Deep Learning

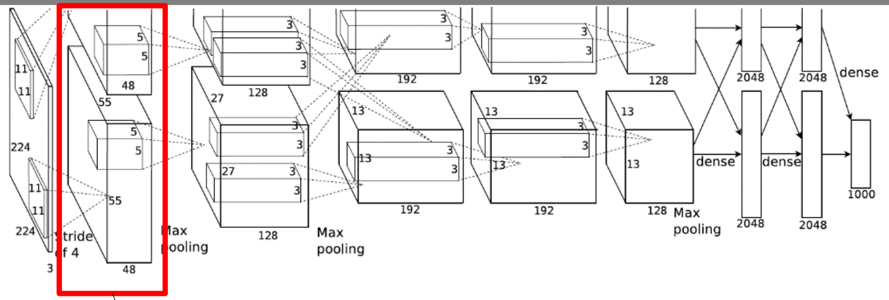
07- Deep Convolutional Neural Networks and Beyond

Marco Piastra

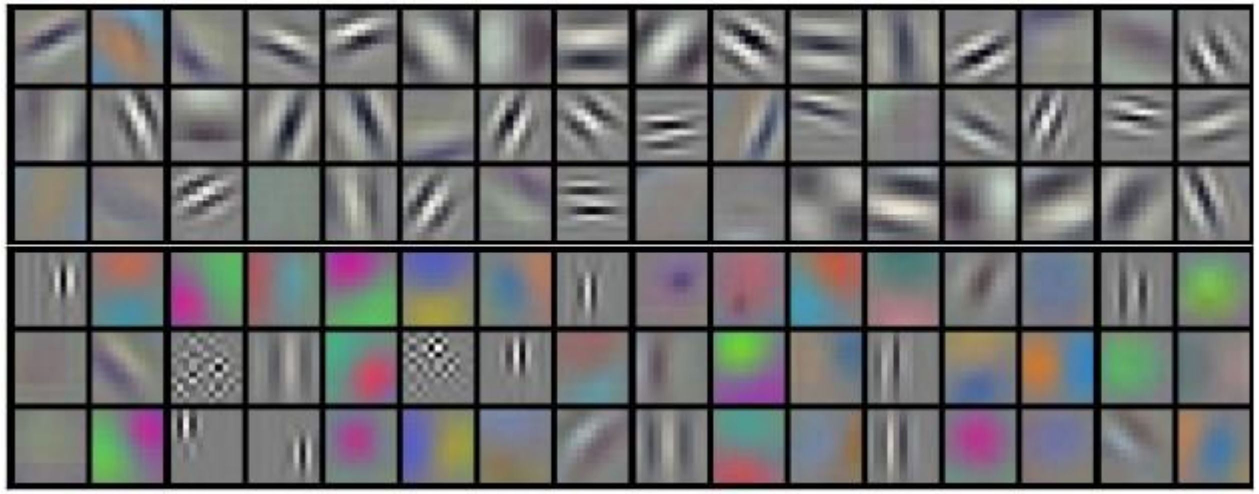
This presentation can be downloaded at:
http://vision.unipv.it/DL

Inside AlexNet (after training)

AlexNet Filters (after training)



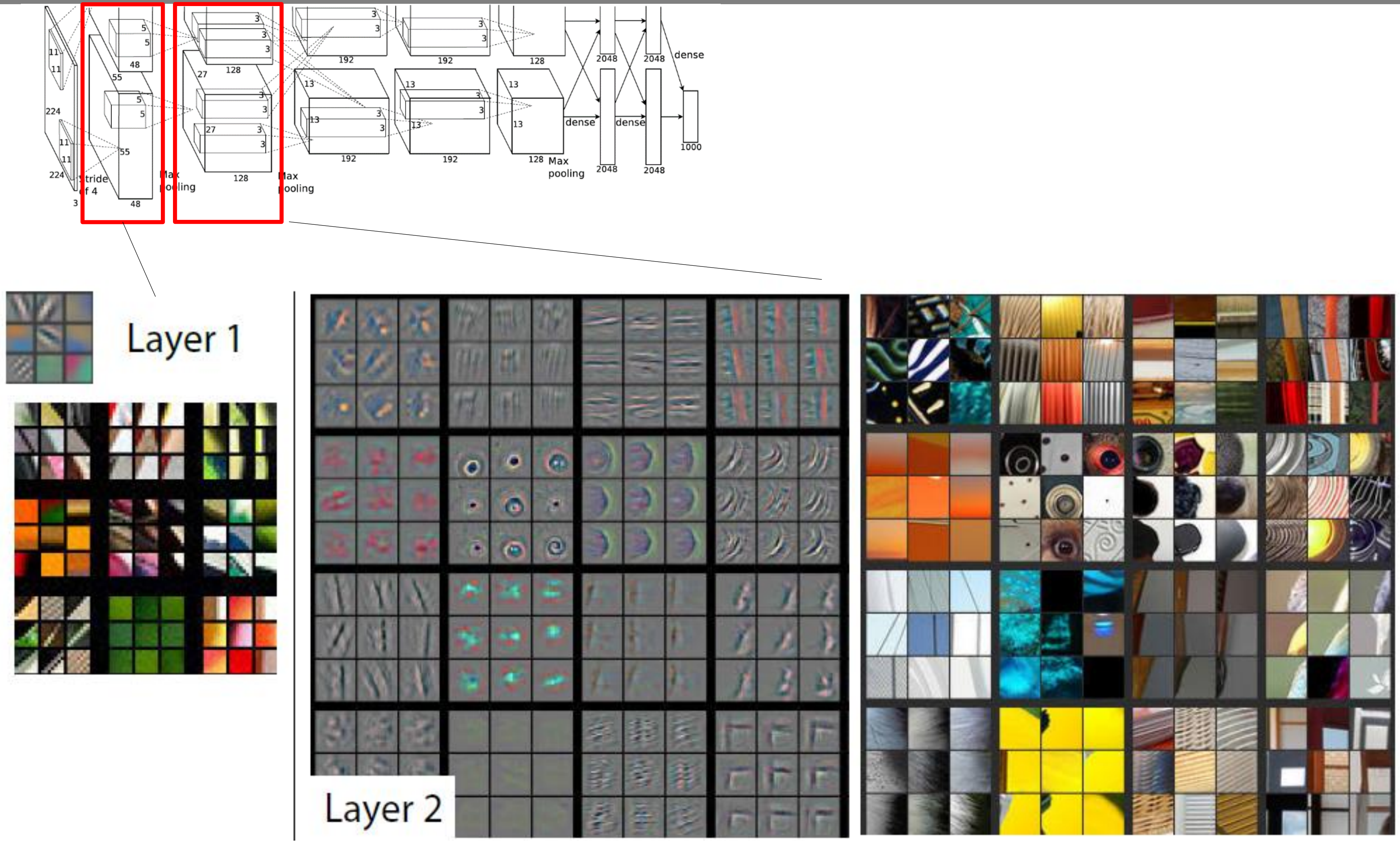
Layer 1



These are 96 real examples of convolutive filters for RGB images

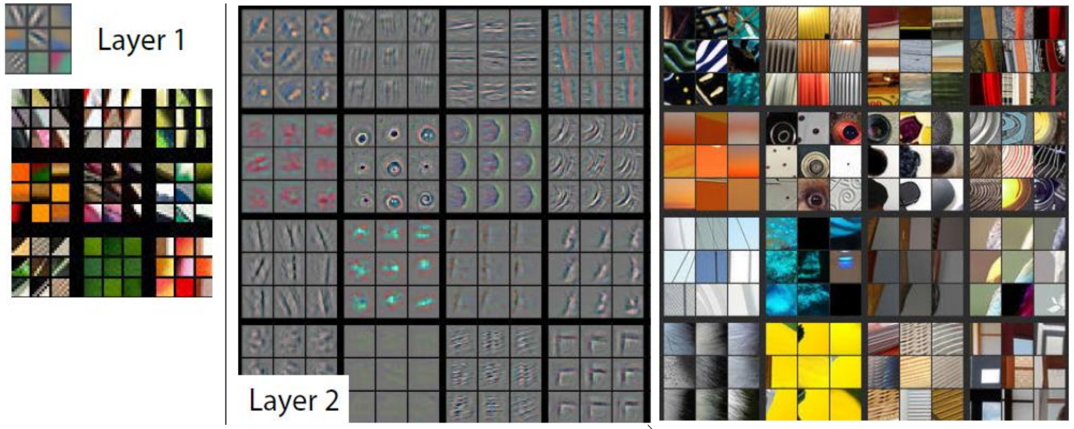
[image from http://cs231n.github.io/convolutional-networks/]

AlexNet Filters- DeconvNet



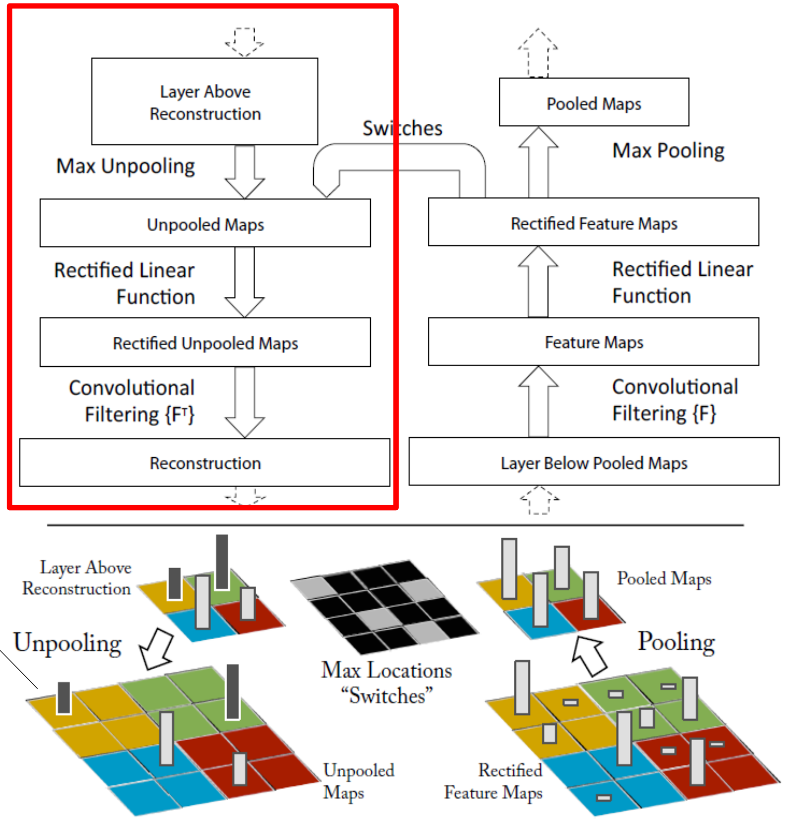
[images from https://arxiv.org/pdf/1311.2901.pdf]

AlexNet Filters- DeconvNet



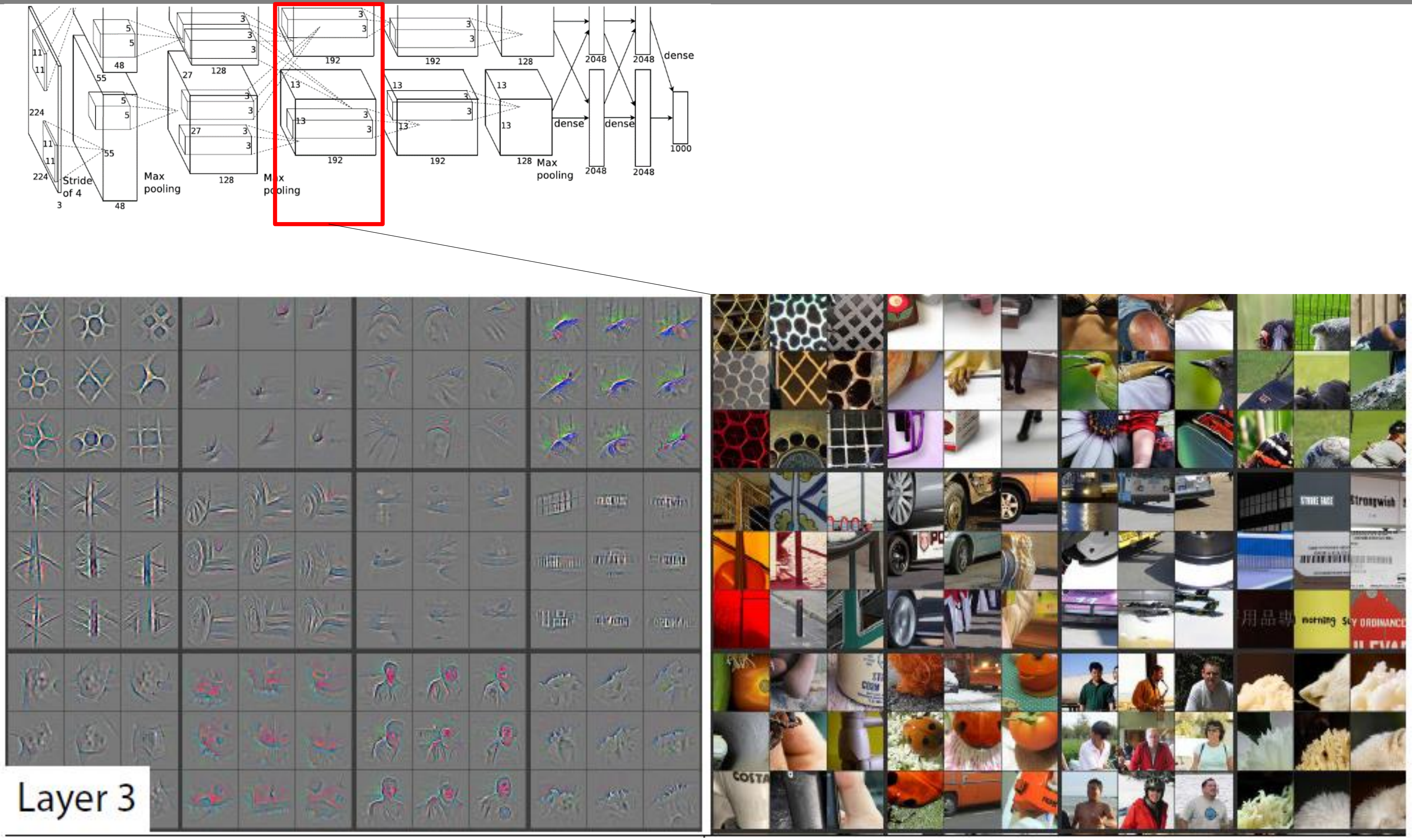
DeconvNet:
using a DCNN in reverse

DeconvNet



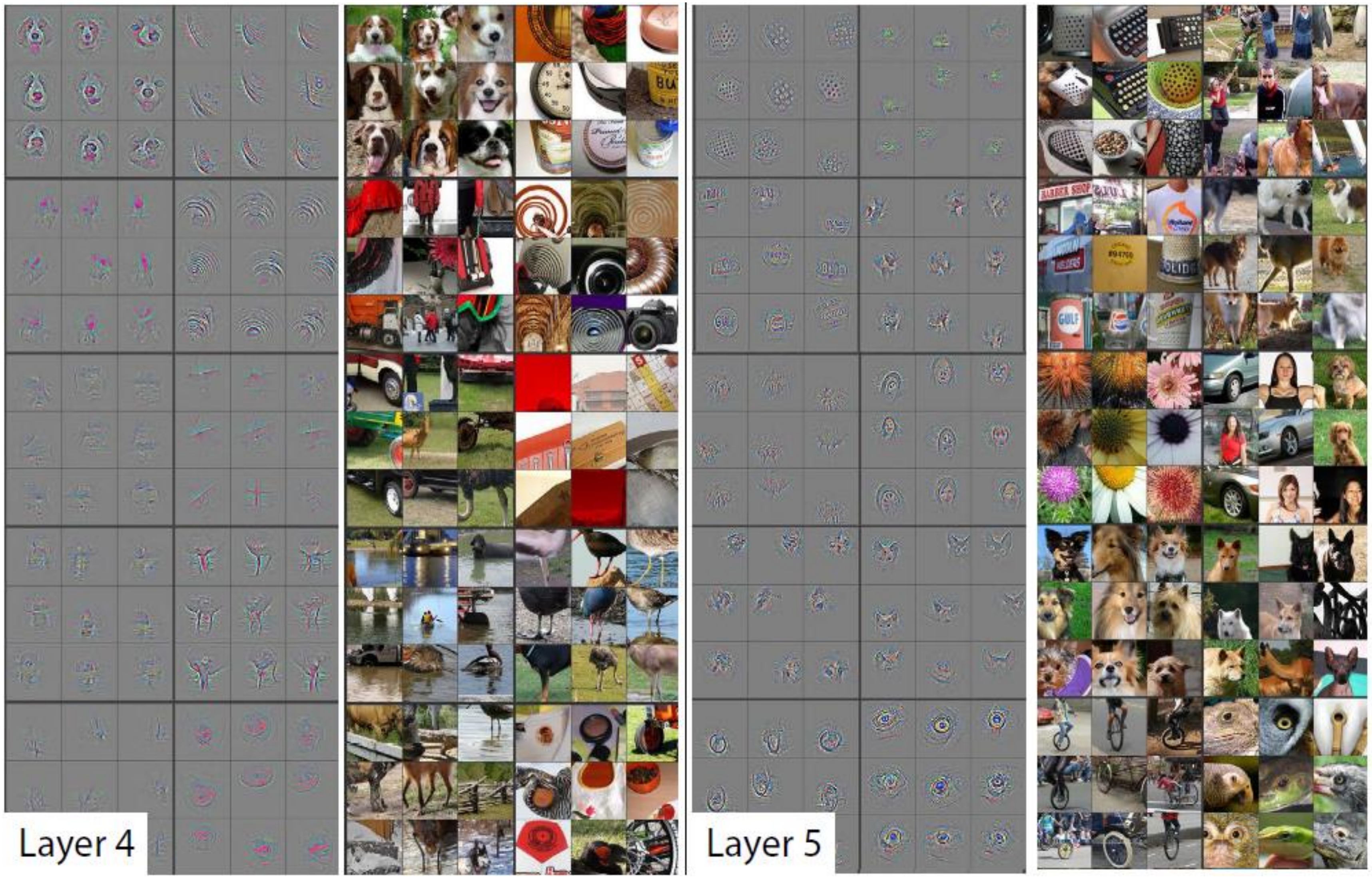
[images from https://arxiv.org/pdf/1311.2901.pdf]

