CS7150 Deep Learning

Jiaji Huang

https://jiaji-huang.github.io

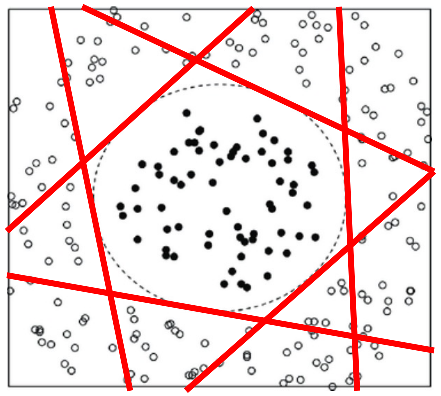
01/27/2024

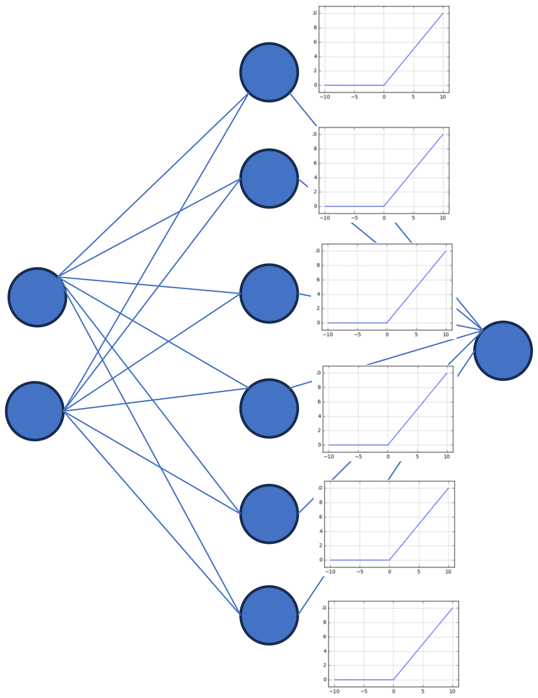
Announcement

- Homework Submission via Canvas
  - Derivations, discussions in pdf format
  - Code submission via
- 01/29: Last day to drop without a W grade
- Presentations start from next lecture (02/03)
  - (randomized) order to present
  - No presentation on days of exam and project report
  - 30~45min with discussion

Recap of Last Lecture

- Supervised Learning, e.g., classification
- Non-parametric method, e.g., Nearest Neighbor Classifier
- Parametric methods
  - Logistic regression
  - Softmax classifier
  - MLP (feed-forward network)



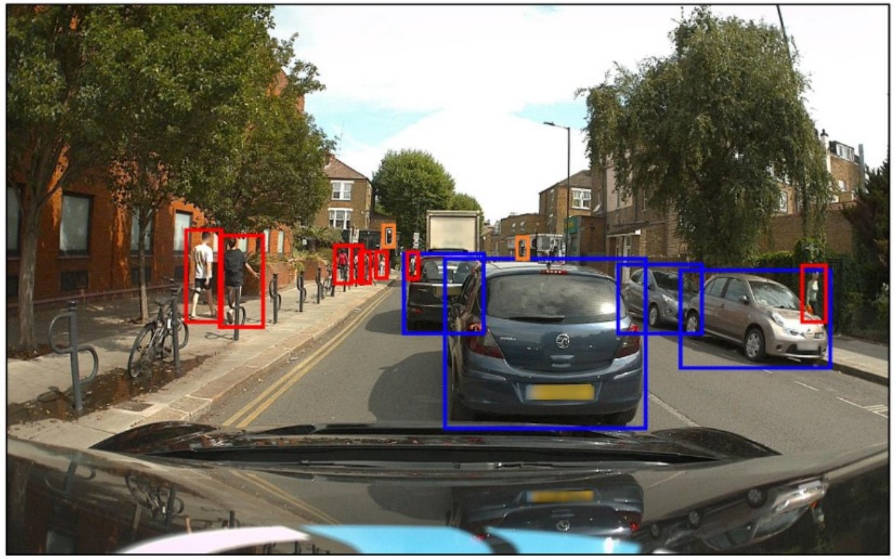


Agenda

- Introduction: Vision Tasks
- Building Blocks
- Convolutional Networks
- Beyond Image Classification

Vision Tasks: Understanding

- Image classification
- Object detection

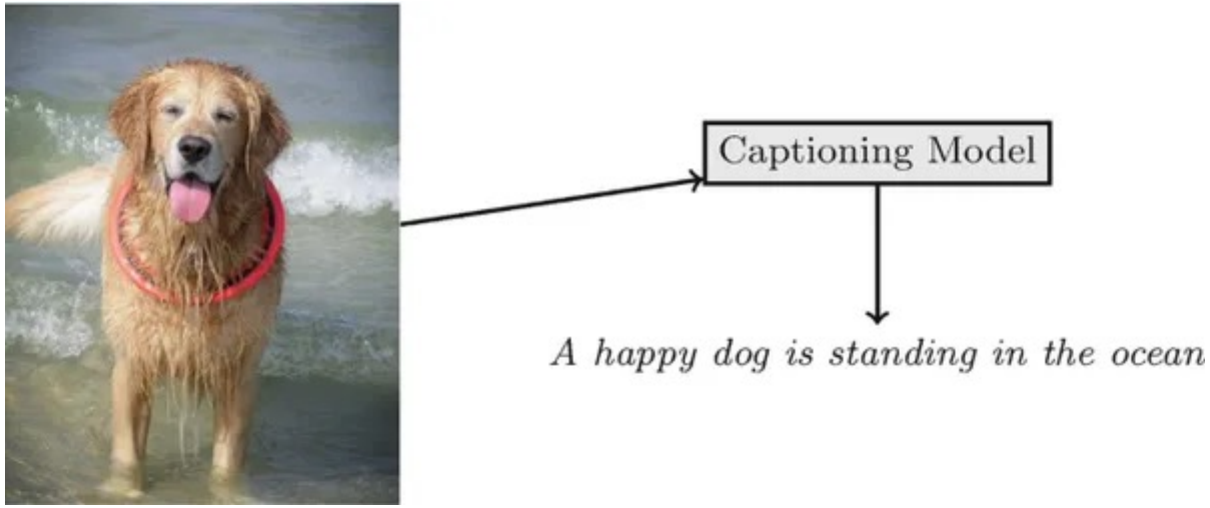


- Image segmentation

Illustrations from Chapter 10 of [Deep Learning Foundations and Concepts]

Vision Tasks: Generation

- Image captioning



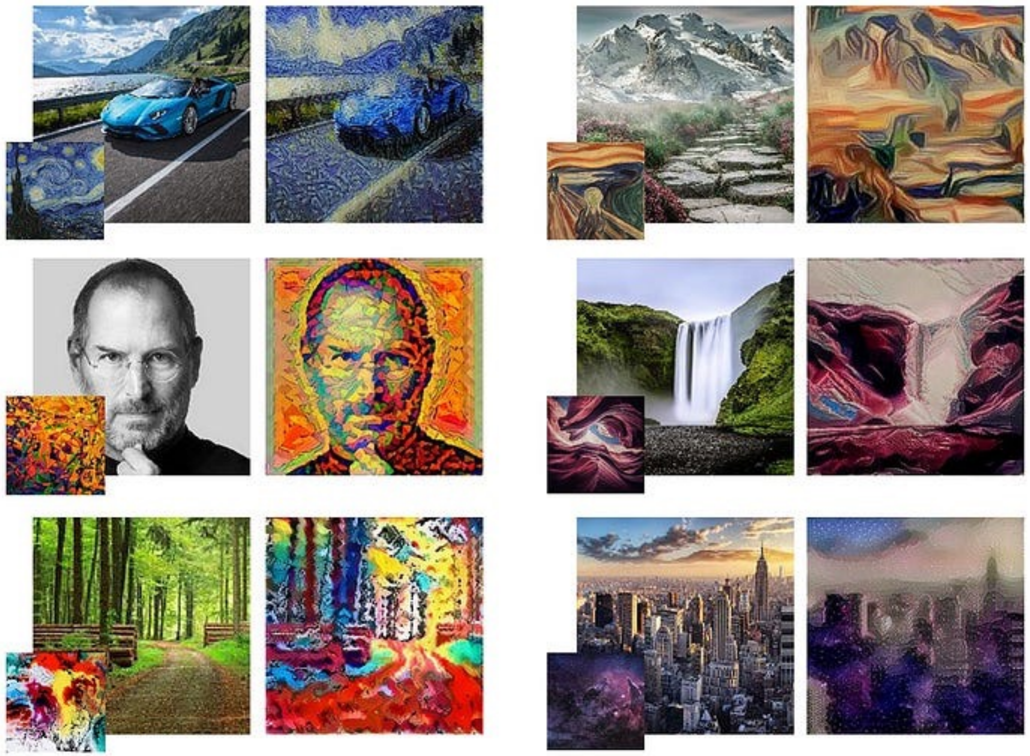
- Creating Image from text

an armchair in the shape of an avocado

Illustrations from this [article] and [DALL-E]

Vision Tasks: Other

- Style transfer



- Super Resolution

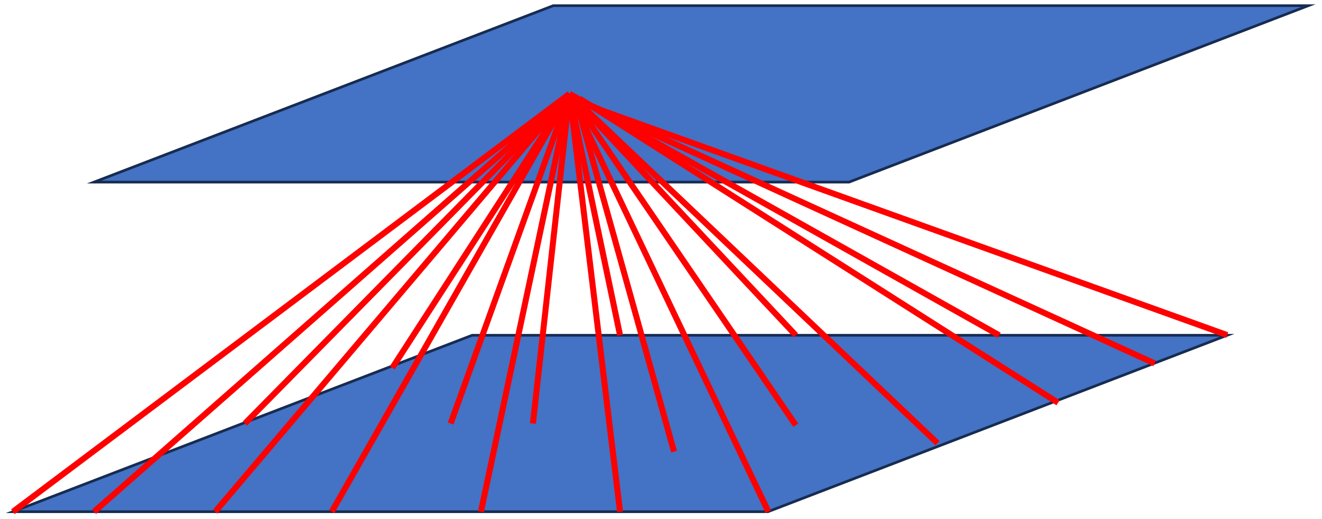
Illustrations from this blogpost and chapter 20 of Deep Learning Foundations and Concepts

Agenda

- Introduction: Vision Tasks
- Building Blocks
- Convolutional Networks
- Beyond Image Classification

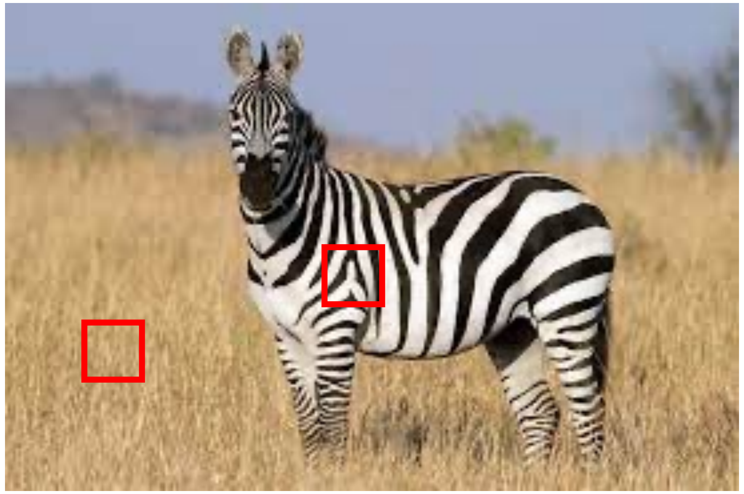
MLP for image tasks?

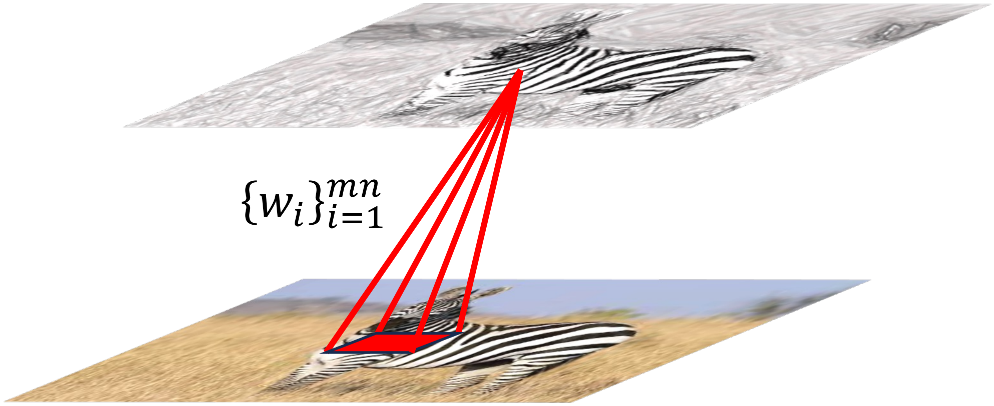
- Not necessarily the best choice
- Images are big, $H \times W$ image has HW pixels
- Mapping to same dimension requires $O((HW)^2)$parameters
- **Receptive field**: the input pixels that the output depends
  - Each output depends all input
  - Very large receptive field



Motivating

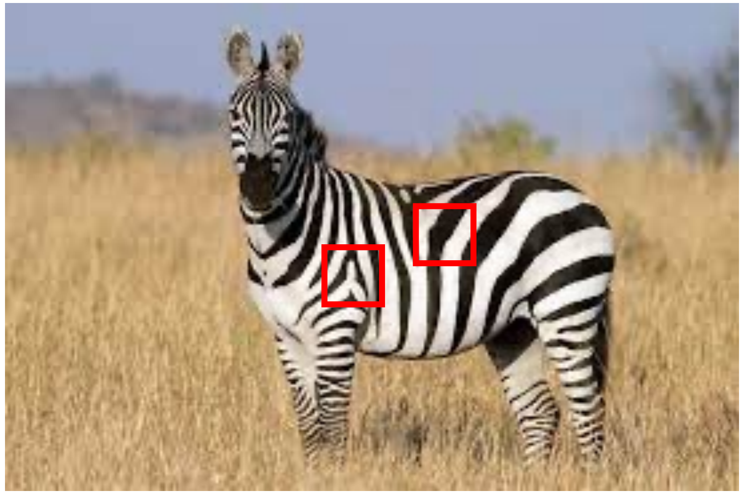
- Task: decide if zebra exists in an image
- The receptive field
  - no need to look into the grass
  - be sensitive to zebra texture
- One idea:
  - depending on neighborhood only
  - parameter reduced to $O(HW \times mn)$

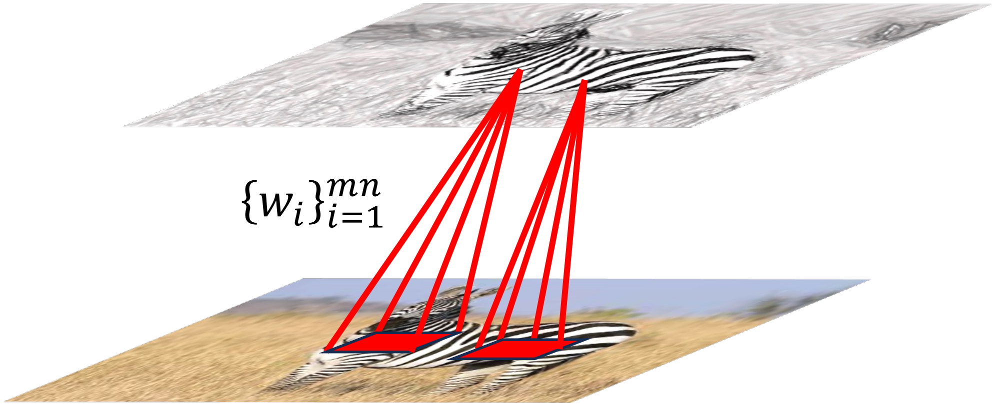




Motivating

- Further, Local structures (e.g., edges) can be repeated
- Share parameters at different locations
  - $O(mn)$ parameters

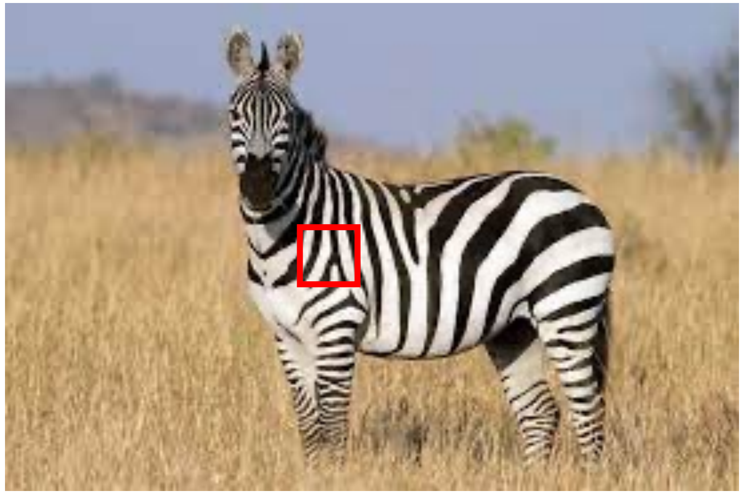




Convolution Operator

- **Convolutional kernel/mask:** the $m \times n$ weights in the small neighborhood
- The mask may look like  , to pick up the zebra
- Stride the kernel inside the image, at each position (x, y):
  - Multiply the pixel values and kernel weights, element-wisely
  - Sum the products

≈ 0



$=$ Big value!

