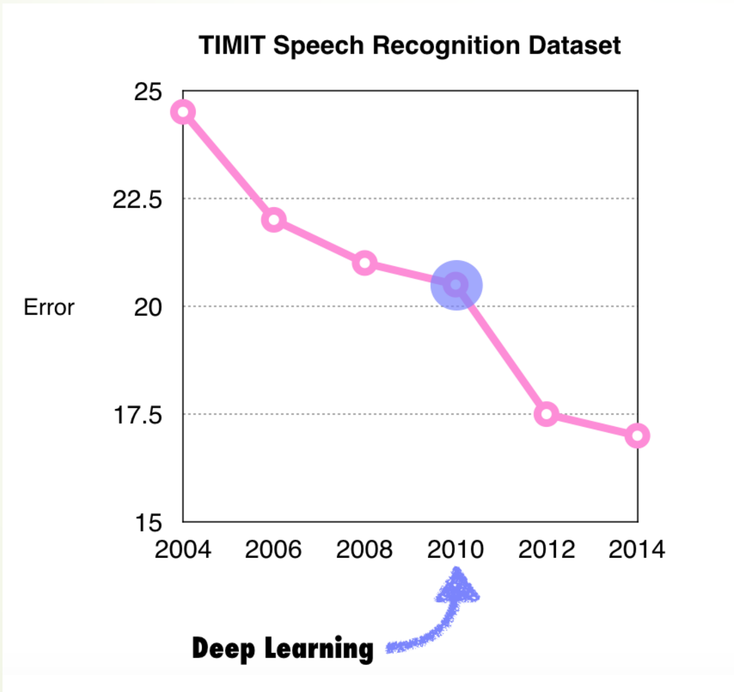
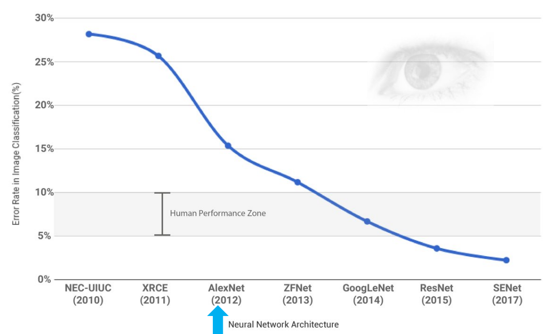
Around the year of 2012...

Speech Recognition: TIMIT



Computer Vision: ImageNet

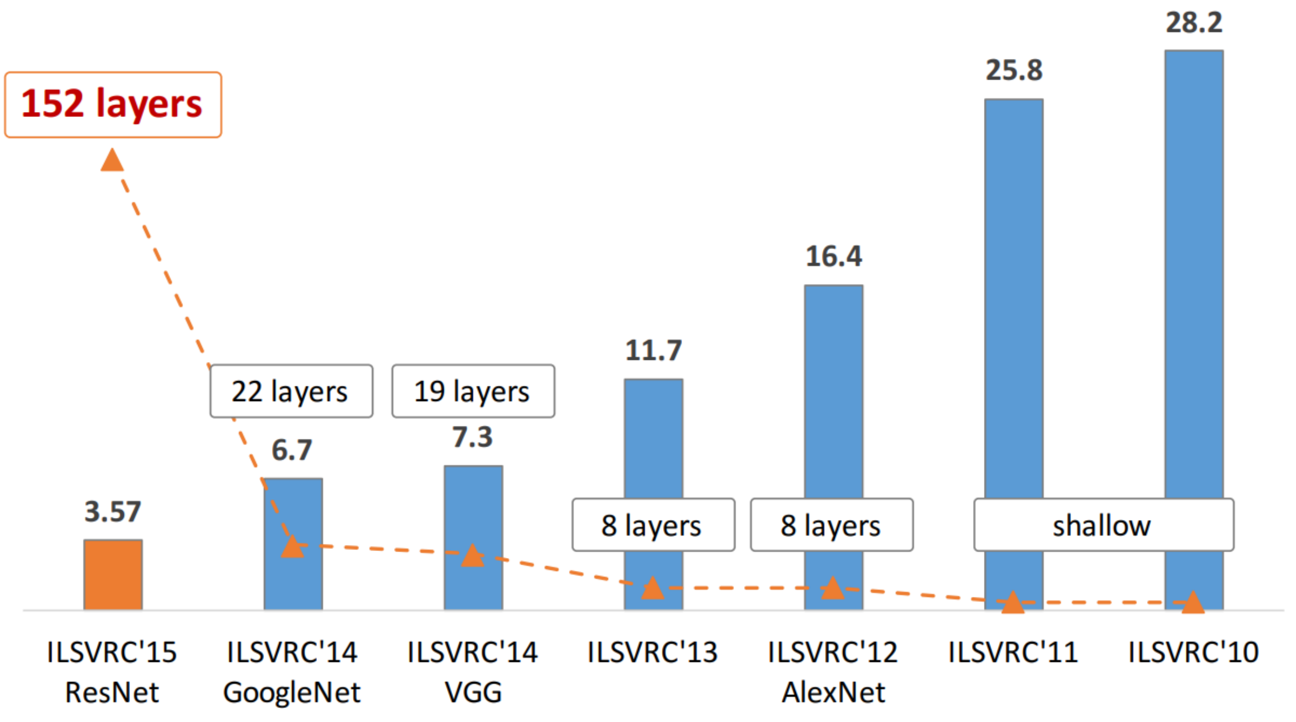
- ImageNet (subset):
  - 1.2 million training images
  - 100,000 test images
  - 1000 classes
- ImageNet large-scale visual recognition Challenge



source: https://www.linkedin.com/pulse/must-read-path-breaking-papers-image-classification-muktabh-mayank

Deep Learning

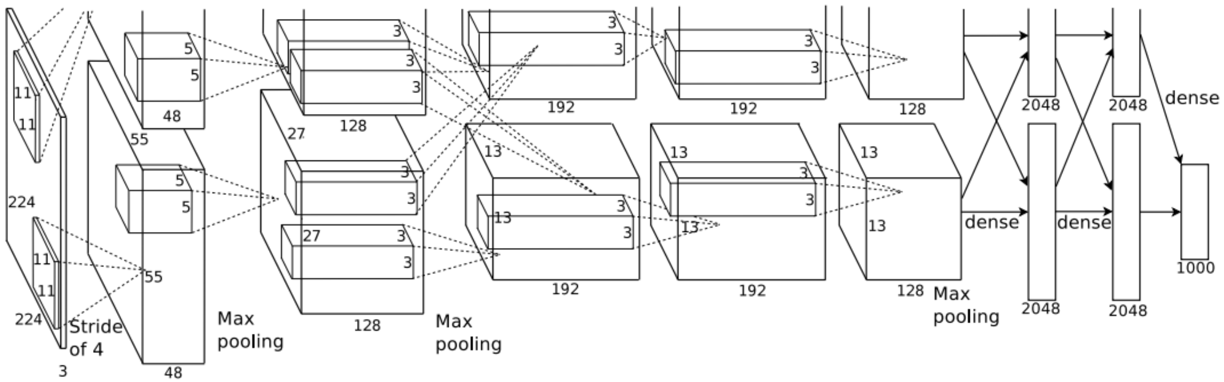
Depth as function of year



[He et al., 2016]