AlexNet Filters- DeconvNet



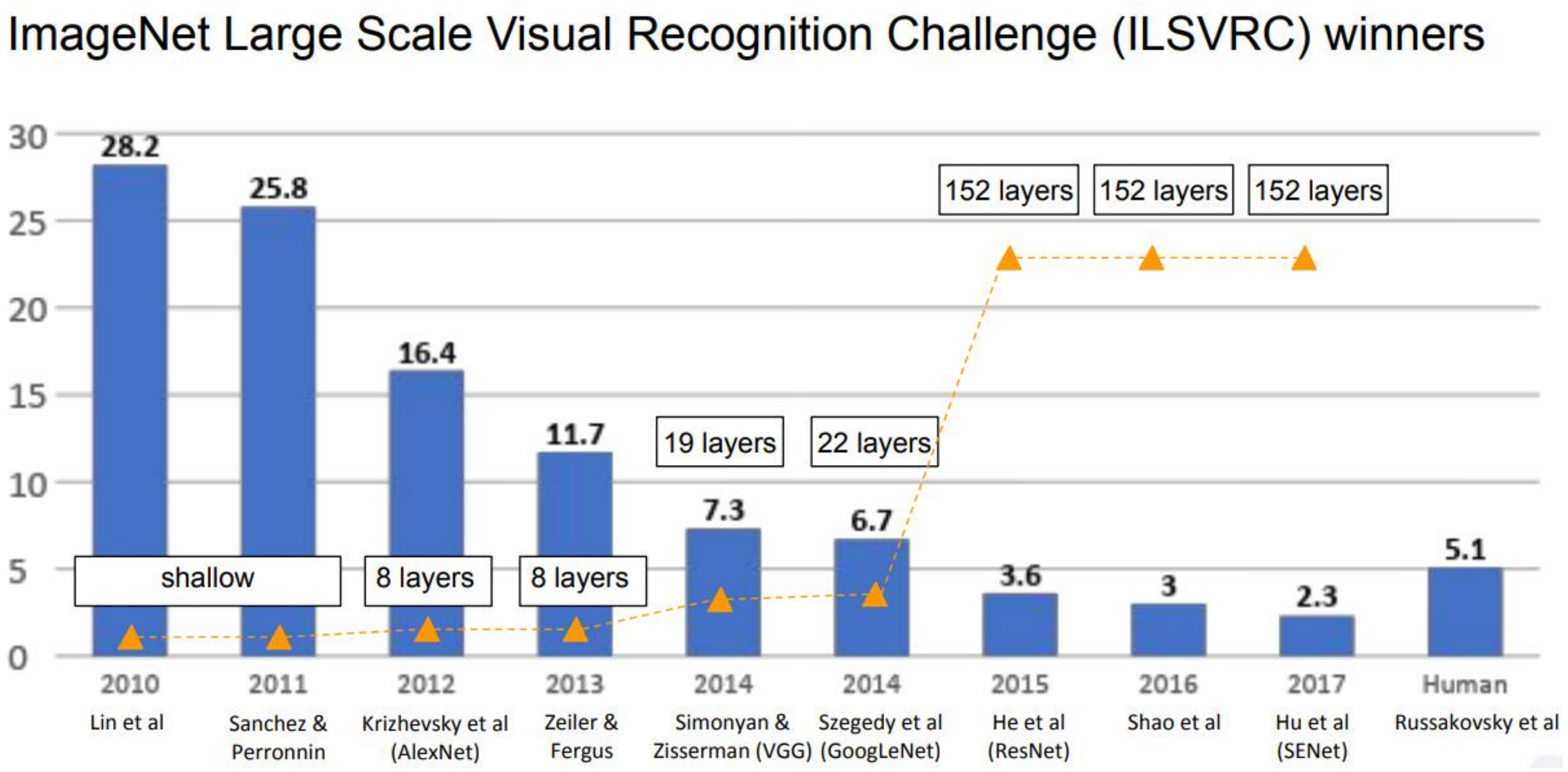
[images from https://arxiv.org/pdf/1311.2901.pdf]

AlexNet Filters- DeconvNet



[images from https://arxiv.org/pdf/1311.2901.pdf]

Beyond AlexNet: The DCNN storm



The challenge is now over

Image from
[http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture09.pdf]

VGG Architecture

Several variants

Only 3x3 convolutional
filters used
(each with ReLU)

LRN used in only one variant

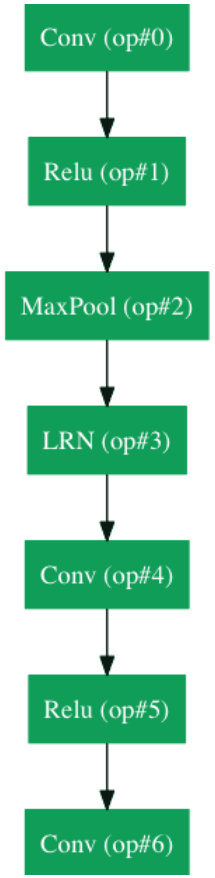
Image from
[https://arxiv.org/pdf/1409.1556.pdf]

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					

Inception Acrhitecture

- The ImageNet Large Scale Visual Recognition Challenge

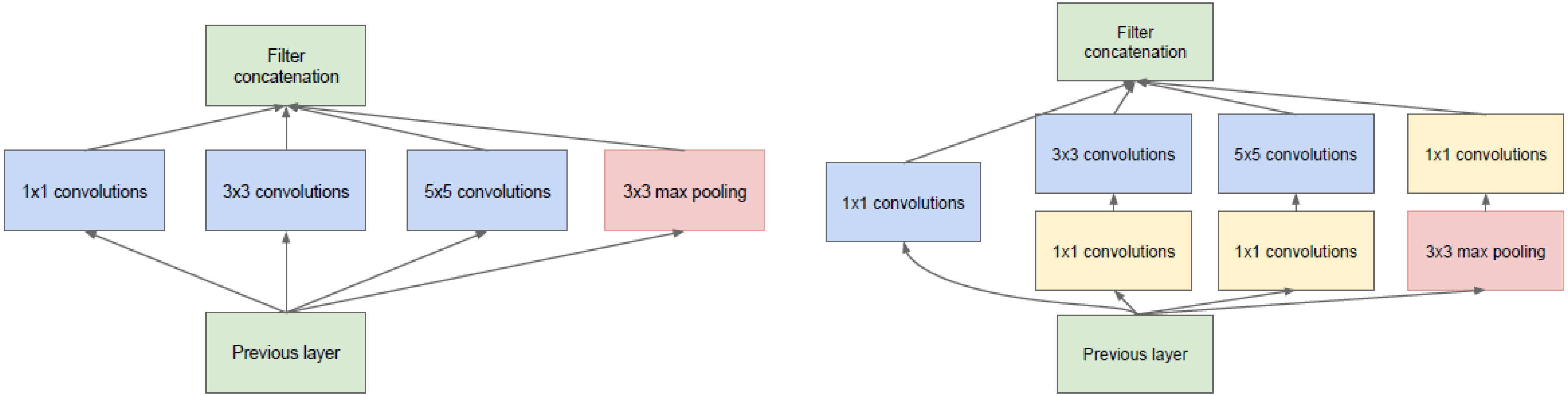
How deep is a deep neural network, for a task like this?



GoogLeNet (Inception v4) winner of two out of three categories in 2014: 154 network layers

Inception Architecture

- Inception modules

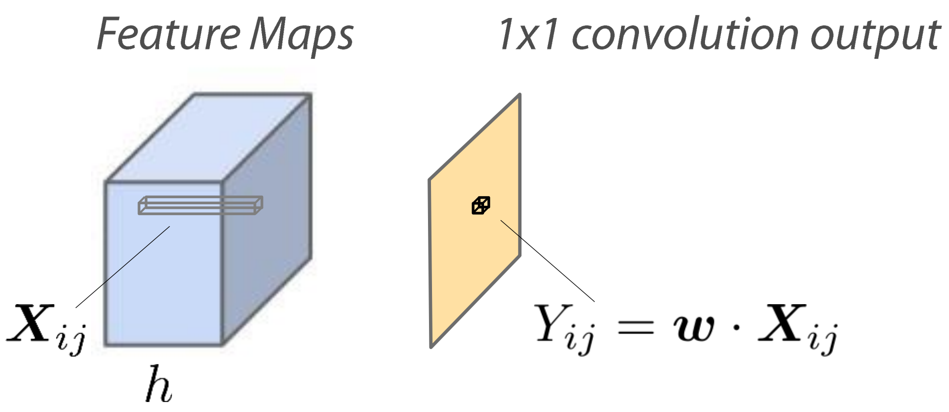


(a) Inception module, naïve version (b) Inception module with dimension reductions

Image from [https://arxiv.org/pdf/1409.4842.pdf]

Inception Architecture

- 1x1 convolution?



(It is a kind of misnomer)

Each filter has dimension $1 \times 1 \times h$

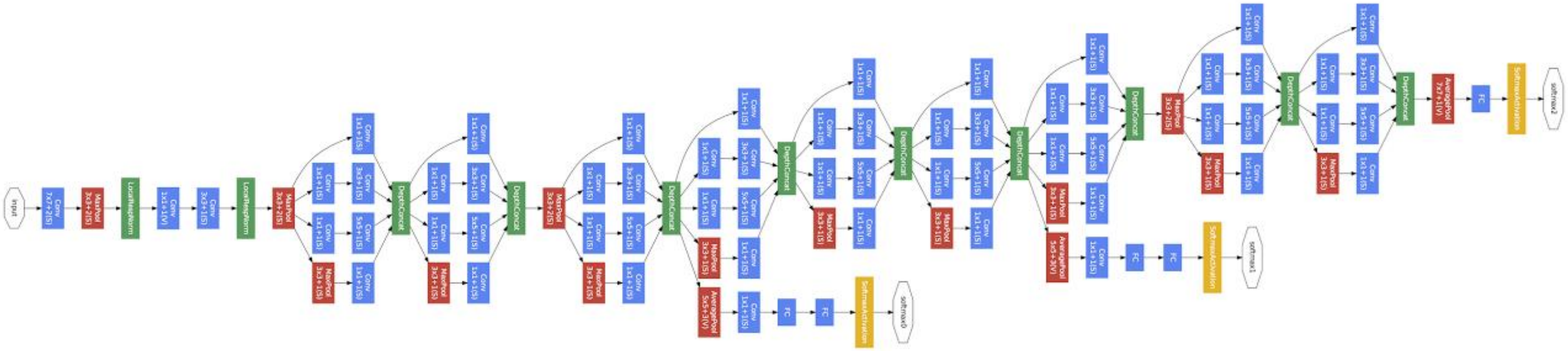
where h is the depth of the set of filter maps

Using d 1x1 convolution filters allows changing depth h into d

Clearly the assumption is $d < h$

It mimics a fully connected layer (across channels)

- GoogLeNet architecture

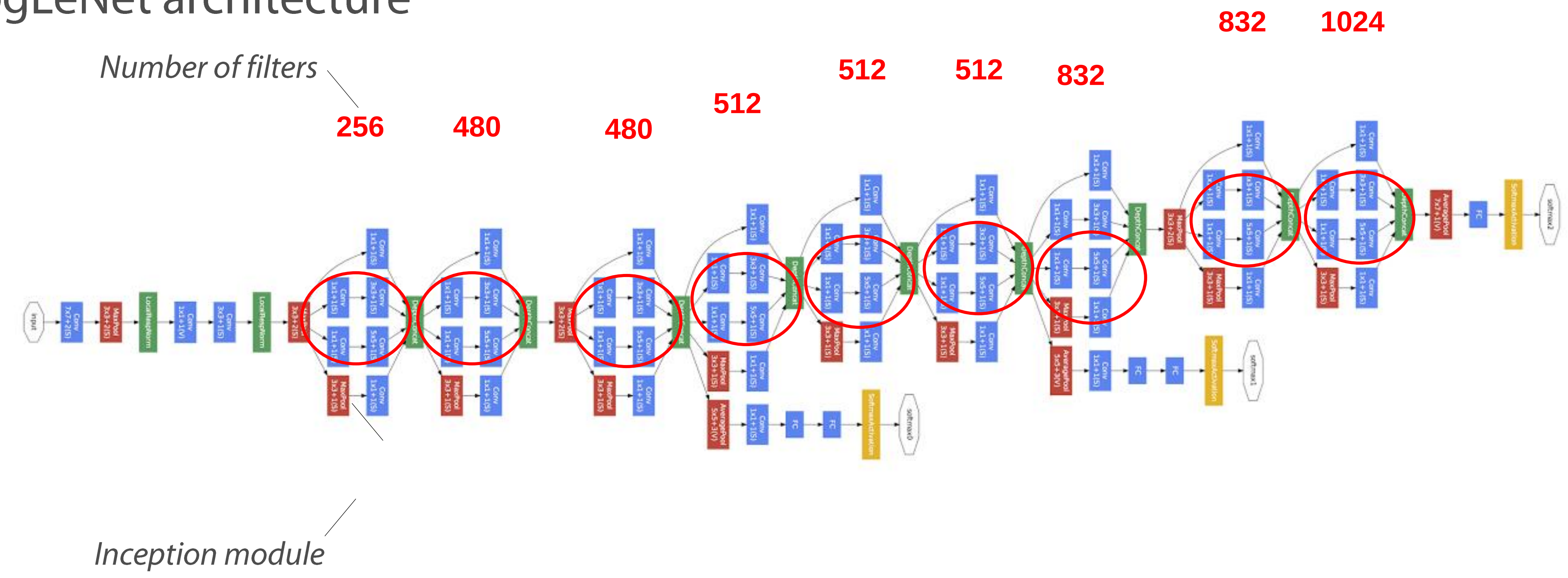


Convolutive
Max Pool
Softmax
Filter Concat

Image from [https://arxiv.org/pdf/1409.4842.pdf]

Inception Architecture

- GoogLeNet architecture

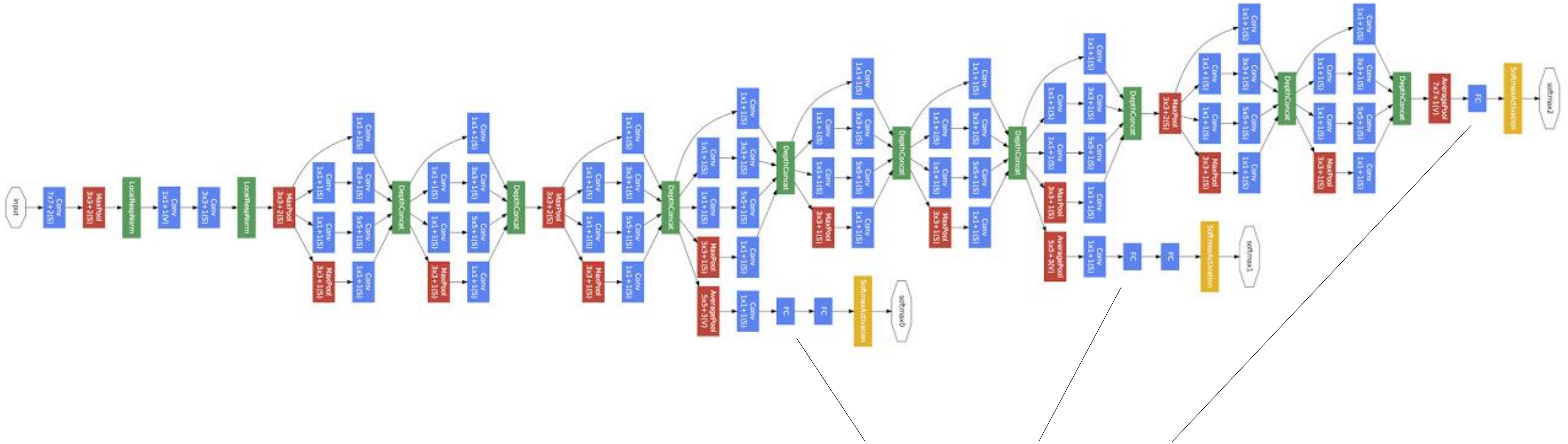


Convolutive
Max Pool
Softmax
Filter Concat

Image from [https://arxiv.org/pdf/1409.4842.pdf]

Inception Architecture

- GoggLeNet architecture



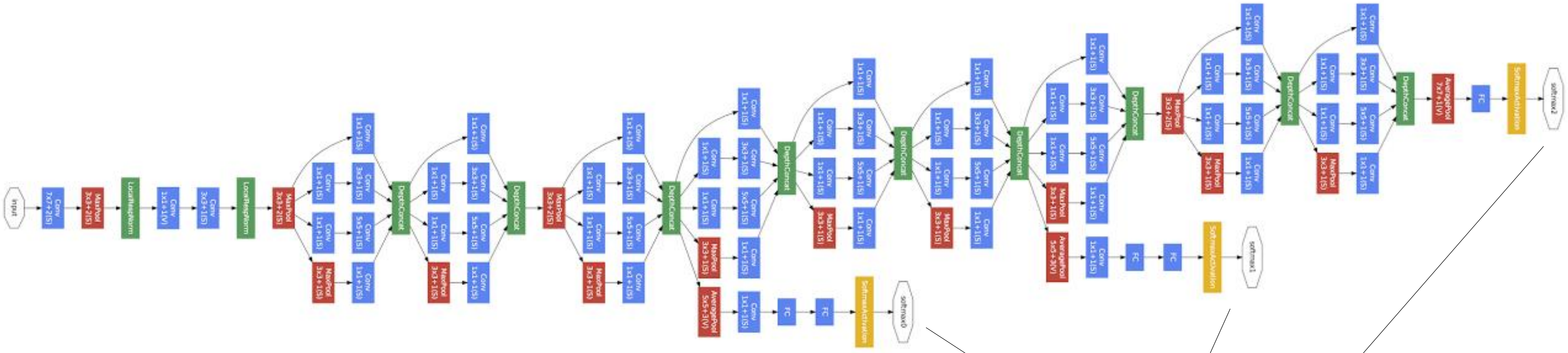
Much smaller FC layers

Convolutive
Max Pool
Softmax
Filter Concat

Image from [https://arxiv.org/pdf/1409.4842.pdf]

Inception Architecture

- GoogLeNet architecture



Three softmax outputs

They are trained to produce
the same output, simultaneously

Convolutive
Max Pool
Softmax
Filter Concat

Image from [https://arxiv.org/pdf/1409.4842.pdf]

- ResNet block

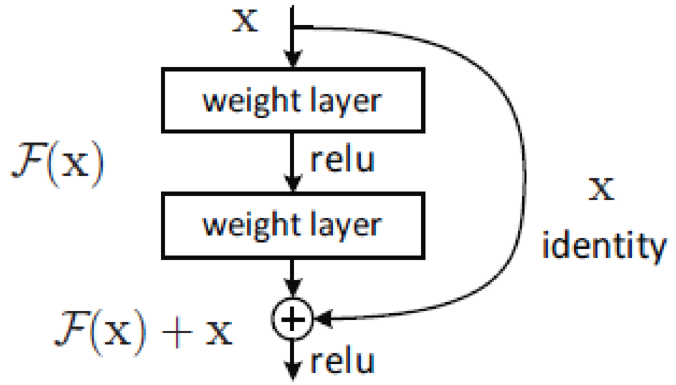


Figure 2. Residual learning: a building block.