1D Convolution: Continuous Case

- Signal $f(t)$, kernel $g(t)$, convolution is defined as

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

- $f * g = g * f$
- How to compute:
  - Express using dummy variable τ
  - Reflect $g(\tau)$ to $g(-\tau)$
  - Move to offset t
  - Integrate their product (where they overlap)

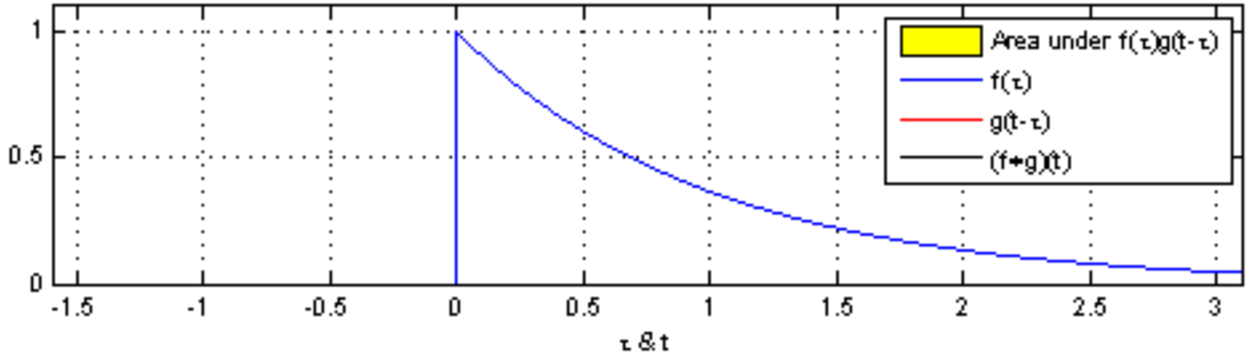


Illustration from [wikipage]

1D Convolution: Discrete Case

- kernel g_n, signal f_n

$$(f * g)_n = \sum_{k=-\infty}^{\infty} f_k g_{n-k}$$

$$= \sum_{k=-\infty}^{\infty} f_{n-k} g_k = (g * f)_n$$

- Finite length g: defined on $n = -M, \ldots, M$

$$(f * g)_n = \sum_{k=-M}^{M} f_{n-k} g_k$$

- Exercise: If f has length N, length of $f * g$?
  $N + M - 1$

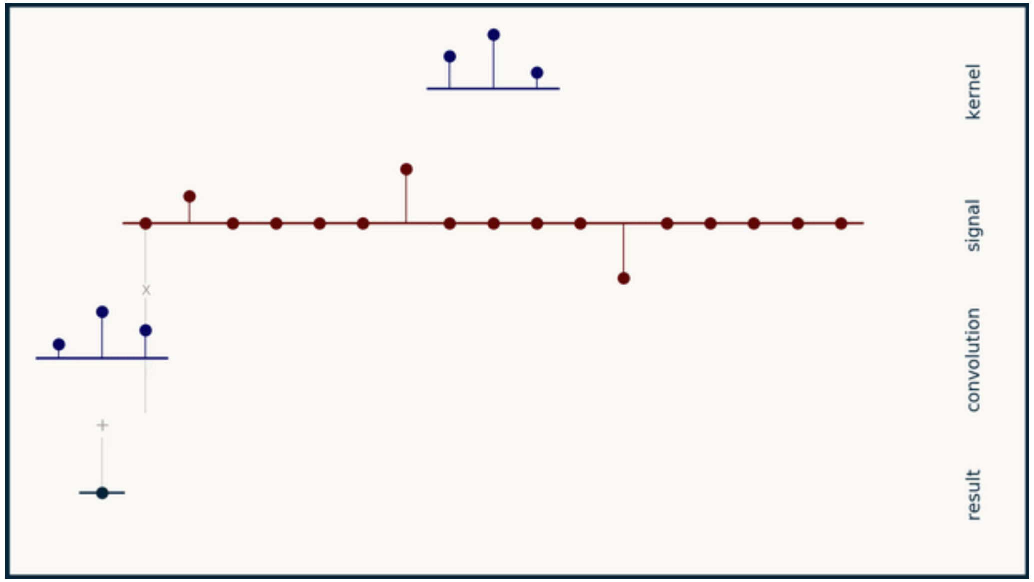


Illustration from [here]

Convolution v.s. correlation

- Correlation doesn't flip f or g

$$f \star g(\tau) = \int_{t=-\infty}^{\infty} f(t)g(t+\tau)$$

$$f \star g[k] = \sum_{k=-\infty}^{\infty} f[n]g[n+k]$$

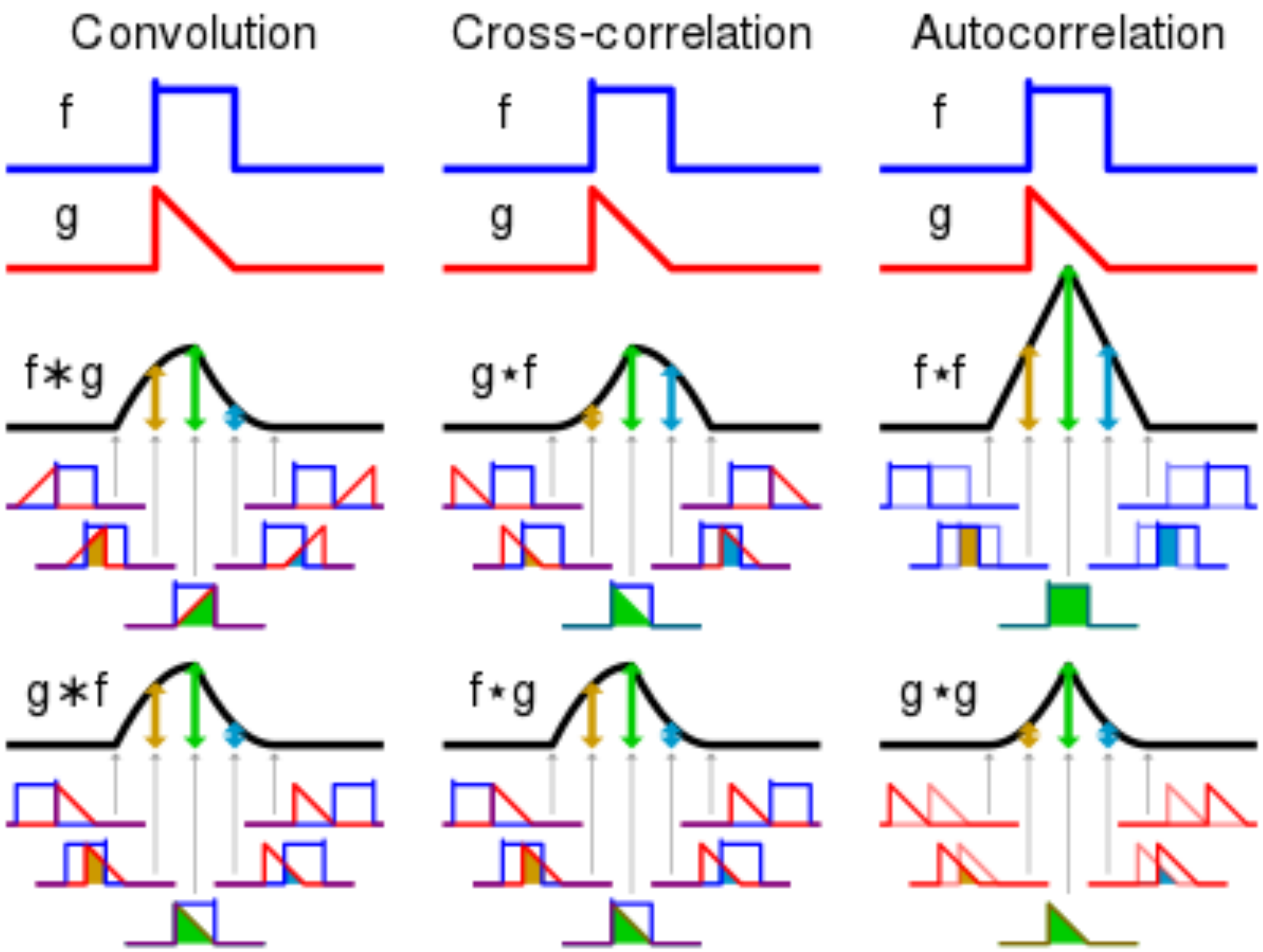
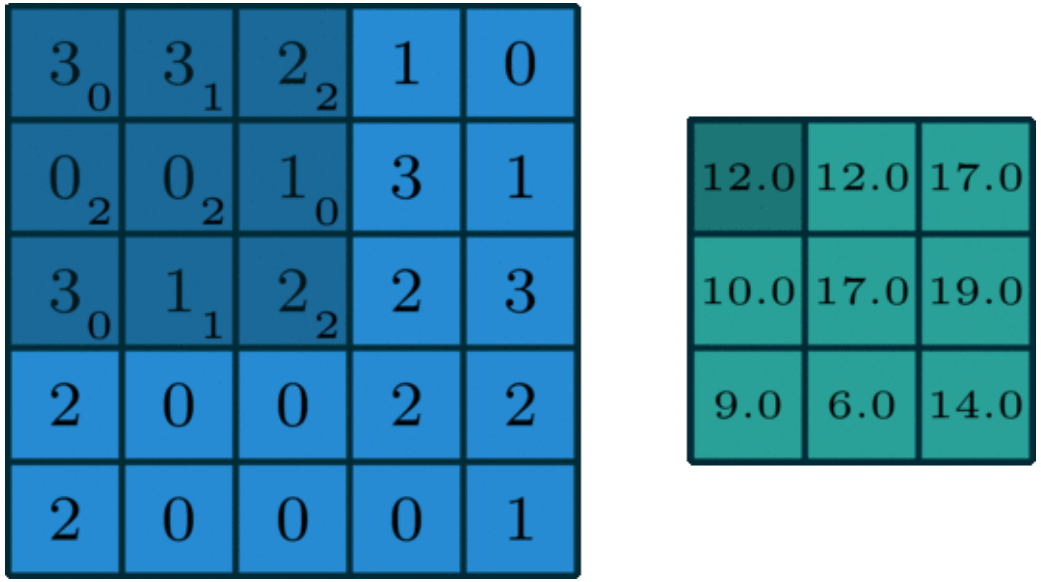


Illustration from [wiki]

2D convolution: Discrete Case

- 2D kernel κ for input image $\boldsymbol{I}$, compute

$$\kappa \star \boldsymbol{I} = \sum_{i=-(m-1)/2}^{(m-1)/2} \sum_{j=-(n-1)/2}^{(n-1)/2} \kappa(i,j)\boldsymbol{I}(x+i, y+j)$$



- Note:
  - A kernel is also called a filter
  - Convolution flips the kernel. But in deep learning we don't. In fact, we are calculating correlation. Why?

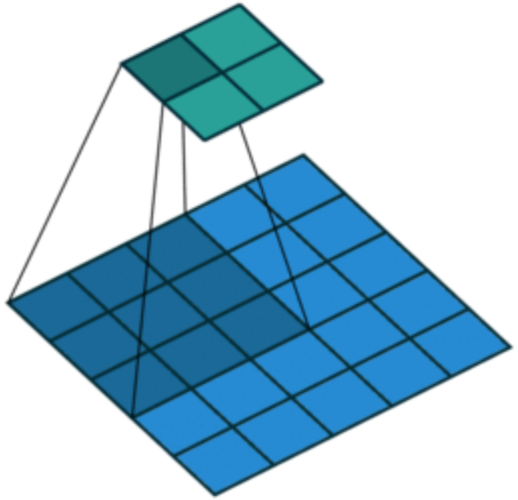
Illustration from here

<u>Exercise</u>: $H \times W$ image, kernel size $m \times m$, what is the output size?

$$(H - m + 1) \times (W - m + 1)$$

Variants

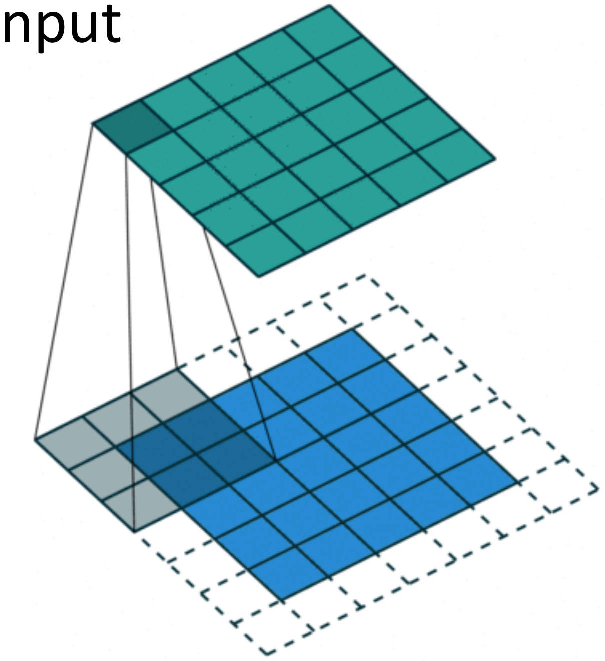
- Stride
  - We move the mask by “stride” positions each time
  - Stride > 1 down samples the image



Stride = 2

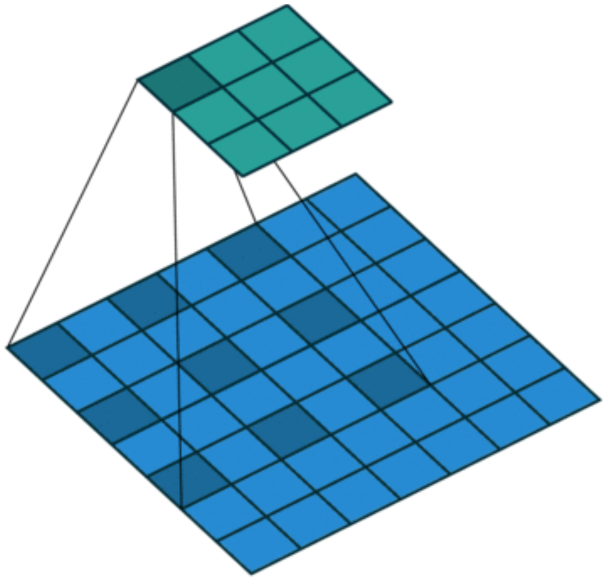
Illustration from [here]

- Padding
  - Previous page: ”valid” padding
  - “same” padding: pad zeros outside boundary so output image size is same as input



Variants

- Dilation: the space between kernel elements



Dilation=2

Illustration from here

bias:

$$\kappa * \boldsymbol{I} = \sum_{i=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{j=-\frac{n-1}{2}}^{\frac{n-1}{2}} \kappa(i,j)\boldsymbol{I}(x+i, y+j) + b$$

<u>Exercise:</u> $H \times W$ image, kernel size $m \times m$, padding="valid", stride=s,

Dilation=d, what is the output image size?