AlexNet (2012): Architecture

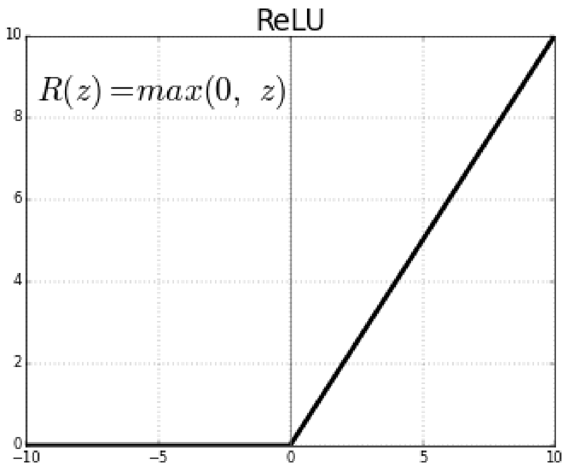
- 8 layers: first 5 convolutional, rest fully connected
- ReLU nonlinearity
- Local response normalization
- Max-pooling
- Dropout



Source: [Krizhevsky et al., 2012]

AlexNet (2012): ReLU

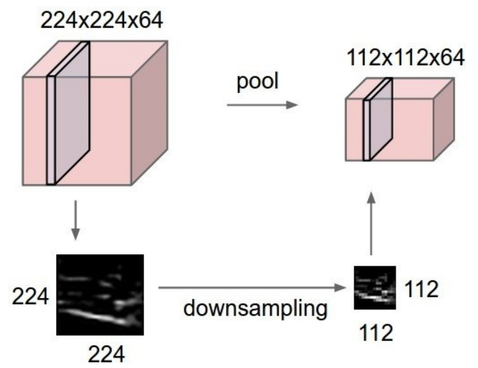
- Non-saturating function and therefore faster convergence when compared to other nonlinearities
- Problem of dying neurons

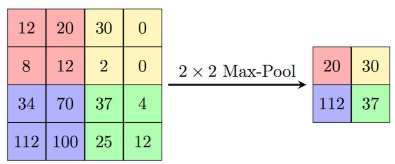


Source: `https://ml4a.github.io/ml4a/neural_networks/`

AlexNet (2012): Max Pooling

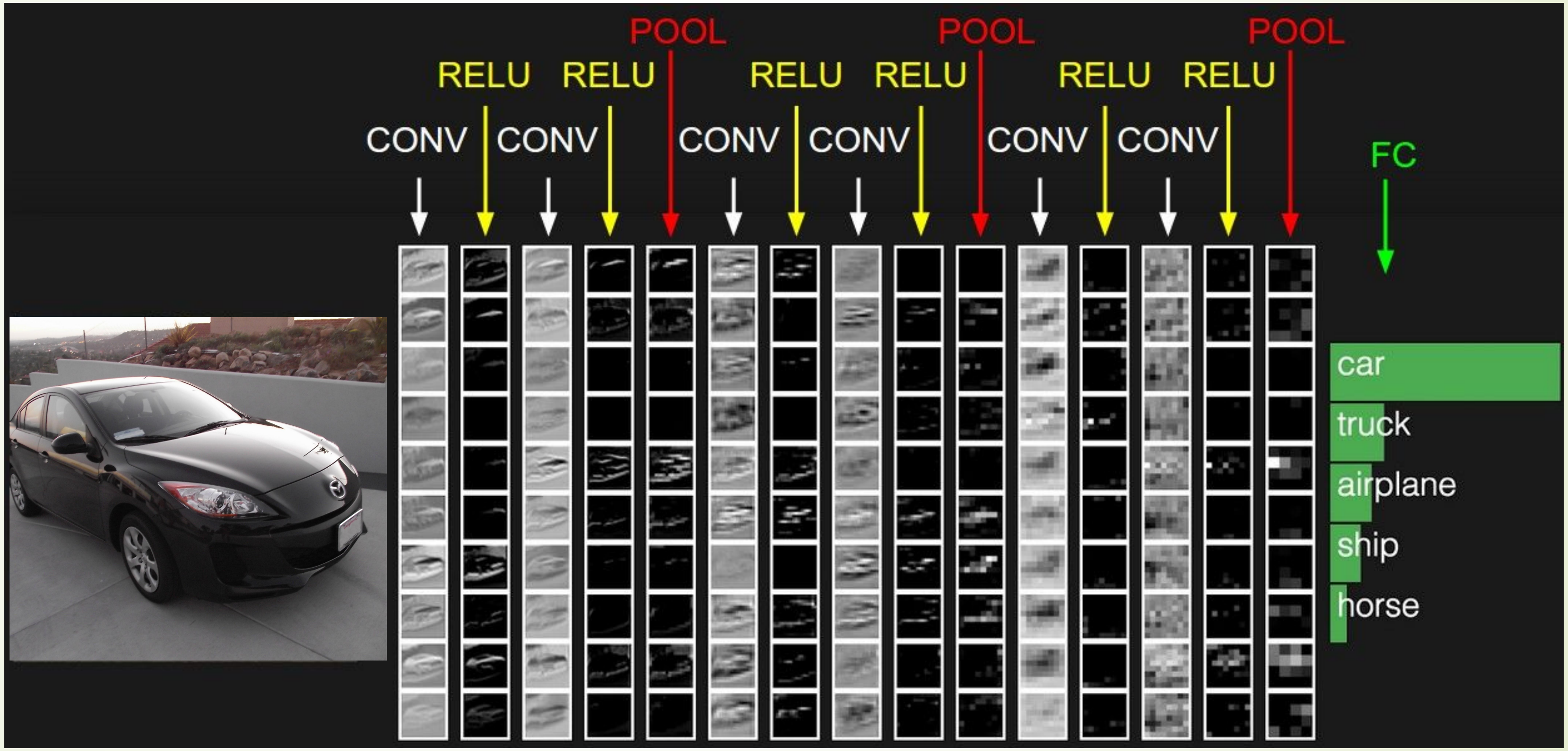
- Chooses maximal entry in every non-overlapping window of size 2×2, for example