Image from [https://arxiv.org/pdf/1512.03385.pdf]

ResNet Architecture

- ResNet architecture

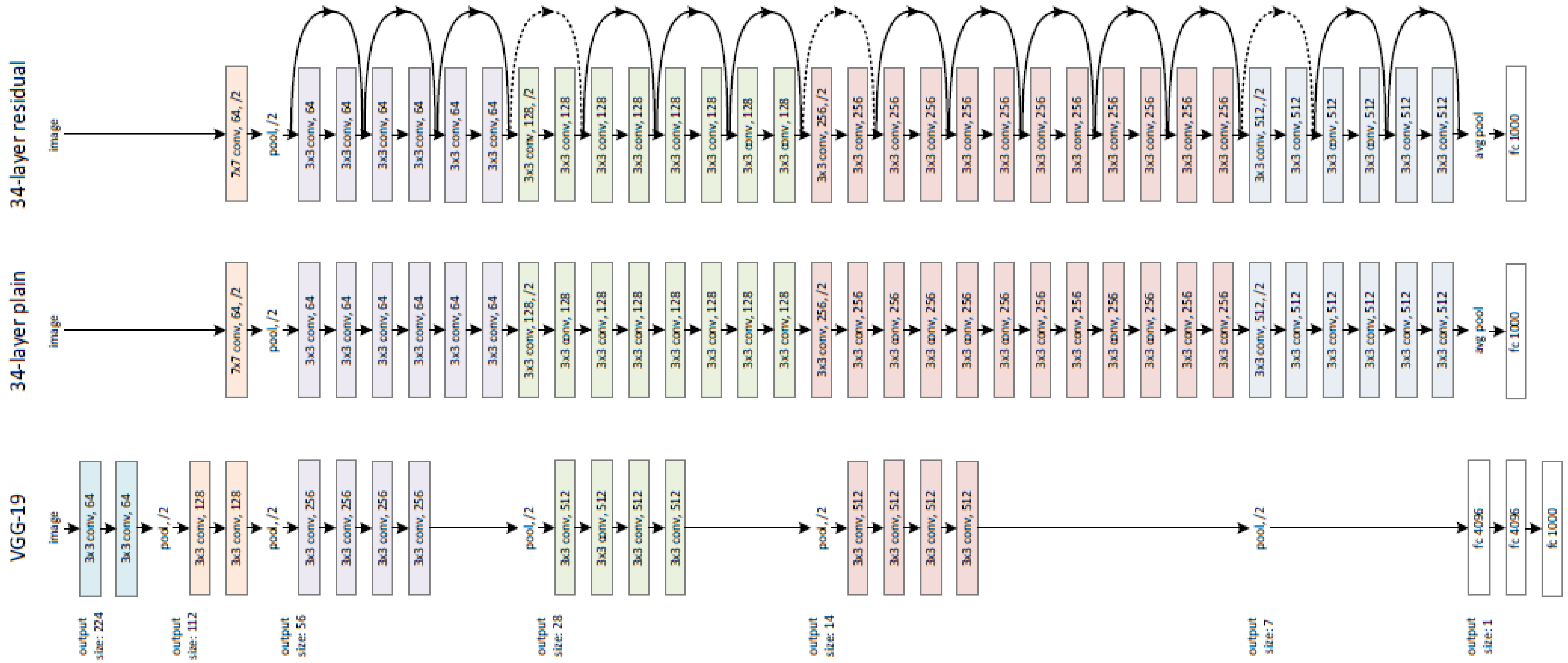
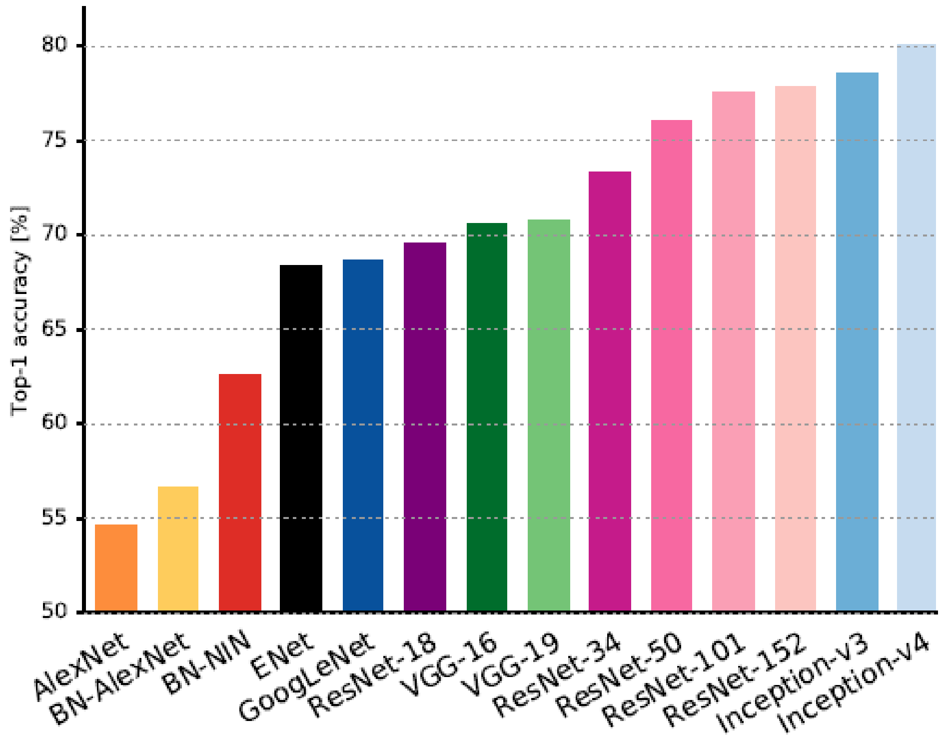


Image from [https://arxiv.org/pdf/1512.03385.pdf]

Comparing Different DCNNs

- Comparative charts at Top-1 accuracy

  i.e. how often the DCNN is right with ImageNet with its top prediction



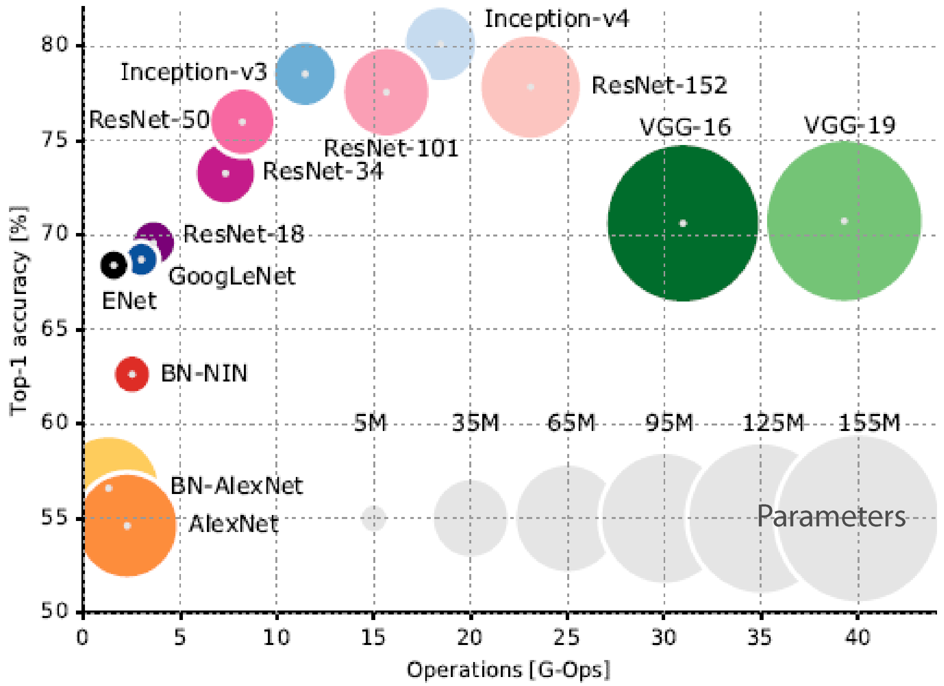


Image from [https://arxiv.org/abs/1605.07678, 2017]

Comparing Different DCNNs

(Same chart, a more recent version)

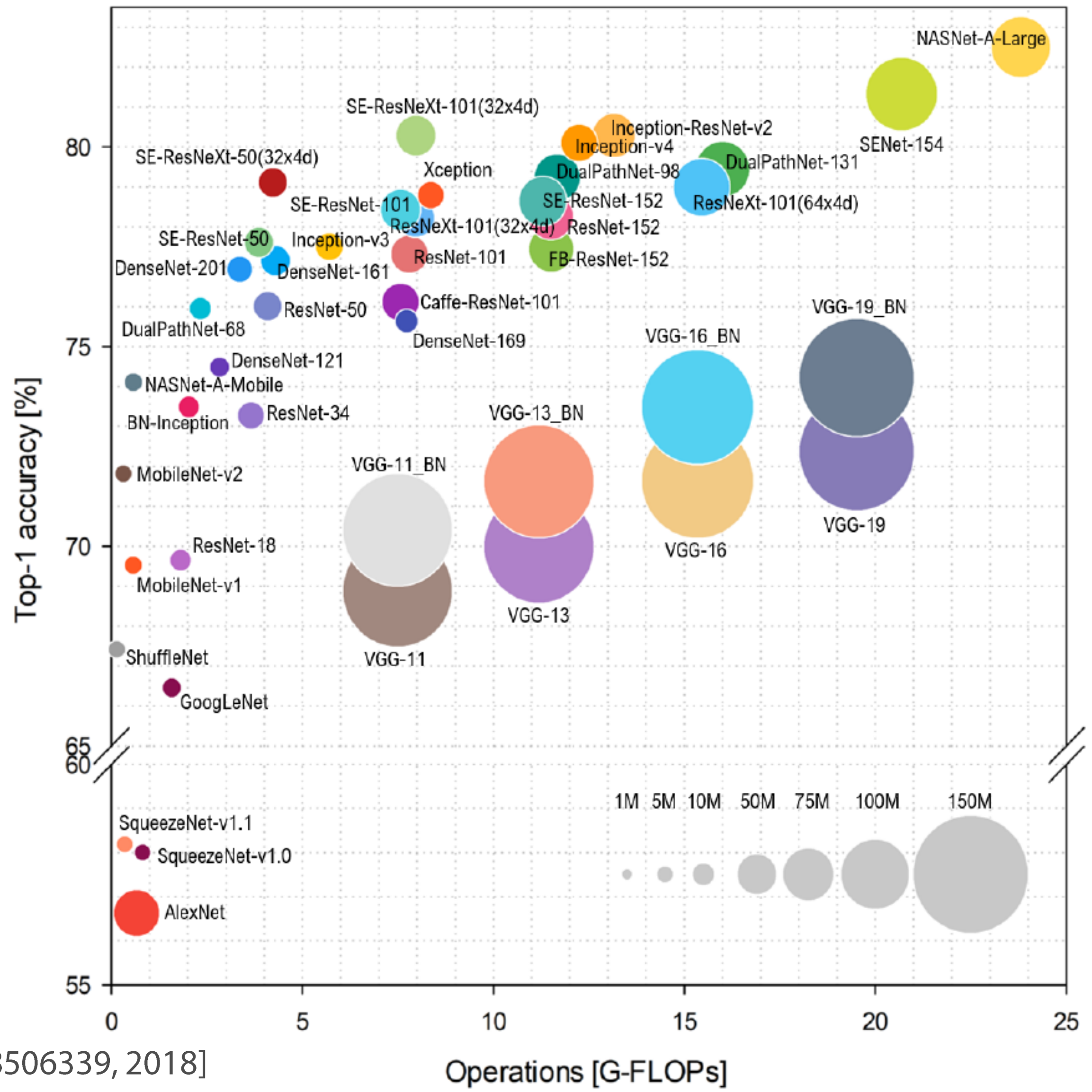
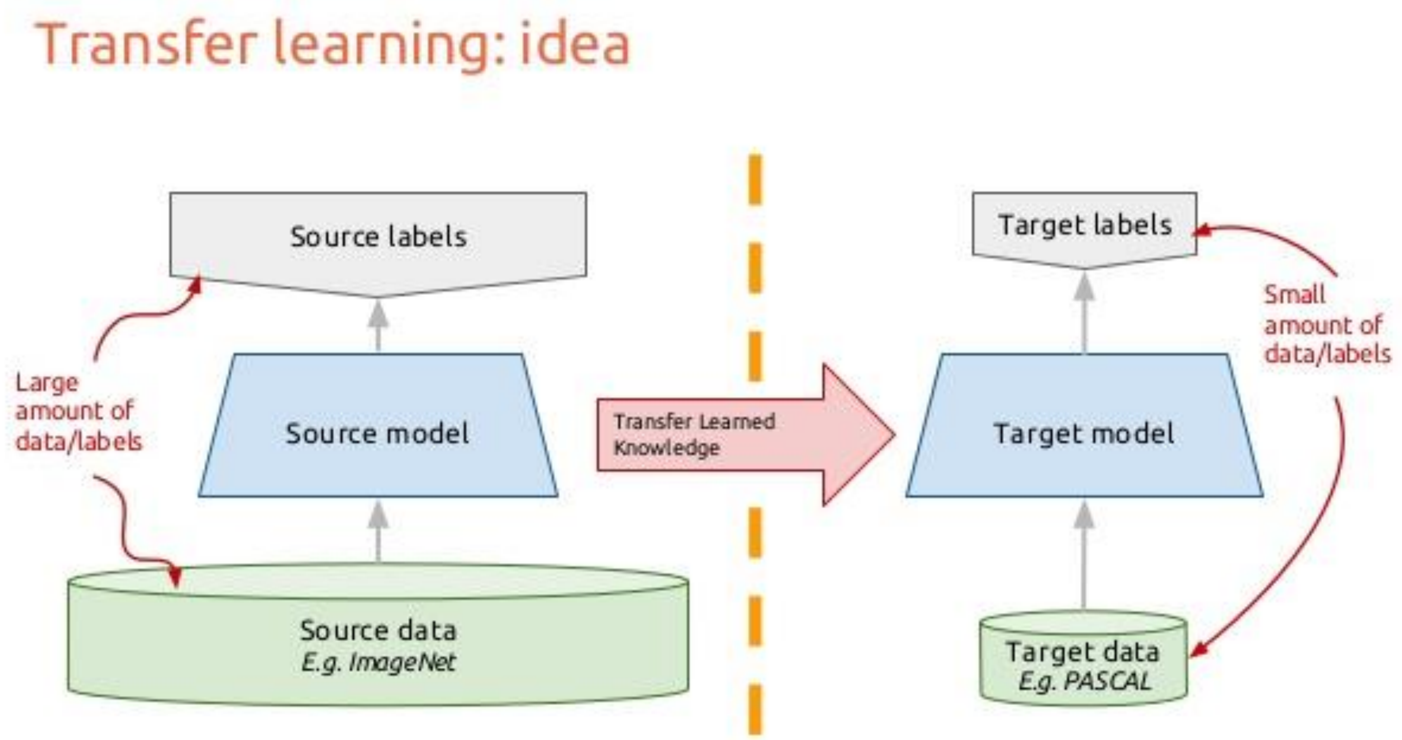


Image from [https://ieeexplore.ieee.org/document/8506339, 2018]

Transfer Learning


Transfer learning: idea
Source labels
Large amount of data/labels
Source model
Source data
E.g. ImageNet
Transfer Learned Knowledge
Target labels
Small amount of data/labels
Target model
Target data
E.g. PASCAL

Do DCNNs Dream of Electric Sheep?

Can DCNNs 'dream'?

sign in | become a supporter | subscribe | search

jobs dating more International edition

theguardian

UK world sport football opinion culture business lifestyle fashion environment tech travel | browse all sections

home › tech

Artificial intelligence (AI)

Yes, androids do dream of electric sheep

Google sets up feedback loop in its image recognition neural network - which looks for patterns in pictures - creating hallucinatory images of animals, buildings and landscapes which veer from beautiful to terrifying

This article is 1 year old

109,591 445

Alex Hern

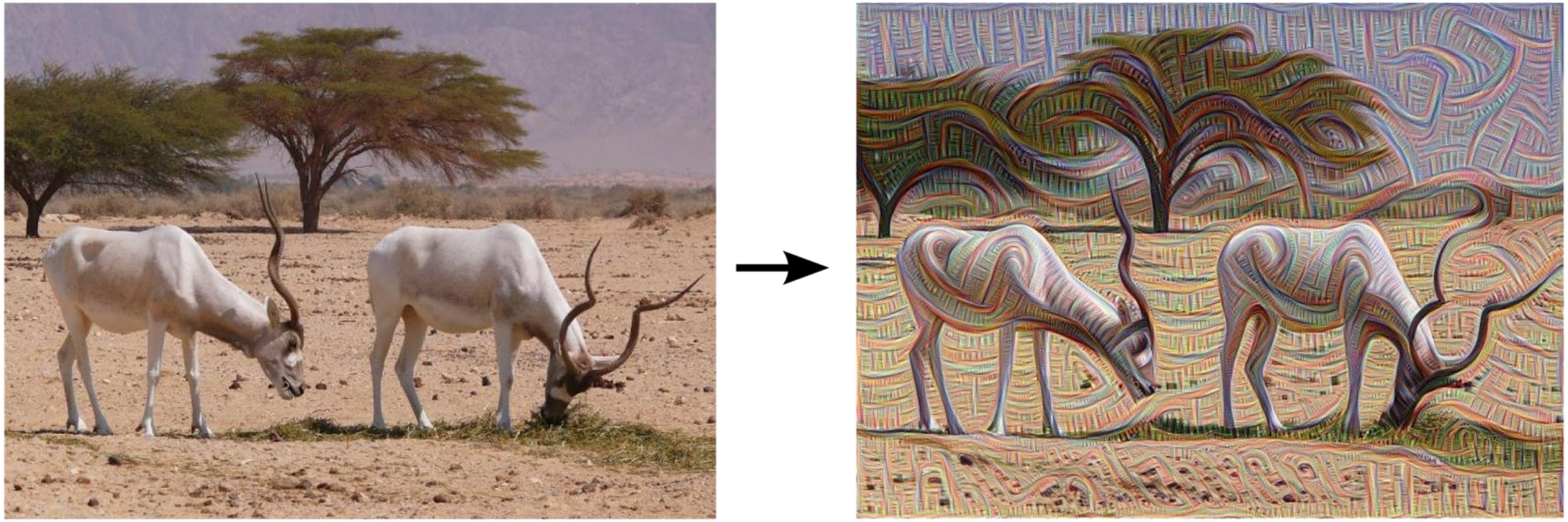
@alexhern

Thursday 18 June 2015 12.57 BST

A hallucinatory filter over a red tree. Spot the animals. Photograph: Google

Can DCNNs 'dream'?

Enhancing lower layers



[images from https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html]

Feature Enhancement

- **Image Space Gradient Descent**

  Define

  $$\mathbf{\Phi}_{k,l}(\boldsymbol{I})$$

  as the response of a DCNN at a layer k, filter l to an image $\boldsymbol{I}$

  Given a specific image $\hat{\boldsymbol{I}}$, we define the loss function

  $$L(\hat{\boldsymbol{I}}, \boldsymbol{I}) := \|\gamma\, \mathbf{\Phi}_{k,l}(\hat{\boldsymbol{I}}) - \mathbf{\Phi}_{k,l}(\boldsymbol{I})\|^2$$

  Amplification factor

  The optimization problem

  $$\boldsymbol{I}^* := \text{argmin}_{\boldsymbol{I}} \left(L(\hat{\boldsymbol{I}}, \boldsymbol{I}) + \lambda \|\boldsymbol{I}\|^2 \right)$$

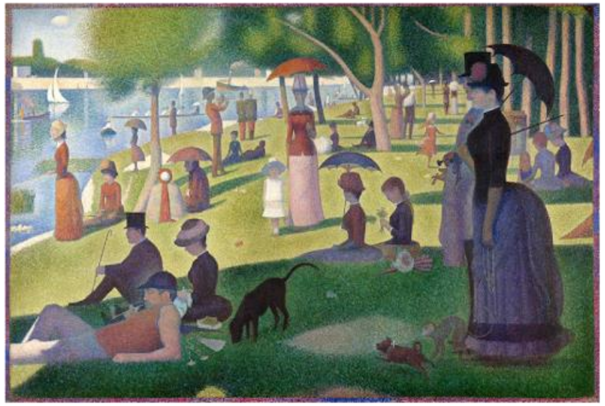
  is solved via gradient descent by computing

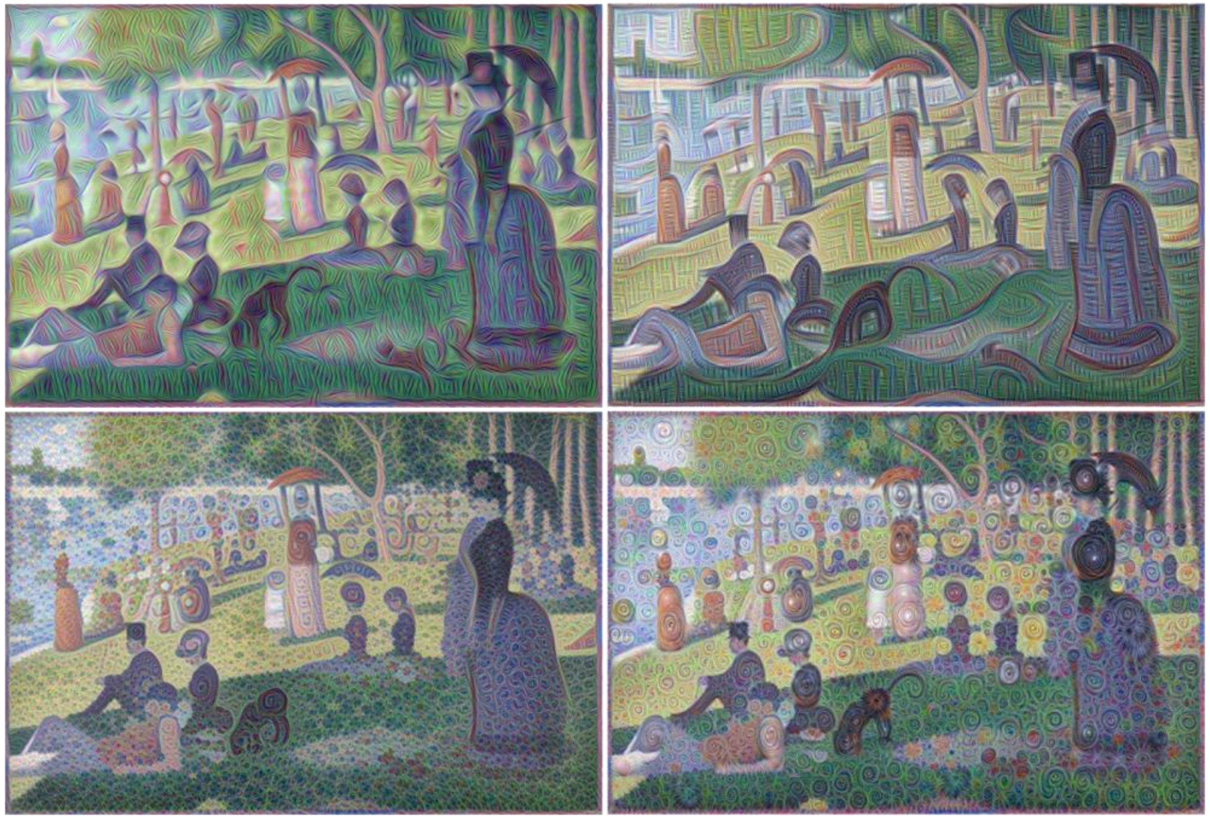
  $$\frac{\partial}{\partial \boldsymbol{I}} \left(L(\hat{\boldsymbol{I}}, \boldsymbol{I}) + \lambda \|\boldsymbol{I}\|^2 \right)$$

  and starting from $\boldsymbol{I}^{(0)} = \hat{\boldsymbol{I}}$

Can DCNNs 'dream'?