Some typical 2D kernels

- Low-pass filter

1	1	1
1	1	1
1	1	1

1	1	1
1	2	1
1	1	1

1	2	1
2	4	2
1	2	1

  - Gaussian kernel

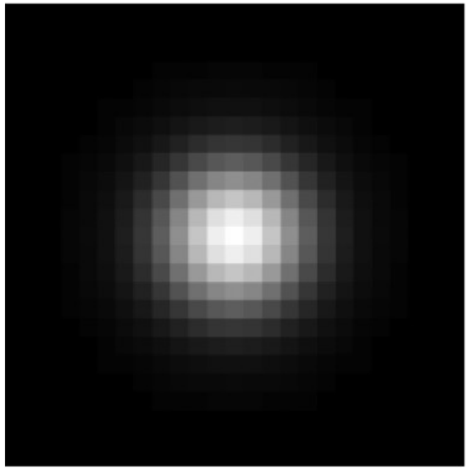
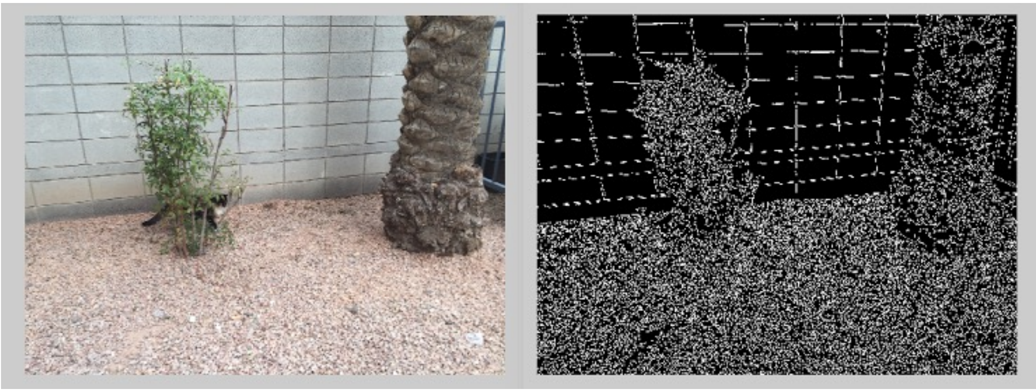


Illustration from [wiki]

- High-pass filter

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1



Original image (left) and image after passing through edge-detecting filter (right)

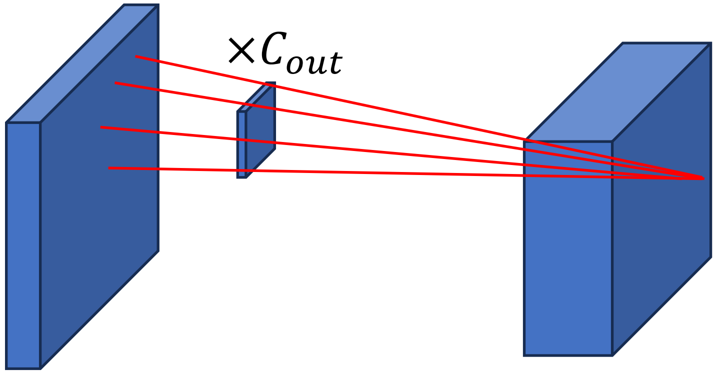
Multi-channel Input and output

- Generally, input image size $C_{in}\times H\times W$ (C_{in}=3 for RGB image)
- Kernel size $C_{in}\times m\times n$

$$\sum_{c=1}^{C_{in}} \kappa[c] * \boldsymbol{I}[c]$$

- Multi-channel output, kernel size $C_{out}\times C_{in}\times m\times n$
  - The c'-th output channel is

$$\sum_{c=1}^{C_{in}} \kappa[c', c] * \boldsymbol{I}[c]$$



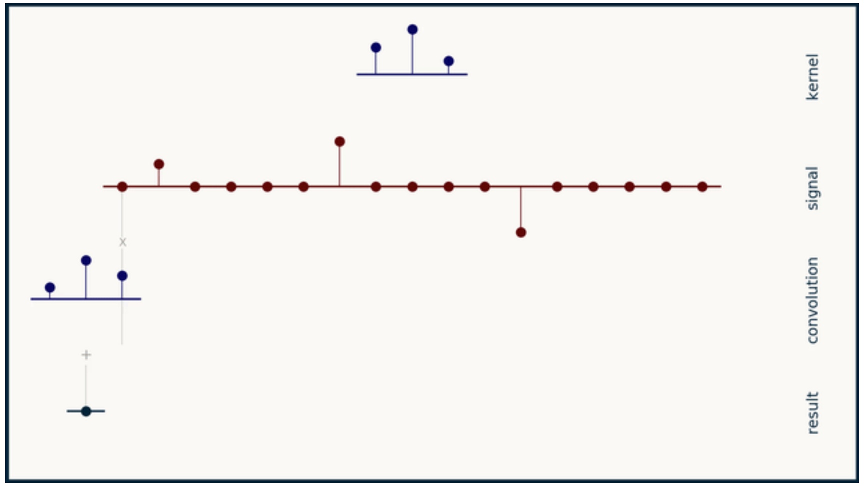
Connection with matrix multiplication

- 1D conv as Toeplitz matrix multiplication

$$(f * g)_n = \sum_{k=-M}^{M} f_{n-k} g_k$$

e.g., example in the right

$$= \begin{bmatrix} g_{-1} & 0 & \dots & & \\ g_0 & g_{-1} & 0 & \dots & \\ g_1 & g_0 & g_{-1} & 0 & \\ & & \ddots & & \\ 0 & & g_1 & g_0 & g_{-1} \\ 0 & & 0 & g_1 & g_0 \\ 0 & & 0 & 0 & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-1} \\ f_N \end{bmatrix}$$



- Exercise: 2D conv?

Translational Equivalence

- Shift the input by δ grids, output is also shifted by δ
- Proof in 1D continuous case: we know

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Consider $\tilde{f}(t) = f(t-\delta)$,

$$\begin{aligned}
\left(\tilde{f} * g\right)(t) &= \int_{-\infty}^{\infty} \tilde{f}(\tau)g(t-\tau)d\tau \\
&= \int_{-\infty}^{\infty} f(\tau-\delta)g(t-\tau)d\tau \\
&= \int_{-\infty}^{\infty} f(\tau)g\big((t-\delta)-\tau\big)d\tau \\
&= (f * g)(t-\delta)
\end{aligned}$$

Translational Equivalence

- Why we need that?
- For example objective detection
- If the the object shifts by δ
- Then the decision box should also shift by δ

Gradient w.r.t. Convolutional Kernel

- Consider single channel in, single channel out

$$\boldsymbol{O}[x, y] = \sum_{i,j \in \mathcal{N}} \kappa[i, j] \cdot \boldsymbol{I}[x + i, y + j]$$

$$\frac{\partial \boldsymbol{O}[x,y]}{\partial \kappa[i,j]} = \boldsymbol{I}[x + i, y + j]$$

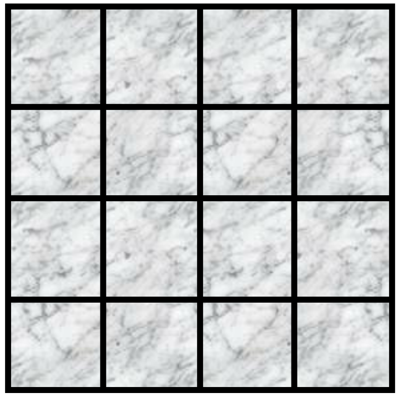
- Chain rule

$$\frac{\partial \ell}{\partial \kappa[i, j]} = \sum_{x,y} \frac{\partial \ell}{\partial O[x, y]} \cdot \boldsymbol{I}[x + i, y + j]$$

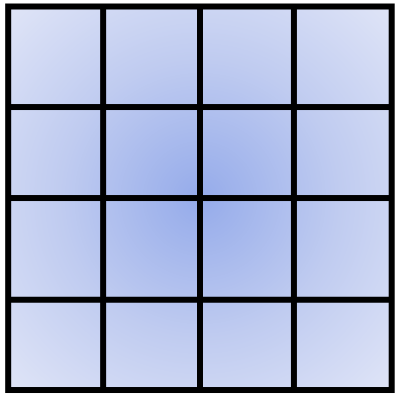
Gradient w.r.t. Convolutional Kernel

$$\frac{\partial \ell}{\partial \kappa[i,j]} = \sum_{x,y} \frac{\partial \ell}{\partial O[x,y]} \cdot \boldsymbol{I}[x+i, y+j]$$

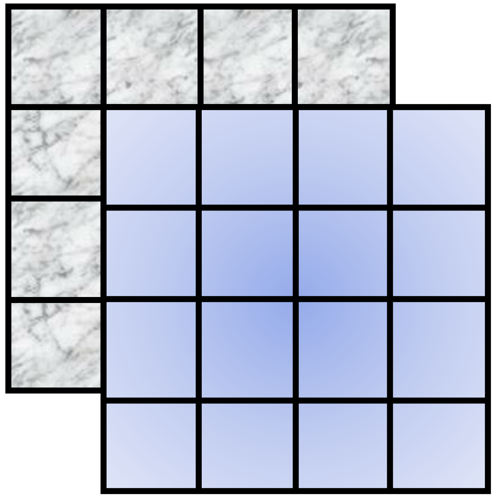
- Consider $\frac{\partial \ell}{\partial \boldsymbol{O}}$ as an image, correlating $\frac{\partial \ell}{\partial \boldsymbol{O}}$ and $\boldsymbol{I}$ for $i, j \in \mathcal{N}$



$\boldsymbol{I}$



$\frac{\partial \ell}{\partial \boldsymbol{O}}$



$$\frac{\partial \ell}{\partial \kappa[1,1]} = \left(\frac{\partial \ell}{\partial \boldsymbol{O}} \star \boldsymbol{I}\right)[1,1]$$

Gradient w.r.t. input

- To derive $\frac{\partial \ell}{\partial \boldsymbol{I}[x,y]}$, note

$$\boldsymbol{O}[x,y] = \sum_{i,j\in\mathcal{N}} \kappa[i,j]\cdot \boldsymbol{I}[x+i,y+j]$$

$$\Leftrightarrow \boldsymbol{O}[x-i,y-i] = \sum_{i,j\in\mathcal{N}} \kappa[i,j]\cdot \boldsymbol{I}[x,y]$$

$$\frac{\partial \ell}{\partial \boldsymbol{I}[x,y]} = \sum_{i,j\in\mathcal{N}} \frac{\partial \ell}{\partial \boldsymbol{O}[x-i,y-j]}\cdot \kappa[i,j]$$

- Consider $\frac{\partial \ell}{\partial \boldsymbol{O}}$ as an image, convolve $\frac{\partial \ell}{\partial \boldsymbol{O}}$ with κ

Pooling

- Max pooling and average pooling
- Down samples the image
- Variants
  - Stride
  - Padding
  - Dilation

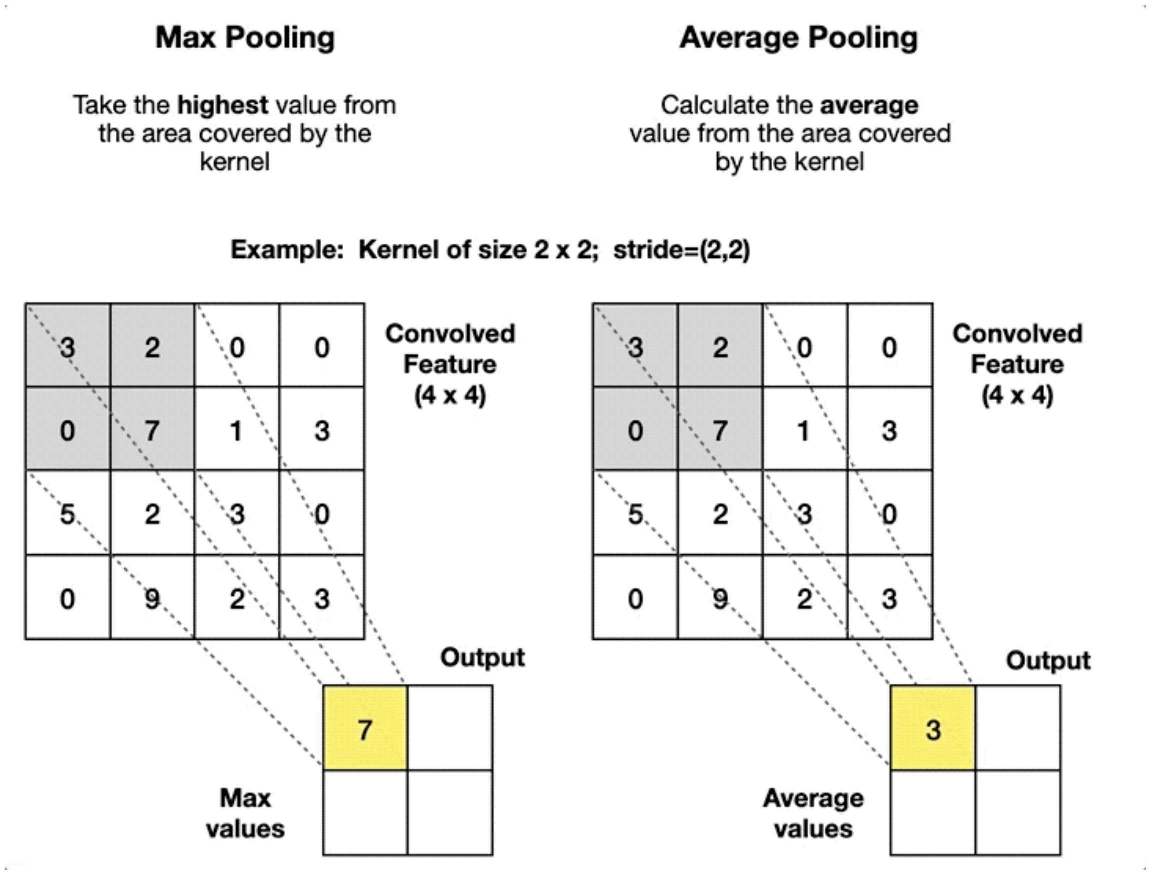


Illustration from [here]

Translation Invariance

- Input translated by δ, output doesn't change
- Translate the zebra should maintain a decision that it exists
- Max Pooling achieves translational invariance (at least partly)

3	2	0	0
0	7	1	3
5	2	3	0
0	9	2	3

↑ by 1

0	7	1	3
5	2	3	0
0	9	2	3
6	4	1	4

7	3
9	3

7	3
9	4

All input values change,
But only 1 of 4 output values change

Gradient of Pooling

- $y = \max\{x_1, \ldots, x_i, \ldots\}$
- Suppose $\arg\max_i x_i = k$
- Then $\frac{\partial y}{\partial x_i} = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases}$

Only the max position receives a gradient!

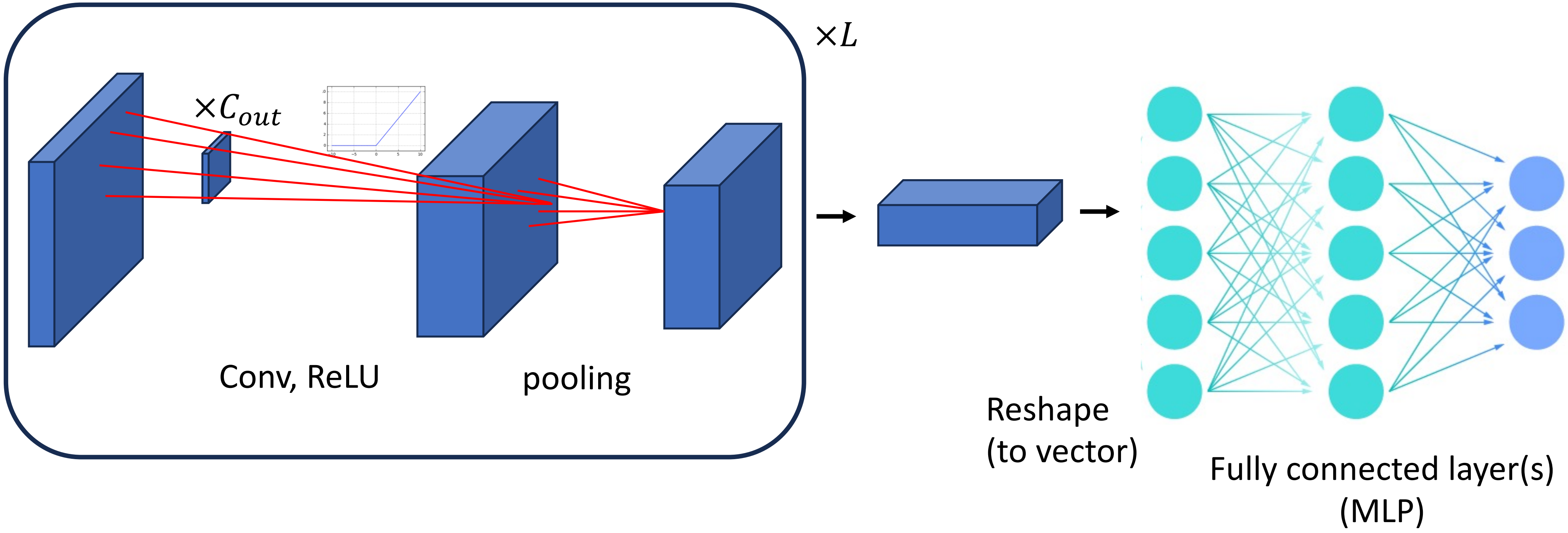
3	2	0	0
0	7	1	3
5	2	3	0
0	9	2	3

Agenda

- Introduction: Vision Tasks
- Building Blocks
- Convolutional Networks
- Beyond Image Classification

Basic network architecture

- Alternate between conv + activation and pooling

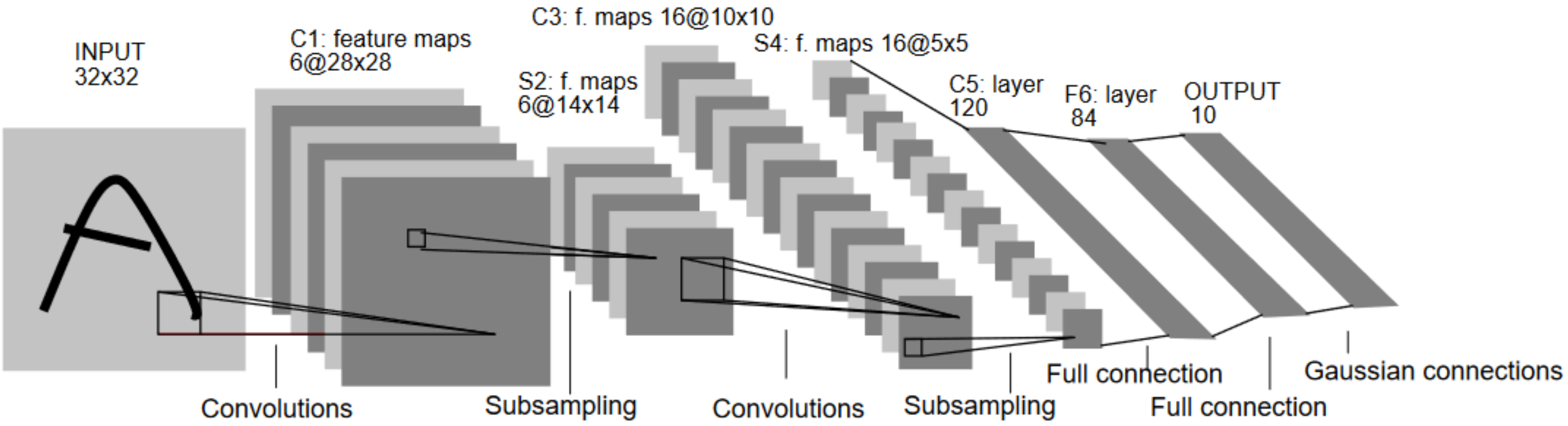


LeNet

Institute of Electrical and Electronics Engineers
https://ieeexplore.ieee.org › document

Gradient-based learning applied to document recognition

by Y Lecun · 1998 · Cited by 61889 — This **paper reviews** various methods applied to handwritten character recognition and compares them on a standard handwritten digit...