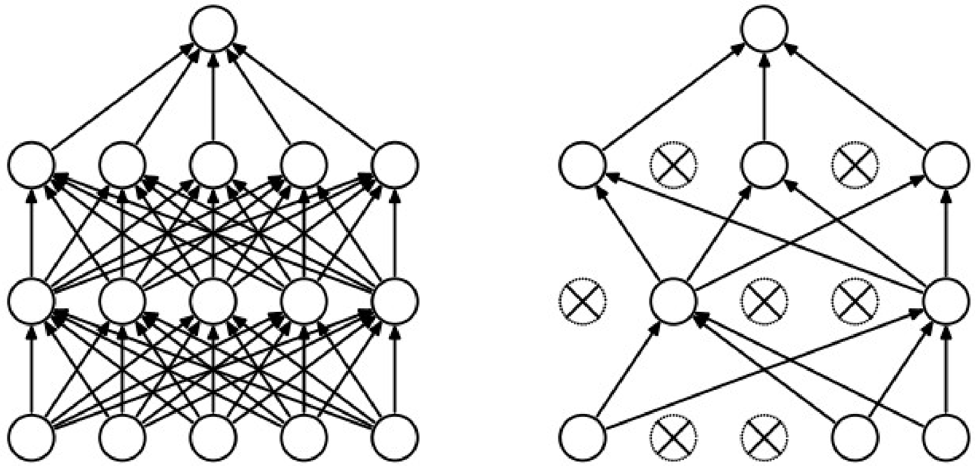


Source: Stanford's CS231n github

CNN for Classification



AlexNet (2012): Dropout

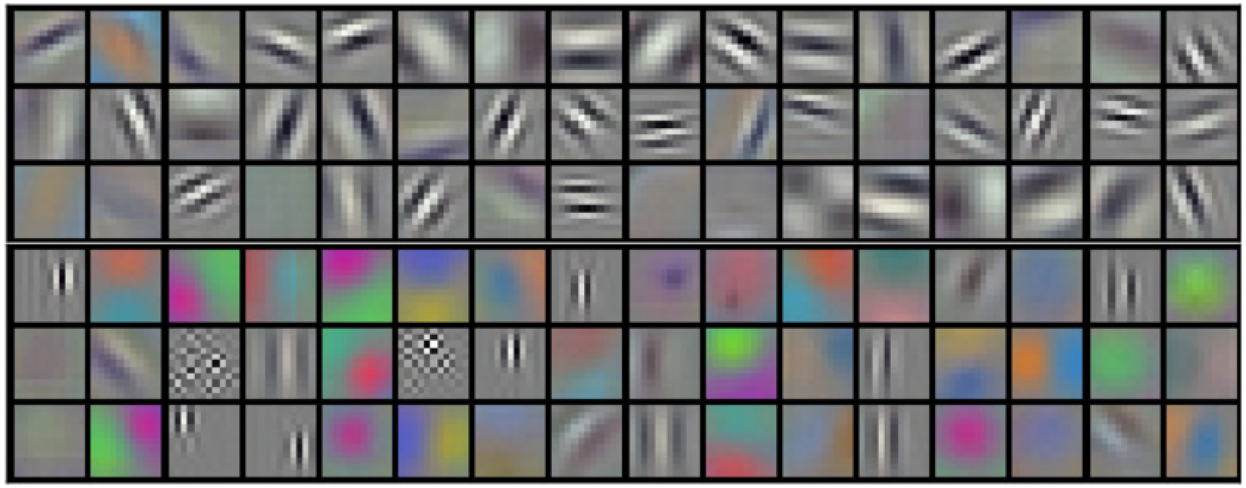


Source: [Srivastava et al., 2014]

- Zero every neuron with probability $1 - p$
- At test time, multiply every neuron by p

AlexNet (2012): Training

- Stochastic gradient descent
- Mini-batches
- Momentum
- Weight decay (ℓ_2 prior on the weights)

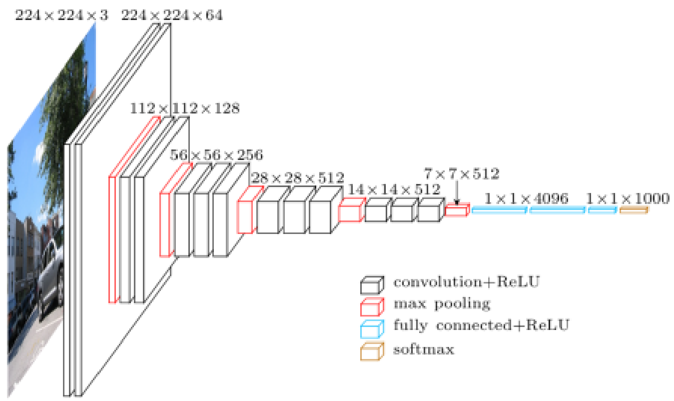


Filters trained in the first layer

Source: [Krizhevsky et al., 2012]

VGG (2014) [Simonyan-Zisserman'14]

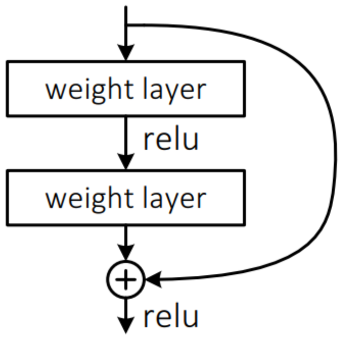
- Deeper than AlexNet: 11-19 layers versus 8
- No local response normalization
- Number of filters multiplied by two every few layers
- Spatial extent of filters 3×3 in all layers
- Instead of 7×7 filters, use three layers of 3×3 filters
  - Gain intermediate nonlinearity
  - Impose a regularization on the 7×7 filters

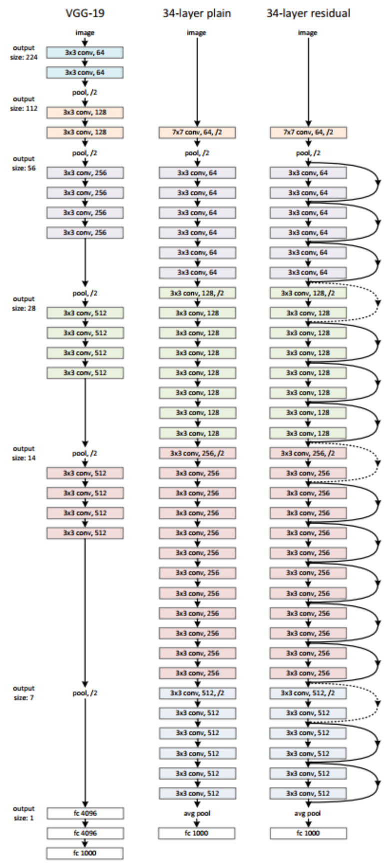


Source: `https://blog.heuritech.com/2016/02/29/`

ResNet (2015) [HGRS-15]

- Solves problem by adding skip connections
- Very deep: 152 layers
- No dropout
- Stride
- Batch normalization