Enhancing lower layers

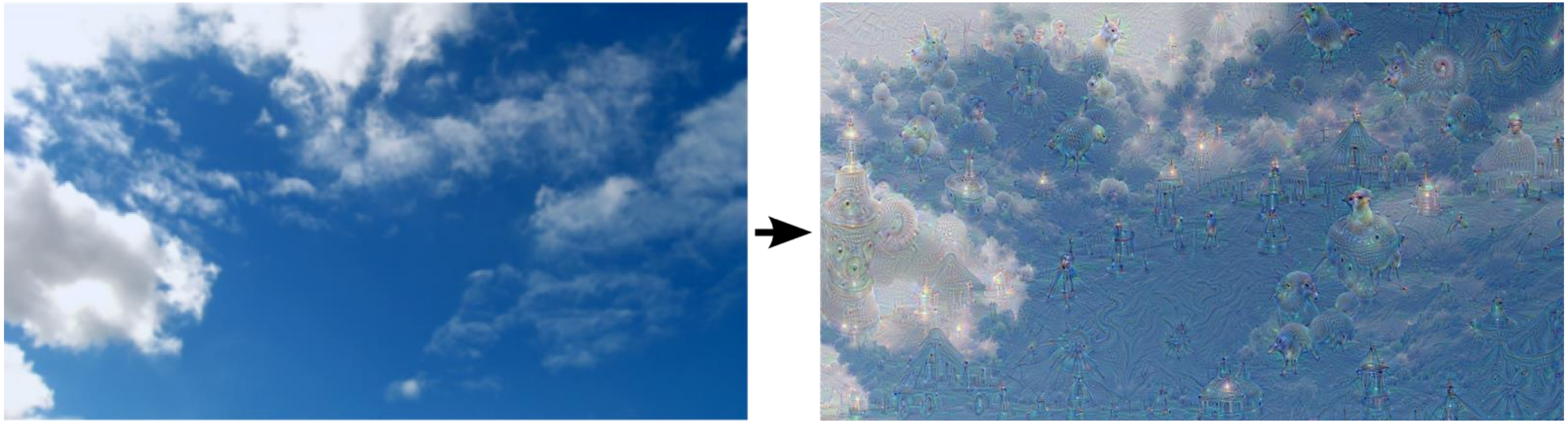




[images from https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html]

Can DCNNs 'dream'?

Enhancing upper layers



[images from https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html]

Can DCNNs 'dream'?

Letting the DCNN go on its own

[images from https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html]

Can DCNNs 'dream'?

Letting the DCNN go on its own

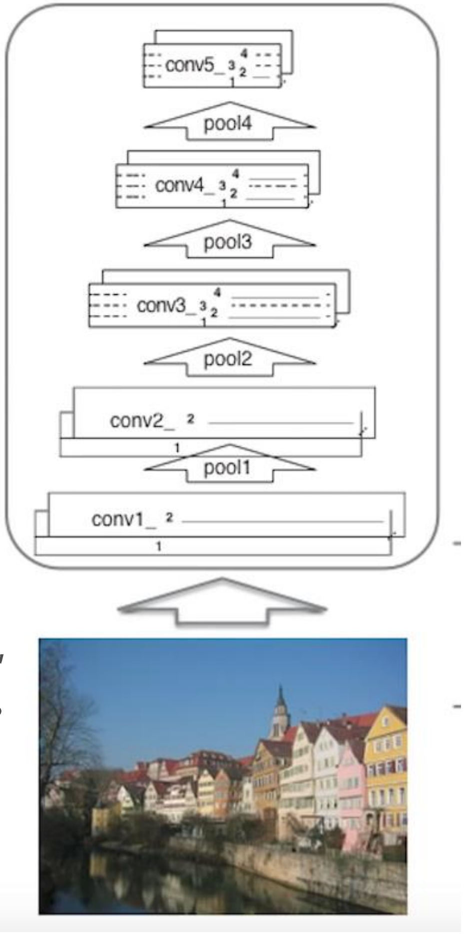
[images from https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html]

The Power of Abstraction
(in layers)

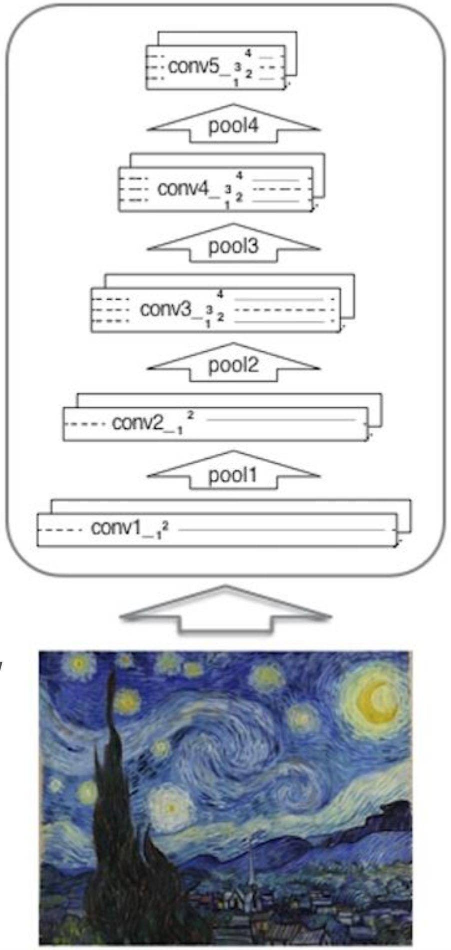
- ***Different Layers of a Deep Convolutional Neural Network***

  What kind of information does each layer 'store'?

'Content' image



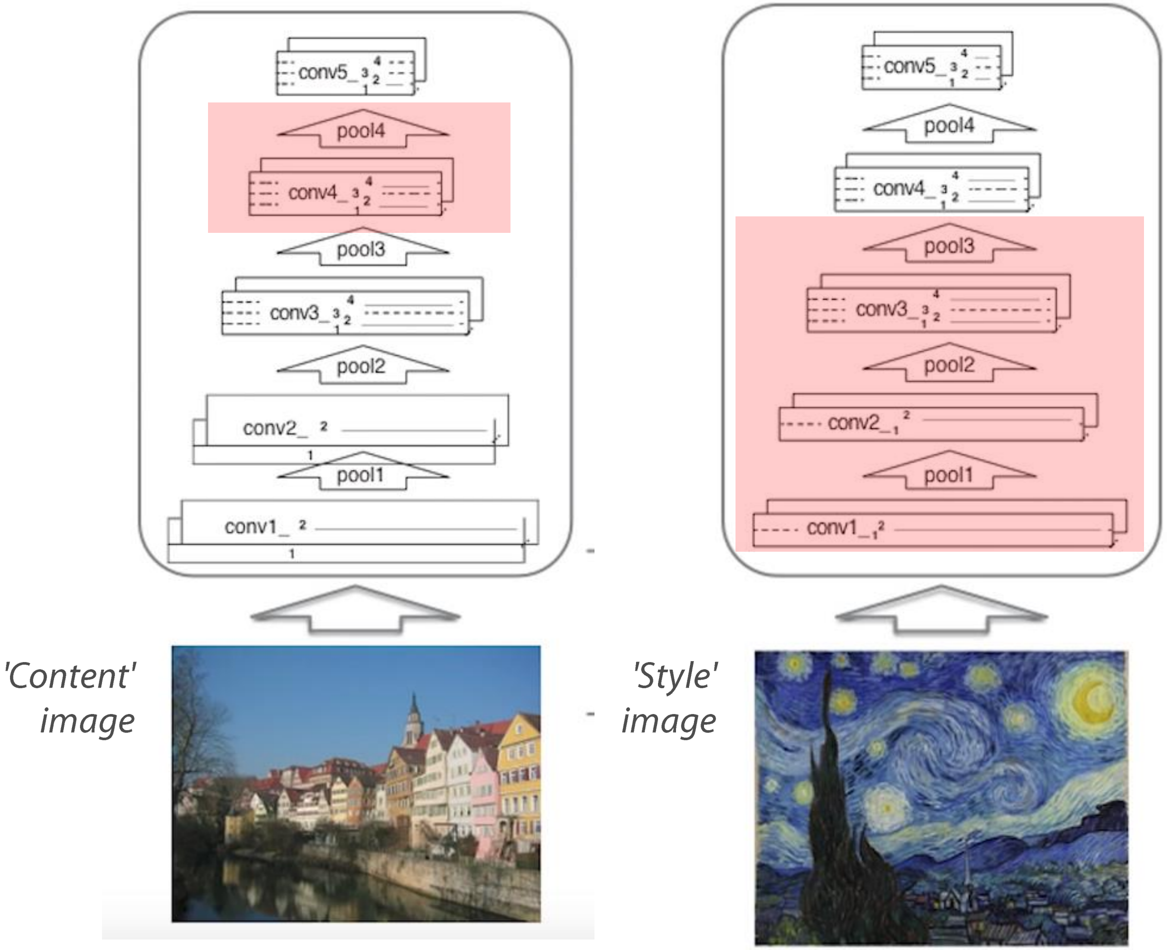
'Style' image



[images from https://www.youtube.com/watch?v=N14_w2RG1A8]

- ***Different Layers of a Deep Convolutional Neural Network***

  What kind of information does each layer 'store'?



Create a new image by combining more of the 'Content' top layer and more of the 'Style' low layers

[images from https://www.youtube.com/watch?v=N14_w2RG1A8]

Mixing Two Images

- **Image Space Gradient Descent**

  Define

  $$\boldsymbol{\Phi}_{k,l}(\boldsymbol{I})$$

  as the response of a DCNN at a layer k, filter l to an image $\boldsymbol{I}$

  Given a specific image $\hat{\boldsymbol{I}}_1$ and $\hat{\boldsymbol{I}}_2$, we define the loss function

  $$L(\hat{\boldsymbol{I}}, \boldsymbol{I}) := \sum_{k,l} \|\boldsymbol{M}_{k,l}(\boldsymbol{\Phi}_{k,l}(\hat{\boldsymbol{I}}_2), \boldsymbol{\Phi}_{k,l}(\hat{\boldsymbol{I}}_1)) - \boldsymbol{\Phi}_{k,l}(\boldsymbol{I})\|^2$$

  Weighted Merge Function

  The optimization problem

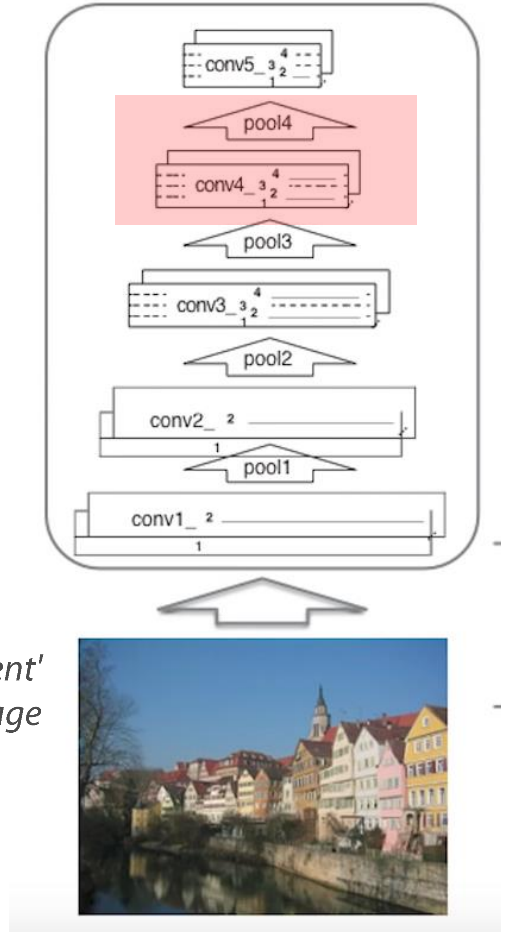
  $$\boldsymbol{I}^* := \operatorname{argmin}_{\boldsymbol{I}} \left(L(\hat{\boldsymbol{I}}, \boldsymbol{I}) + \lambda \|\boldsymbol{I}\|^2 \right)$$

  is solved via gradient descent starting from $\boldsymbol{I}^{(0)} = \hat{\boldsymbol{I}}_1$

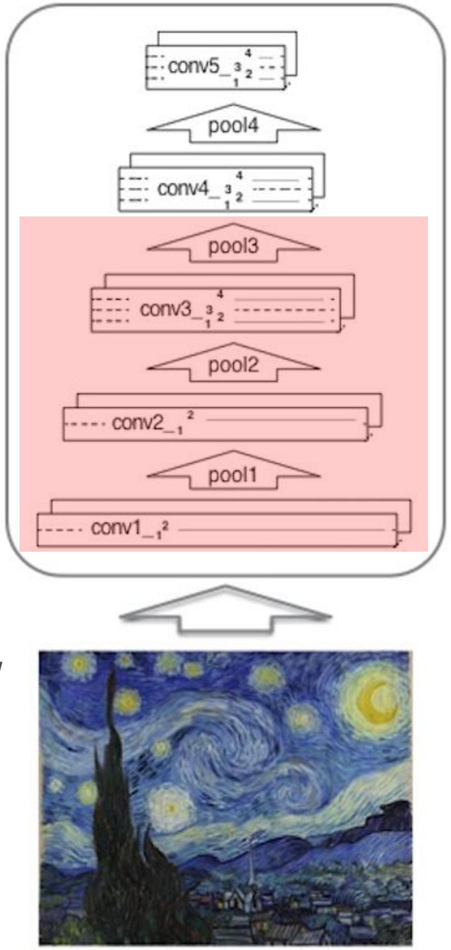
- ***Different Layers of a Deep Convolutional Neural Network***

  What kind of information does each layer 'store'?

'Content' image



'Style' image



Create a new image by combining more of the 'Content' top layer and more of 'Style' low layers

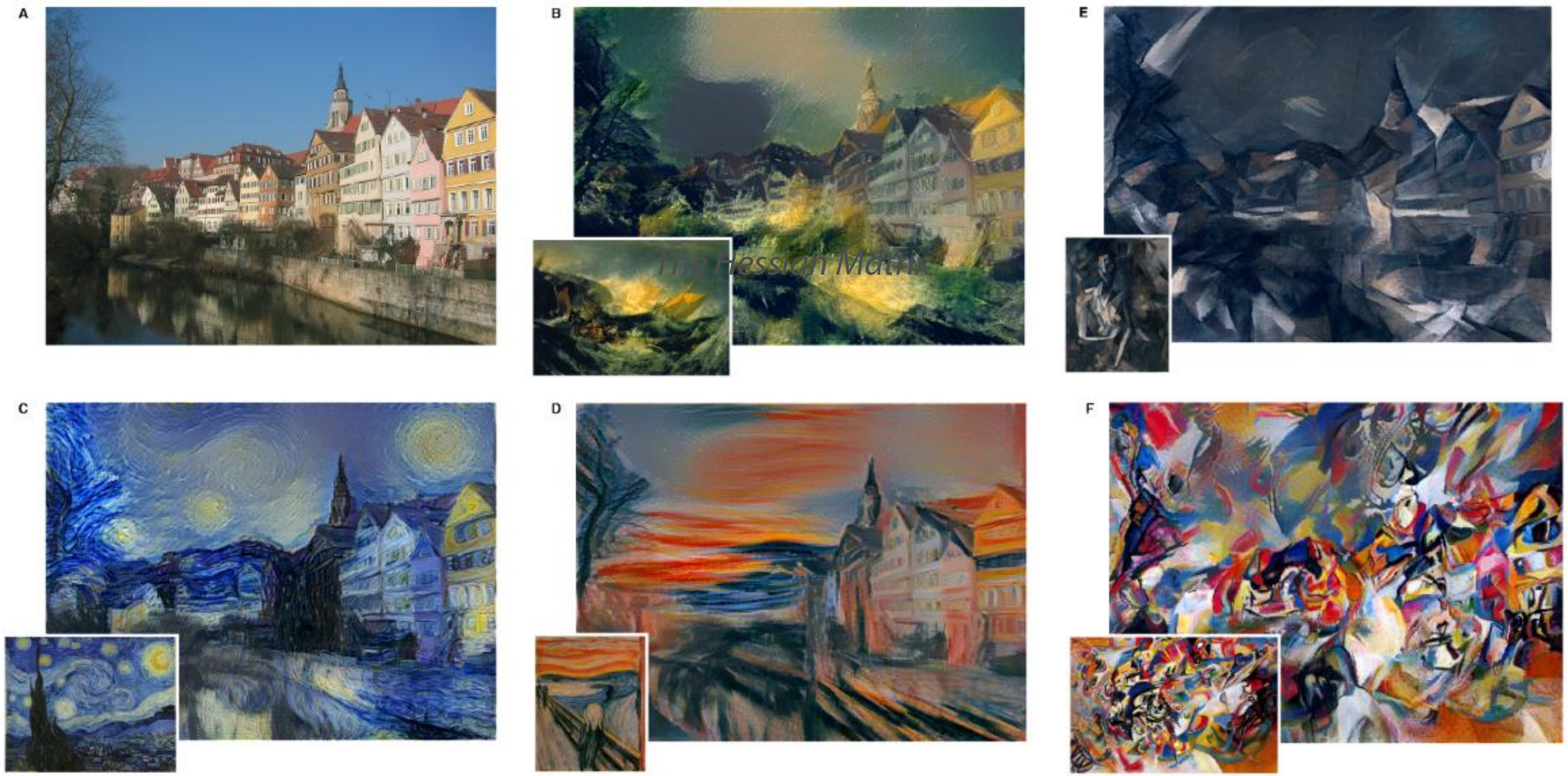
This is the result

[images from https://www.youtube.com/watch?v=N14_w2RG1A8]