Video, colab

AlexNet

Neural Information Processing Systems
https://proceedings.neurips.cc › paper › 4824-i... PDF

ImageNet Classification with Deep Convolutional Neural ...

by A Krizhevsky · Cited by 124370 — The specific contributions of this **paper** are as follows: we trained one of the largest convolutional neural networks to date on the subsets of ImageNet...
9 pages

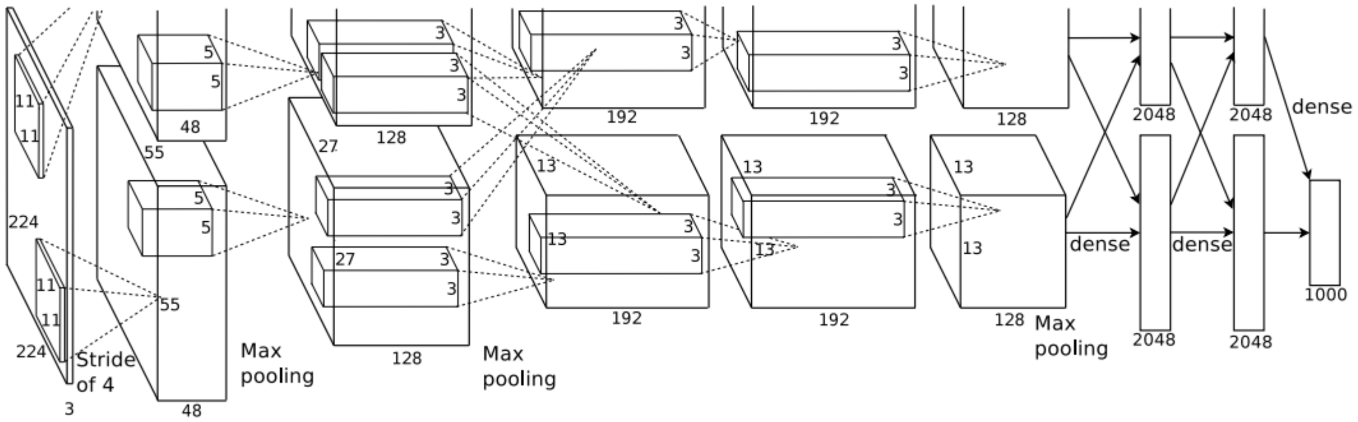


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Model	**Top-1**	**Top-5**
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

AlexNet

- Key innovations:
  - Local response normalization
  - Data augmentation
  - Dropout on layer 7 and 8
  - Momentum SGD

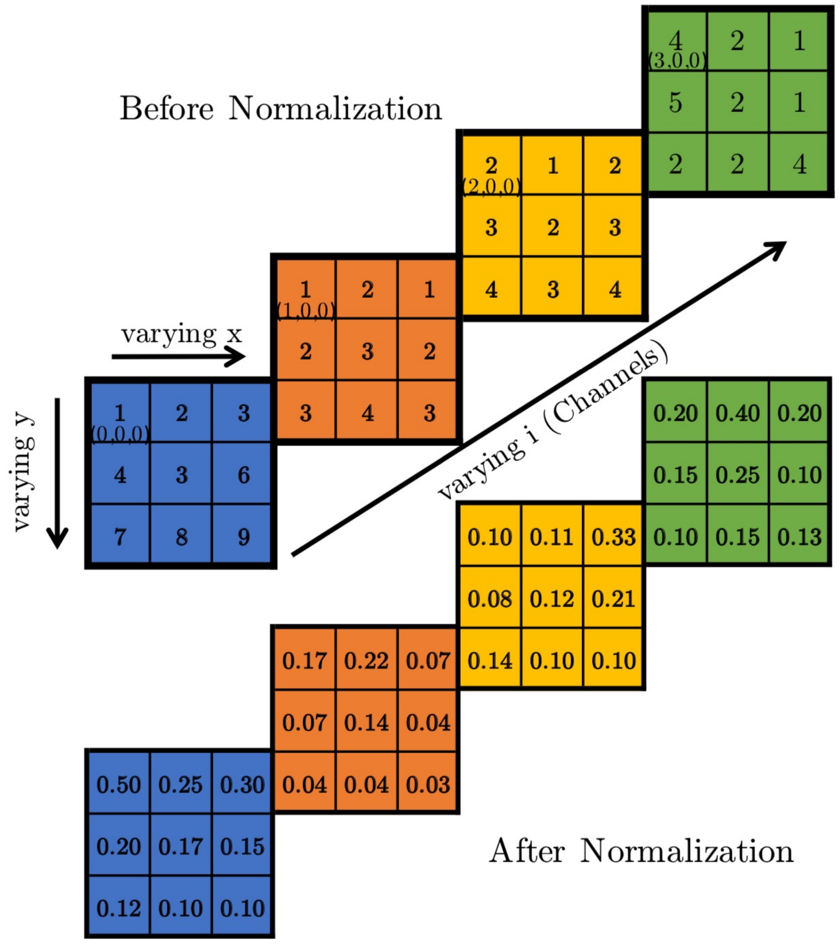
Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Local response normalization

- For each spatial location (x, y), at channel i
- normalize over adjacent n channels

$$b_{x,y}^{i} = a_{x,y}^{i} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^{j})^2 \right)^{\beta}$$

- Improve contrast in neighborhood
- dampen constant neighborhood
- Motivation in [lateral inhibition]

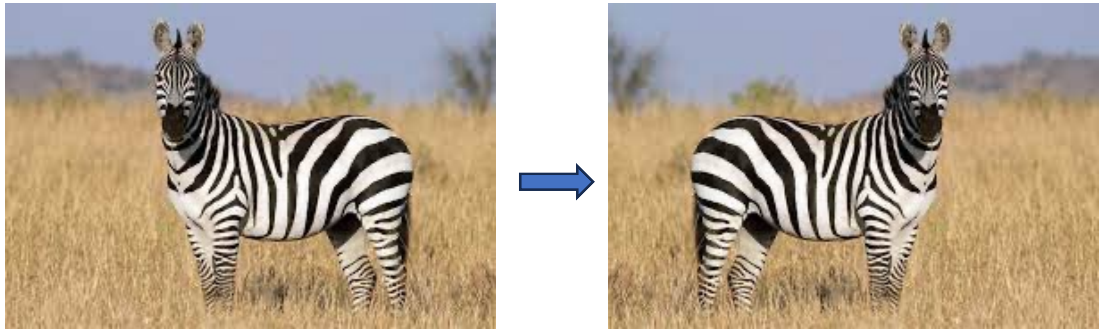


$(k, \alpha, \beta, n) = (0,1,1,2)$

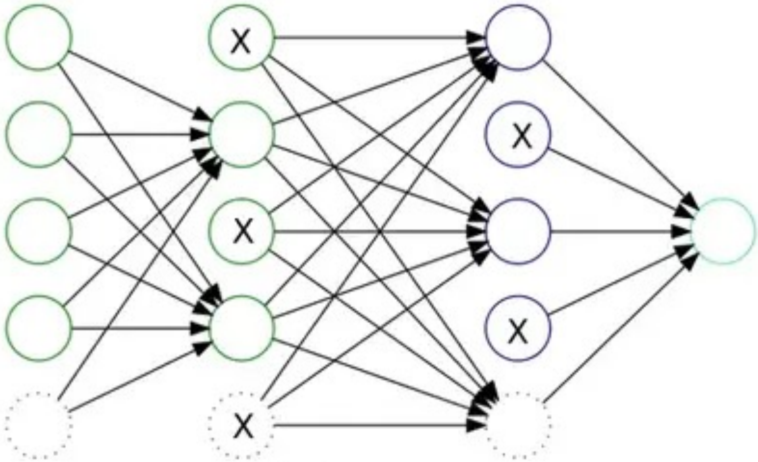
Illustration from [here]

Reduce Overfitting

- Data Augmentation: introduce variations in the input image
  - Reflection, shift, …
  - Other augmentations?



- Dropout
  - At training: Randomly zeros some activations
  - At testing: disabled
  - Effect: Ensemble many models



Momentum SGD

- Weighted moving average of gradient

$$\Delta_s = \beta \cdot \Delta_{s-1} + \nabla \ell(\boldsymbol{w})$$
$$\boldsymbol{w}_s = \boldsymbol{w}_{s-1} - \gamma \cdot \Delta_s$$

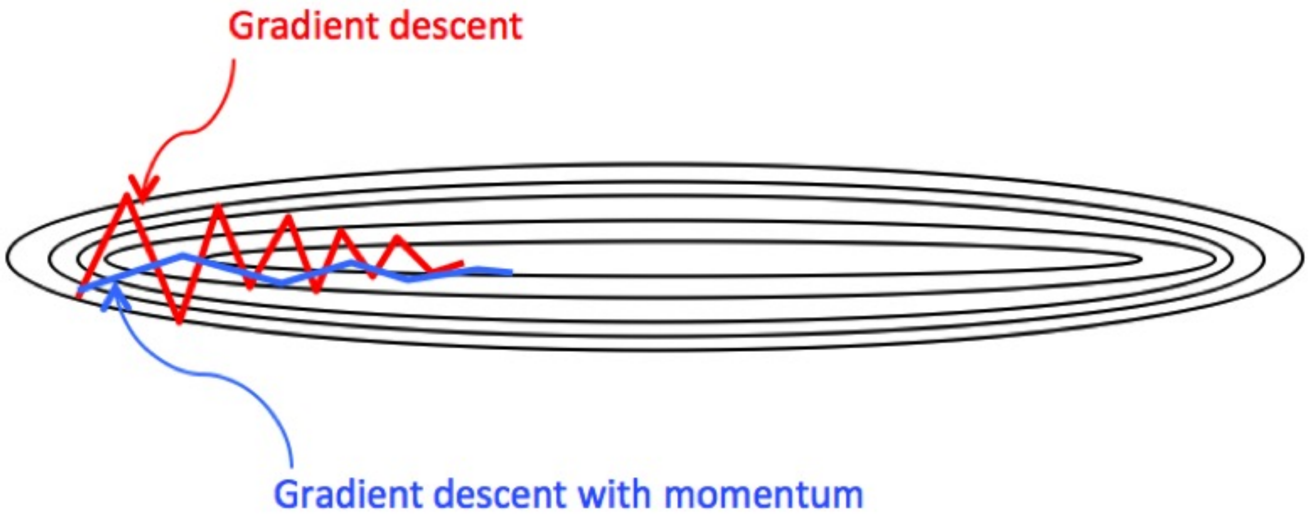
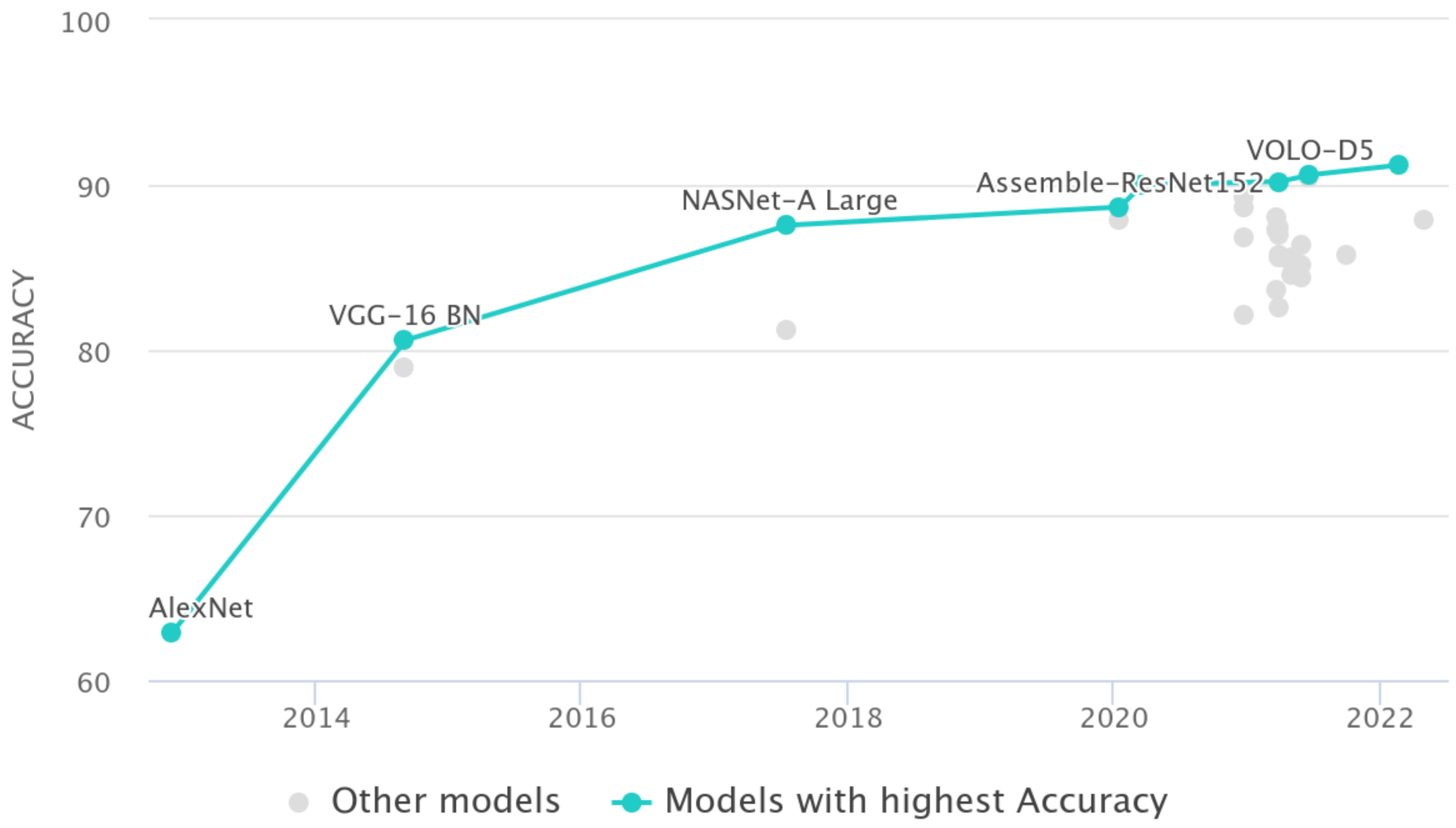


Illustration from this [blogpost]

Since AlexNet …



Trend according to [paperwithcode]

VGG

arXiv
https://arxiv.org › cs

Very Deep Convolutional Networks for Large-Scale Image ...

by K Simonyan · 2014 · Cited by 117289 — Our main contribution is a thorough evaluation of **networks** of increasing depth using an architecture with **very** small (3x3) **convolution** filters, ...

- smaller conv kernel, deeper
- Same receptive field size
  - More ReLU's (nonlinearity)
  - Fewer parameters
  - E.g, two 3x3 conv-layer v.s.

  one 5x5 conv-layer

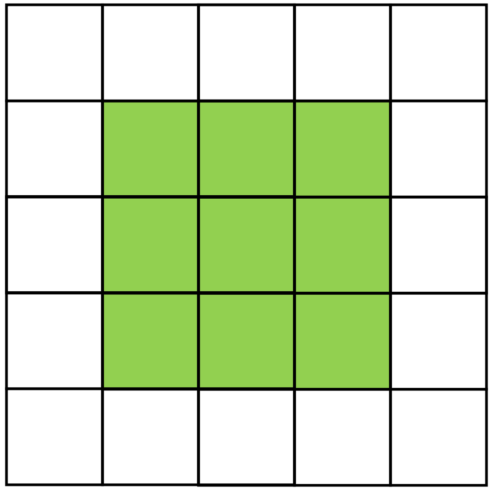


Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

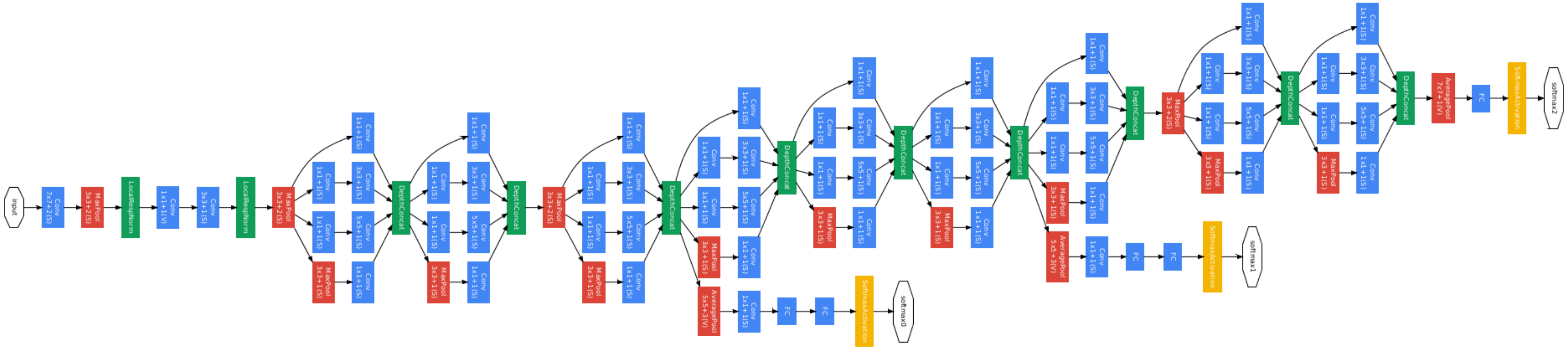
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

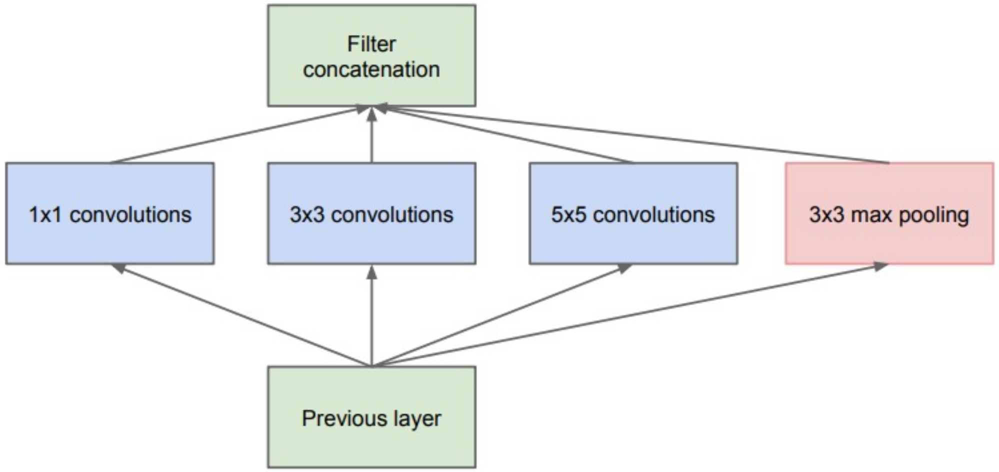
Going deeper

arXiv
https://arxiv.org › cs

[1409.4842] Going Deeper with Convolutions

by C Szegedy · 2014 · Cited by 56587 — We propose a deep convolutional neural **network** architecture codenamed "**Inception**", which was responsible for setting the new state of the art ...