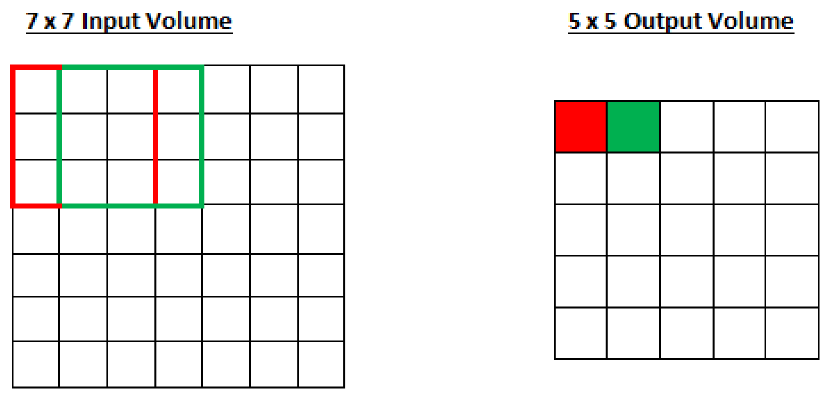




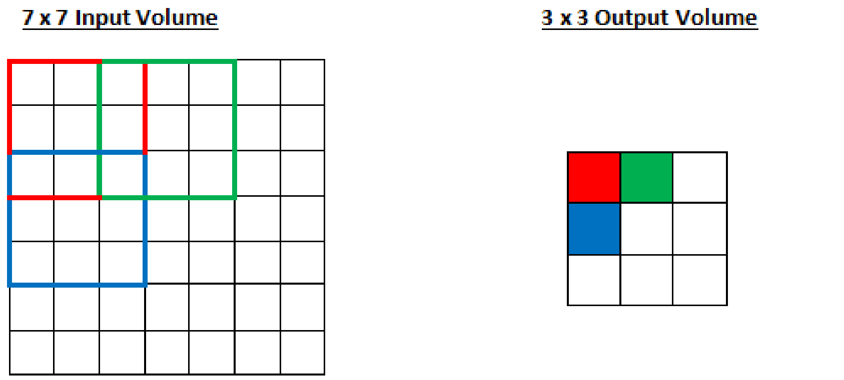
Source: Deep Residual Learning for Image Recognition

Stride

Stride 1



Stride 2



Batch Normalization

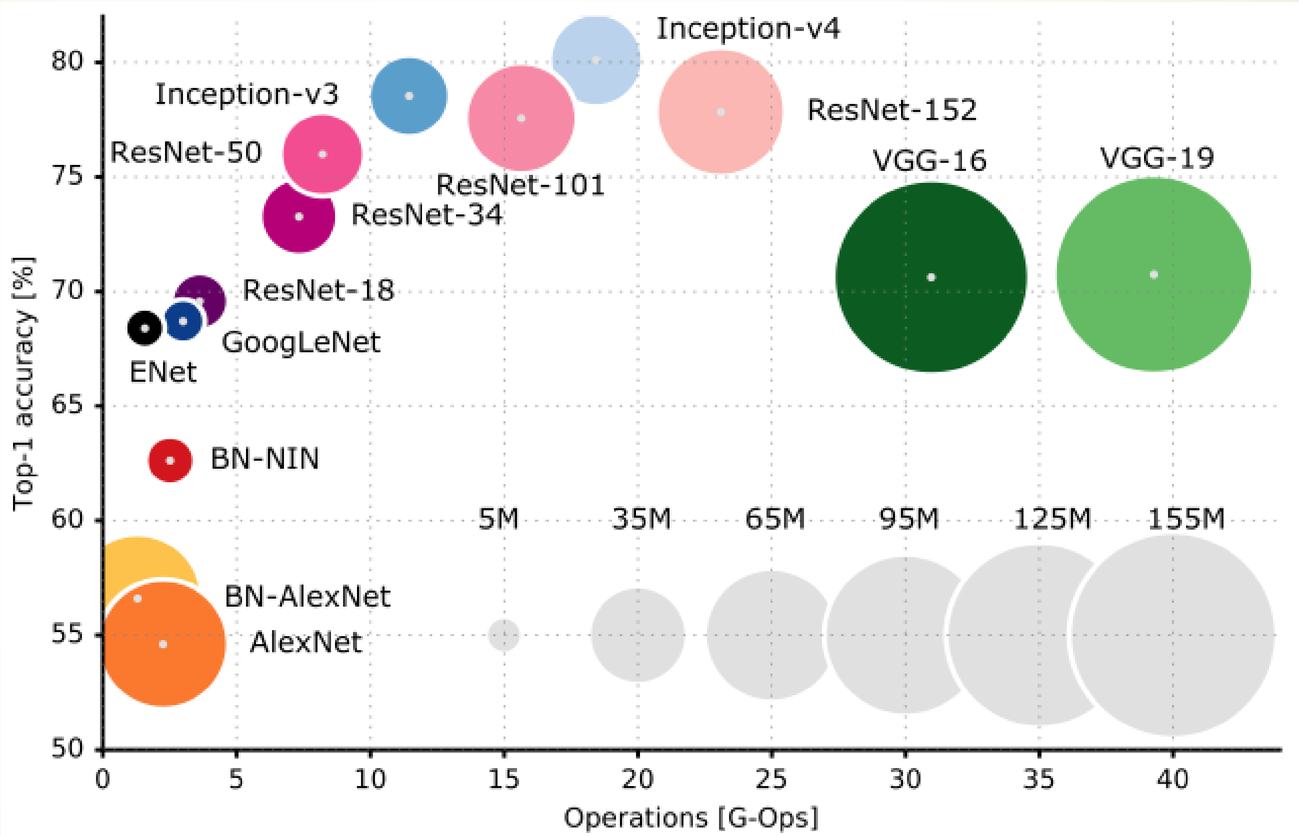
Algorithm 2 Batch normalization [Ioffe and Szegedy, 2015]

Input: Values of x over minibatch $x_1 \dots x_B$, where x is a certain channel in a certain feature vector
Output: Normalized, scaled and shifted values $y_1 \dots y_B$

1: $\mu = \frac{1}{B} \sum_{b=1}^{B} x_b$
2: $\sigma^2 = \frac{1}{B} \sum_{b=1}^{B} (x_b - \mu)^2$
3: $\hat{x}_b = \frac{x_b - \mu}{\sqrt{\sigma^2 + \epsilon}}$
4: $y_b = \gamma \hat{x}_b + \beta$

- Accelerates training and makes initialization less sensitive
- Zero mean and unit variance feature vectors

Complexity vs. Accuracy of Different Networks



Deep Learning Softwares

- Pytorch (developed by Yann LeCun and Facebook):
  - http://pytorch.org/tutorials/
- Tensorflow (developed by Google based on Caffe)
  - https://www.tensorflow.org/tutorials/
- Theano (developed by Yoshua Bengio)
  - http://deeplearning.net/software/theano/tutorial/
- Keras (based on Tensorflow or Pytorch)
  - https://www.manning.com/books/deep-learning-with-python?a_aid=keras&a_bid=76564dff

Visualizing NN and Transfer Learning

Visualizing Deep Neural Networks

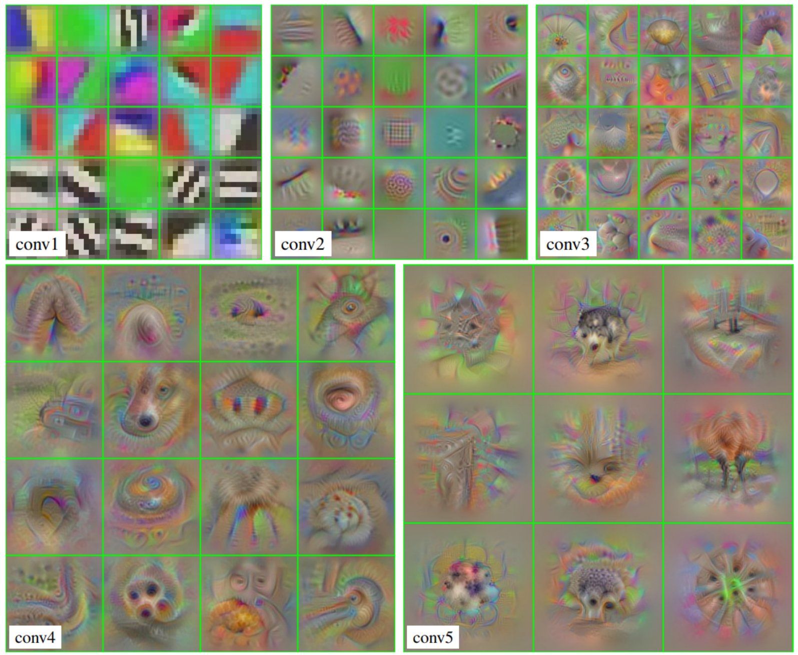
- Filters in first layer of CNN are easy to visualize, while deeper ones are harder
- *Activation maximization* seeks input image maximizing output of the i-th neuron in the network
- Objective

$$x^* = \arg\min_x \mathcal{R}(x) - \langle \Phi(x), e_i \rangle$$

- e_i is indicator vector
- $\mathcal{R}(x)$ is simple natural image prior

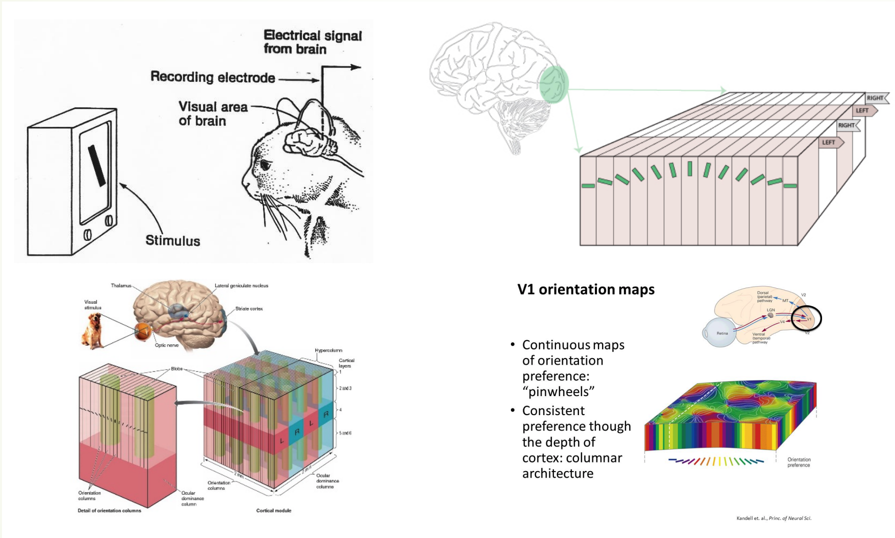
Visualizing VGG

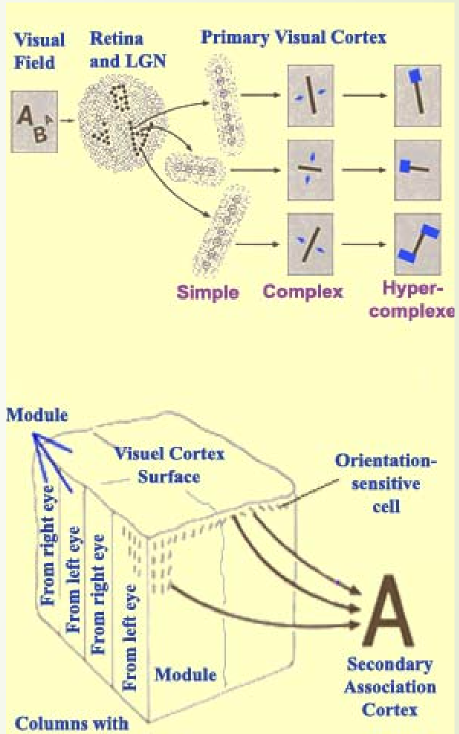
- Gabor-like images in first layer
- More sophisticated structures in the rest



[Mahendran and Vedaldi, 2016]

Visual Neuroscience: Hubel/Wiesel, …





Olshausen and Field 1996

Experimental Neuroscience uncovered the

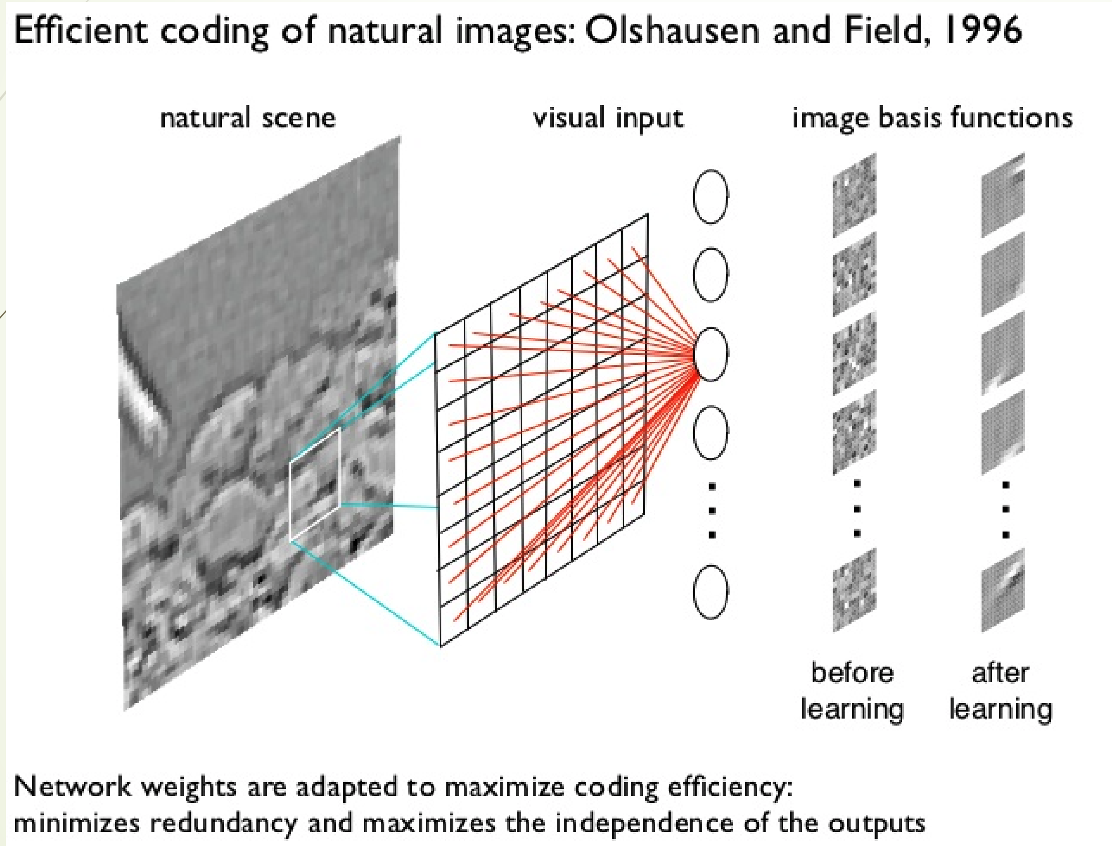
- ... neural architecture of Retina/LGN/V1/V2/V3/ etc
- ... existence of neurons with weights and activation functions (simple cells)
- ... pooling neurons (complex cells)

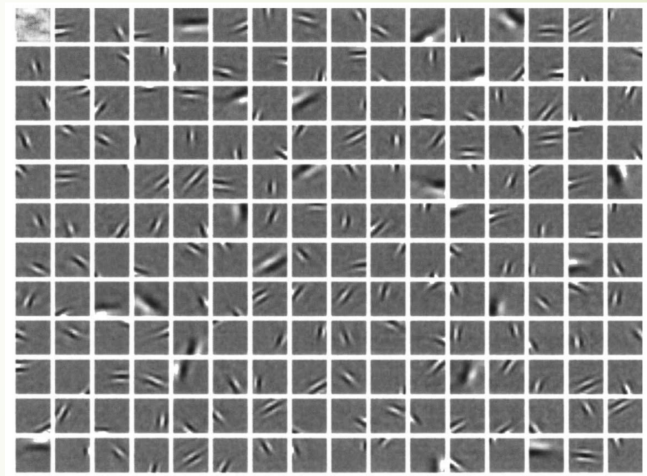
All these features are somehow present in today's sucessful Deep Learning systems

Neuroscience	Deep Network
Simple cells	First layer
Complex celle	Pooling Layer
Grandmother cells	Last layer

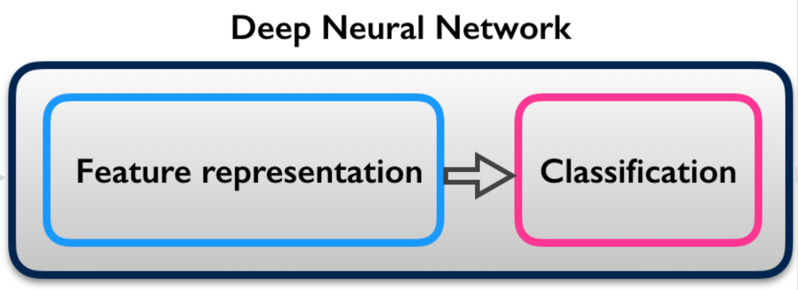
Theorists Olshausen and Field (Nature, 1996) demonstrated that receptive fields learned from image patches

First layers learned …





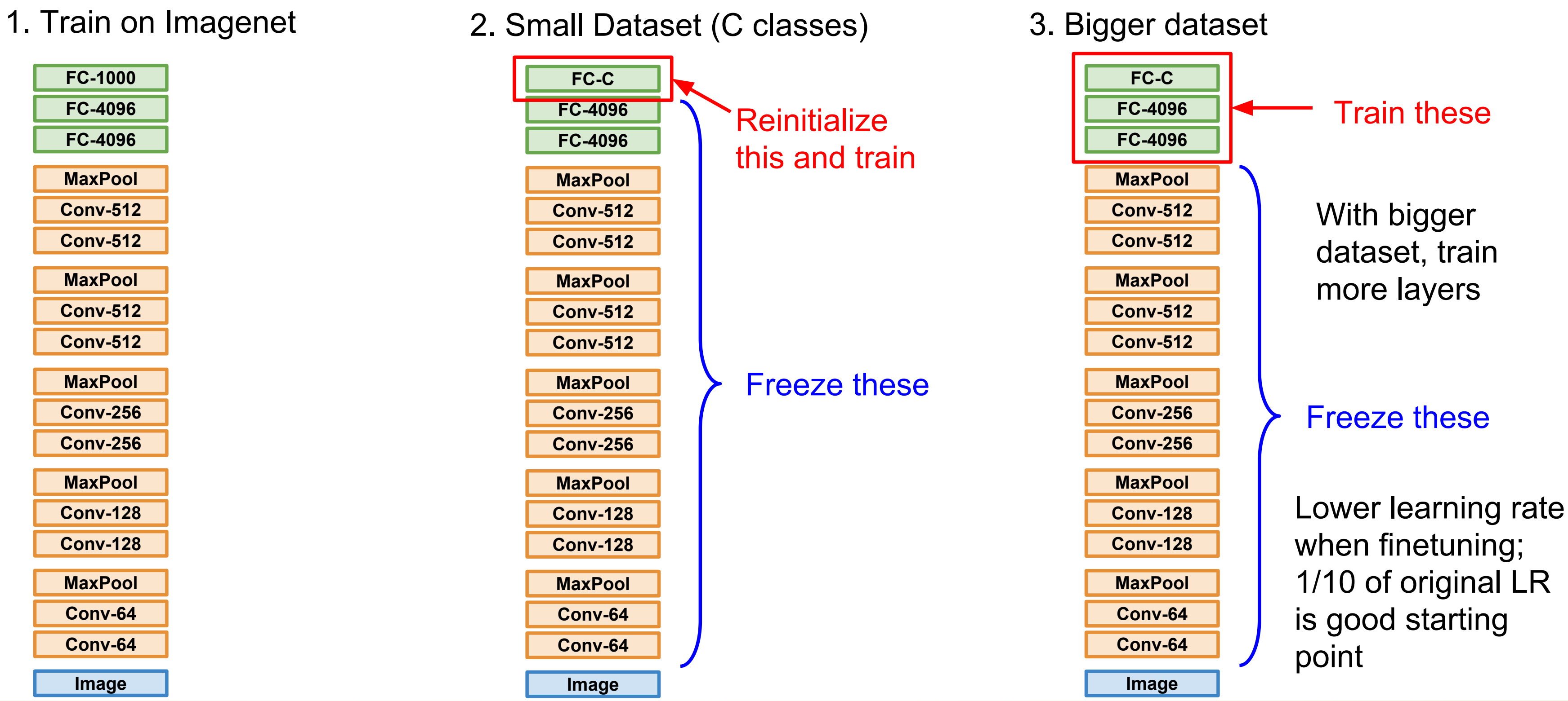
Transfer Learning?

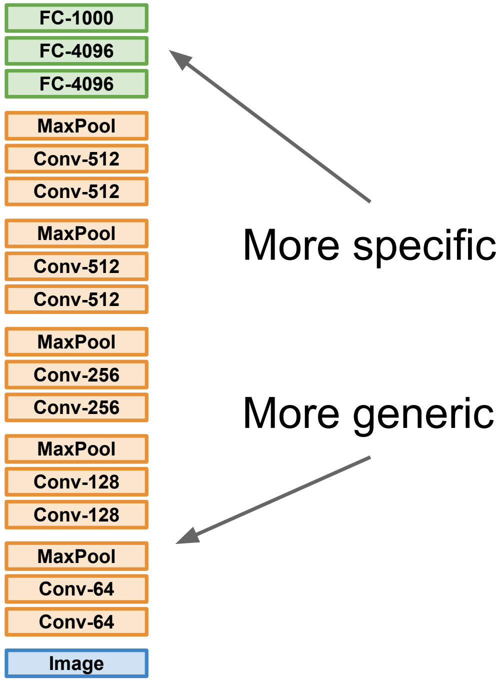


- Filters learned in first layers of a network are transferable from one task to another
- When solving another problem, no need to retrain the lower layers, just fine tune upper ones
- Is this simply due to the large amount of images in ImageNet?
- Does solving many classification problems simultaneously result in features that are more easily transferable?
- Does this imply filters can be learned in unsupervised manner?
- Can we characterize filters mathematically?

Transfer Learning with CNNs

Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014





	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You’re in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Takeaway for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

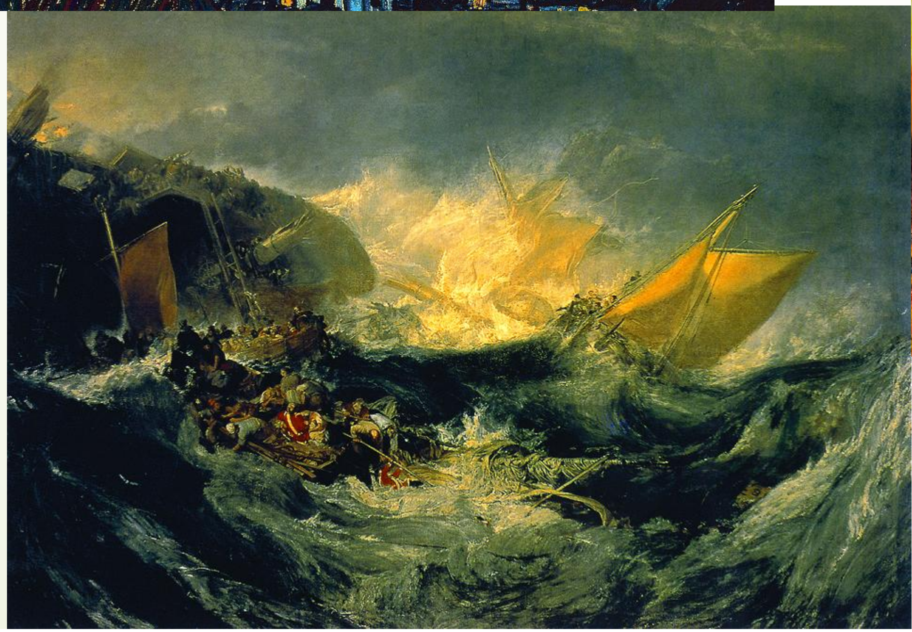
Deep learning frameworks provide a "Model Zoo" of pretrained models so you don't need to train your own

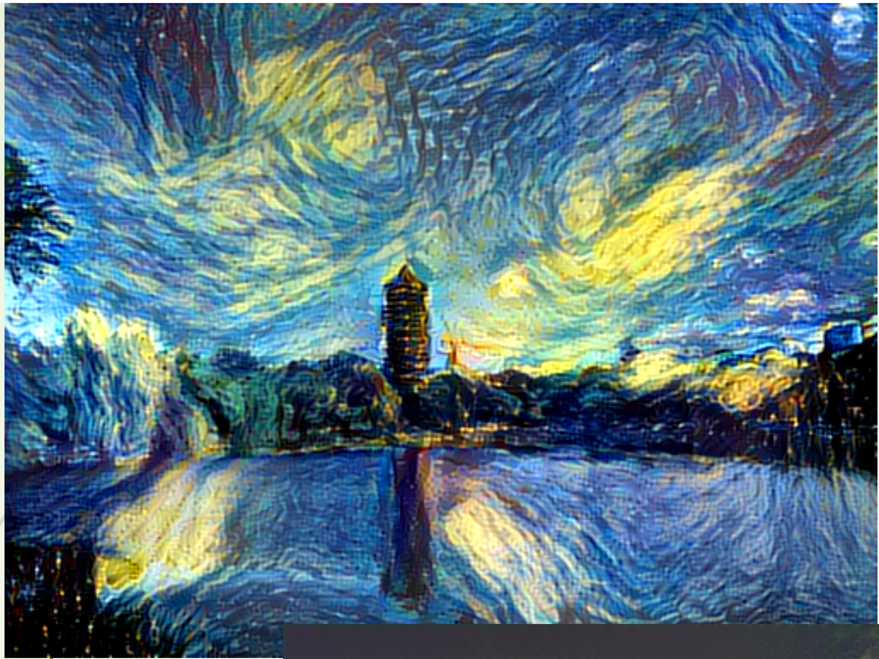
Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

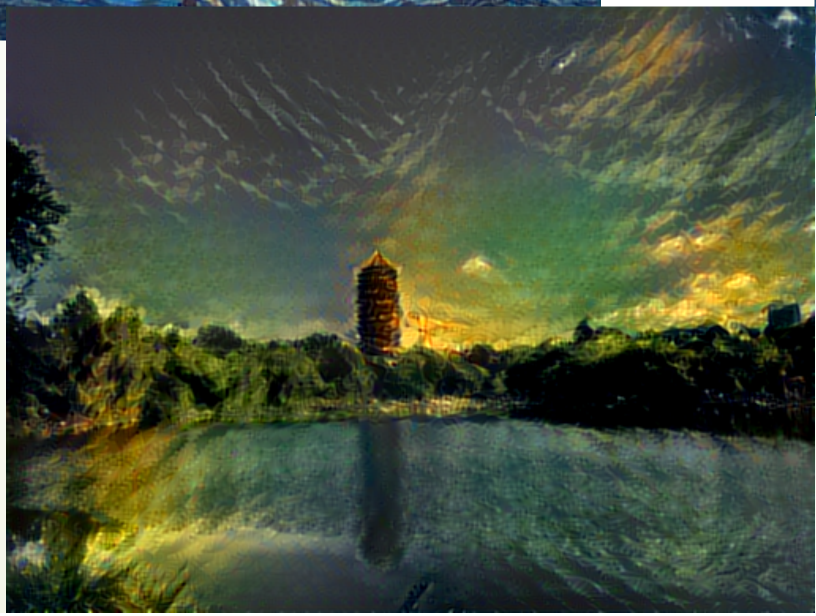
Style-Content Features

Example: The Noname Lake in PKU

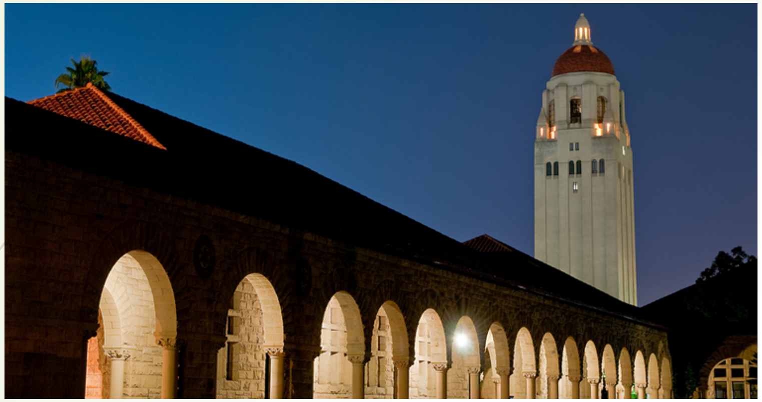
Left: *Vincent Van Gogh*, Starry Night
Right: *Claude Monet*, Twilight Venice
Bottom: *William Turner*, Ship Wreck







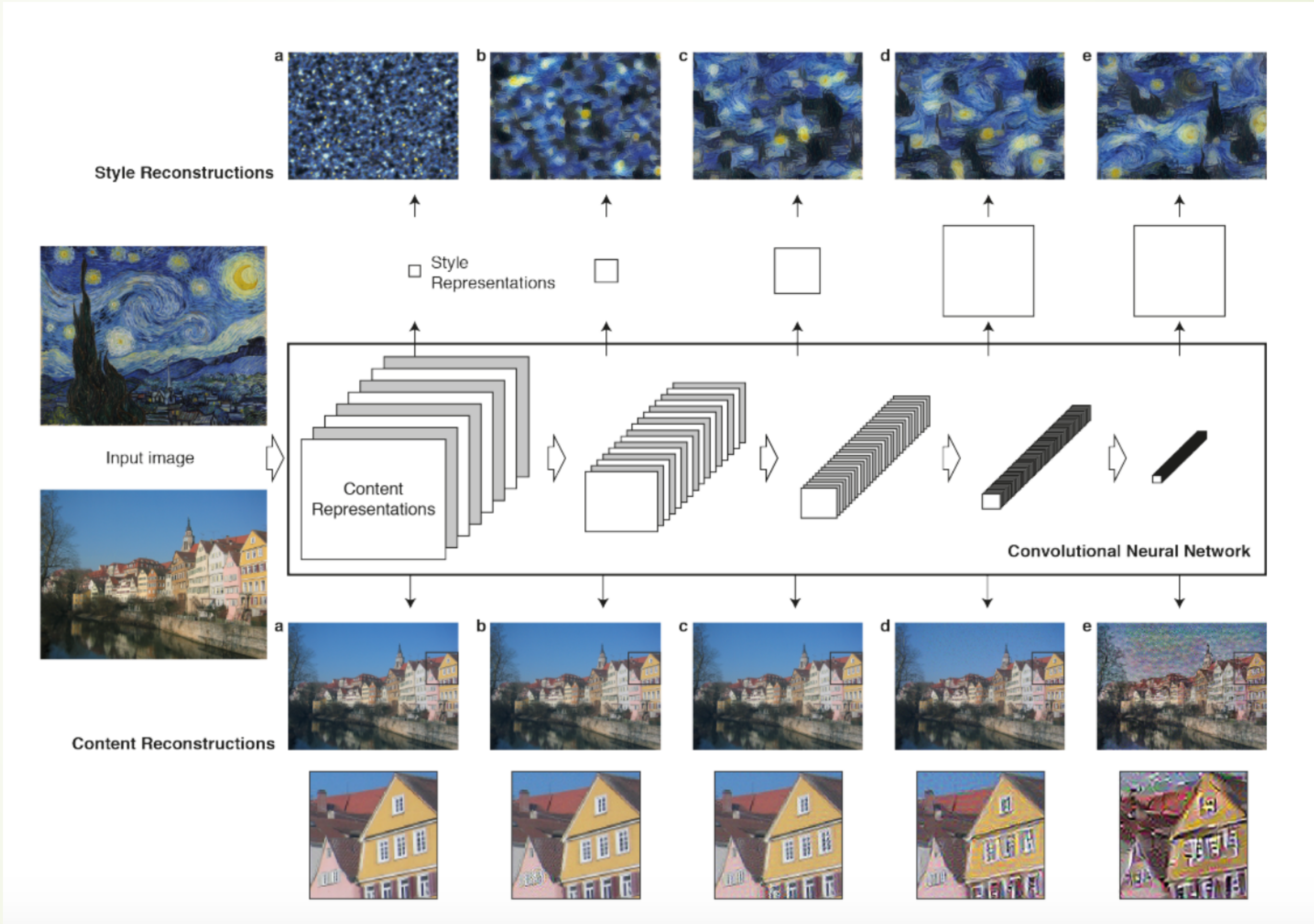
Application of Deep Learning:
Content-Style synthetic
pictures
By "neural-style"



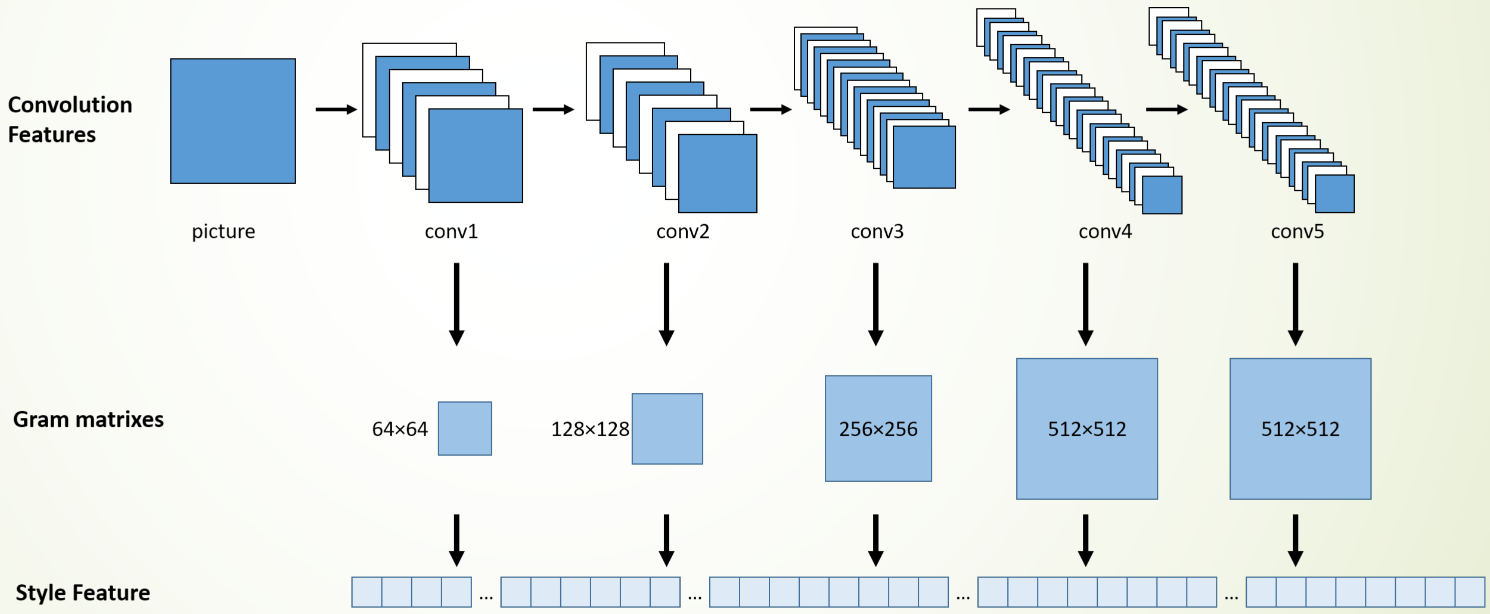

Neural Style

- J C Johnson's Website: https://github.com/jcjohnson/neural-style
- A torch implementation of the paper
  - *A Neural Algorithm of Artistic Style*,
  - by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge.
  - http://arxiv.org/abs/1508.06576

Style-Content Feature Extraction



Style Features as Second Order Statistics



Loss for Content (1^{st} order statistics) and Style (2^{nd} order statistics of outputs)

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2 \; .$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij}\right)^2$$

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}.$$

Thank you!