The Power of Abstraction

- ***Different Layers of a Deep Convolutional Neural Network***

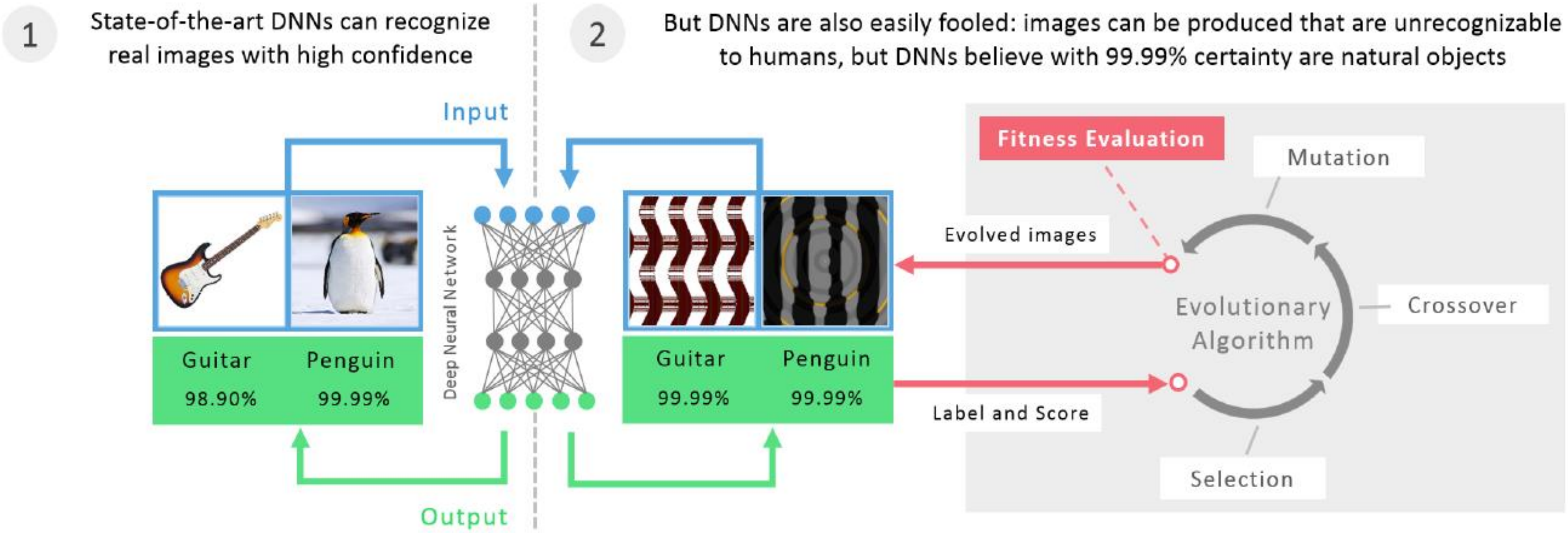
  Further examples:



[images from https://arxiv.org/pdf/1508.06576.pdf]

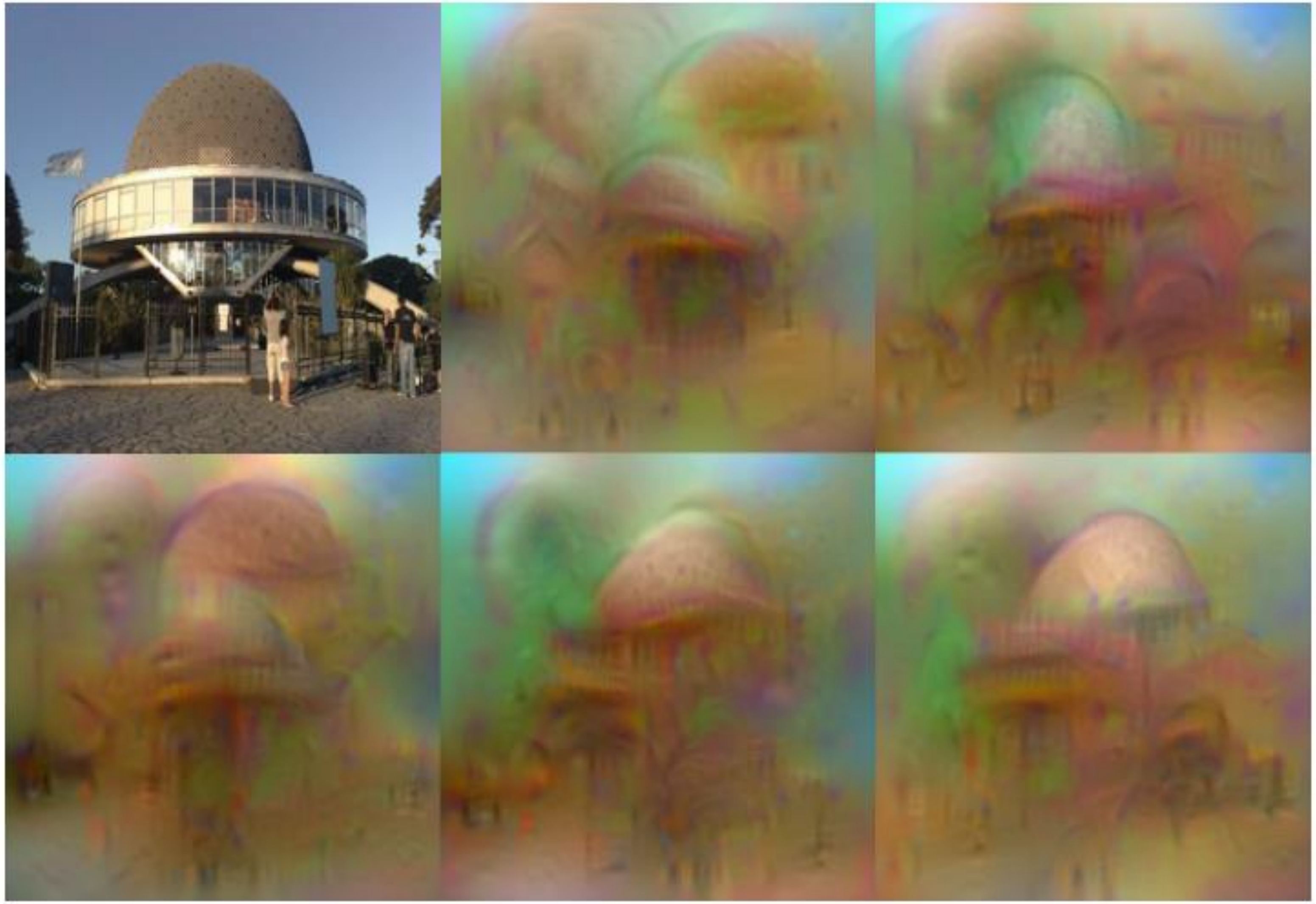
Human-like Vision?
No way

A DCNN can be fooled…



[images from https://arxiv.org/pdf/1412.1897v4.pdf]

Reconstructing Images from Feature Maps



[images from https://arxiv.org/pdf/1412.0035v1.pdf]

Reconstructing Images from Feature Maps

- **Image Space Gradient Descent**

Define

$$\boldsymbol{\Phi}_{k,l}(\boldsymbol{I})$$

as the response of a DCNN at a layer k, filter l to an image $\boldsymbol{I}$

Given a specific image $\hat{\boldsymbol{I}}$, we define the loss function

$$L(\hat{\boldsymbol{I}}, \boldsymbol{I}) := \|\boldsymbol{\Phi}_{k,l}(\hat{\boldsymbol{I}}) - \boldsymbol{\Phi}_{k,l}(\boldsymbol{I})\|^2$$

and the optimization problem

$$\boldsymbol{I}^* := \mathrm{argmin}_{\boldsymbol{I}} \left(L(\hat{\boldsymbol{I}}, \boldsymbol{I}) + \rho P(\boldsymbol{I}) + \lambda \|\boldsymbol{I}\|^2 \right)$$

L2 Regularization

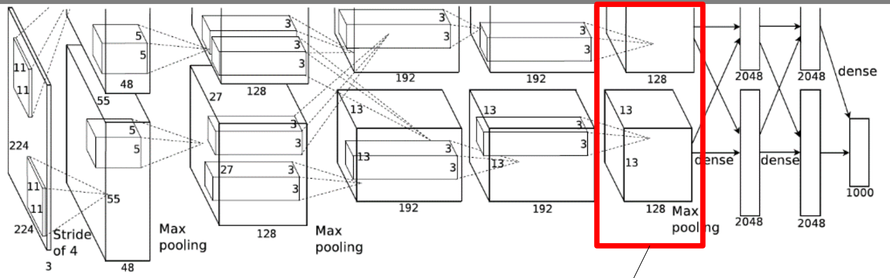
'Statistical Realism'

To solve this, we can compute

$$\frac{\partial}{\partial \boldsymbol{I}} \left(L(\hat{\boldsymbol{I}}, \boldsymbol{I}) + \rho P(\boldsymbol{I}) + \lambda \|\boldsymbol{I}\|^2 \right)$$

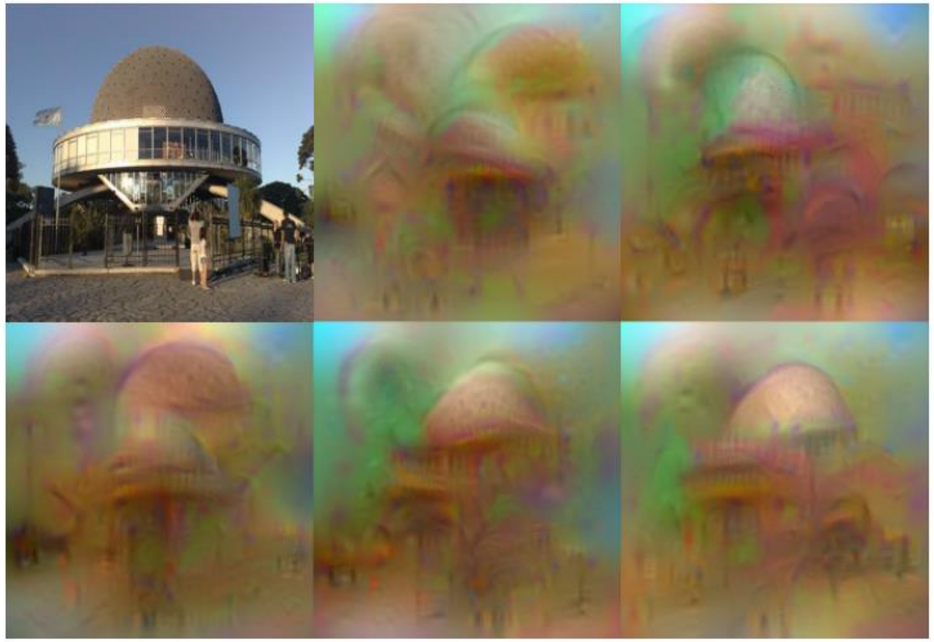
and apply a gradient descent procedure, starting from a random image $\boldsymbol{I}^{(0)}$

Reconstructing Images from Feature Maps



$\Phi_{k,l}(\hat{\boldsymbol{I}})$ is taken here

This is $\hat{\boldsymbol{I}}$



The remaining five images were generated using image space gradient descent with different initial images $\boldsymbol{I}^{(0)}$

[images from https://arxiv.org/pdf/1412.0035v1.pdf]

Just add some little noise ...

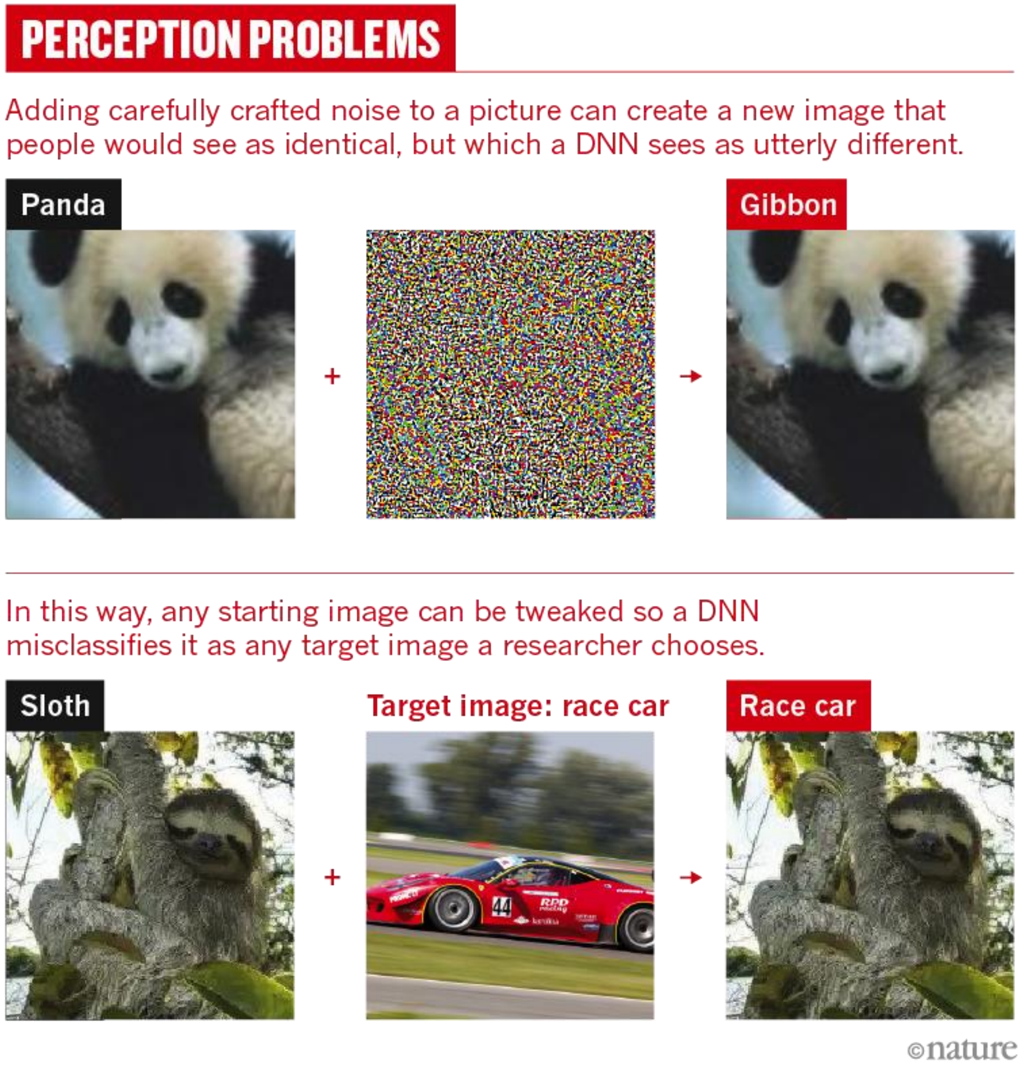
nature

Subscribe

NEWS FEATURE · 09 OCTOBER 2019

Why deep-learning AIs are so easy to fool

Artificial-intelligence researchers are trying to fix the flaws of neural networks.