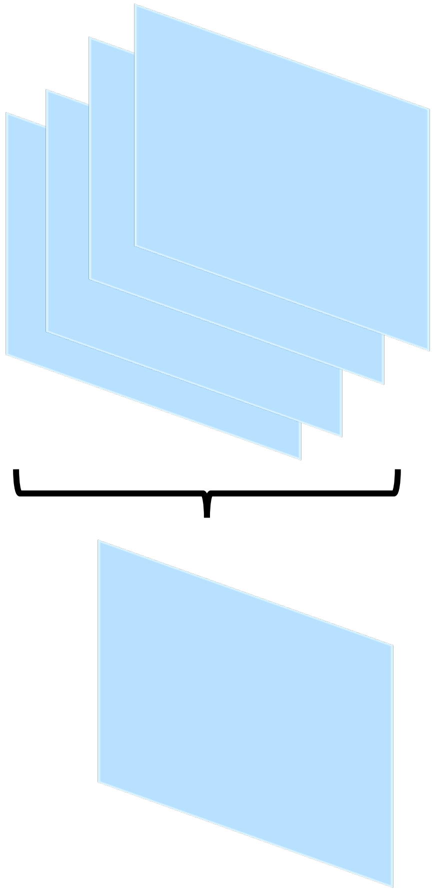




1x1 convolution

arXiv
https://arxiv.org › cs
[1312.4400] Network In Network
by M Lin · 2013 · Cited by 8887 — Abstract:We propose a novel deep network structure called "**Network In Network**" (**NIN**) to enhance model discriminability for local patches ...

- Weighted average of all channels
- Reduce the number of channels in inception net



<u>Exercise</u>

Input: 512×512 RGB image

Layer1: 3×3 filter with 16 output channels,
stride=2, padding="same"

Layer2: 1x1 filter with 8 output channels

What is output shape?

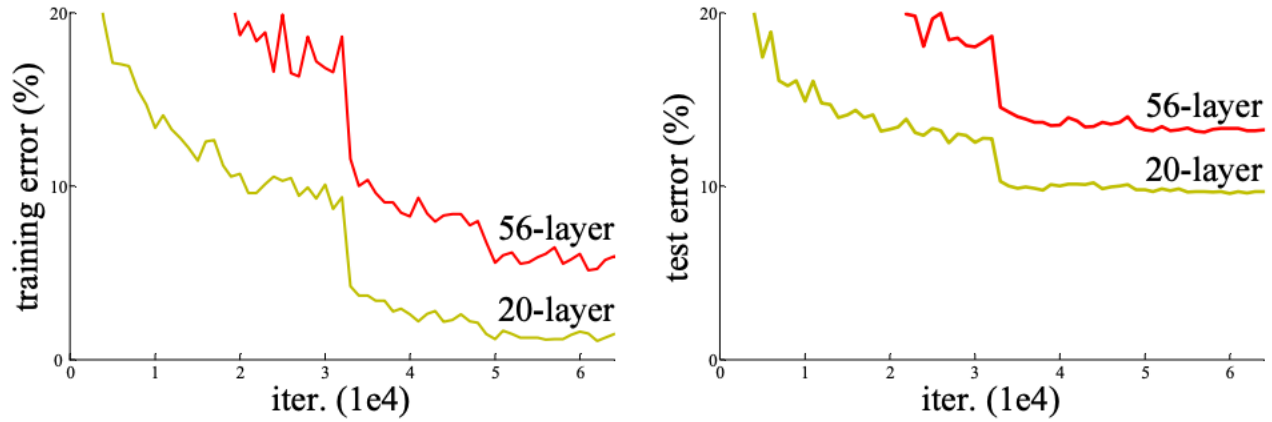
Resnet

arXiv
https://arxiv.org › cs

[1512.03385] Deep Residual Learning for Image Recognition

by K He · 2015 · Cited by 197465 — On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity.

- Deeper may be good
- But harder to train



Resnet

- One conjecture is reducing gradient vanishing
- $\frac{\partial \ell}{\partial x} = \frac{\partial \ell}{\partial y} \cdot \frac{\partial y}{\partial x} = \frac{\partial \ell}{\partial y} \cdot \left(\frac{d\mathcal{F}}{dx} + I\right)$
- But the authors argue
  "optimization difficulty is unlikely

  to be caused by vanishing gradients"

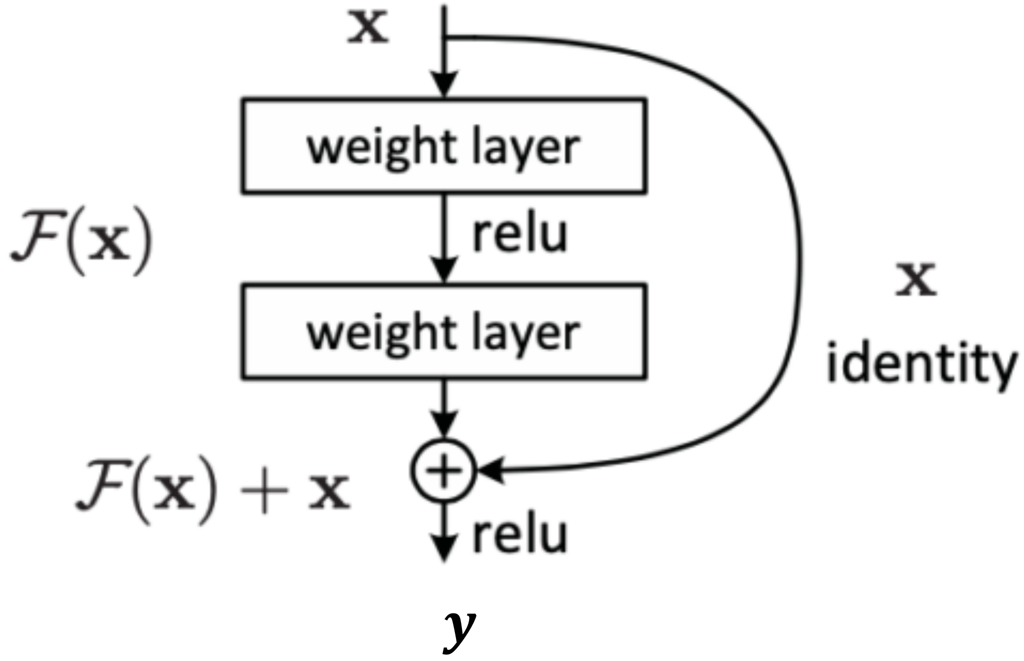


Figure from He *et. al* (2015)

ResNext

- Multiple branches

CVF Open Access
https://openaccess.thecvf.com › papers › Xie_A... PDF

[Aggregated Residual Transformations for Deep Neural ...]

by S Xie · 2017 · Cited by 11501 — A module in our **network** performs a set of **transformations**, each on a low-dimensional embedding, whose outputs are **aggregated** by...

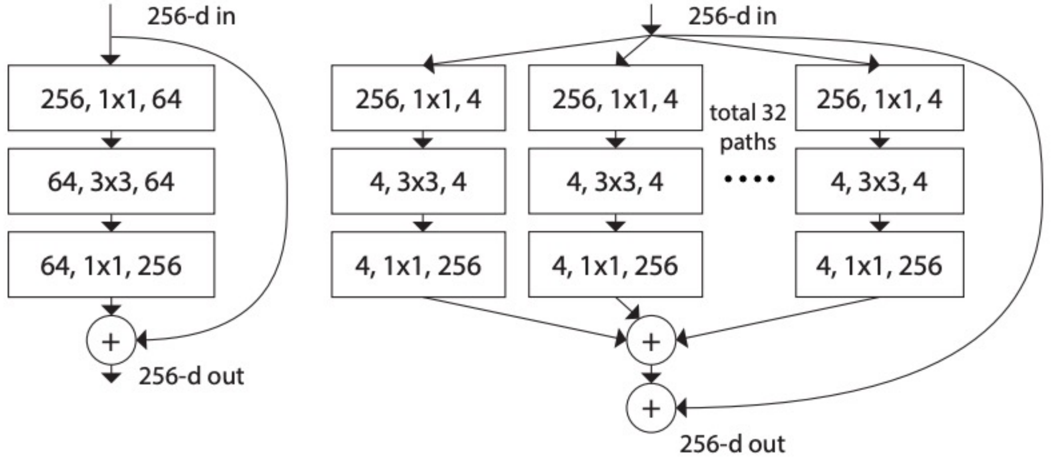
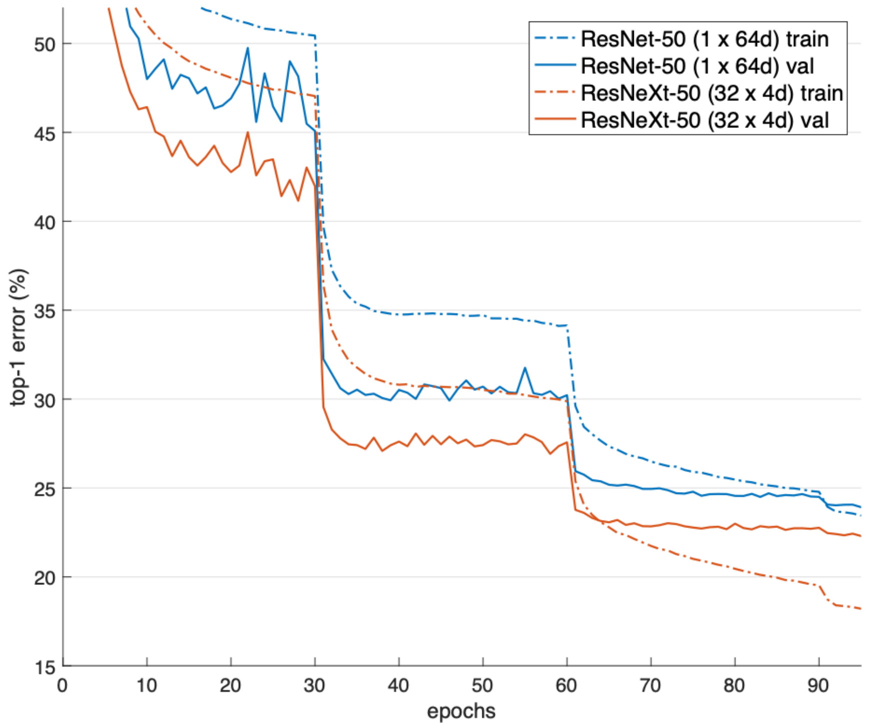


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).



DenseNet

arXiv
https://arxiv.org › cs

[1608.06993] Densely Connected Convolutional Networks

by G Huang · 2016 · Cited by 41724 — **DenseNets** have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature ...

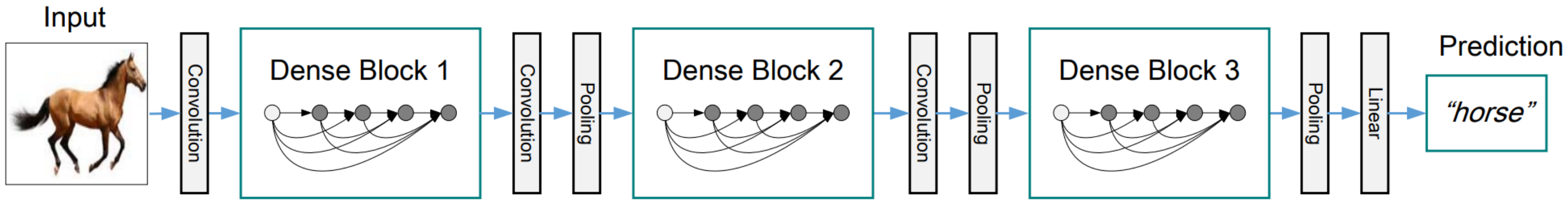


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

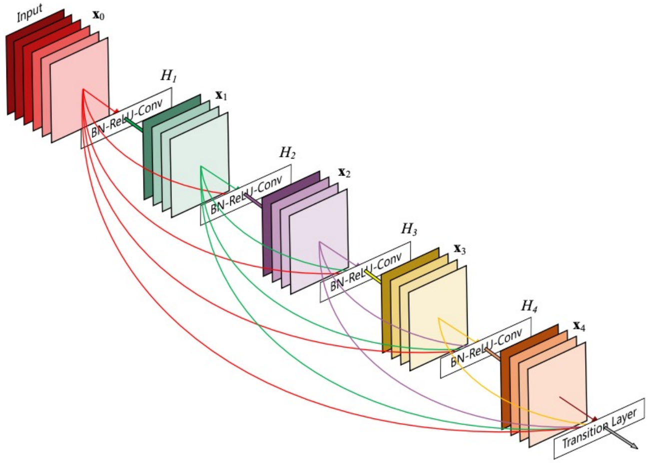
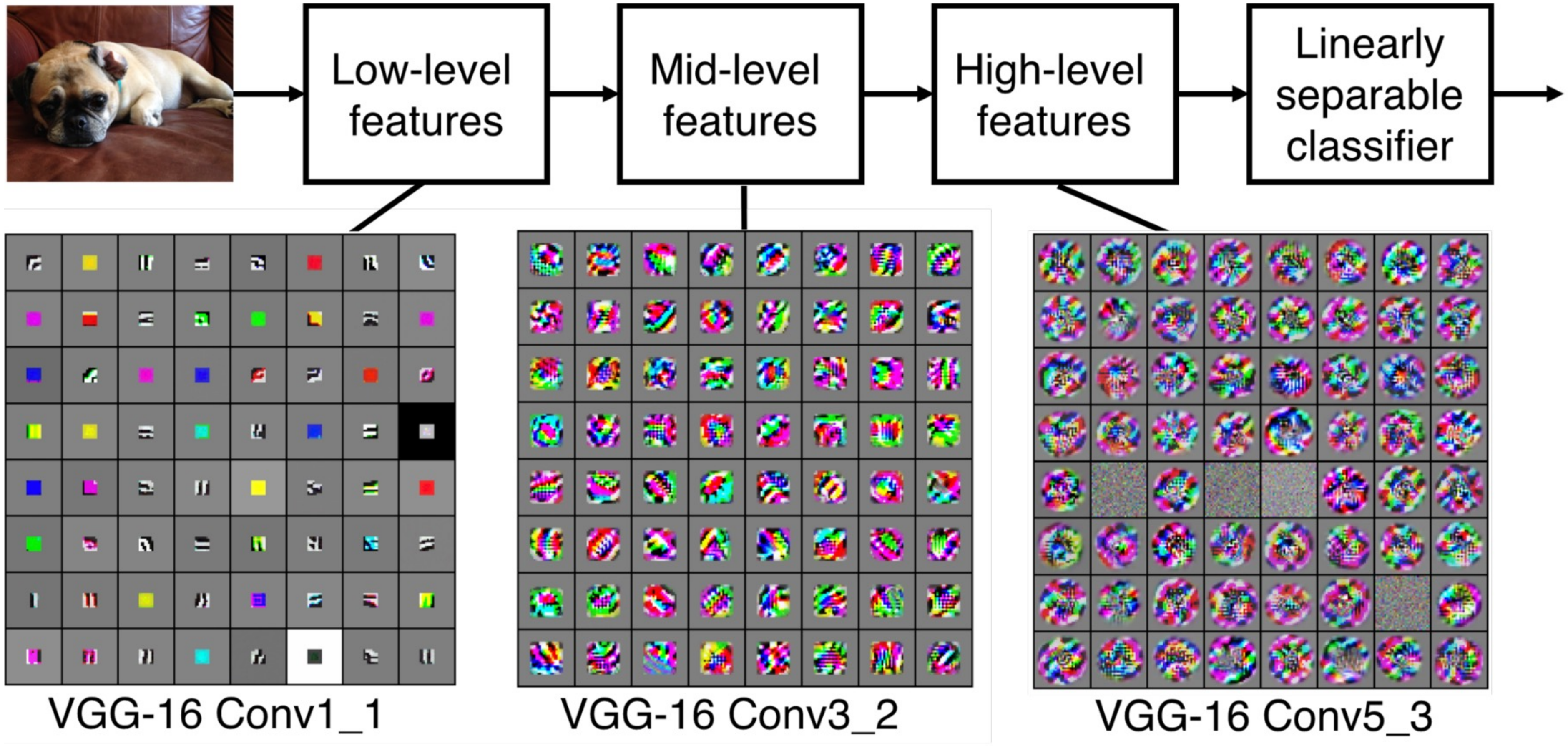


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

What does the network learn?



