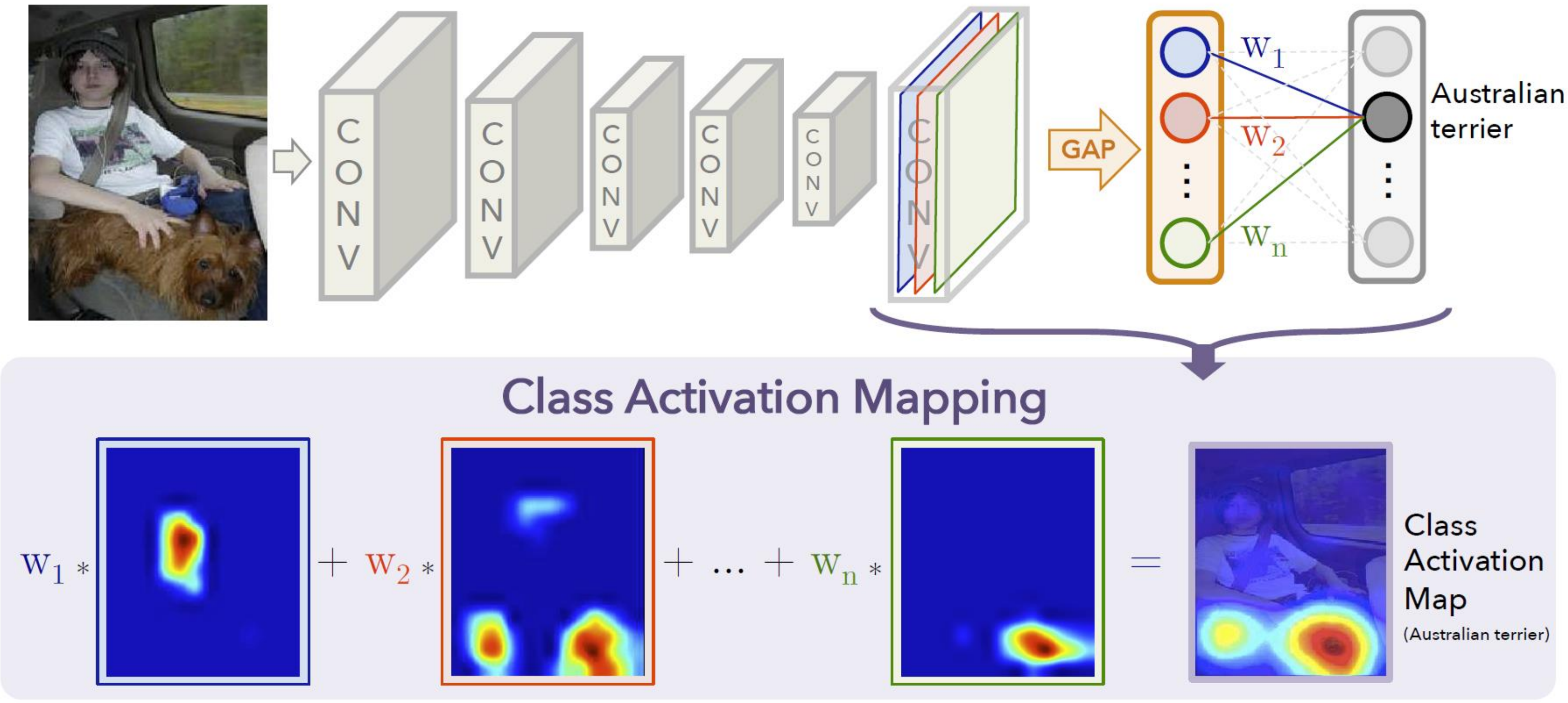
Deconvolution



Deconvolution

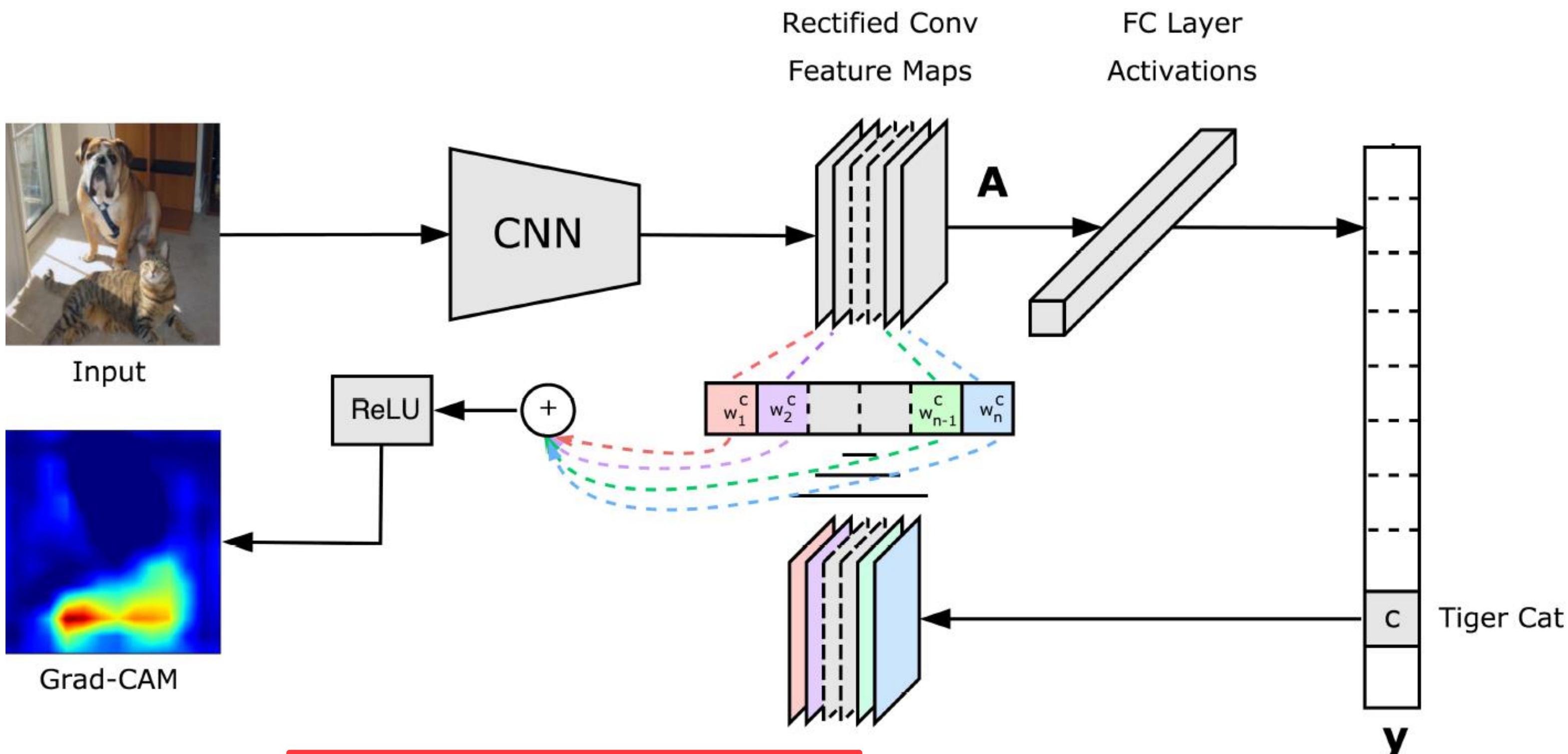


CAM: Class Activation Mapping [CVPR 2016]



$$L^c_{\mathrm{CAM}} = \underbrace{\sum_k w^c_k A^k}_{\text{linear combination}}$$

Grad-CAM [ICCV 2017]