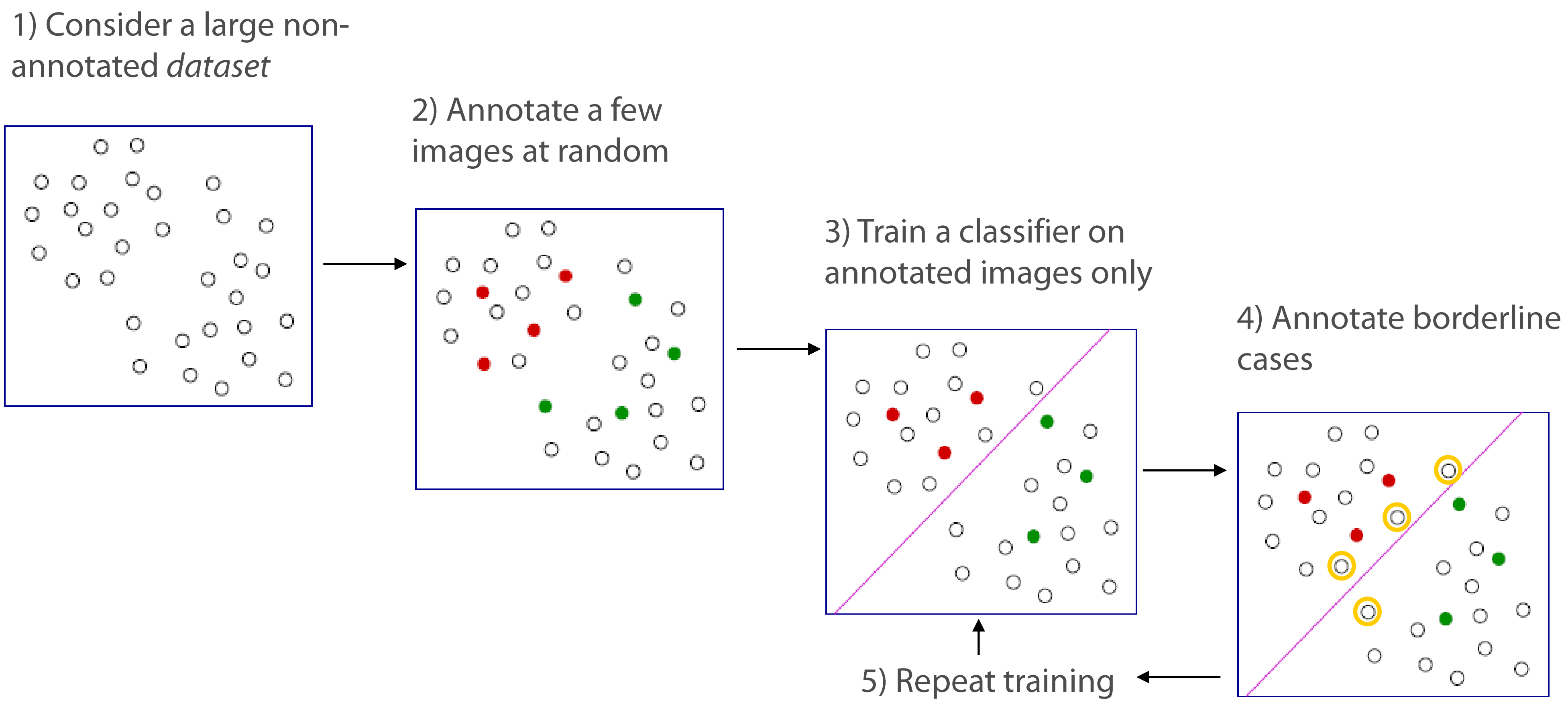


[images from https://www.nature.com/articles/d41586-019-03013-5

No Free Lunch:
creating an annotated dataset

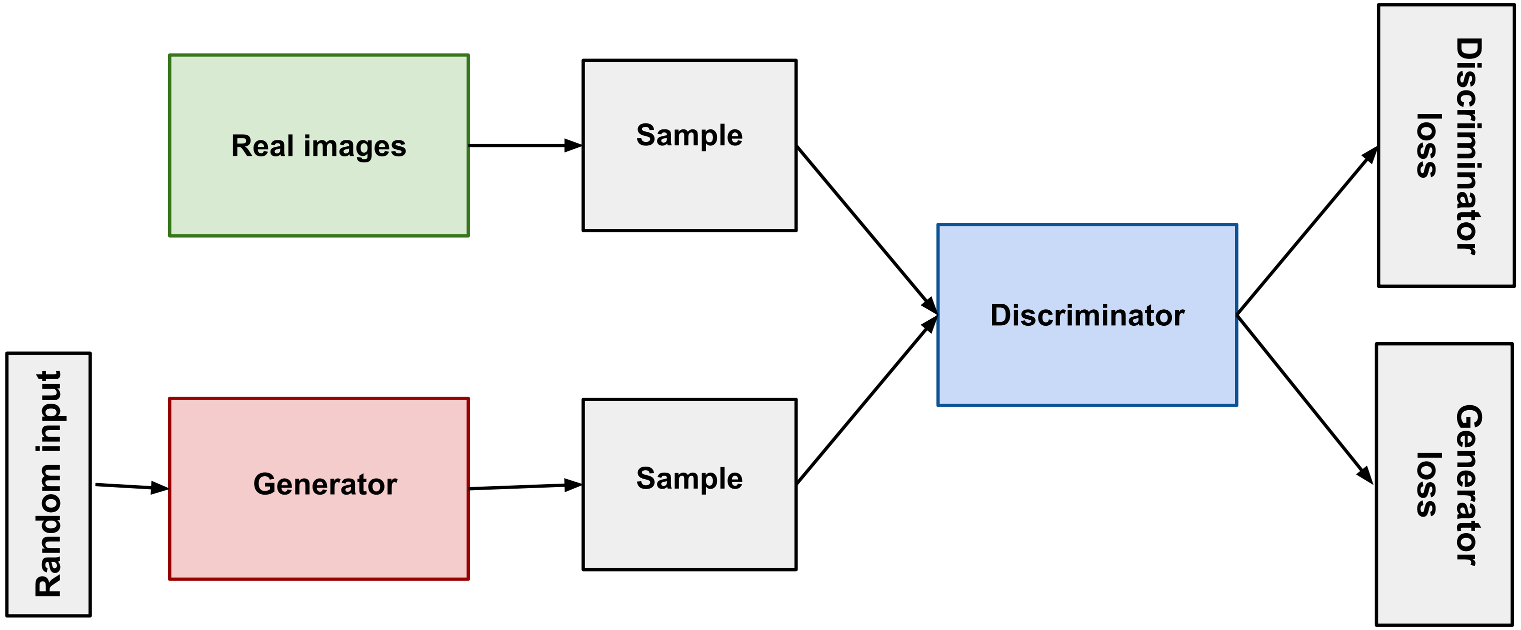
Active Learning

When the network decides which annotations should be made



Generative Adversarial Network

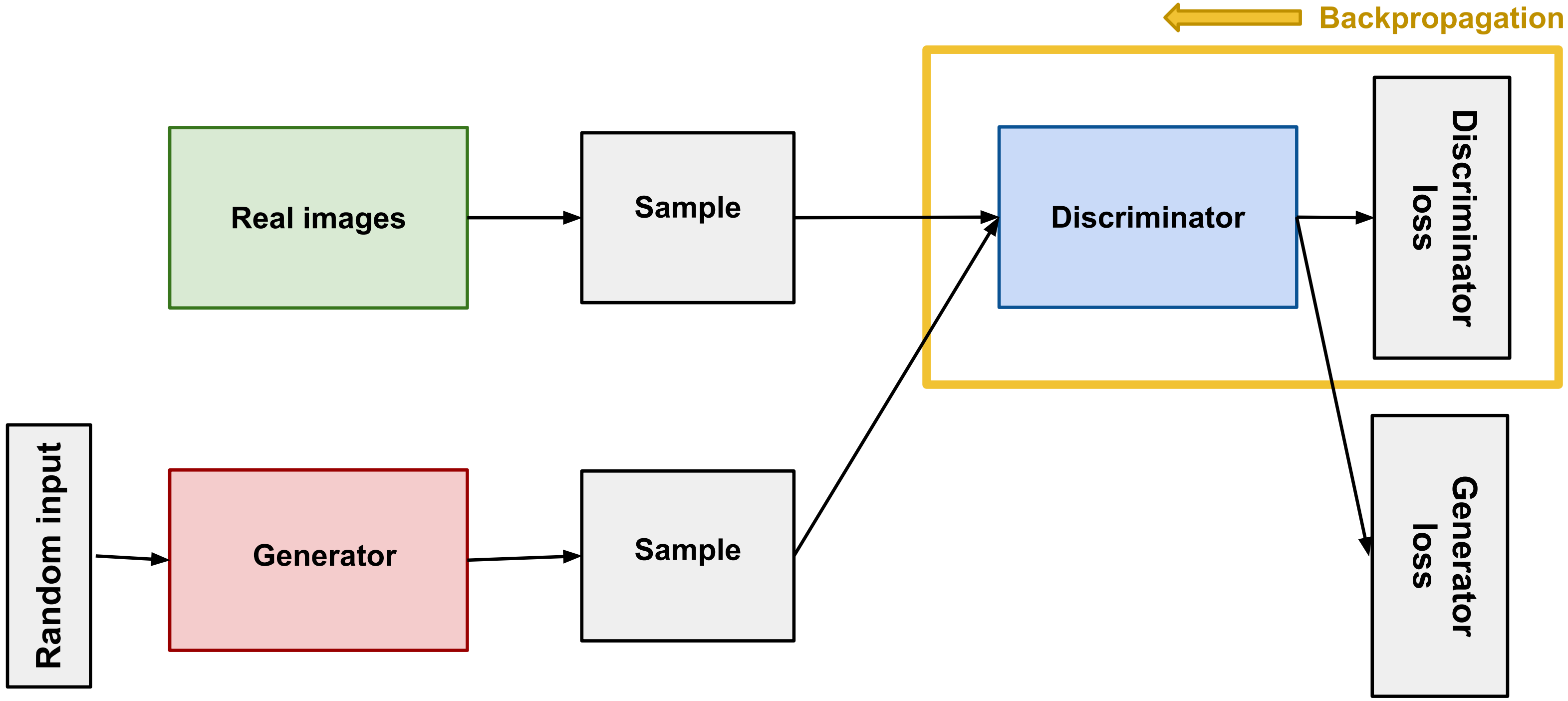
- ***Two competing networks***
  a) A *discriminator* learns to classify images while detecting fake ones
  b) A *generator* learns how to fool the discriminator



[images from https://developers.google.com/machine-learning/gan/gan_structure]

Generative Adversarial Network

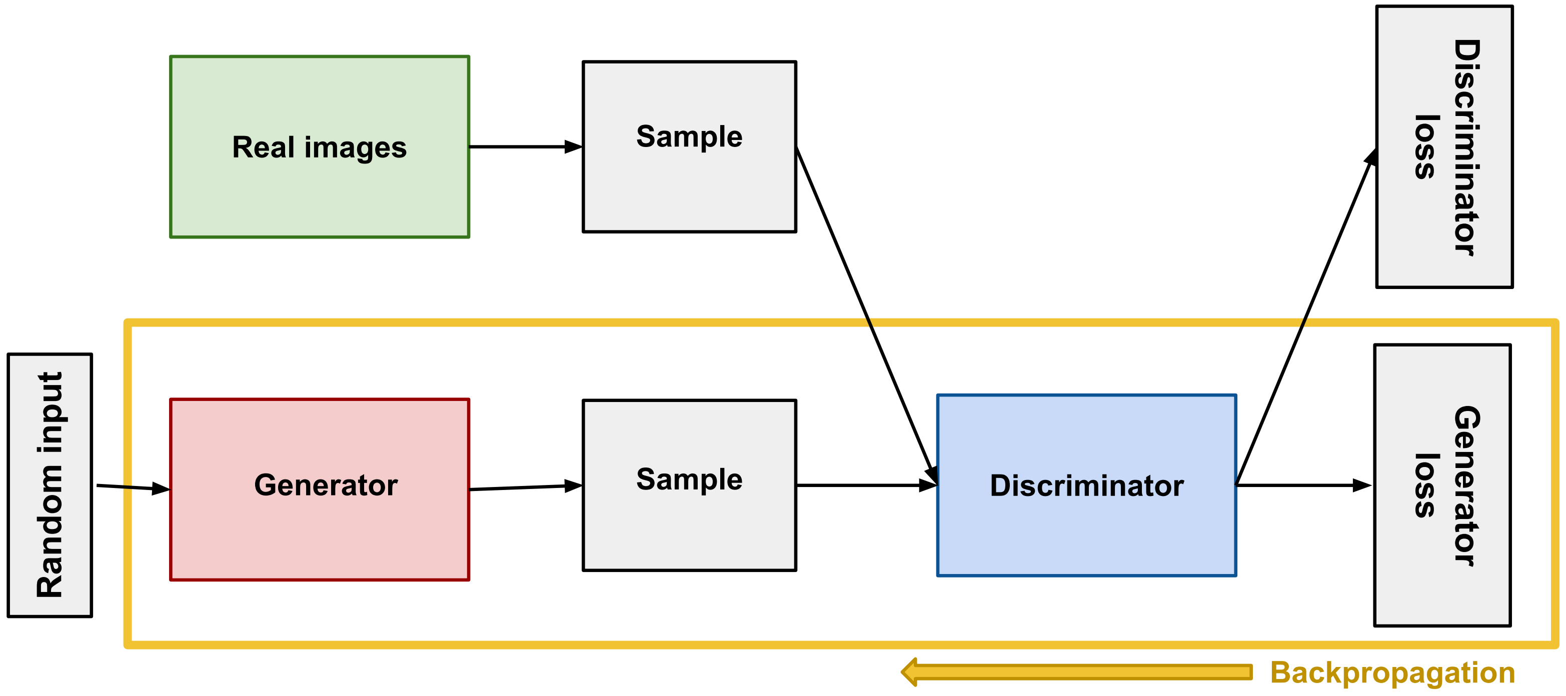
- ***Two competing networks***
  a) A *discriminator* learns to classify images while detecting fake ones
  b) A *generator* learns how to fool the discriminator
  c) The two networks are trained in alternate turns



[images from https://developers.google.com/machine-learning/gan/gan_structure]

Generative Adversarial Network

- ***Two competing networks***
  a) A *discriminator* learns to classify images while detecting fake ones
  b) A *generator* learns how to fool the discriminator
  c) The two networks are trained in alternate turns



[images from https://developers.google.com/machine-learning/gan/gan_structure]

- ***Applications***
  - This method can be used to generate larger datasets from smaller ones
  - Or to generate photo-realistic, yet synthetic images (even 'deep fakes')

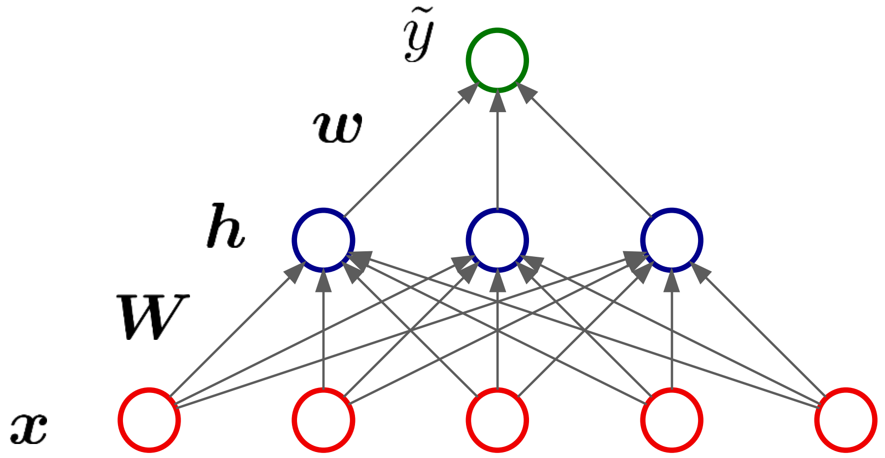
[images from] Karras, Aila, Laine & Lehtinen, 2017

Unsupervised Learning: Auto-Encoders

- **Encoder**

A feed-forward neural network with one hidden layer

$$\tilde{y} = \boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b$$



Auto-Encoders

- **Encoder**

  A feed-forward neural network with one hidden layer

  $$\tilde{y} = \boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b$$

- **Auto-encoder** (*basic idea*)**: encoder + decoder**

  $$\boldsymbol{x}^{[m]} = g(\boldsymbol{W}^{[m]} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + \boldsymbol{b}^{[m]})$$

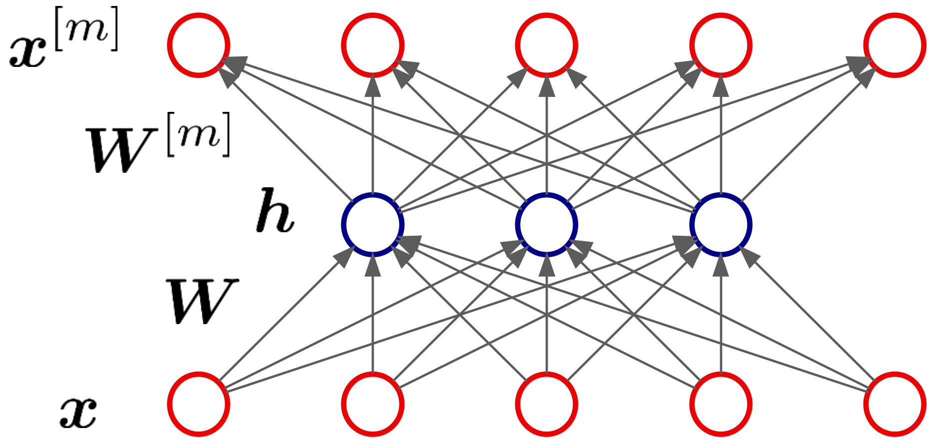
  Loss function (MSE):

  $$L(\boldsymbol{x}^{[m]}, \boldsymbol{x}) = (\boldsymbol{x}^{[m]} - \boldsymbol{x})^2$$

  Initially:

  $$\boldsymbol{W}^{[m]} = \boldsymbol{W}^T$$

  then train the network with each data sample ***onto itself***



- **Auto-encoder** *(More in general)*

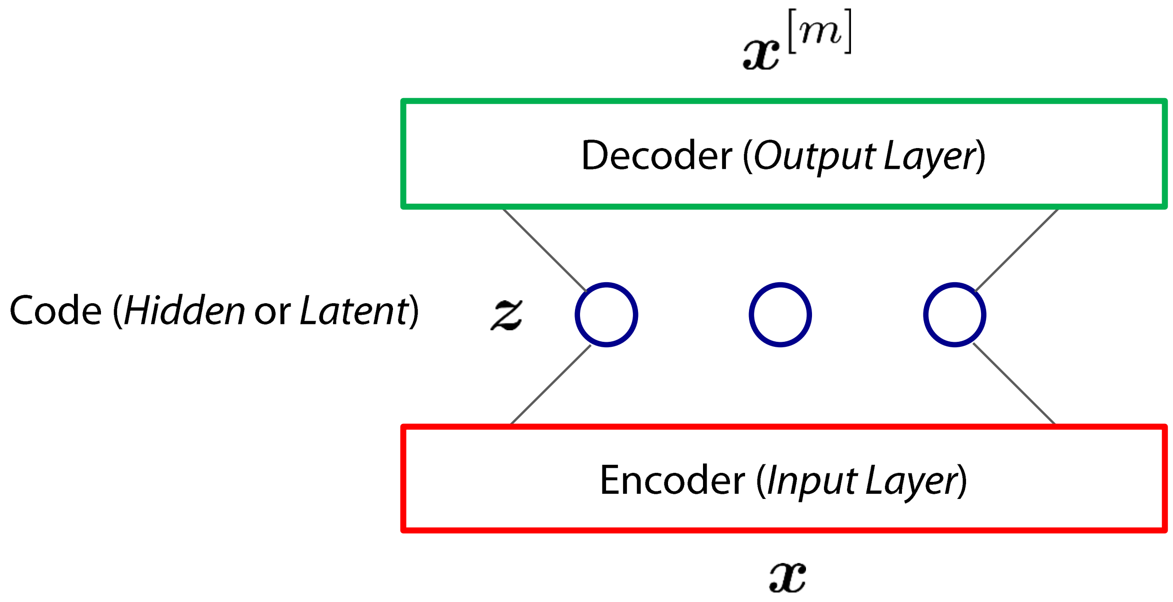
  Two main (composite) layers: **encoder** and **decoder**

  One **hidden** or **latent** layer $\boldsymbol{z}$

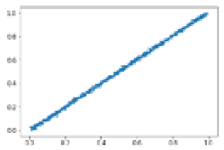
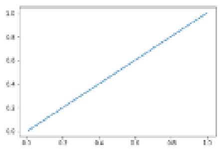
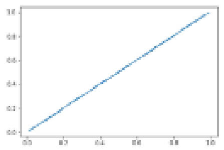
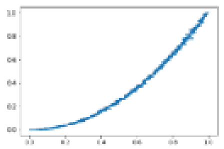
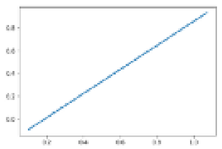
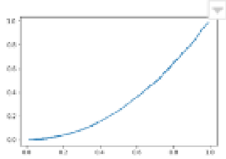
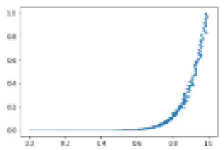
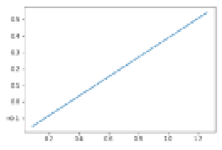
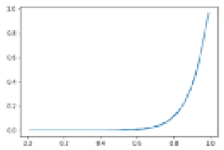
  Each item in the dataset comprises the input only (*Unsupervised Learning*)

$$D := \{(\boldsymbol{x}^{(i)})\}_{i=1}^{N},$$

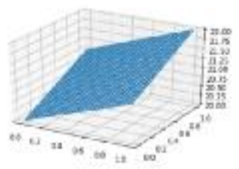
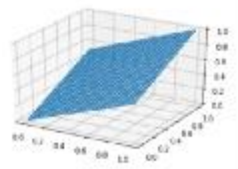
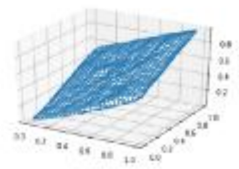
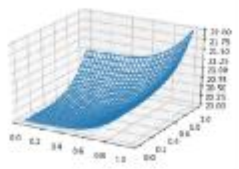
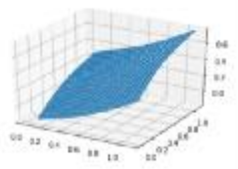
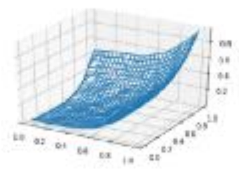
  The result of the optimization is $\boldsymbol{z}$:
  a compact (i.e. lower-dimensional)
  representation of the input $\boldsymbol{x}$



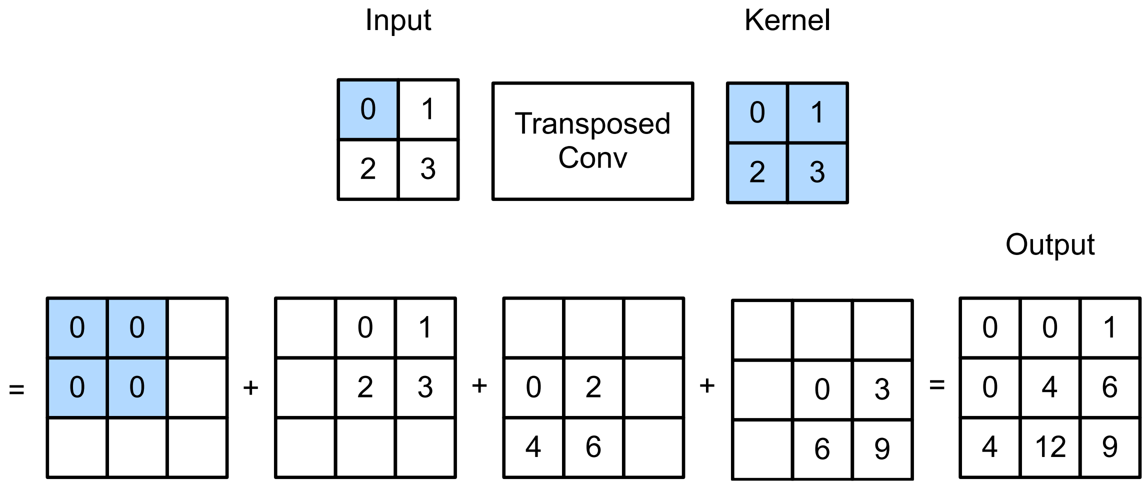
Auto-Encoders vs PCA

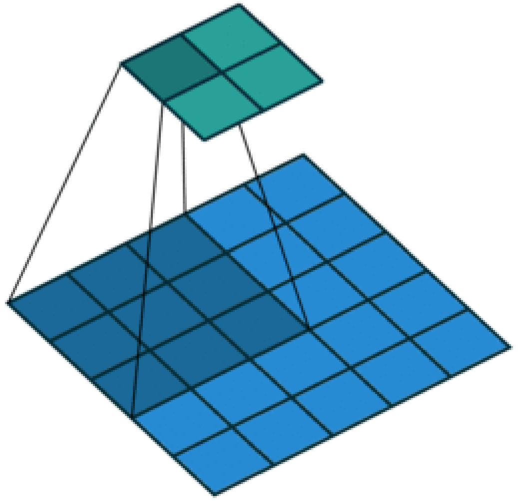
Function	Feature Space	PCA Reconstruction	Auto Encoder Reconstruction
$y=mx+c$			
$y=mx^2+c$			
$y=mx^8+c$			