Slide credit: Fei-fei Li et. al [CS231n]

Visualizing Filters

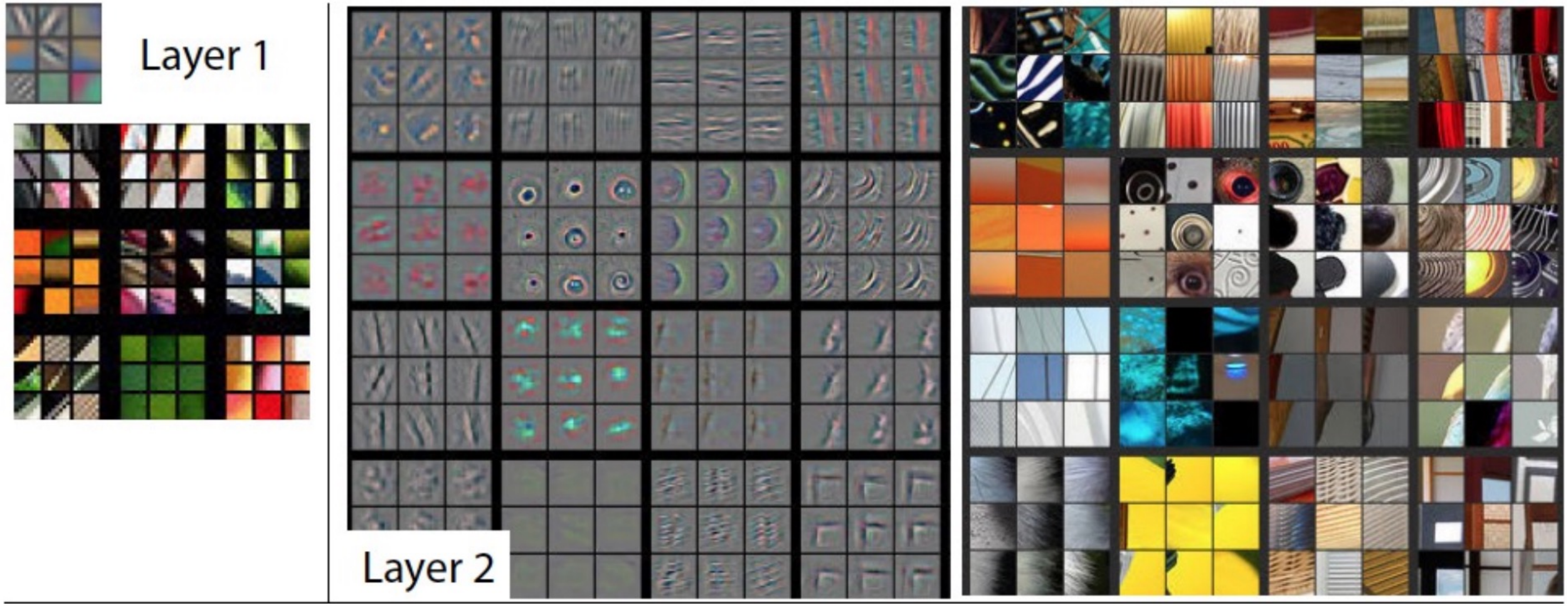
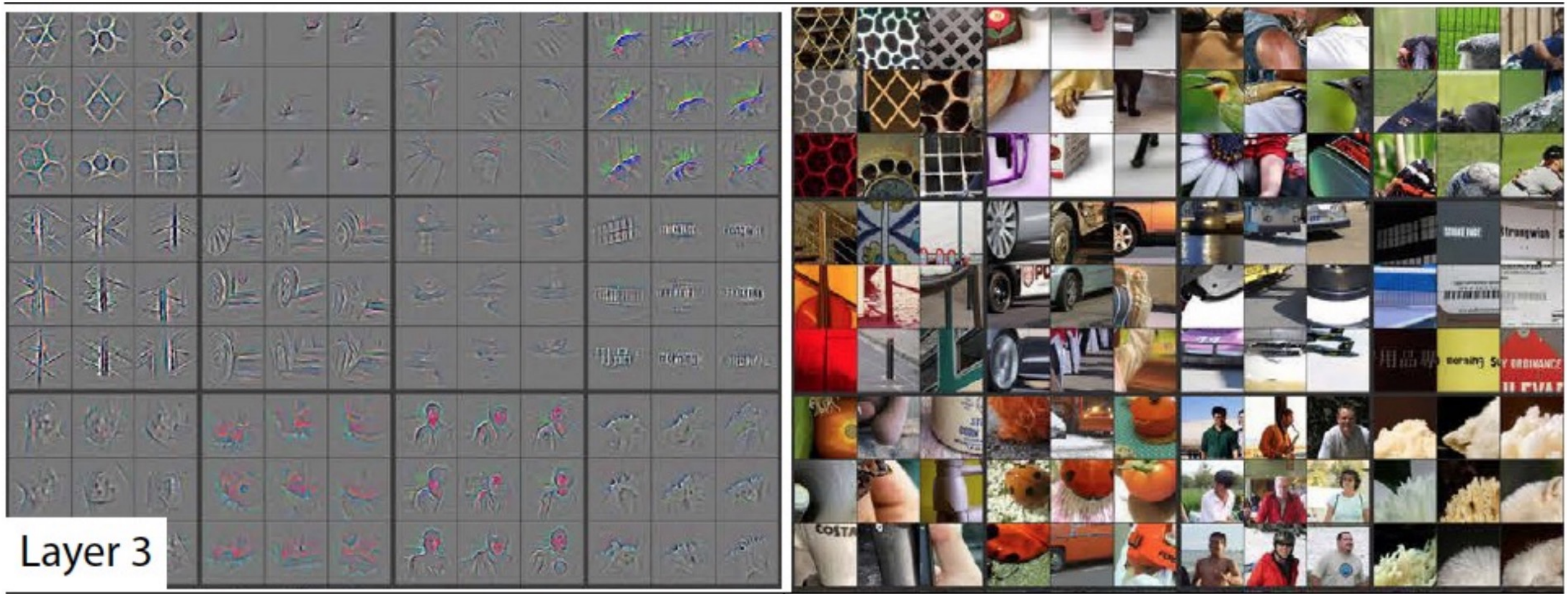
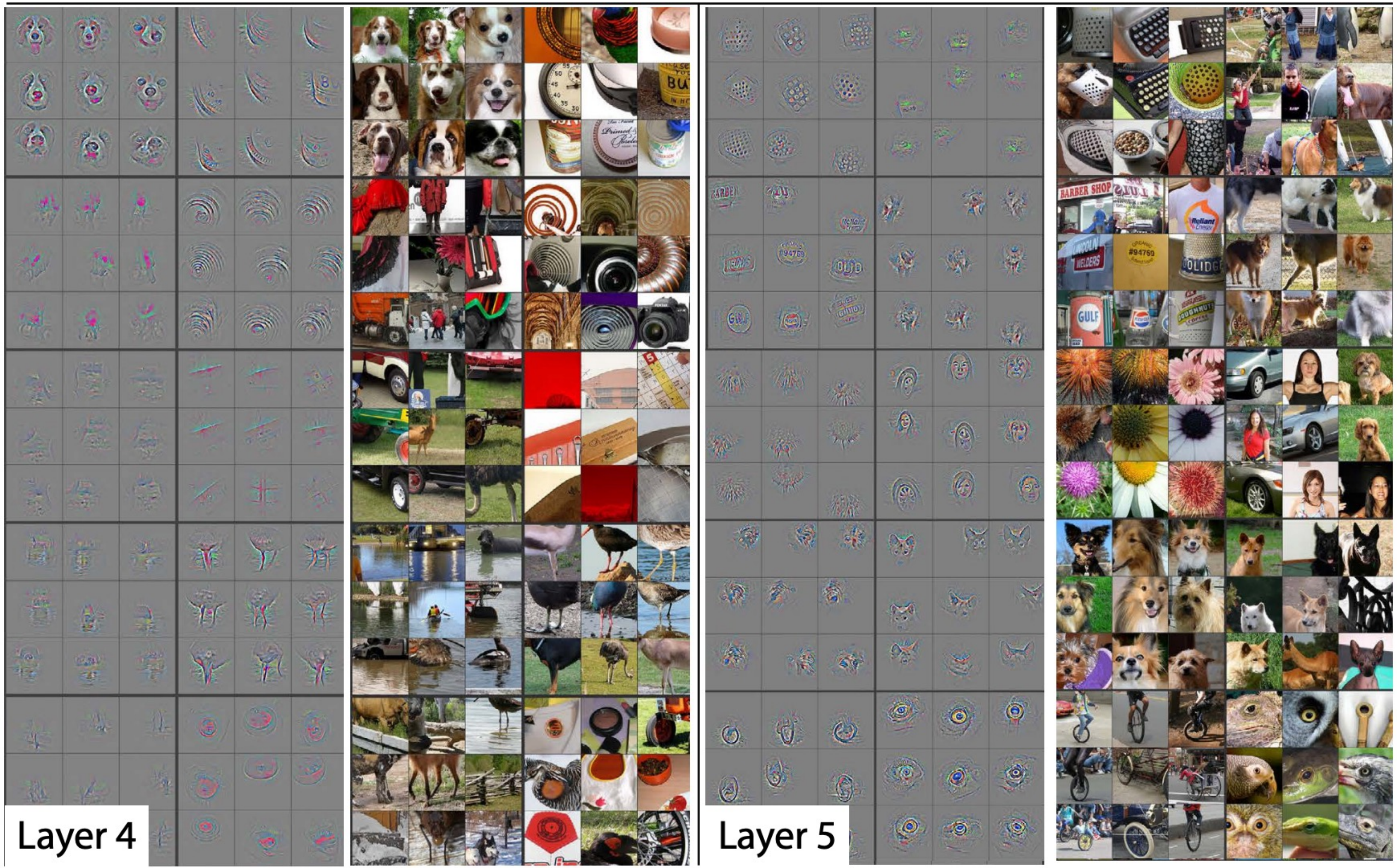


Image from [Zeiler et. al, 2014]

Visualizing Filters

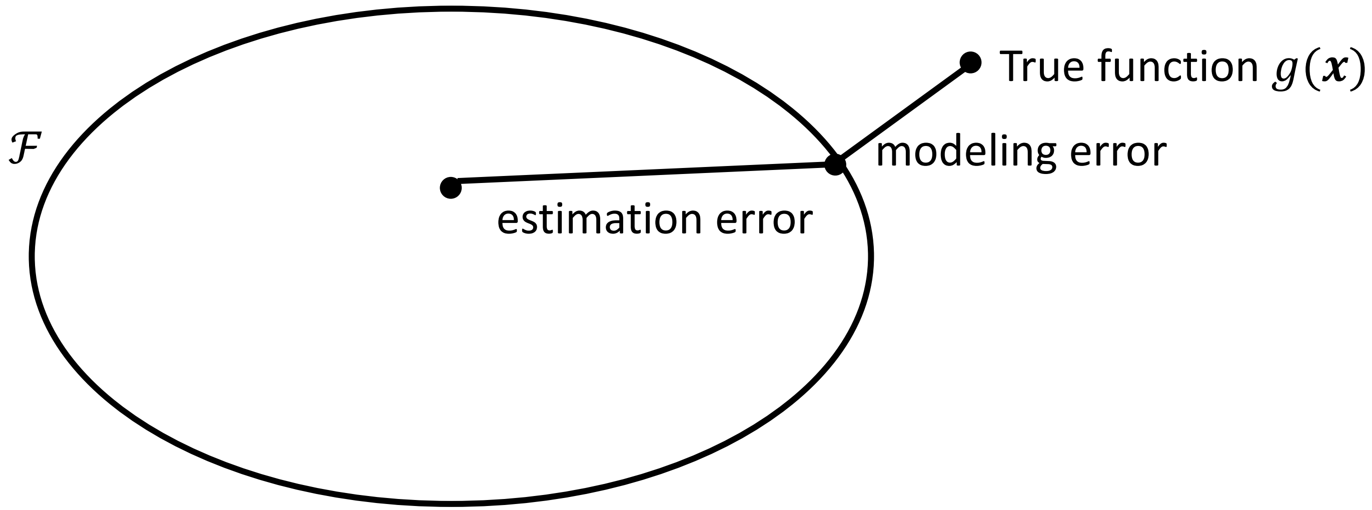


Visualizing Filters



Implications

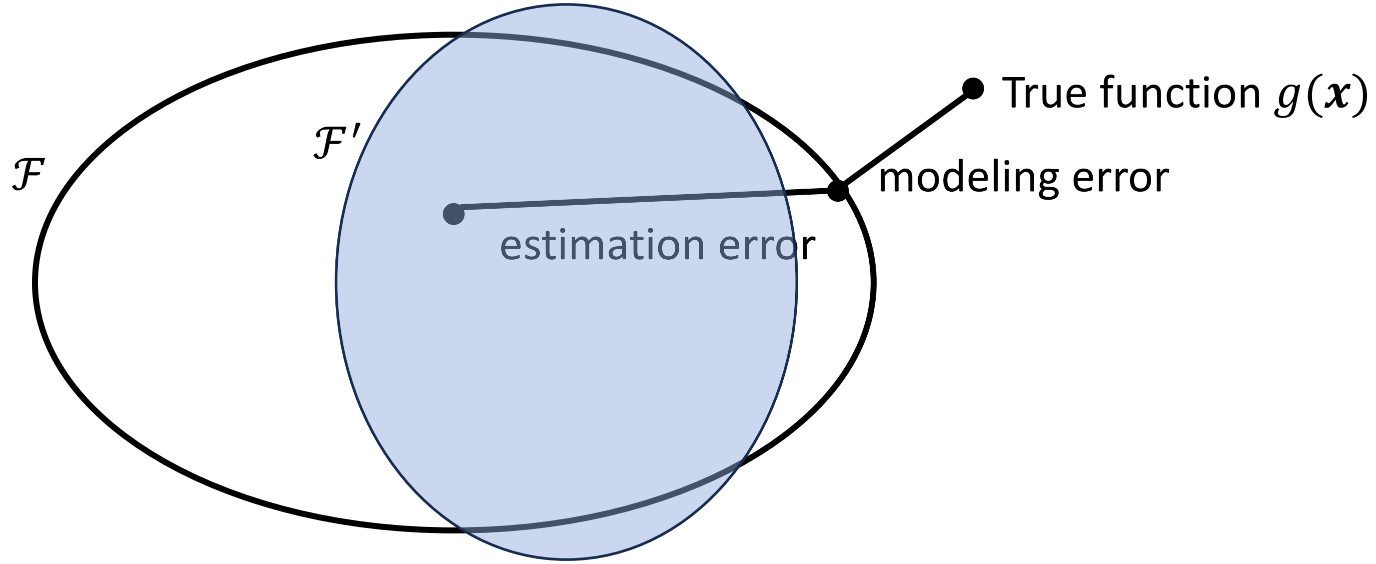
- The filters in lower layers are very “generic”
- They may be reused for another image set
- Recall estimation error (variance), due limited training data
- “reuse” a trained model for another task?

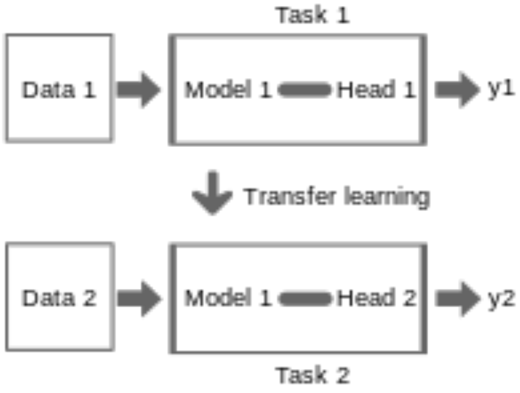


Transfer Learning

"knowledge learned from a task is re-used in order to boost performance on a related task"

Why it may work? Constrain to a smaller $\mathcal{F}' \cap \mathcal{F}$

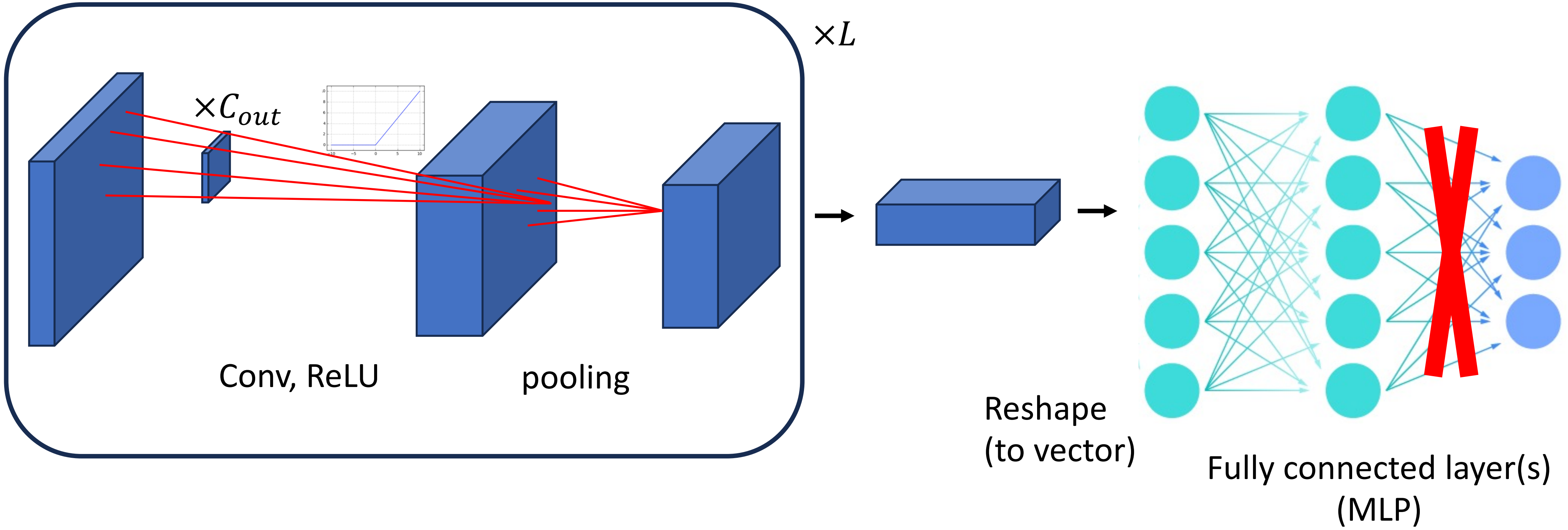




Quote and Illustration from [wiki]