$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

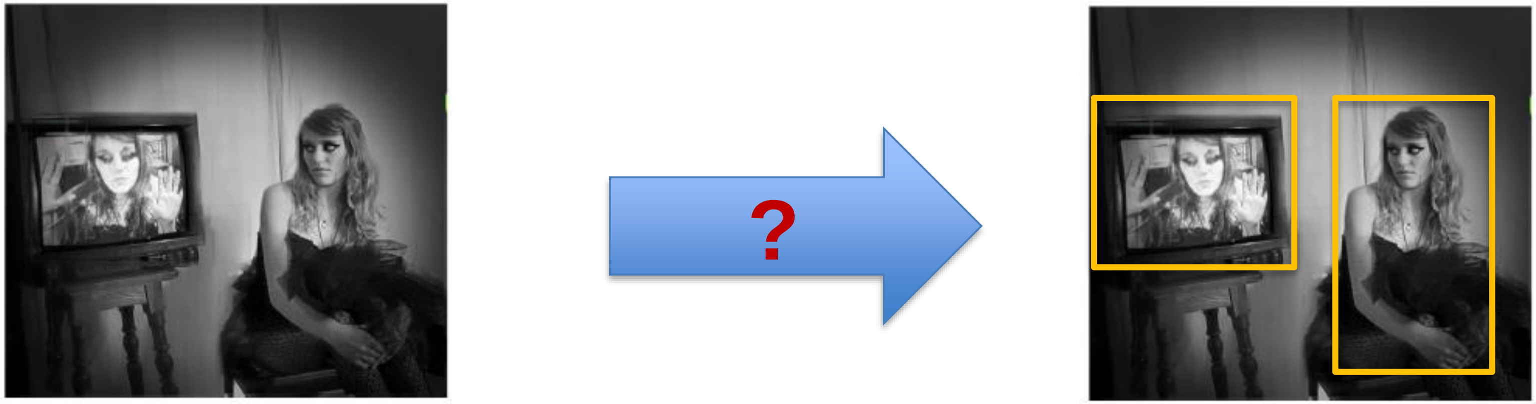
$$L^c_{\text{Grad-CAM}} = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

Region-based CNNs

Object recognition

Input Image

Object Detection (and Segmentation)

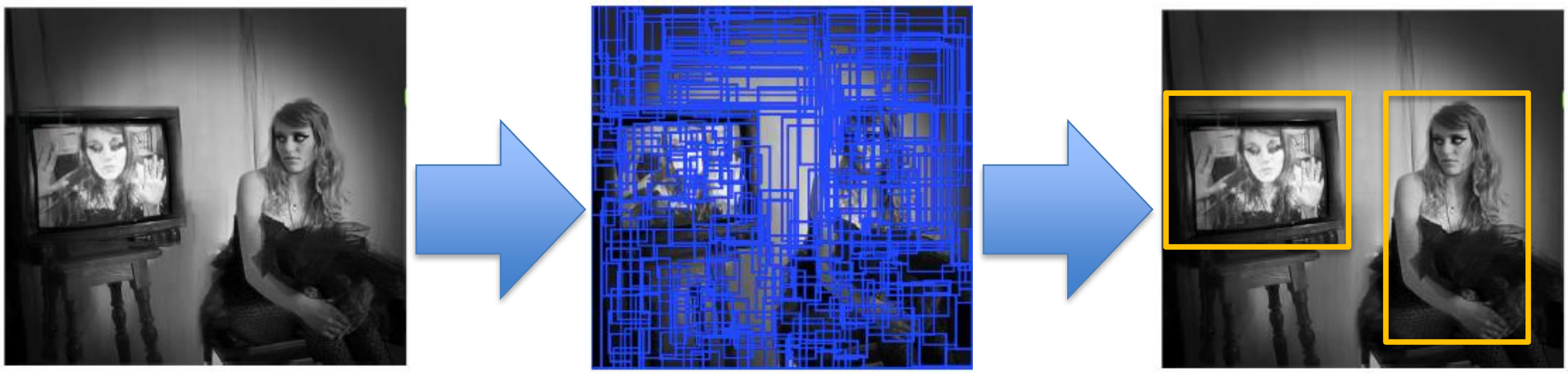


Input image

Detected Objects

One option: Sliding window

Object Detection (and Segmentation)



Input image | Region Proposals | Detected Objects

A better option: Start by Identifying hundreds of region proposals and then apply our CNN object detector

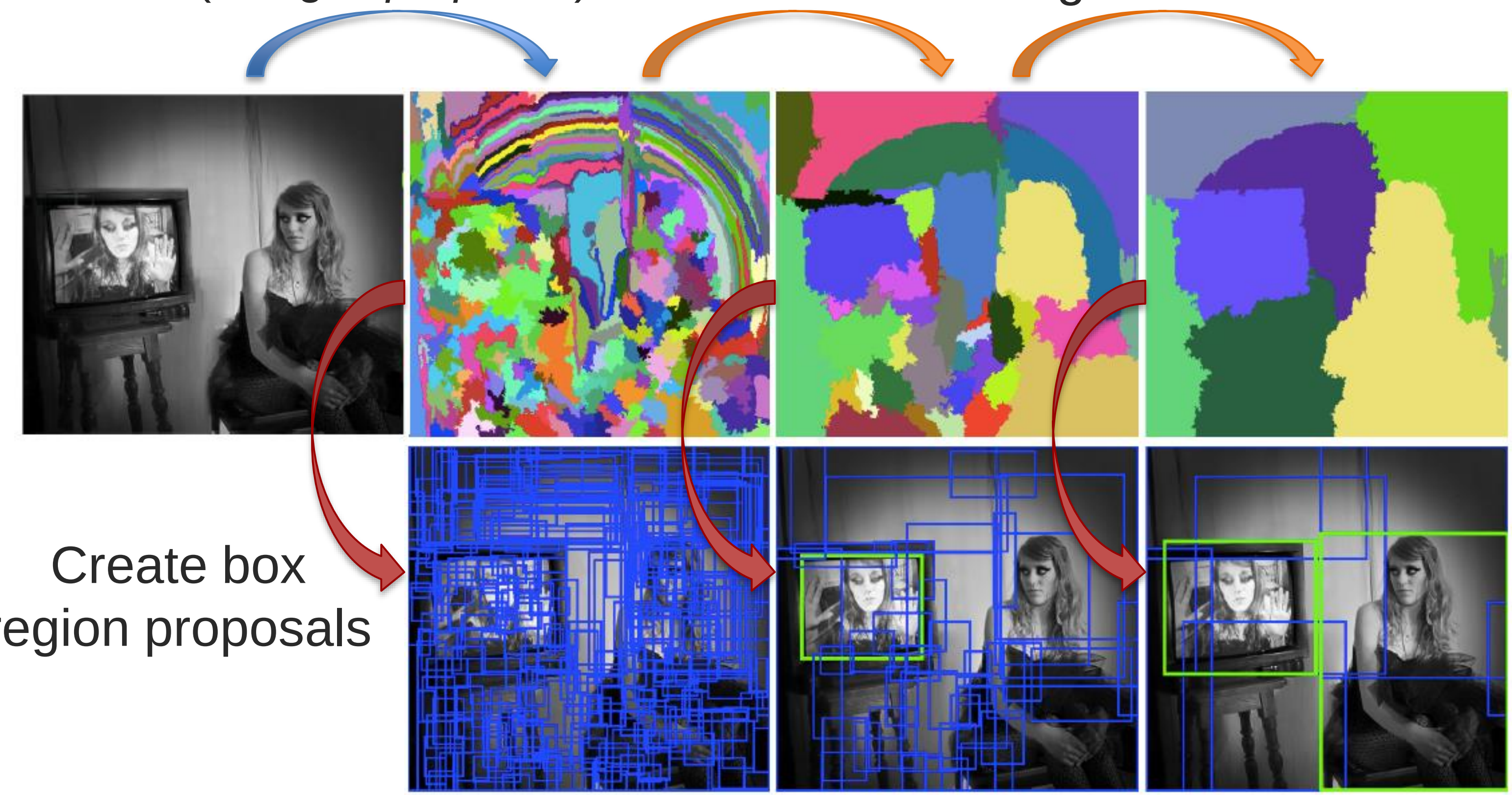
How to efficiently identify region proposals?

Selective Search [Uijlings et al., IJCV 2013]

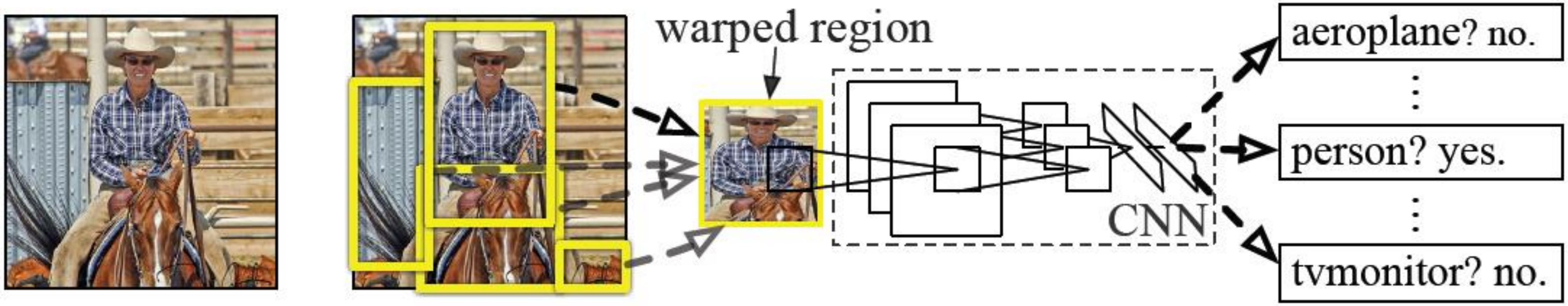
Image segmentation
(using *superpixels*)

And then merge
similar regions

Create box
region proposals



R-CNN [Girshick et al., CVPR 2014]

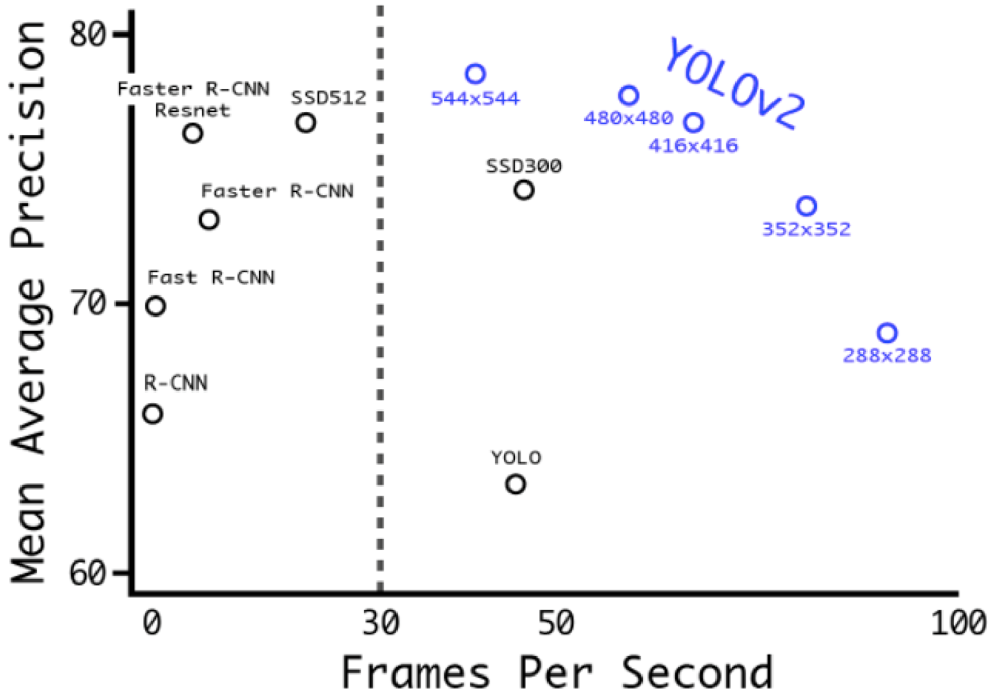


- Select ~2000 region proposals → Time consuming!
- Warp each region
- Apply CNN to each region → Time consuming!

Fast R-CNN: Applies CNN only once, and then extracts regions

Faster R-CNN: Region selection on the Conv5 response map

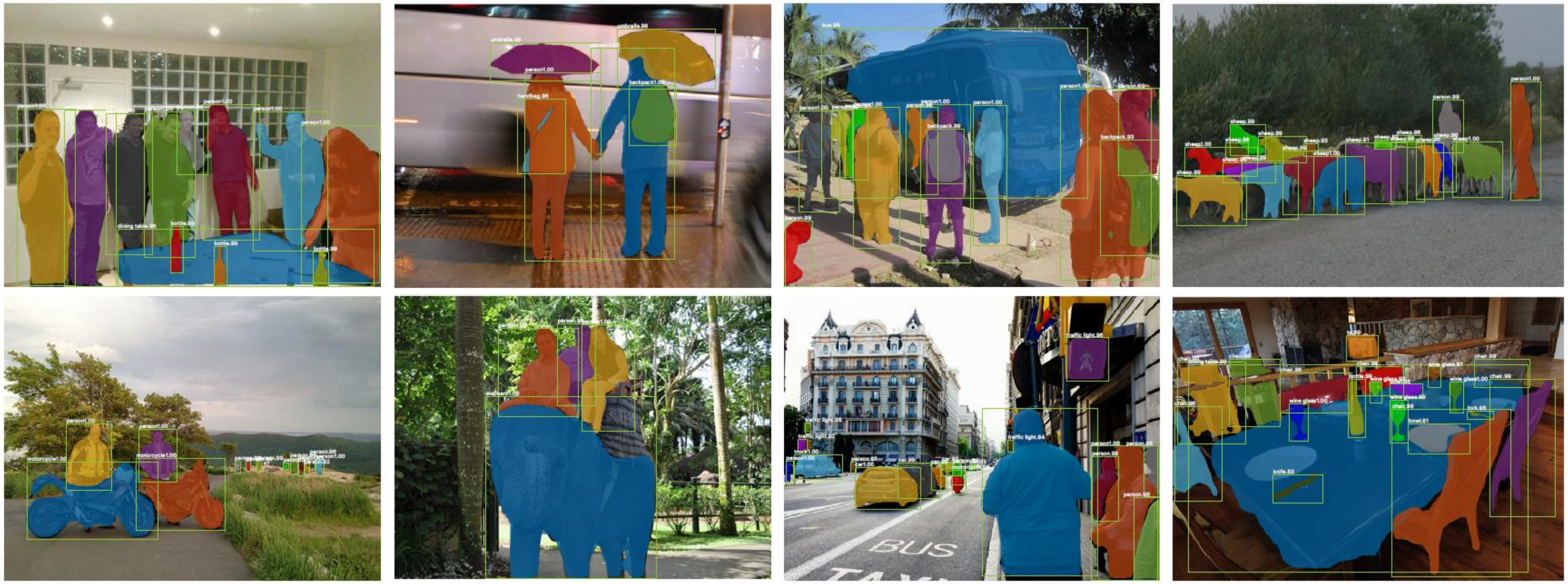
Trade-off Between Speed and Accuracy



YOLO: You Only Look Once (CVPR 2016, 2017)
SSD: Single Shot MultiBox Detector (ECCV 2016)

Mask R-CNN: Detection and Segmentation

(He et al., 2018)



Sequential Modeling with Convolutional Networks

Modeling Temporal and Sequential Data

How to represent a video sequence?

One option: Recurrent Neural Networks

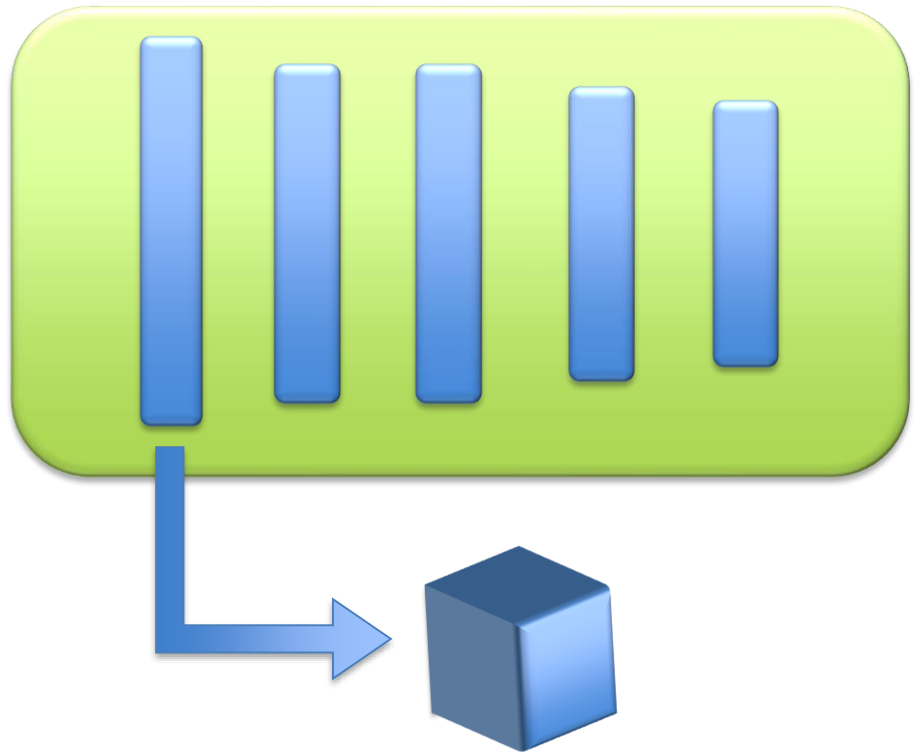
(more about this on Thursday)

3D CNN

Input as a 3D tensor

(stacking video images)

3D CNN



First layer with 3D kernels

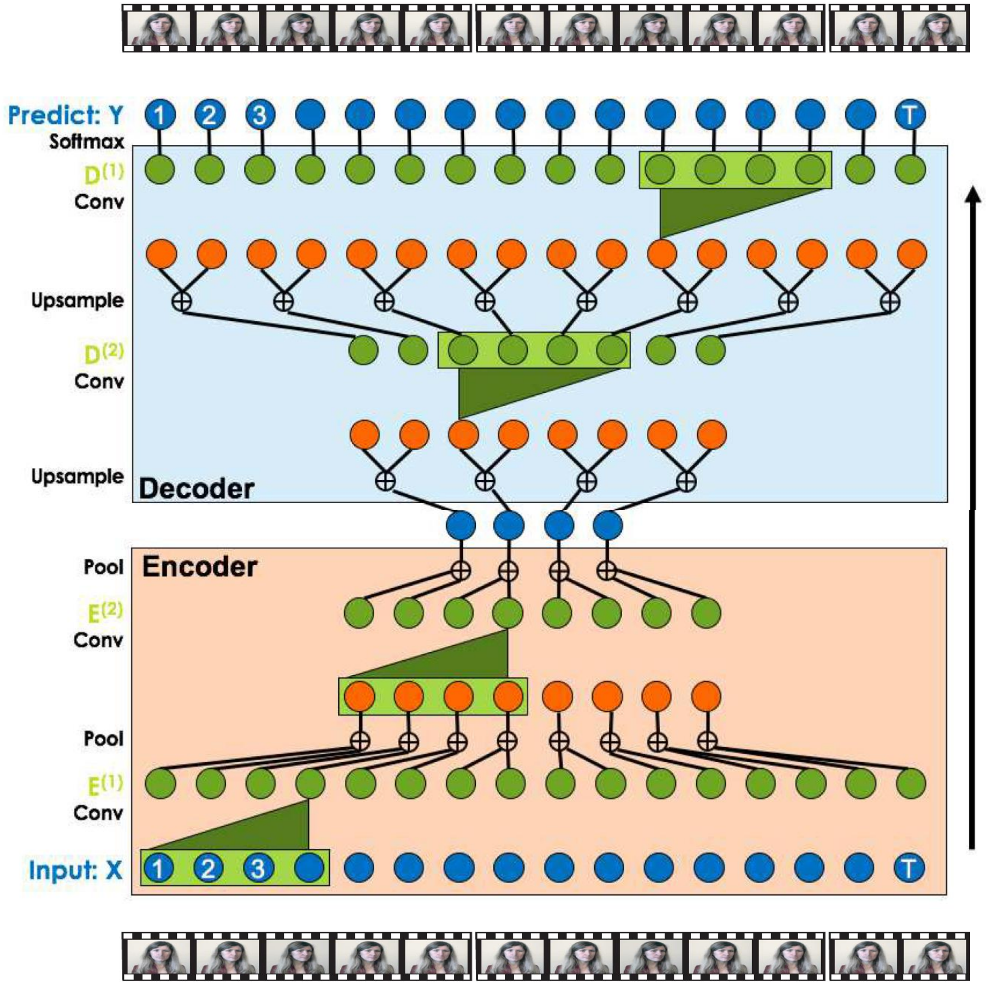
Time-Delay Neural Network

Alexander Waibel, Phoneme Recognition Using Time-Delay Neural Networks, SP87-100, Meeting of the Institute of Electrical, Information and Communication Engineers (IEICE), December, 1987,Tokyo, Japan.

Temporal Convolution Network (TCN) [Lea et al., CVPR 2017]

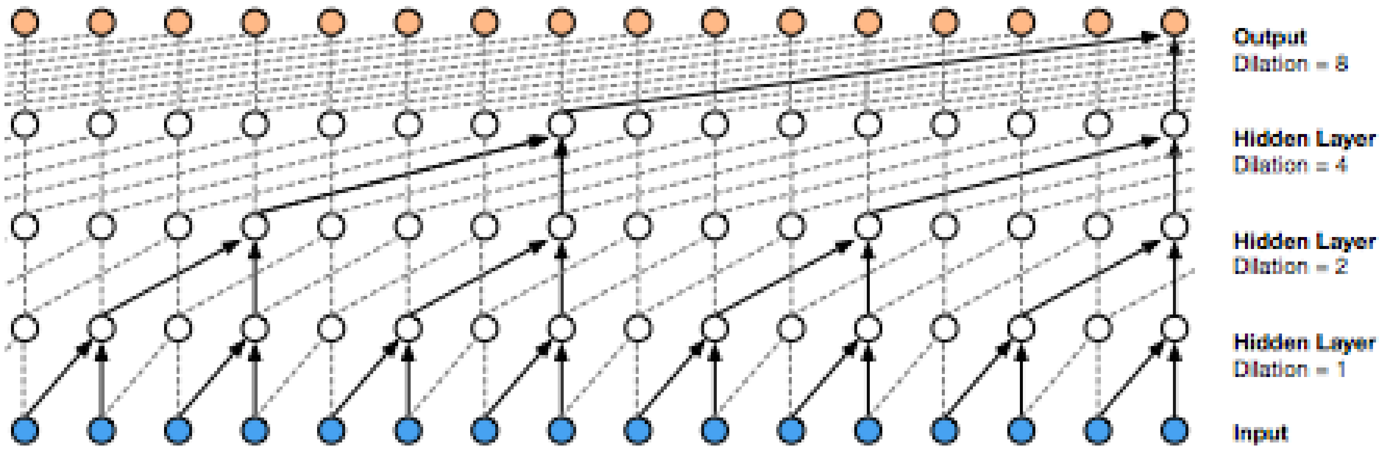
Decoder

Encoder



Dilated TCN Model [Lea et al., CVPR 2017]

Dilated Convolutions



Dilation of 4: Step size of 4 when convoluting

+ Skip connections to help with deep modeling

Dilated TCN Models [Lea et al., CVPR 2017]