When non-linearity matters…

Function	Feature Space	PCA Reconstruction	Auto Encoder Reconstruction
Plane			
Curved Surface			

Transposed Convolution (a.k.a. 'Deconvolution')



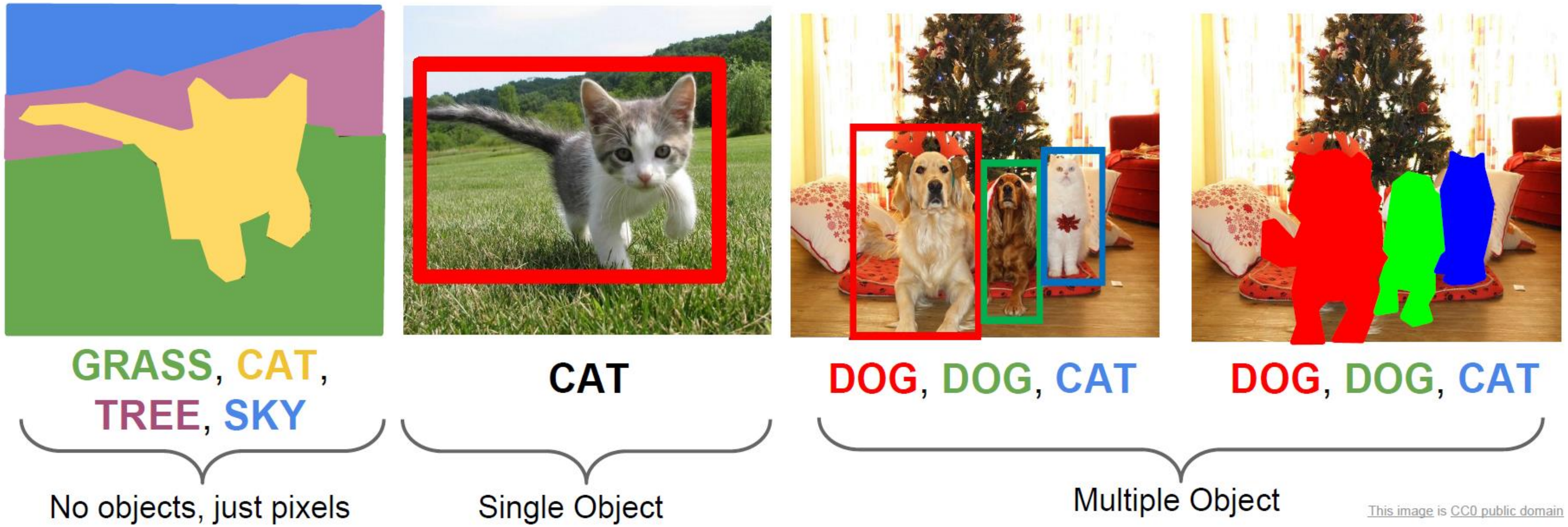


- **For autoencoders based on convolutional layers**
  - Scalar input values are multiplied by the kernel tensor
  - The output feature map is obtained by summing up all contributions

Image Classification
Object Detection
Segmentation

Deep Learning for different imaging tasks

Beyond simple image classification



[images from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf]

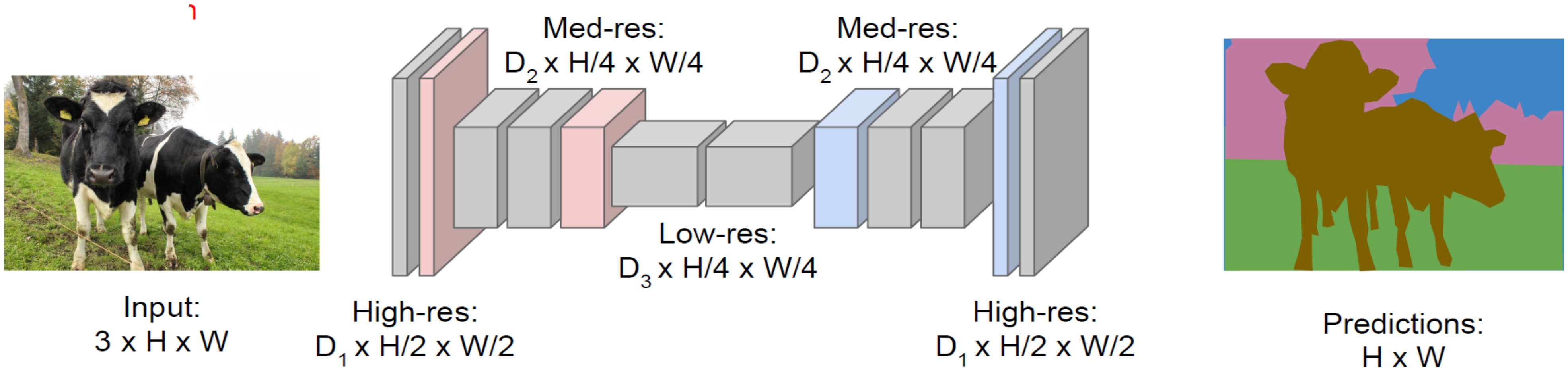
Semantic segmentation

Beyond simple image classification

- ***Similar network architecture, different arrangement***

Fully Convolutional Networks (FCN)

Downsampling first, *upsampling* afterwards

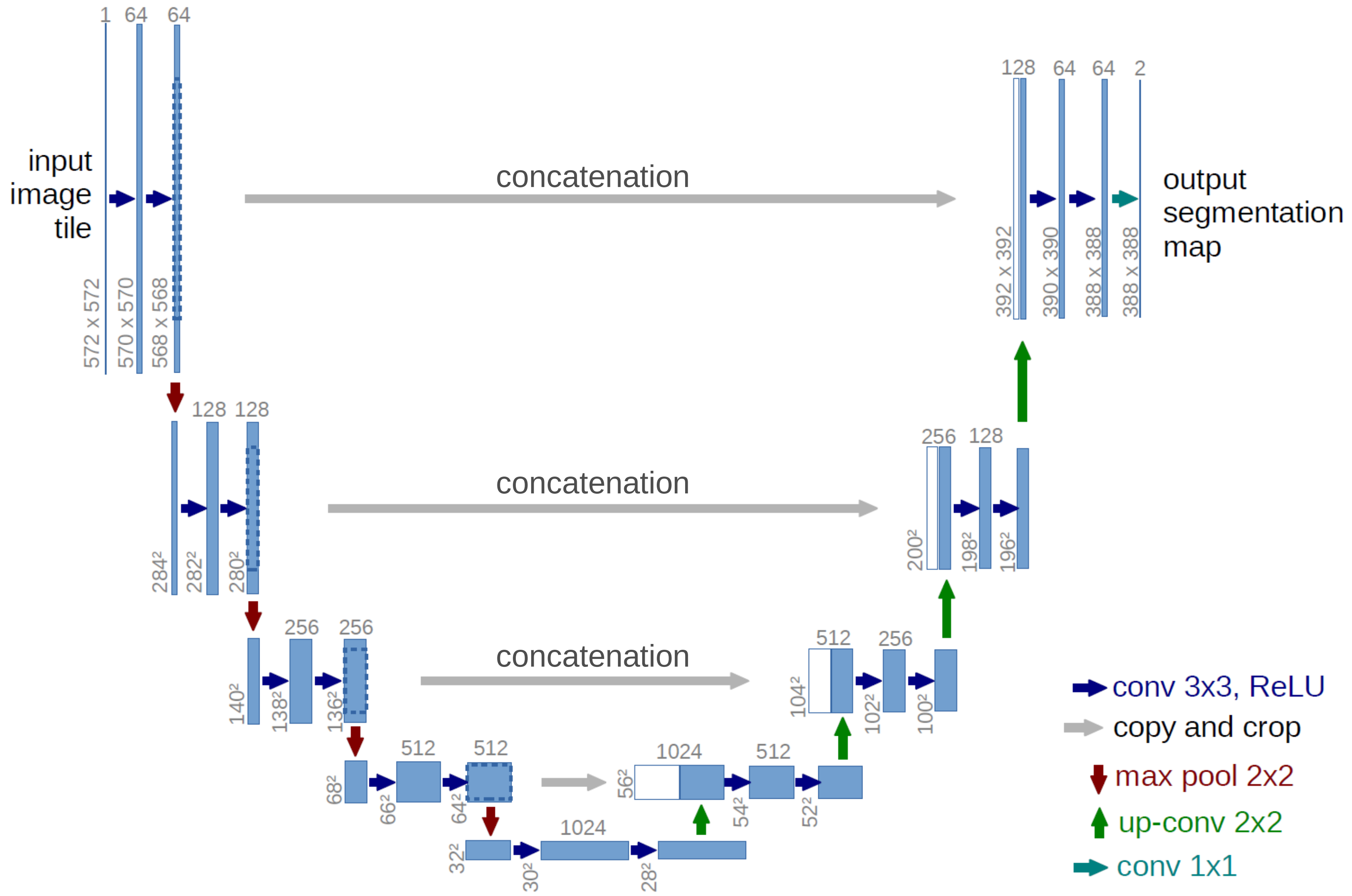


[images from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf]

Semantic segmentation

- ***U-Net* [2015]**

  Great precision
  Fast to train

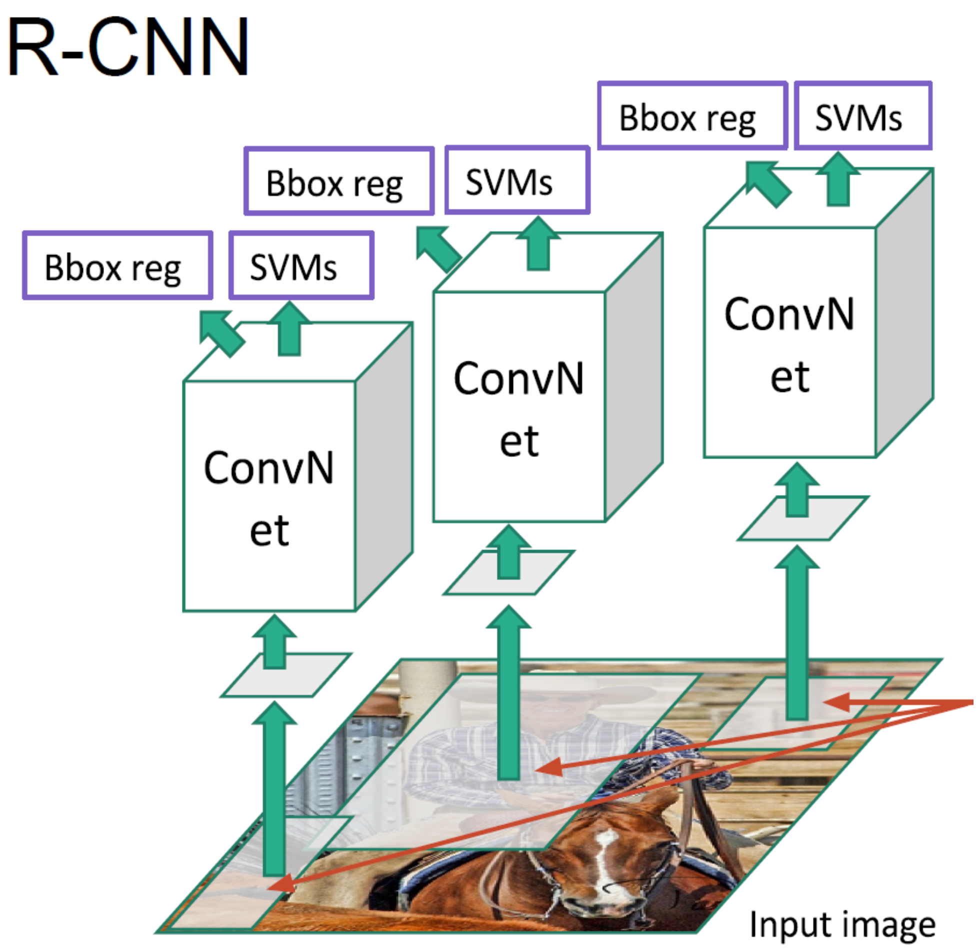


[image from https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/]

Object detection and positioning

Generate boxes and classifications

- ***Two-stage Process***
  - *Generate bounding box candidates*
  - *Pass each candidate through a DCNN*
  - *Select those candidates that are classified with higher certainty*



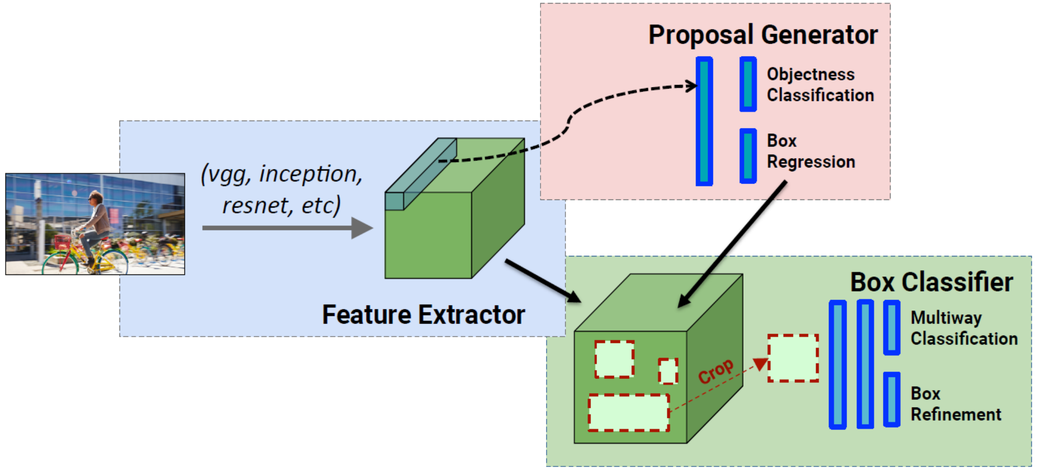
[images from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf]

Generate boxes and classifications

- ***Two-stage to One-stage process***

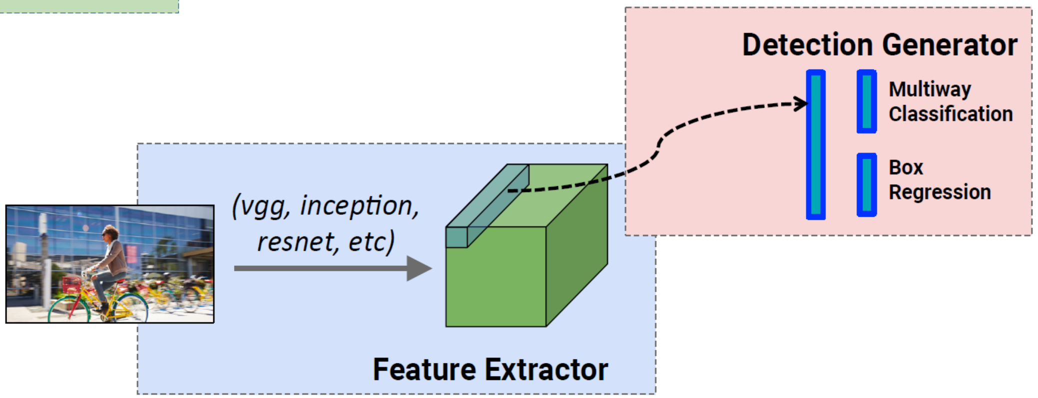
  Generate bounding box candidates and classifications in one go

[images from https://arxiv.org/pdf/1611.10012.pdf



Two stages

One stage



Object detection and positioning

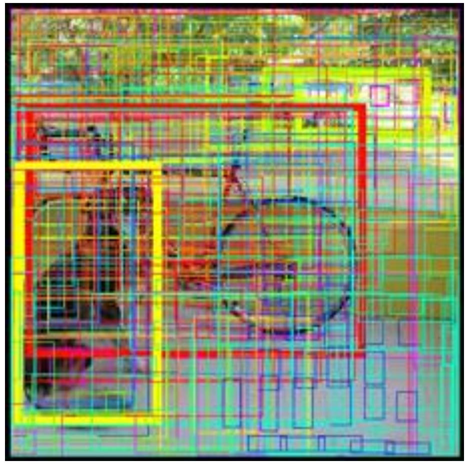
- ***YOLO and SSD: one-pass convolutional network for object detection***

  Generate boxes and classifications *at once*

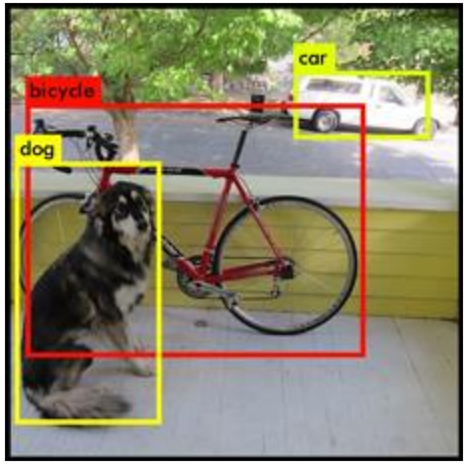
1) Impose a fixed grid over the input image

2) Generate possible bounding boxes

3) Classify each of them

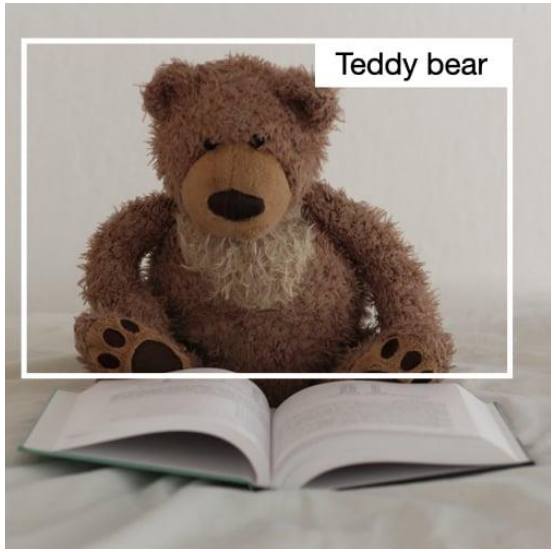


4) Keep the boxes at highest confidence

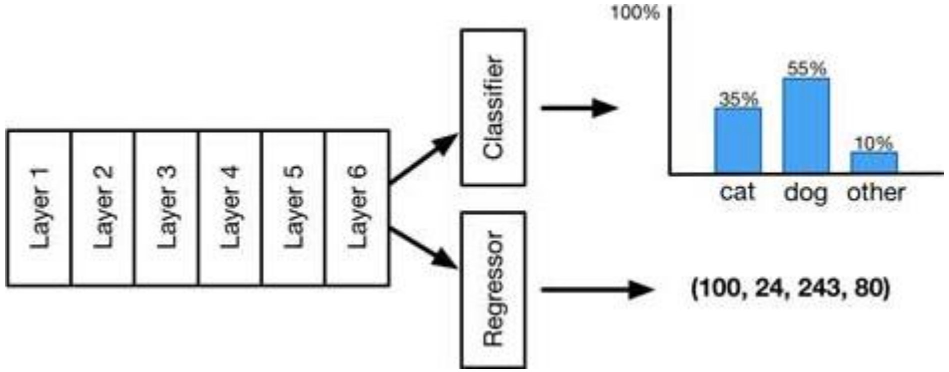


[images from http://machinethink.net/blog/object-detection-with-yolo/]