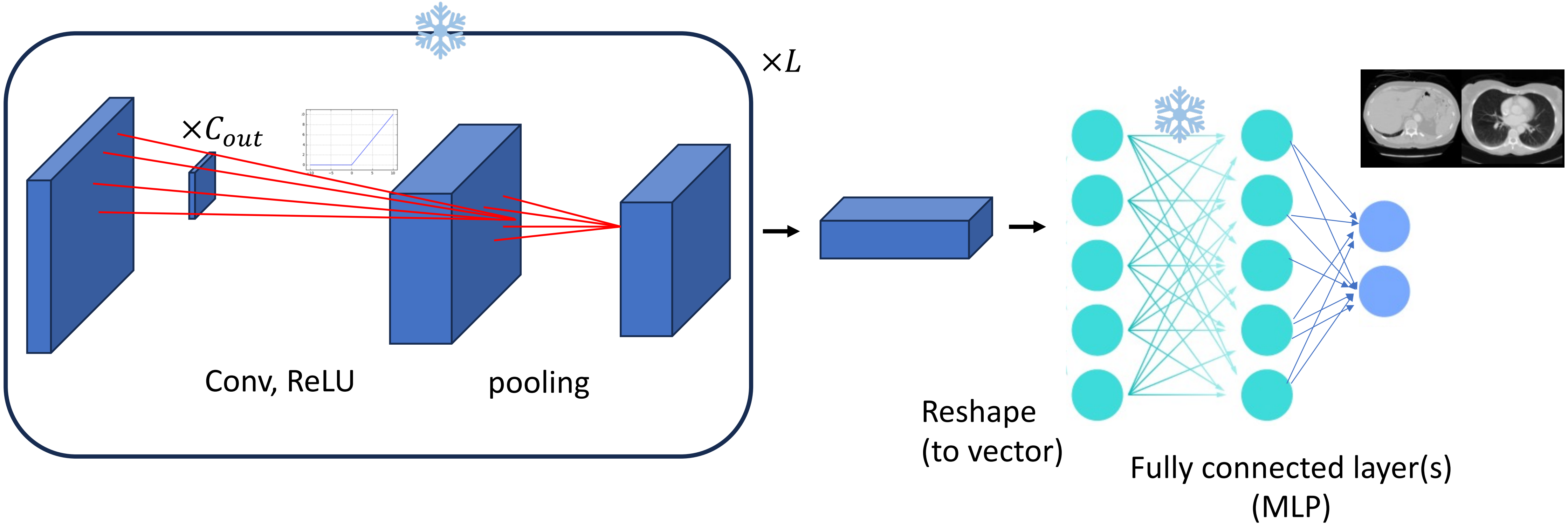
Deep Transfer Learning Recipe

- Step 1: train a model on a big dataset, e.g., imagenet classification
- Step 2: cut the upper layers (at least softmax classifier)



Deep Transfer Learning Recipe

- Step3: Swap in your own upper layer, and train on your own data
  - Option1: freeze some lower layers (e.g., conv layers)
  - Option2: freeze all lower layers

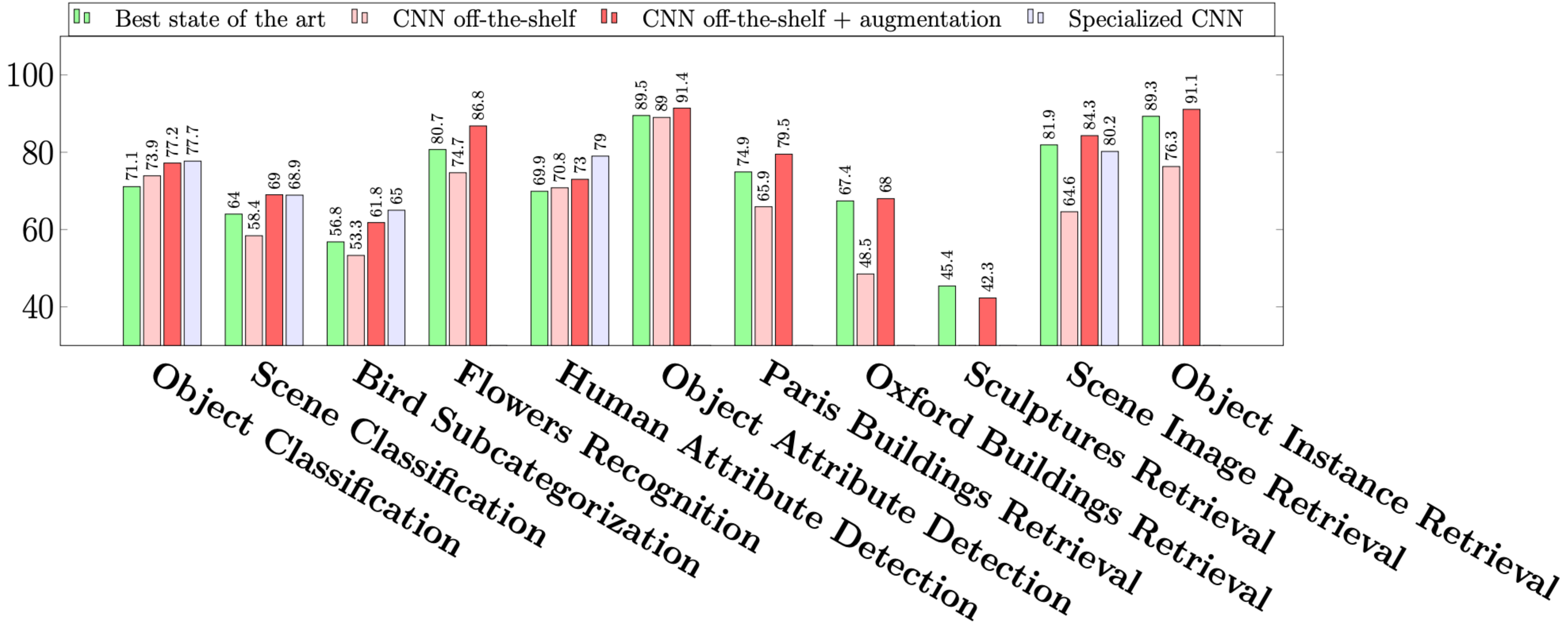


How it works

arXiv
https://arxiv.org › cs

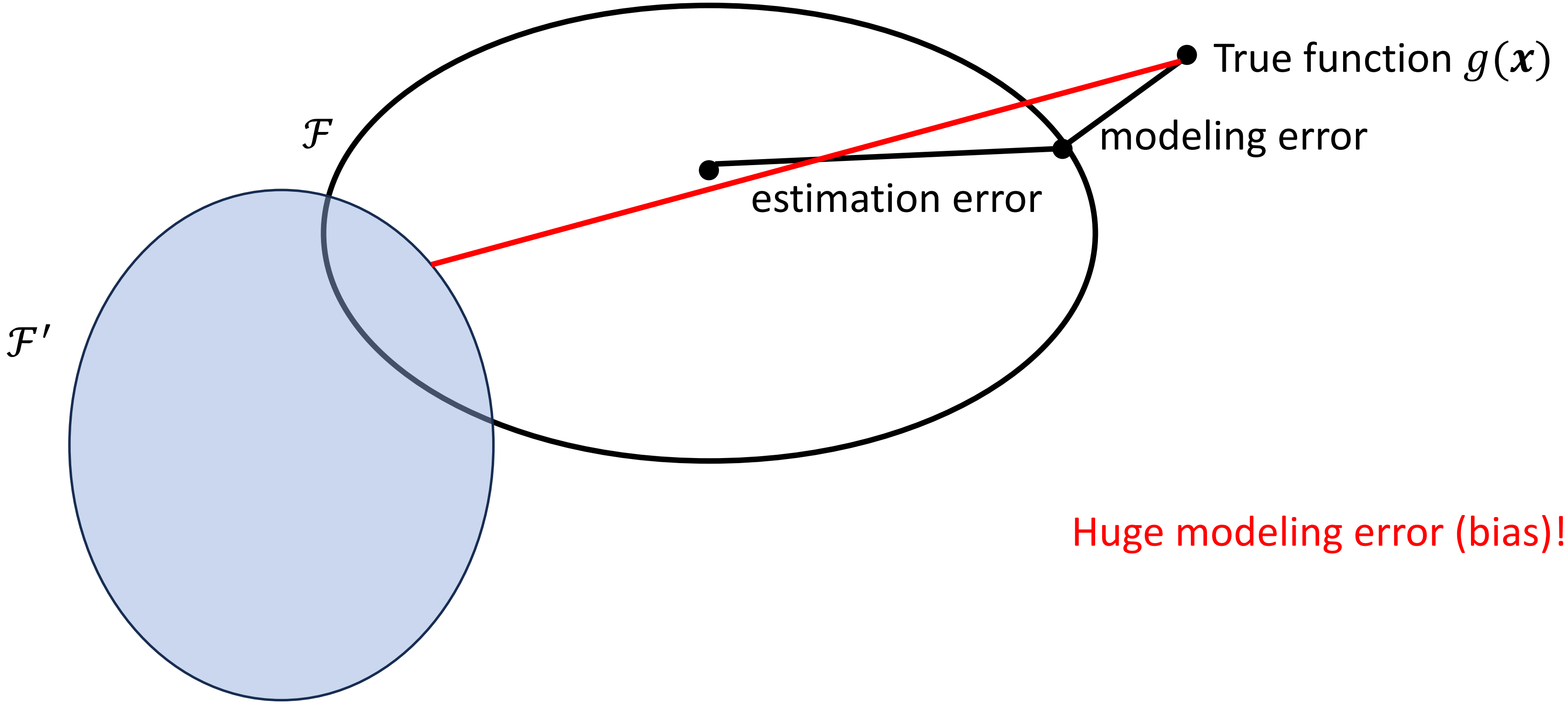
CNN Features off-the-shelf: an Astounding Baseline for ...

by AS Razavian · 2014 · Cited by 5976 — Abstract:Recent results indicate that the generic descriptors extracted from the convolutional neural networks are very powerful.



How it breaks

"when the source and target domains are unrelated, the transfer may fail forcefully"

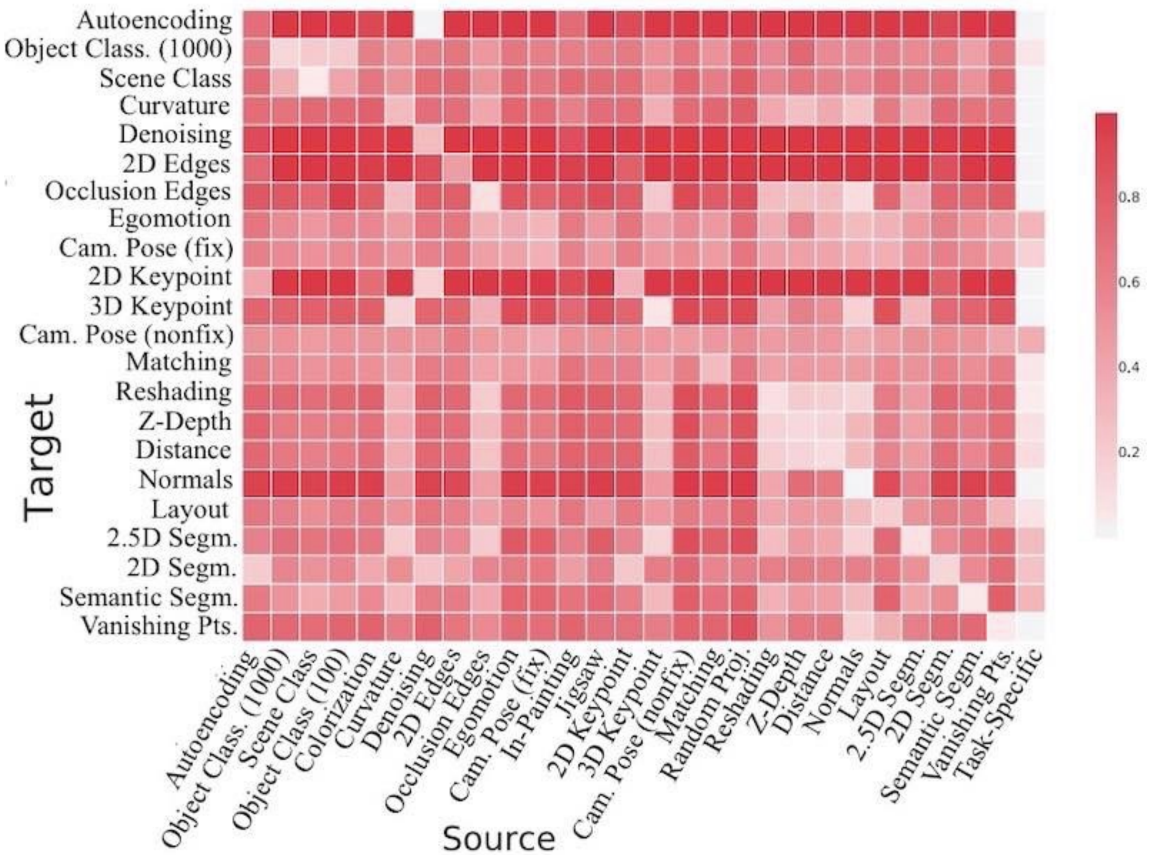


Huge modeling error (bias)!

Quote from [Hosna et. al (2022)]

On Task Relevance

- [Taskonomy](#) by Stanford



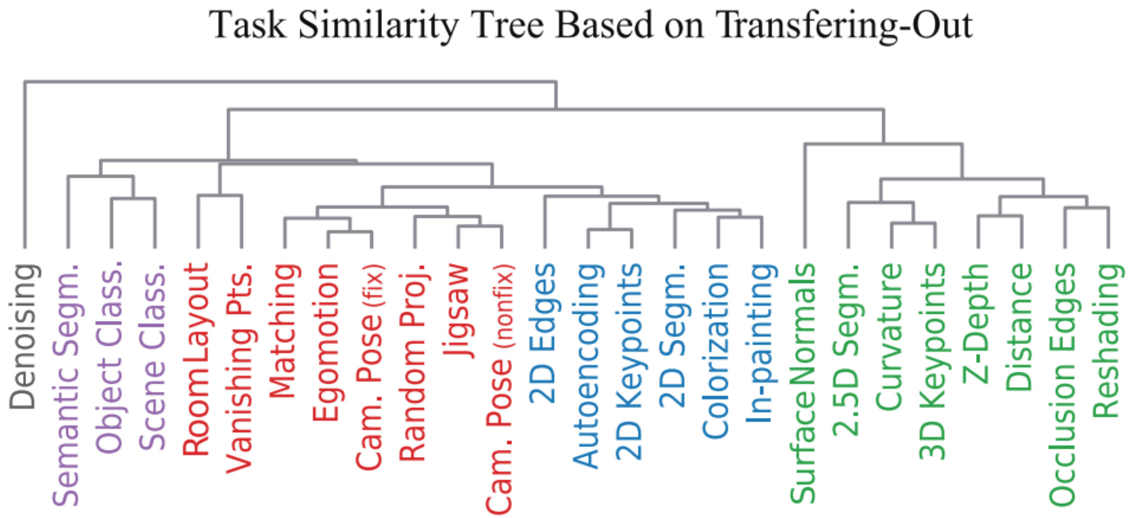
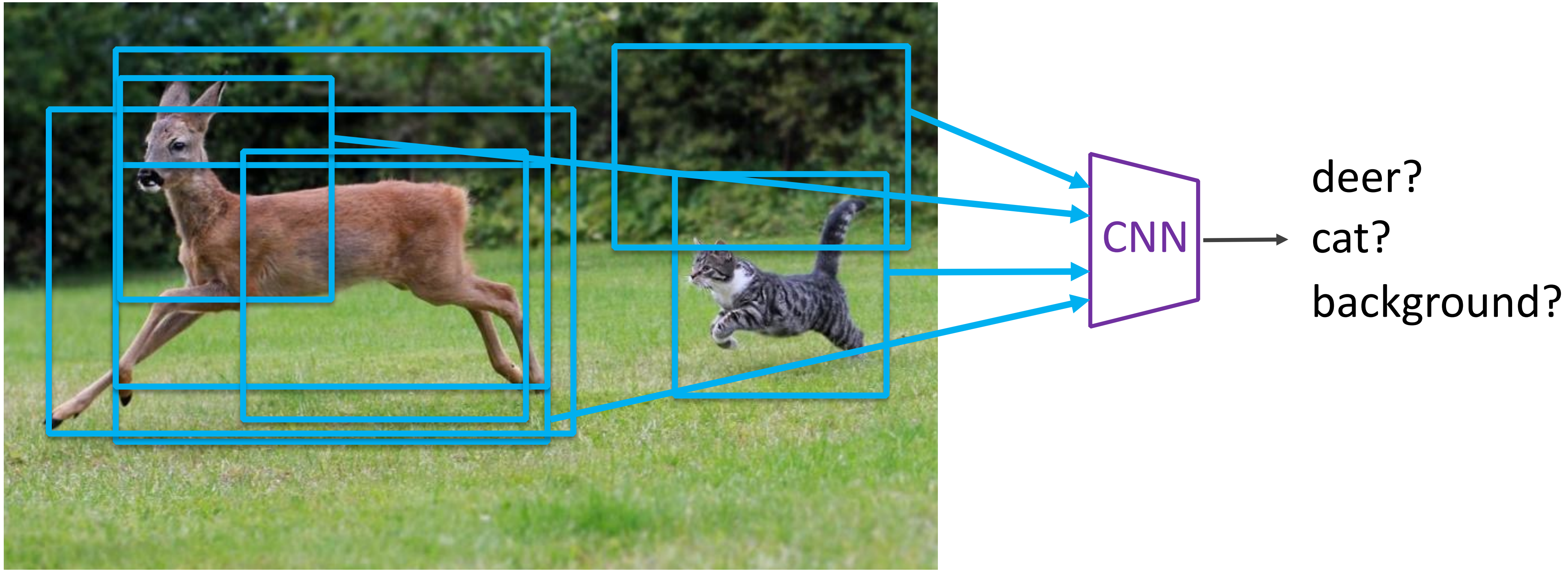


Figure 13: **Task Similarity Tree.** Agglomerative clustering of tasks based on their transferring-out patterns (i.e. using columns of normalized affinity matrix as task features). 3D, 2D, low dimensional geometric, and semantic tasks clustered together using a fully computational approach.

Agenda

- Introduction: Vision Tasks
- Building Blocks
- Convolutional Networks
- Beyond Image Classification

Object Detection based on Proposed Regions



Slide credit: Rice University CS6501

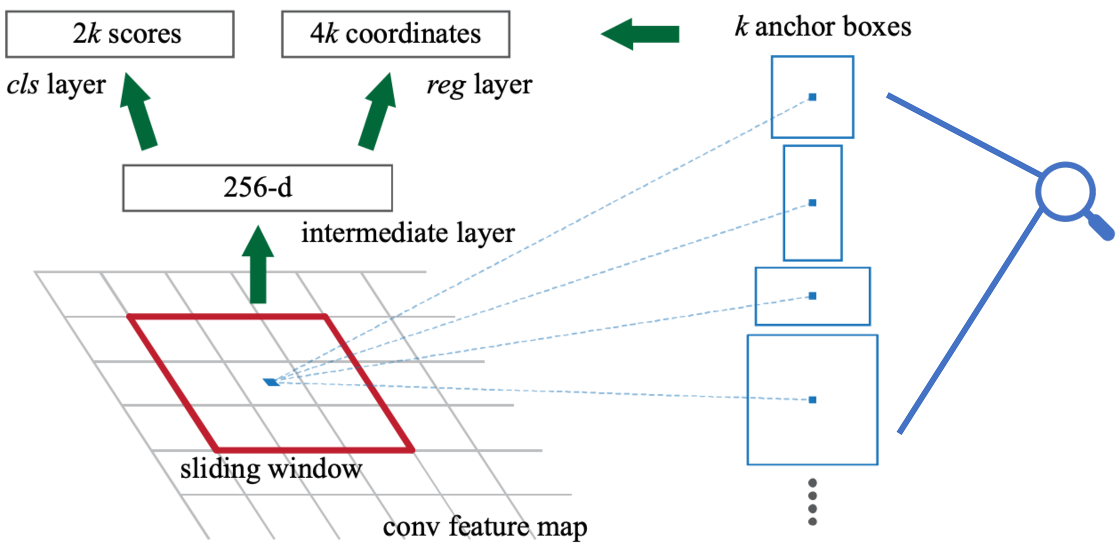
Faster R-CNN

Neural Information Processing Systems
https://papers.nips.cc › paper › 5638-faster-r-cnn-towa...

Faster R-CNN: Towards Real-Time Object Detection with ...

by S Ren · 2015 · Cited by 69884 — An RPN is a fully-convolutional network that simultaneously predicts object bounds and objectness scores at each position. RPNs are trained end-to-end…

- How to propose the regions?



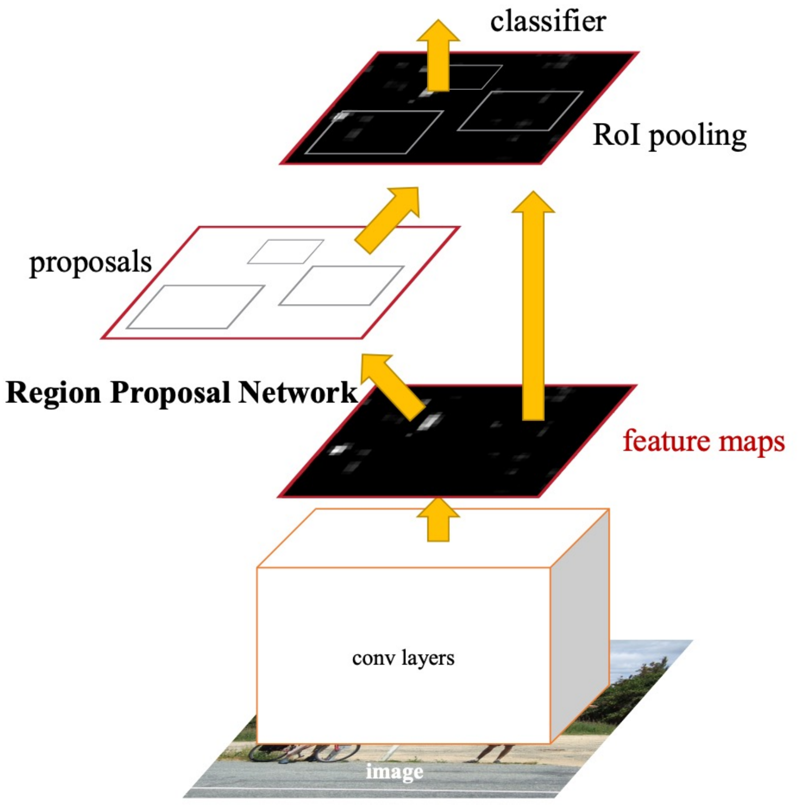


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

- Translation equivalence!

Faster R-CNN

- Train the regional proposal network
- Binary Classifier: object in the region or not
- Regression: compare region vs groundtruth

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*).$$

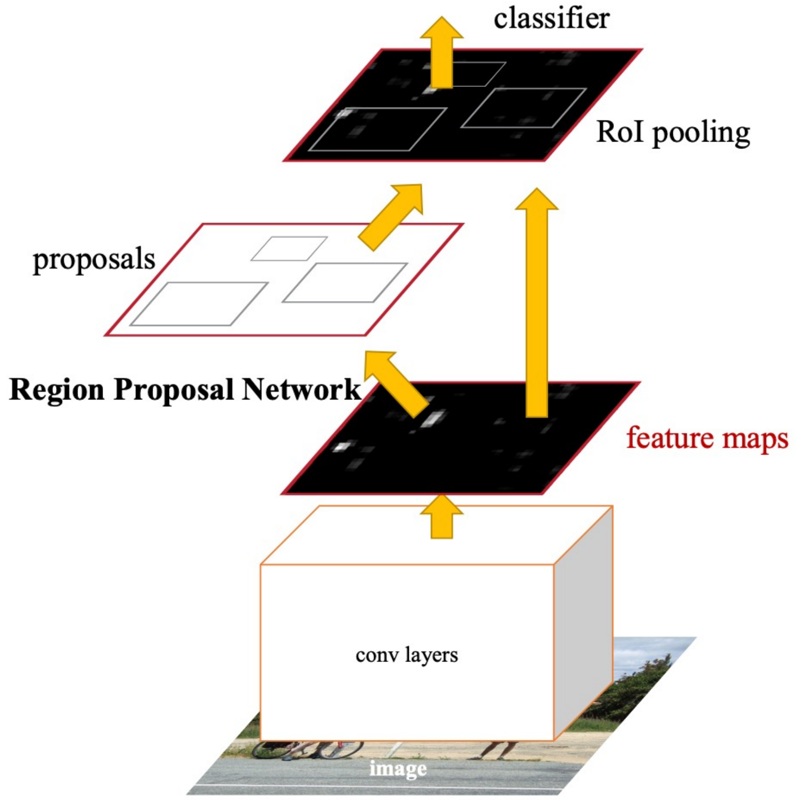
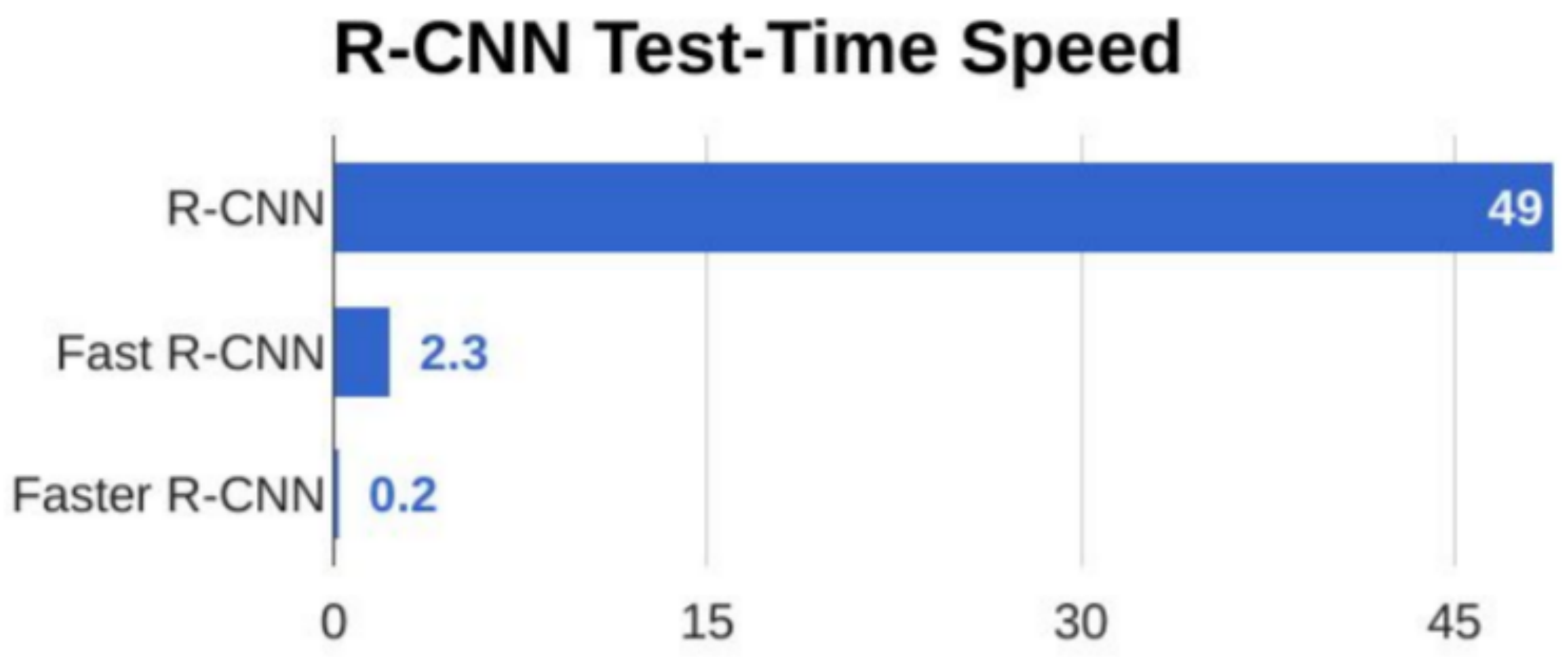


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

How fast?


R-CNN Test-Time Speed
R-CNN
Fast R-CNN
Faster R-CNN
49
2.3
0.2
0
15
30
45

Even faster

arXiv
https://arxiv.org › cs

You Only Look Once: Unified, Real-Time Object Detection

by J Redmon · 2015 · Cited by 43213 — Abstract:We present **YOLO**, a new approach to **object detection**. Prior work on **object detection** repurposes classifiers to perform detection.

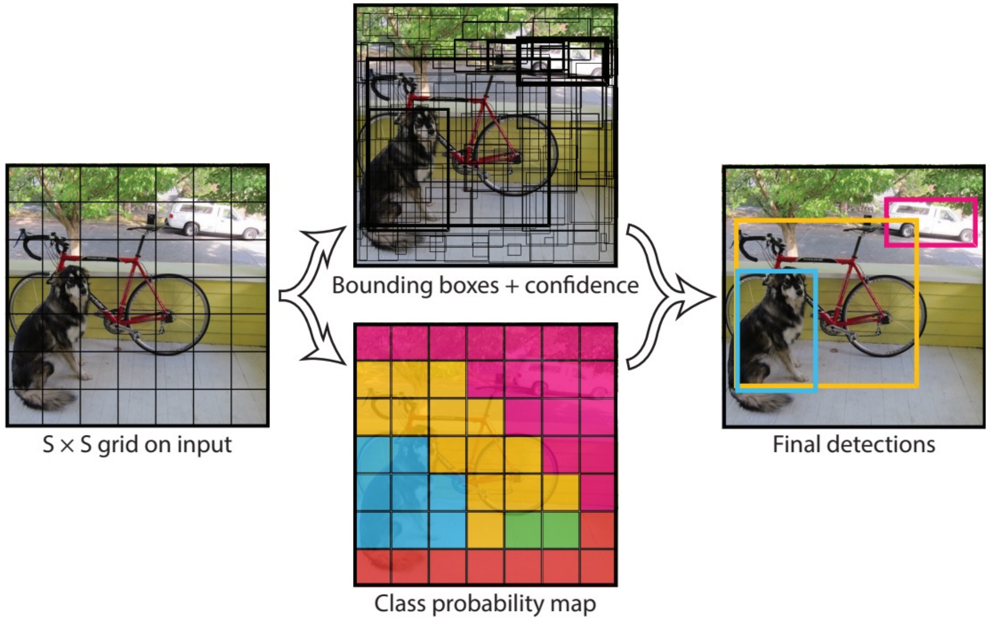
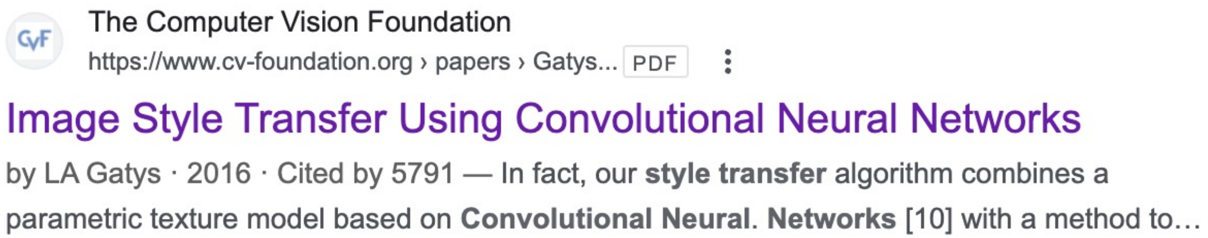
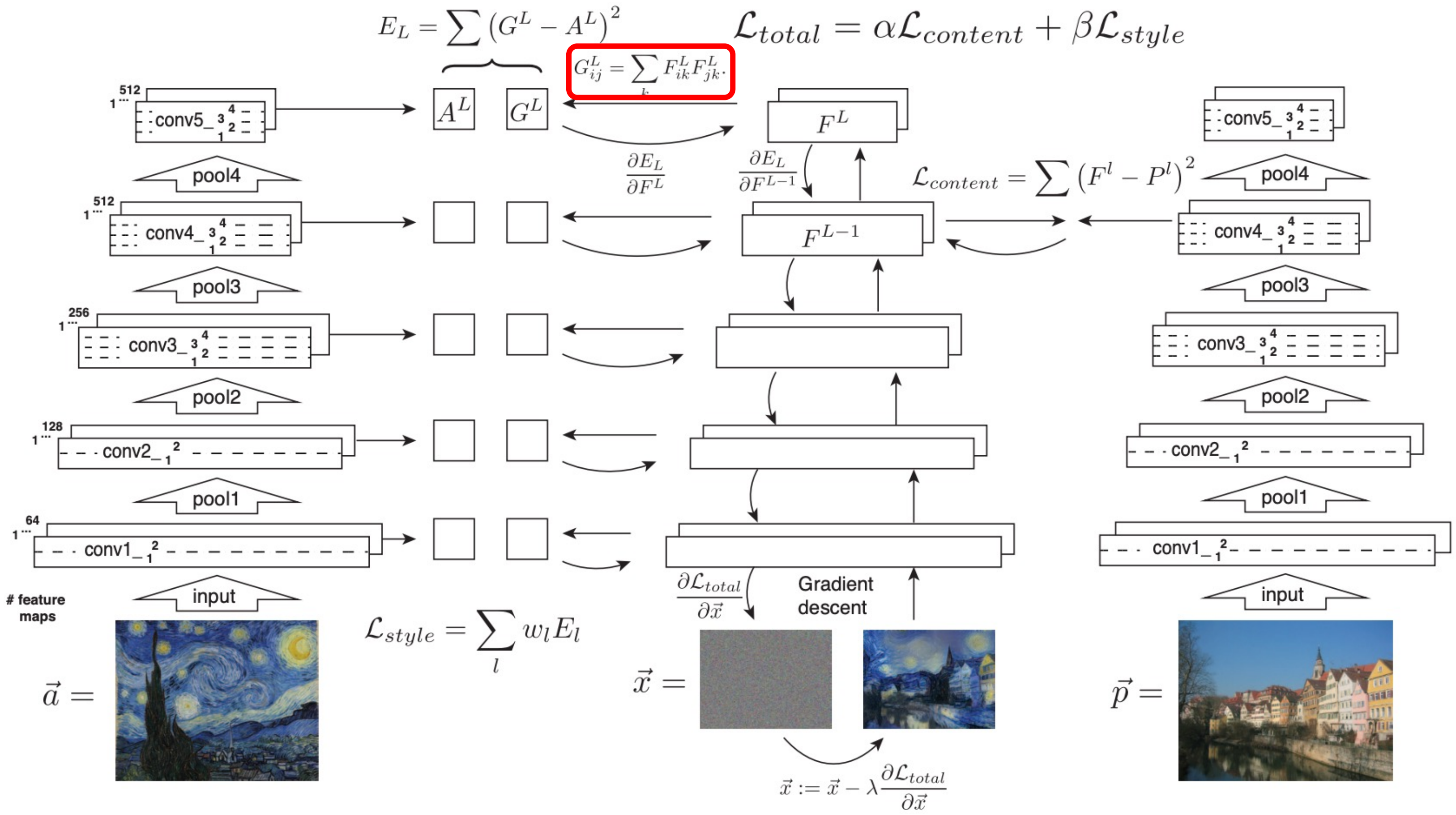


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

- $S = 7$ and $B = 2$ in the paper
- Bounding box is center position (x, y), size (h, w), and confidence
- C the number of object classes
- Faster than faster R-CNN
- But may fail for tiny object

Style Transfer





Discussion

- Why gram matrix captures style?
- What layers to transfer over?
- Why not initialize from content image?
- Multiple style?