- ***From classification to localization***



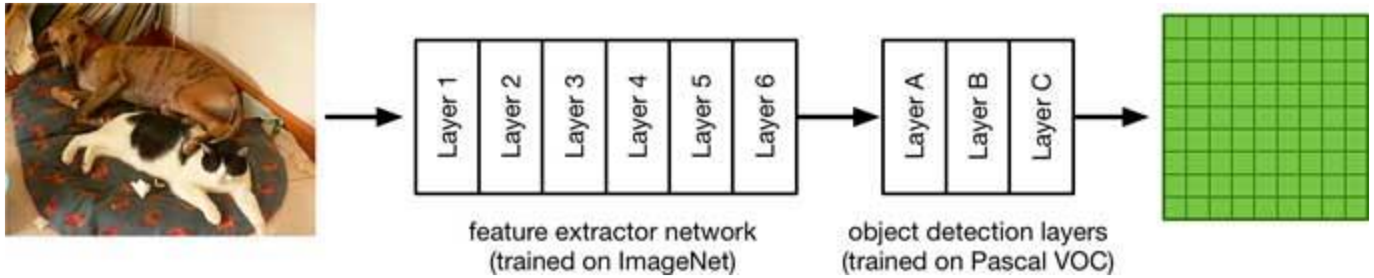
[images from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks]



Loss: cross-entropy

Loss: mean square error

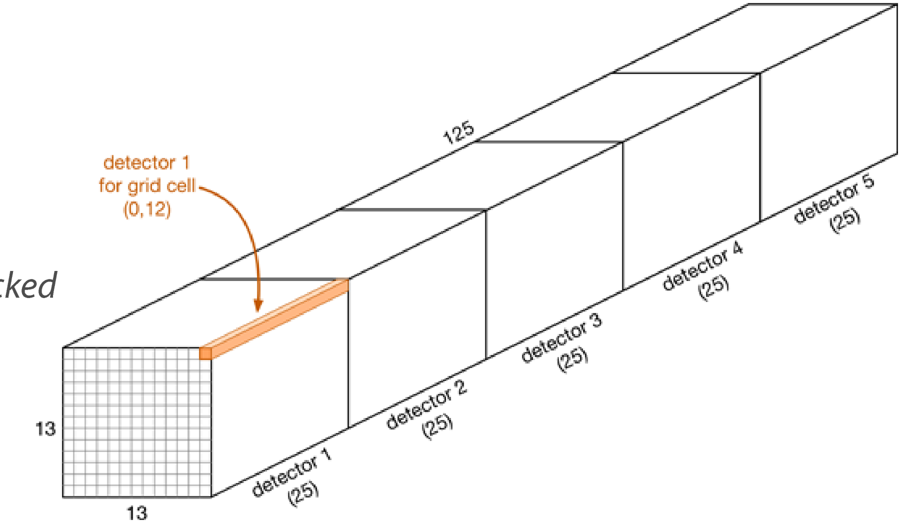
- ***Grid detectors***



Classification (transfer learning)

Multi-class detectors (arranged in a grid)

(Detector blocks are shown as stacked but they work in parallel)



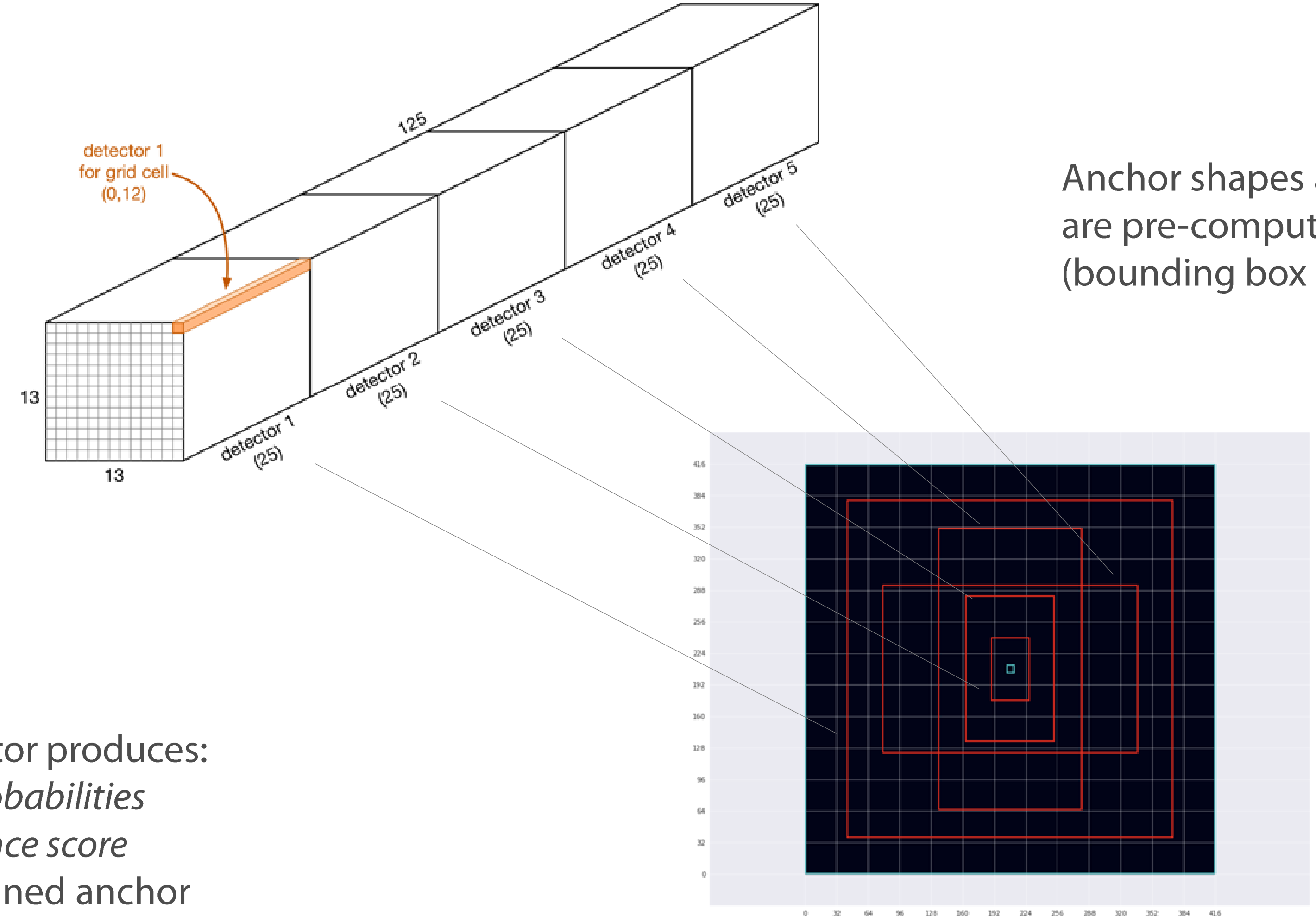
1) Impose a fixed 13 x 13 grid over the input image

2) Assign 5 multiclass detectors to each cell of the grid

3) Each multiclass detector works on a specific *anchor* shape (*see next slide*)

[images from https://machinethink.net/blog/object-detection/]

- ***Grid detectors: one per anchor***

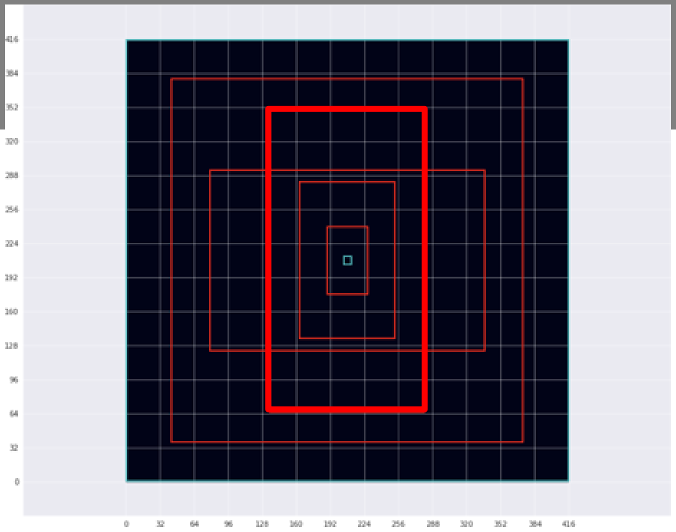


Anchor shapes and sizes
are pre-computed from the training set
(bounding box statistics)

Each detector produces:
- *class probabilities*
- *confidence score*

per its assigned anchor

[images from https://machinethink.net/blog/object-detection/]



- **Given anchor, cell and class**

$$\langle c_x, c_y, p_w, p_h \rangle$$

top-left cell coordinates — anchor sizes

- **Each detector produces**

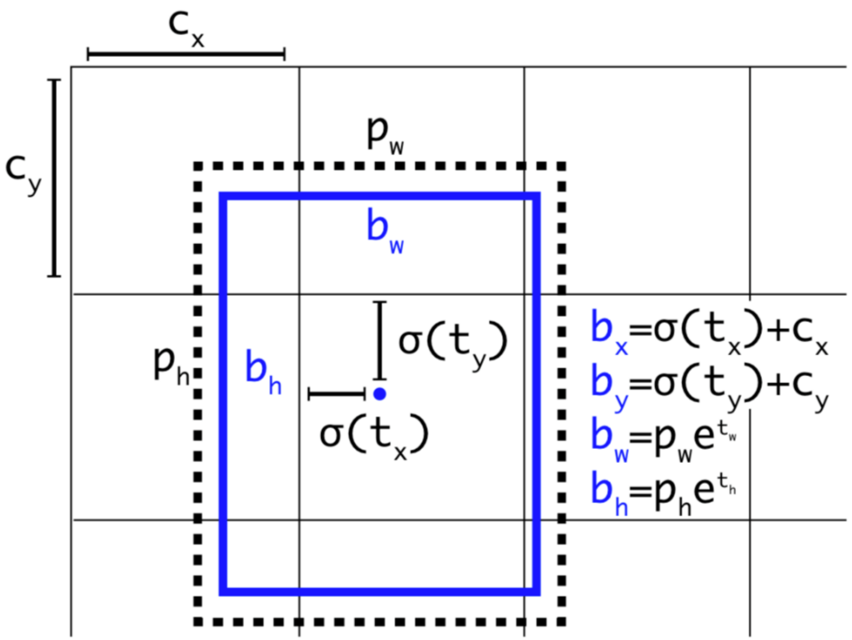
$$\langle t_x, t_y, t_w, t_h, p_o, p_c \rangle$$

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned}$$

bounding box coordinates

p_o 'objectness' probability

p_c class probability

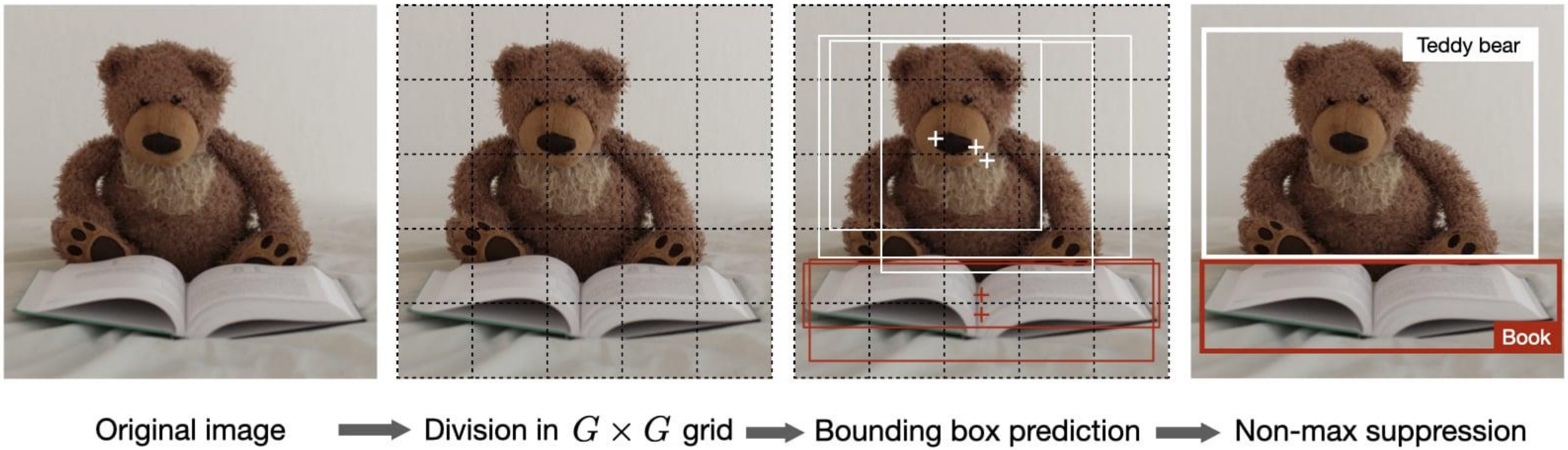


[images from https://wikidocs.net/167697]

Object detection and positioning

- ***From grid boxes to candidate boxes***

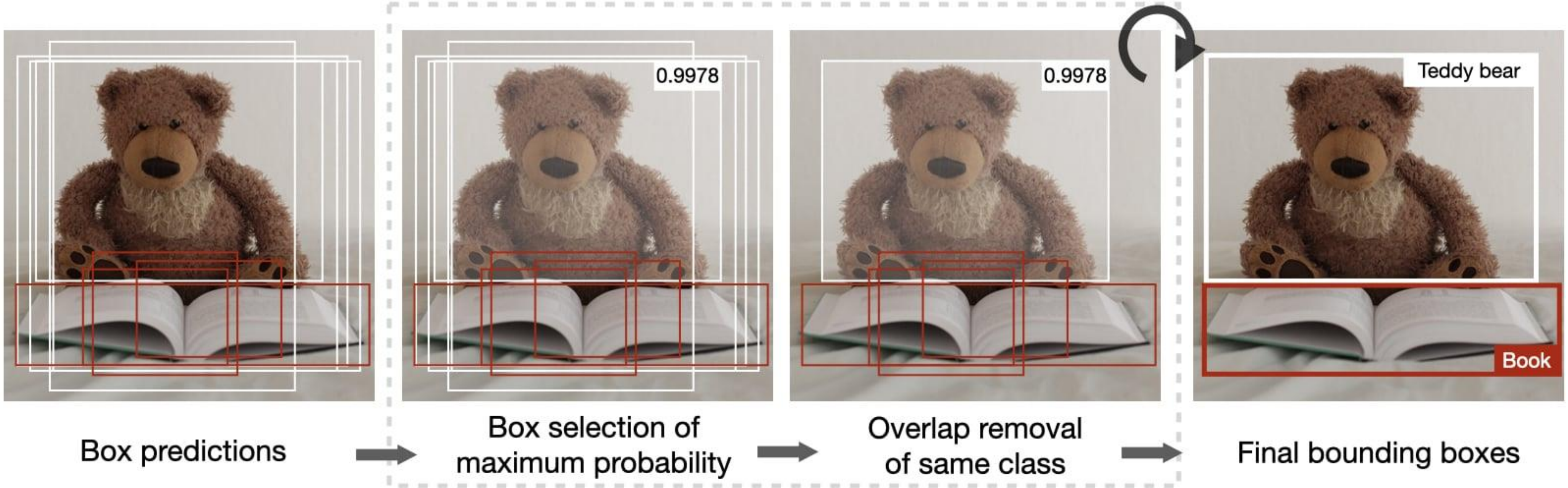
  Merging predictions



[images from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks]

Object detection and positioning

- ***Further processing***



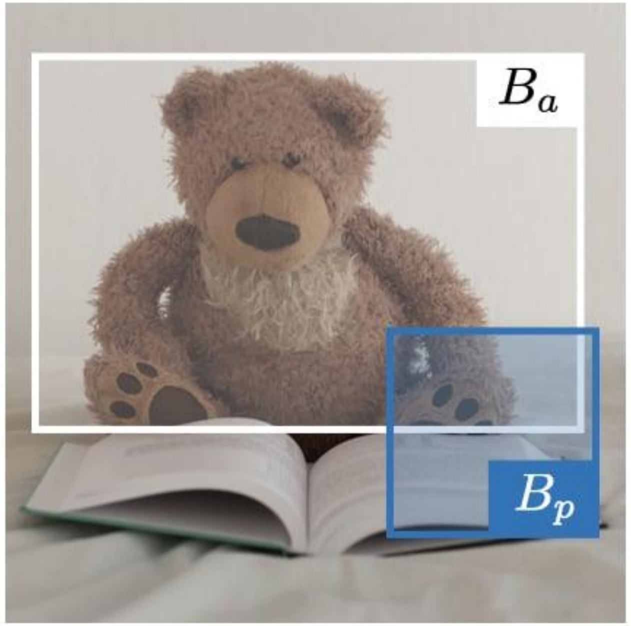
[images from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks]

- ***Measuring object detection accuracy***

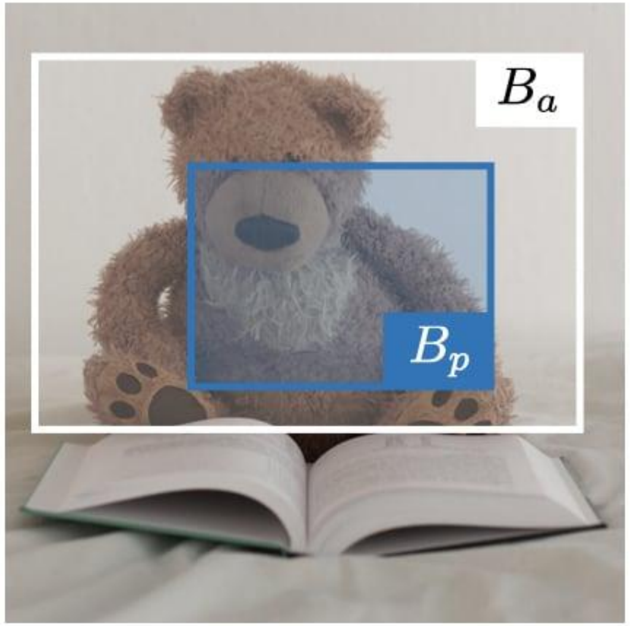
  Intersection over Union (IoU)

$$\mathrm{IoU}(B_p, B_a) := \frac{B_p \cap B_a}{B_p \cup B_a}$$

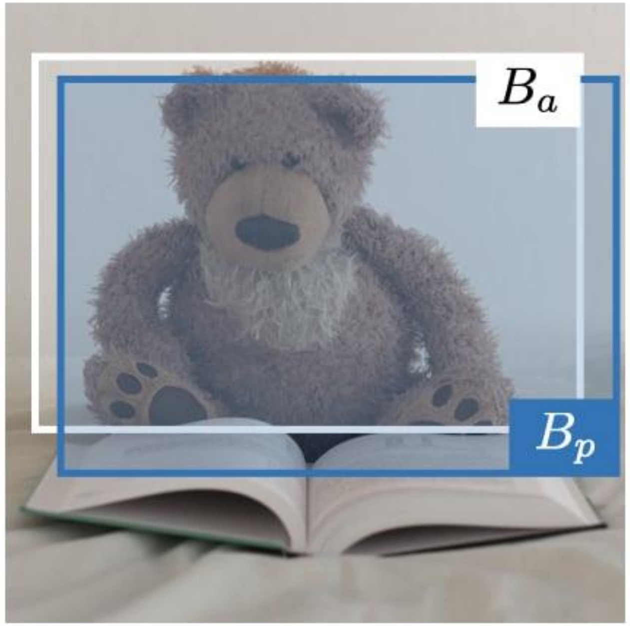
It's a post-localization accuracy measure (not a loss function)



$\mathrm{IoU}(B_p, B_a) = 0.1$



$\mathrm{IoU}(B_p, B_a) = 0.5$



$\mathrm{IoU}(B_p, B_a) = 0.9$

[images from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks]