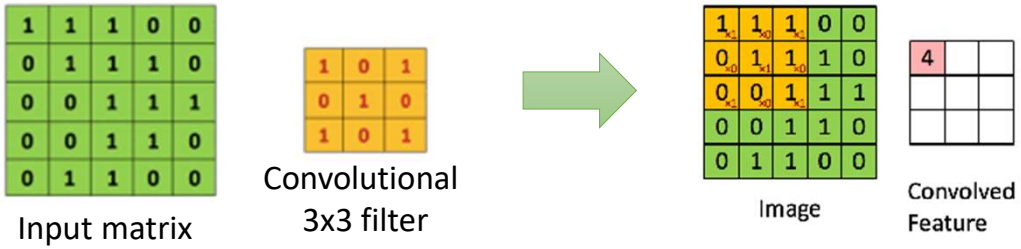
CNN Architectures

- Alexnet
- VGG
- Inception
- Resnet

Convolutional Neural Networks (CNNs)

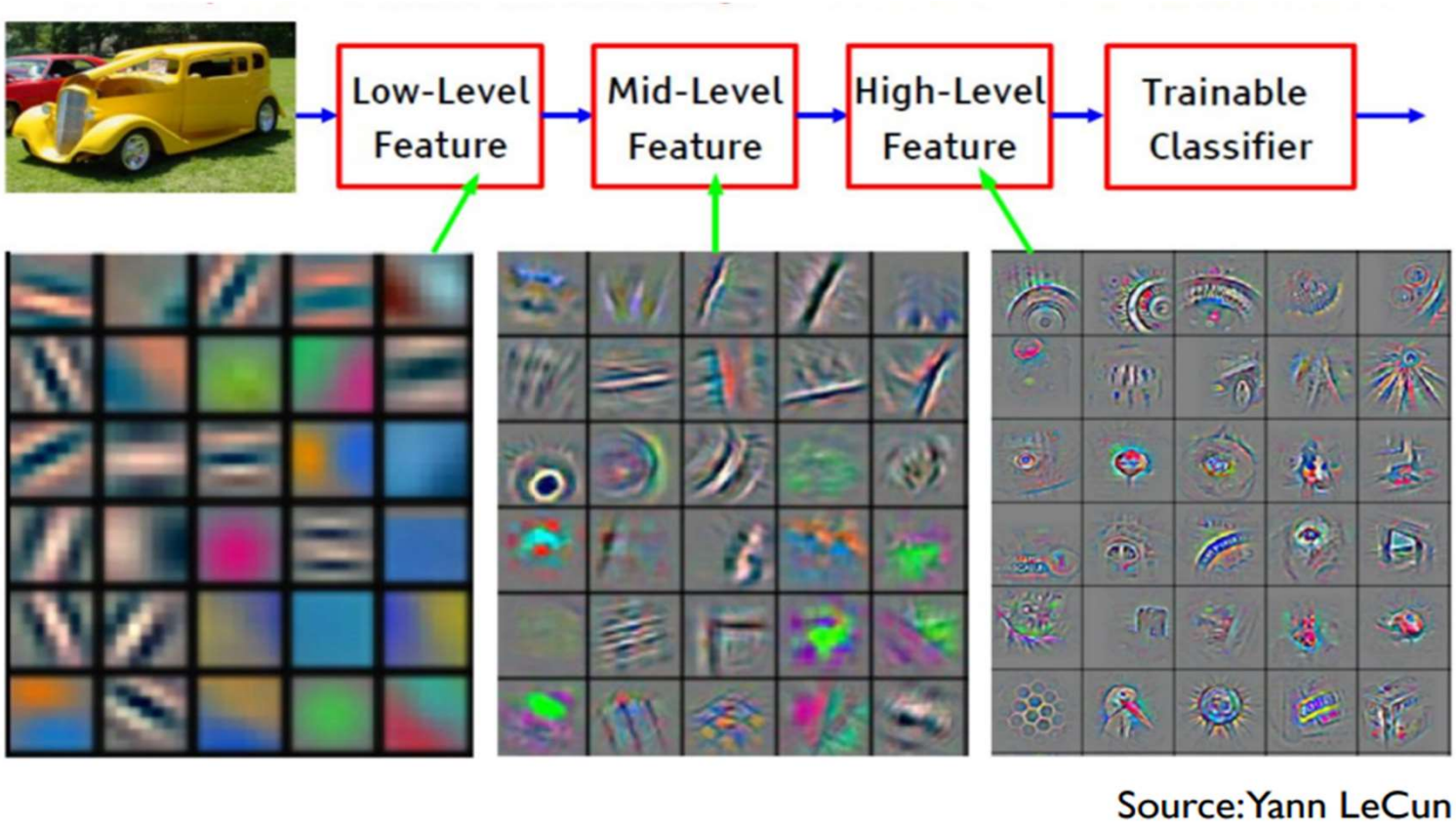
- CNNs were primarily designed for image data
- CNNs use a convolutional operator for extracting data features
  - Allows parameter sharing
  - Efficient to train
  - Have less parameters than NNs with fully-connected layers
- CNNs are robust to spatial translations of objects in images
- A convolutional filter slides (i.e., convolves) across the image



Picture from: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Deep Learnt Features

- It's deep if it has more than one stage of non-linear feature transformation



ImageNet Competition

- **ImageNet** is formally a project aimed at (manually) labeling and categorizing images into almost 22,000 separate object categories for the purpose of computer vision research.

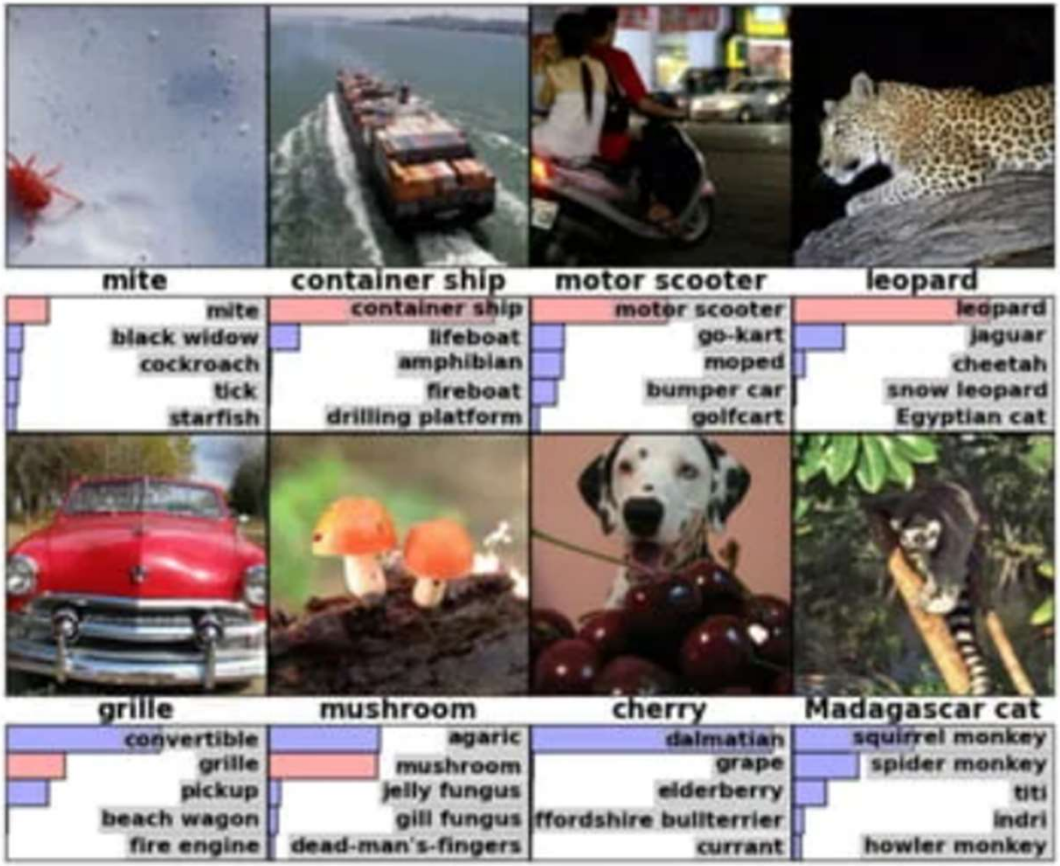
- *"ImageNet"* in the context of deep learning and Convolutional Neural Networks, likely refers to the ***ImageNet Large Scale Visual Recognition Challenge***, or ILSVRC

- The goal of this image classification challenge is to train a model that can correctly classify an input image into 1,000 separate object categories.

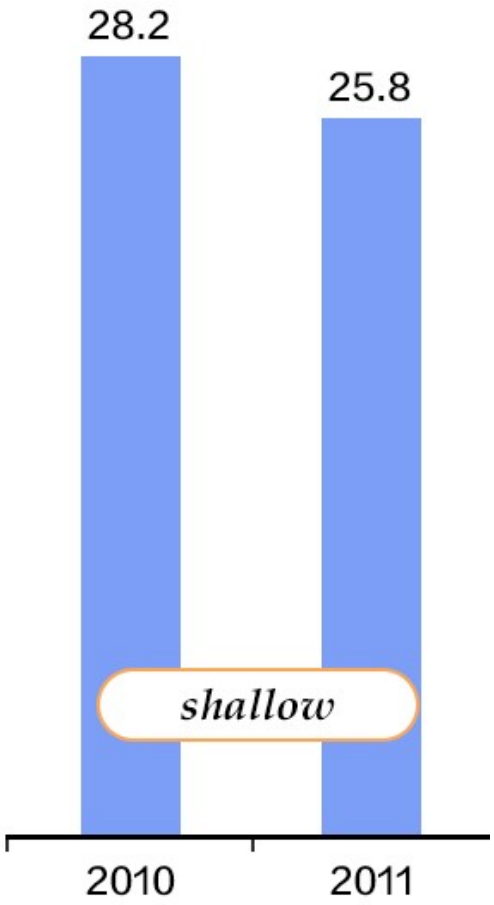
- Models are trained on ~1.2 million training images with another 50,000 images for validation and 100,000 images for testing.

ImageNet Challenge

These 1,000 image categories represent object classes that we encounter in our day-to-day lives, such as species of dogs, cats, various household objects, vehicle types, and much more.



Top-5 misclassification error rate



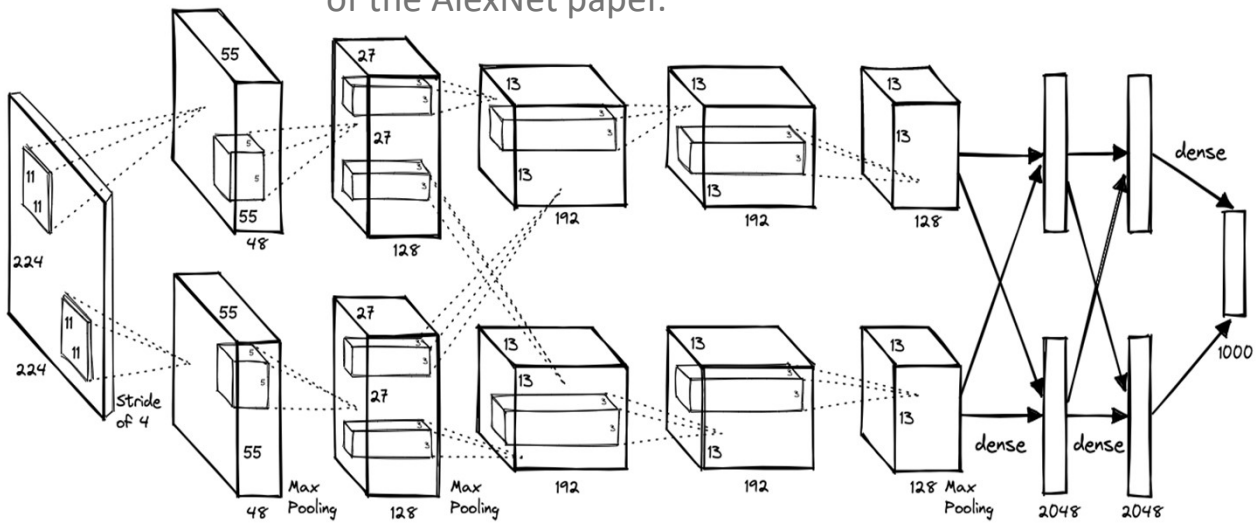
AlexNet

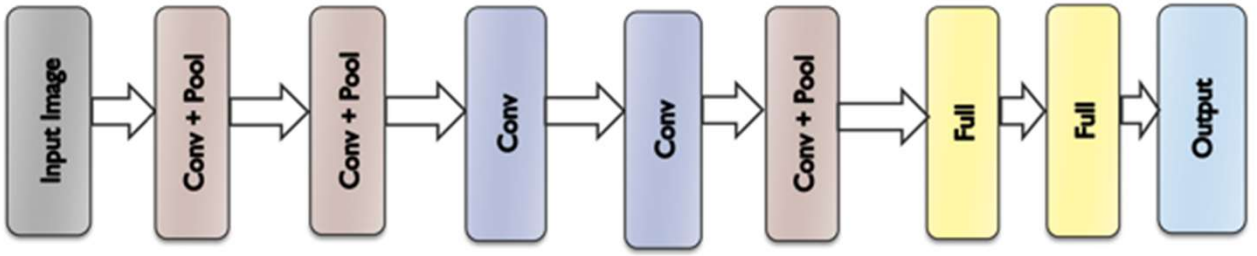
- Winner of ImageNet LSVRC-2010
- Was primarily designed by Alex Krizhevsky and was published with Ilya Sutskever and Krizhevsky's doctoral advisor Geoffrey Hinton [ImageNet Classification with Deep Convolutional Neural Networks]
- Was the first CNN which used GPU to boost performance (Optimized GPU implementation)
- Trained over 1.2M images using SGD with regularization
- Has a deep architecture (around 60 M parameters)

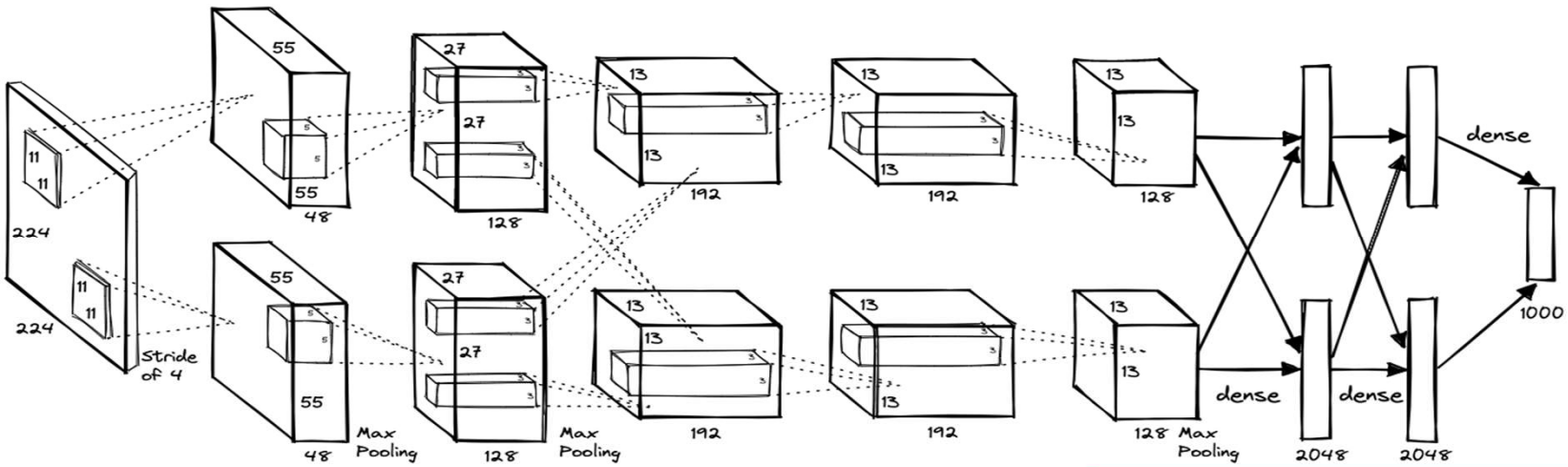
AlexNet's Architecture

- 5 convolutional layers, 3 max-pooling layers, 2 fully connected layers, and 1 softmax layer.
- ReLU used as an activation function
- Used SGD Momentum for optimization with batch size of 128
- The authors introduced overlap in pooling, and saw a reduction in error by about 0.5%

Image credits to Krizhevsky et al., the original authors of the AlexNet paper.





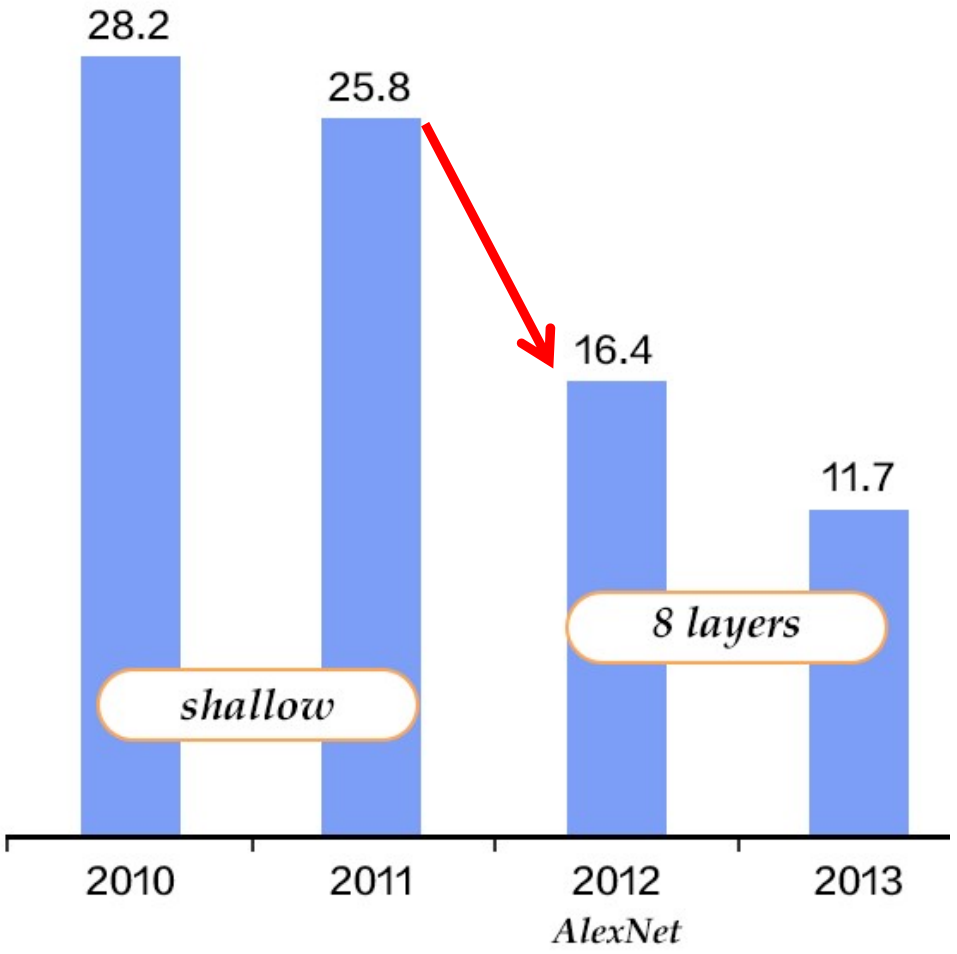


About 57 M parameters are in the fully connected layers

								Total
Parameters /Weights	$[(11 \times 11 \times 3)] \times 96 = 34848$	$2[5 \times 5 \times 48] \times 128 = 307200$	$[3 \times 3 \times 256] \times 384 = 884736$	$2[3 \times 3 \times 192] \times 192 = 663552$	$2[3 \times 3 \times 192] \times 128 = 442368$	37 M	16 M	4 M	60 M
Neurons:	$55 \times 55 \times 96 = 290400$	$27 \times 27 \times 256 = 186624$	$13 \times 13 \times 384 = 64896$	$13 \times 13 \times 384 = 64896$	$13 \times 13 \times 256 = 43264$	4096	4096	1000	0.63 M

- Convolutional layers cumulatively contain about 90-95% of computation, only about 5% of the parameters
- Fully-connected layers contain about 95% of the parameters

Top-5 misclassification error rate



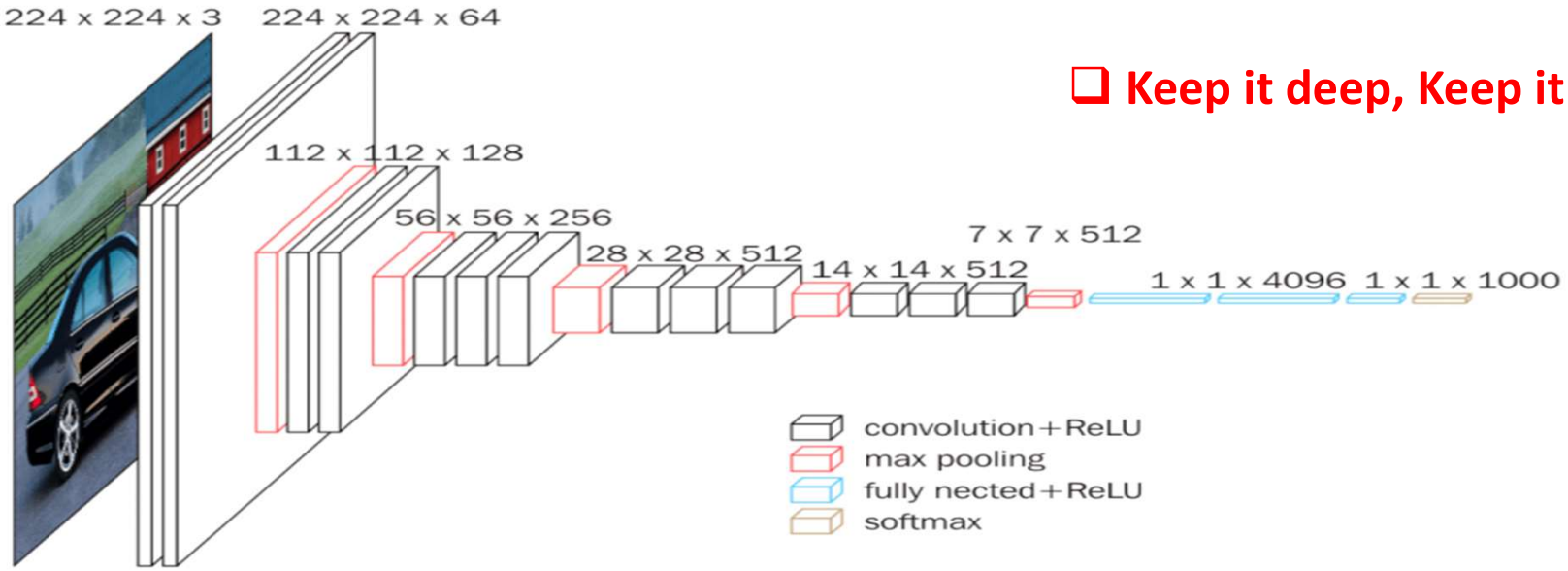
VGGNet – The Deep Convolutional Network

- VGGNet is a Deep Convolutional Neural Network that was proposed by Karen Simonyan and Andrew Zisserman of the University of Oxford in their research work ‘Very Deep Convolutional Neural Networks for Large-Scale Image Recognition’.

- The name of this model was inspired by the name of their research group ‘Visual Geometry Group (VGG)’.

- This convolutional neural network has 19 or 16 layers in its architecture; therefore, it is also named VGG-19/ VGG-16.

Image Source: https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide



- **Keep it deep, Keep it simple**

Going Deeper with Convolutions

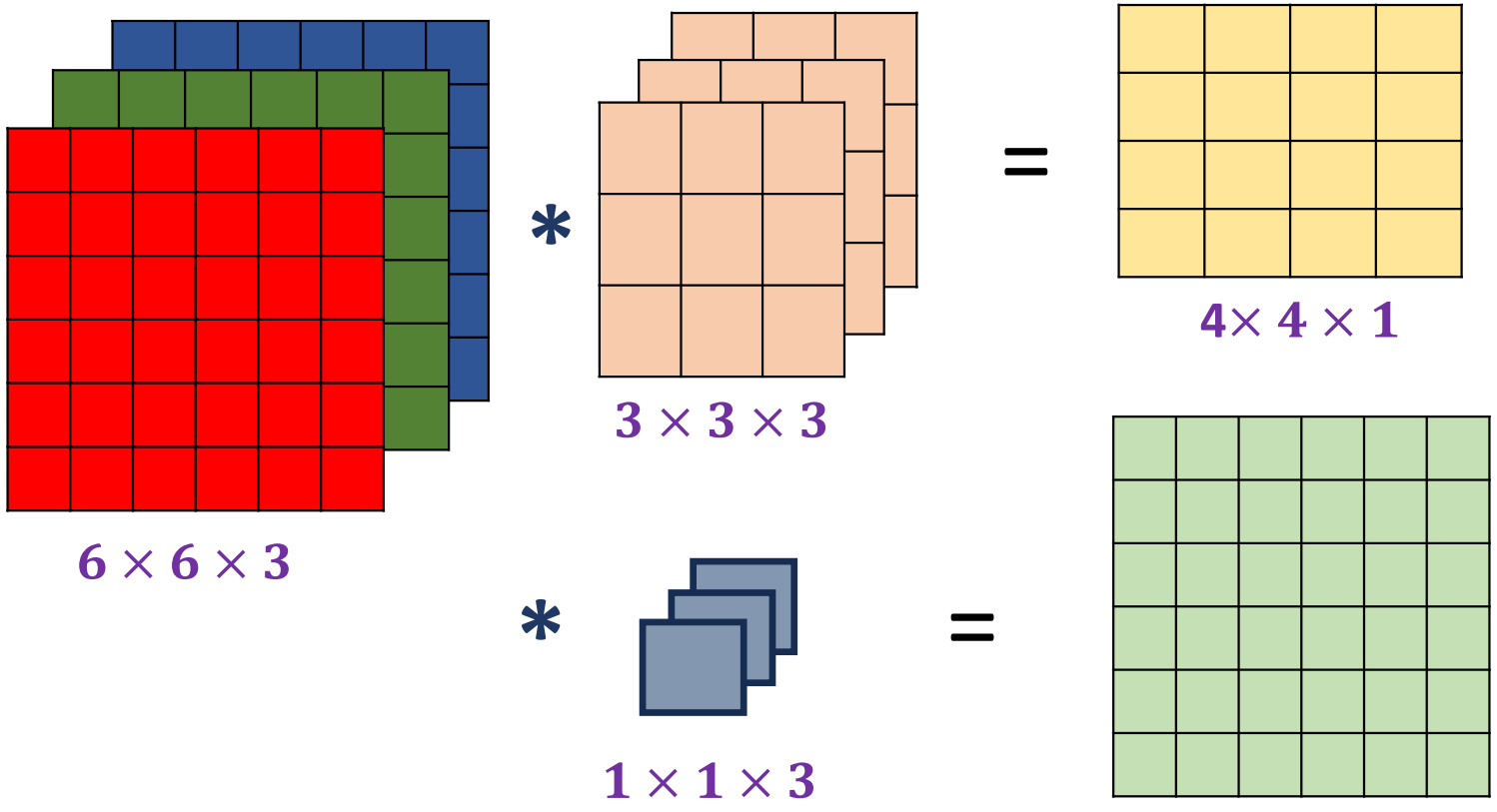
- Salient parts in the image can have extremely large variation in size.



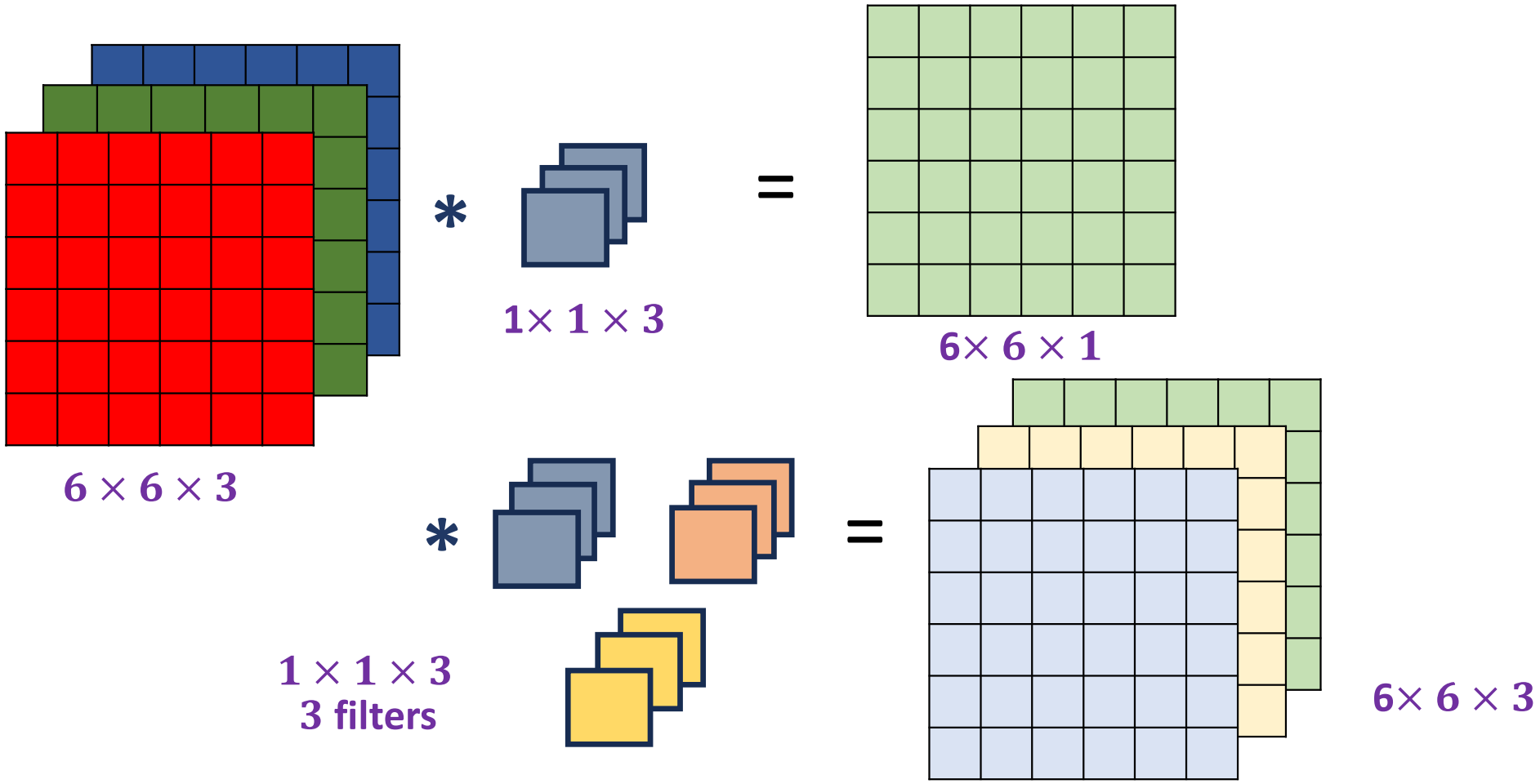
The area occupied by the cat is different in different images

- Choosing the **right kernel size** becomes tough because of this huge variation in the location of the information.
  - A **larger kernel** is preferred for information that is distributed more **globally**.
  - A **smaller kernel** is preferred for information that is distributed more **locally.**

Convolution Layer

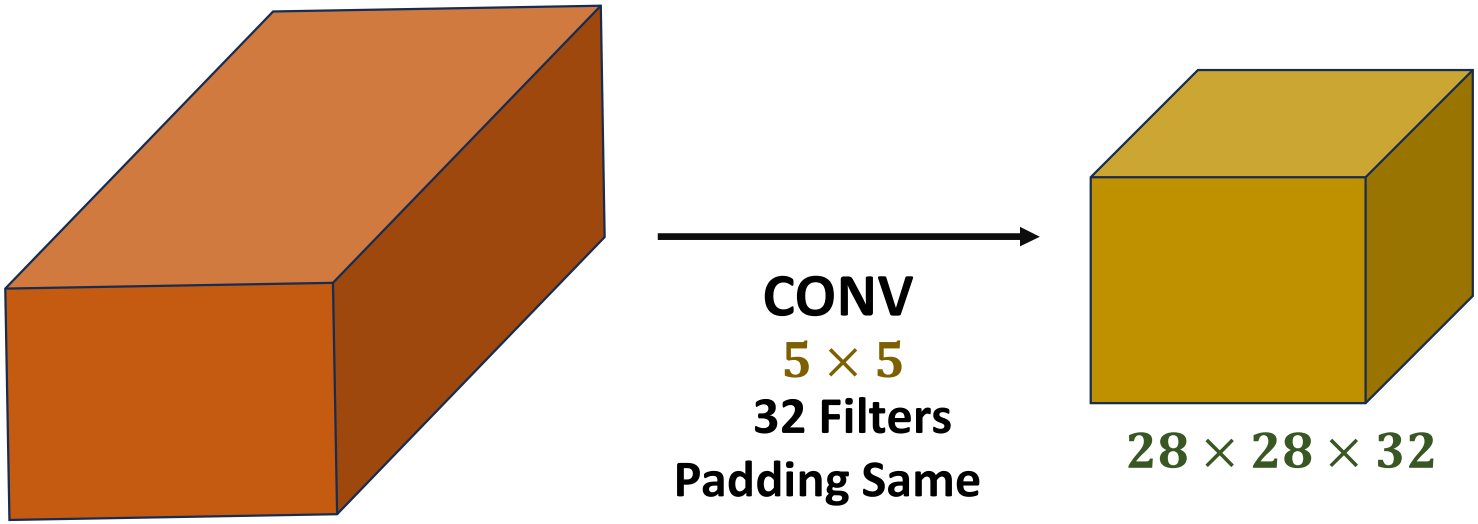


1x1 (Bottleneck) Convolution Layer



1x1 (Bottleneck) Convolution Layer

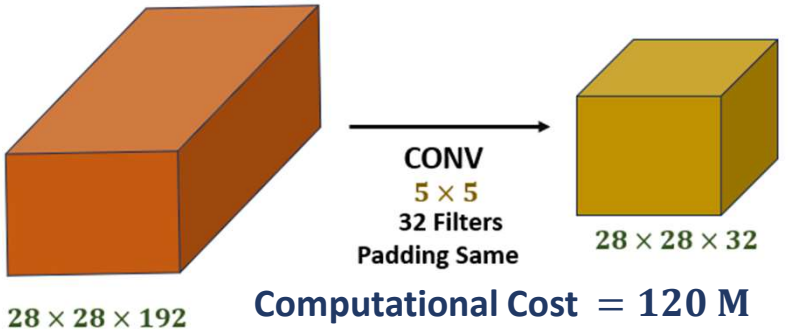
- Enable dimensionality reduction (computational savings)
- Add more non-linearity to representation space



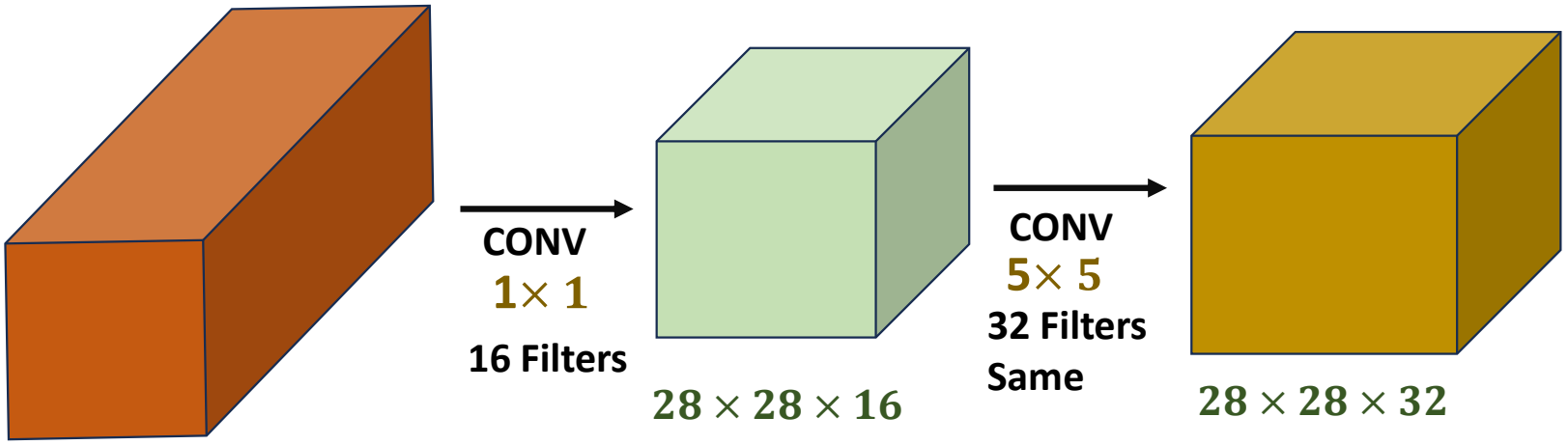
Computational Cost = $(28 \times 28 \times 32) \times (5 \times 5 \times 192) = 120\ \mathbf{M}$

https://www.quora.com/What-is-a-1X1-convolution

1x1 (Bottleneck) Convolution Layer



Computational Cost $= 120\ \mathbf{M}$



Cost2 = $(28 \times 28 \times 32) \times (5 \times 5 \times 16) = 10\ \mathbf{M}$

Cost1 = $(28 \times 28 \times 16) \times (1 \times 1 \times 192) = 2.4\ \mathbf{M}$

Computational Cost = $10\ \mathbf{M} + 2.4\ \mathbf{M} = 12.4\ \mathbf{M}$

https://www.quora.com/What-is-a-1X1-convolution

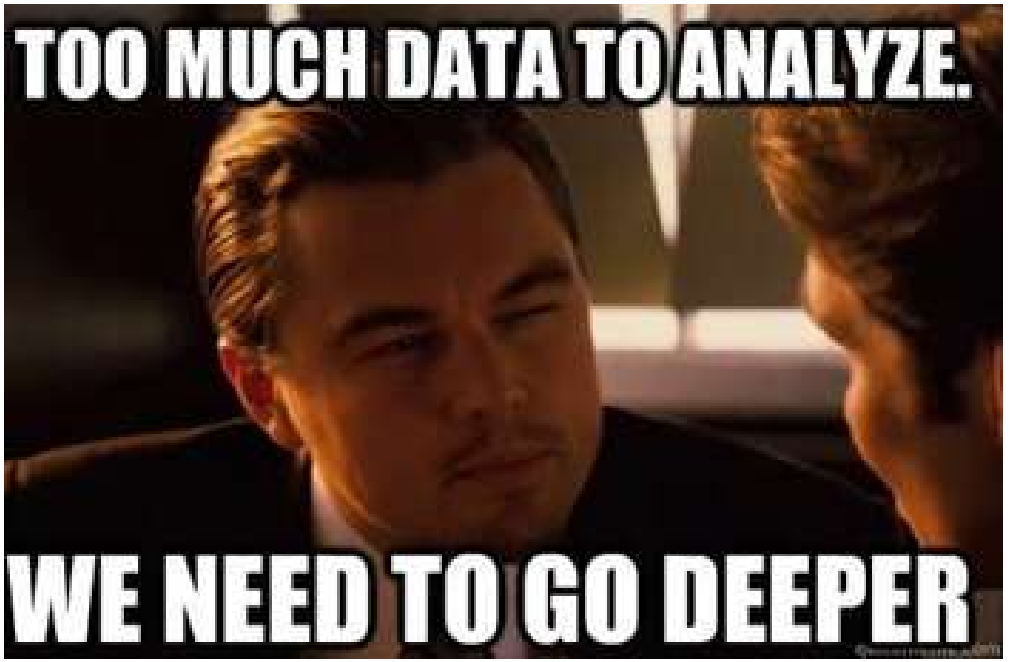
Why not have multiple levels (scales) of filtering ?

- Why not have filters with **multiple sizes** operate on the **same level**?
- The network essentially would get a bit "**wider**" rather than "deeper".

Dreams within Dreams

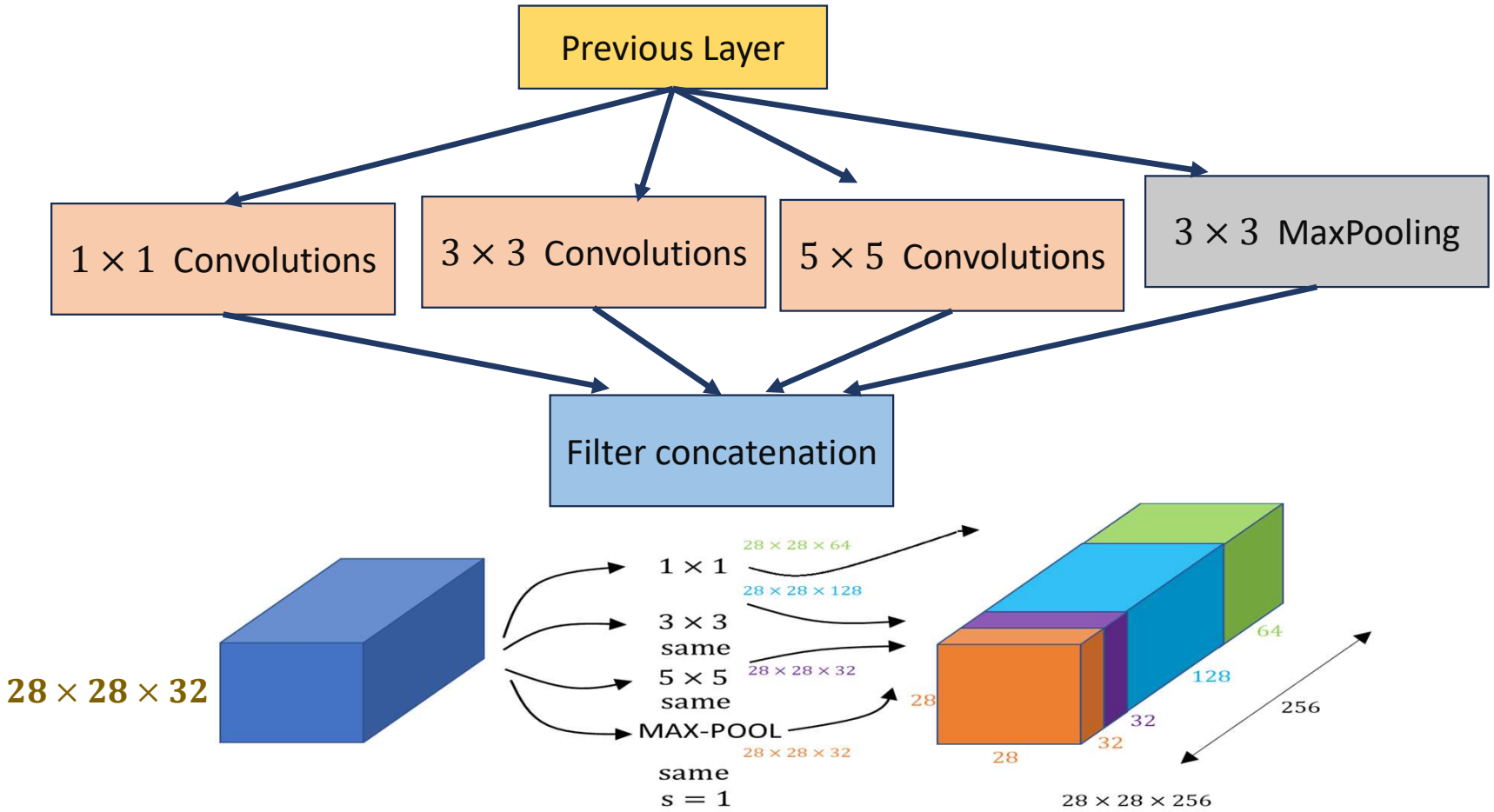
Network in Network

paper published in 2014 by Min Lin et al

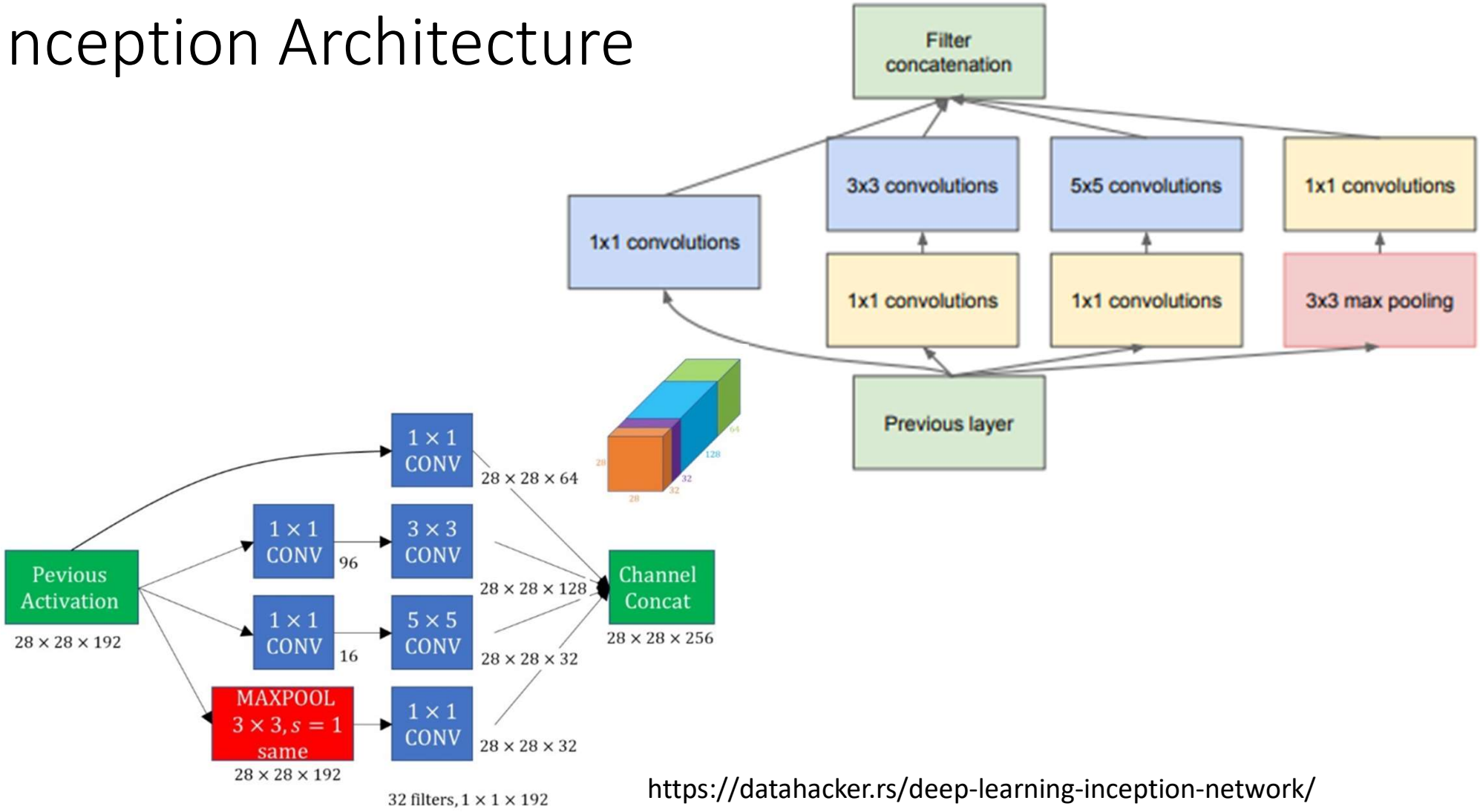


this meme was referenced in the first inception net paper.
https://arxiv.org/pdf/1409.4842v1.pdf?

Inception – v1



Inception Architecture



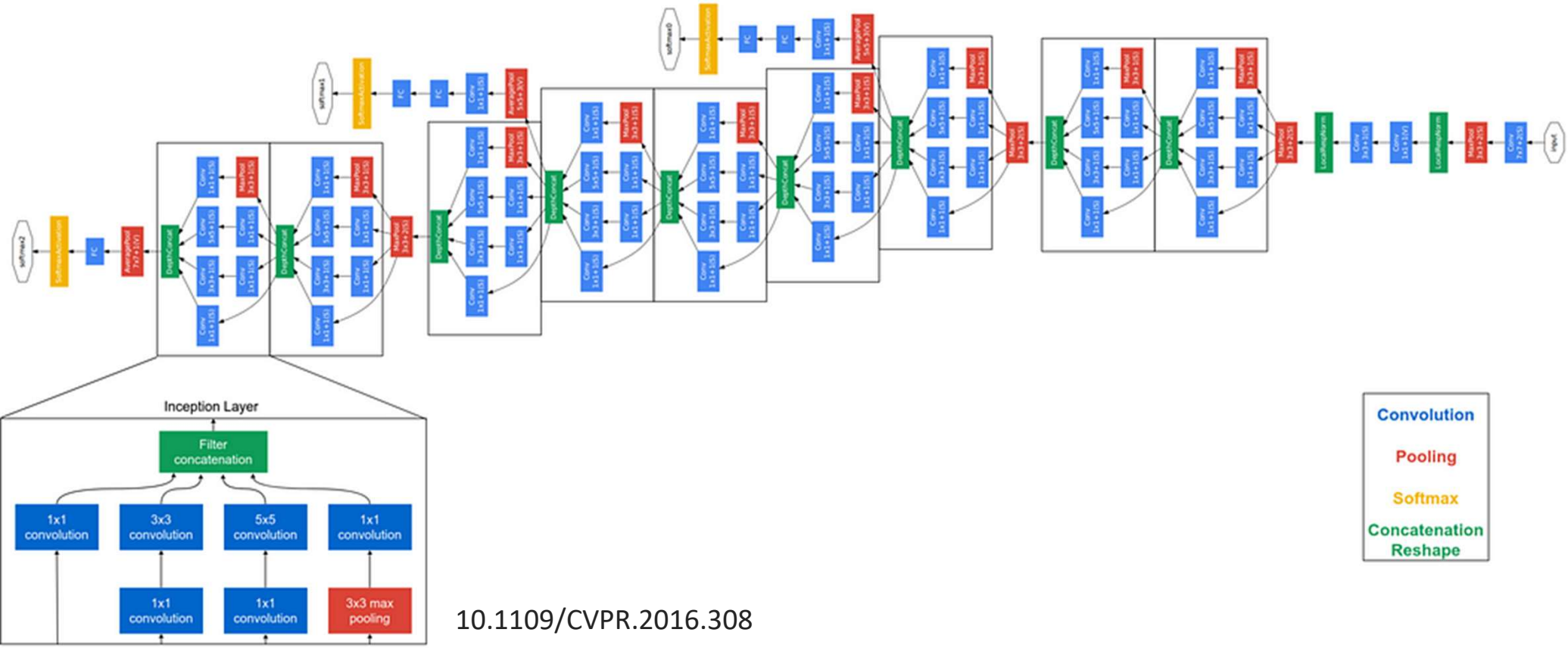
https://datahacker.rs/deep-learning-inception-network/

GoogLeNet

12x less parameters compared to AlexNet !

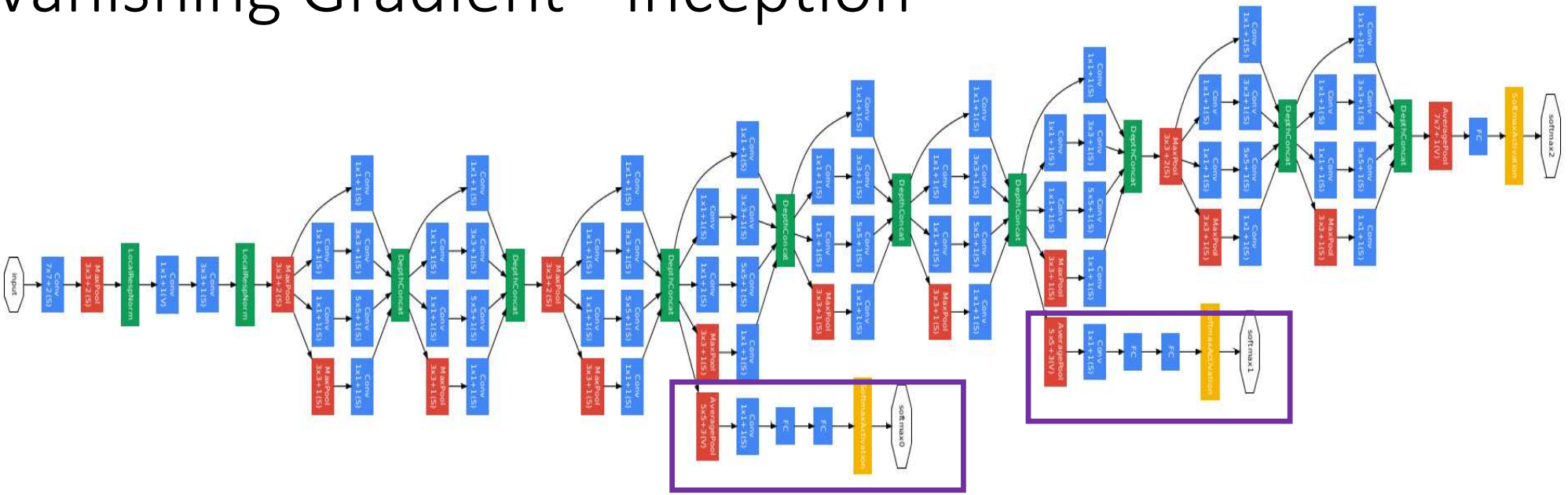
22-layers-deep network, 9 inception modules

Very deep network subjected to
Vanishing Gradient Problem



10.1109/CVPR.2016.308

Vanishing Gradient - Inception

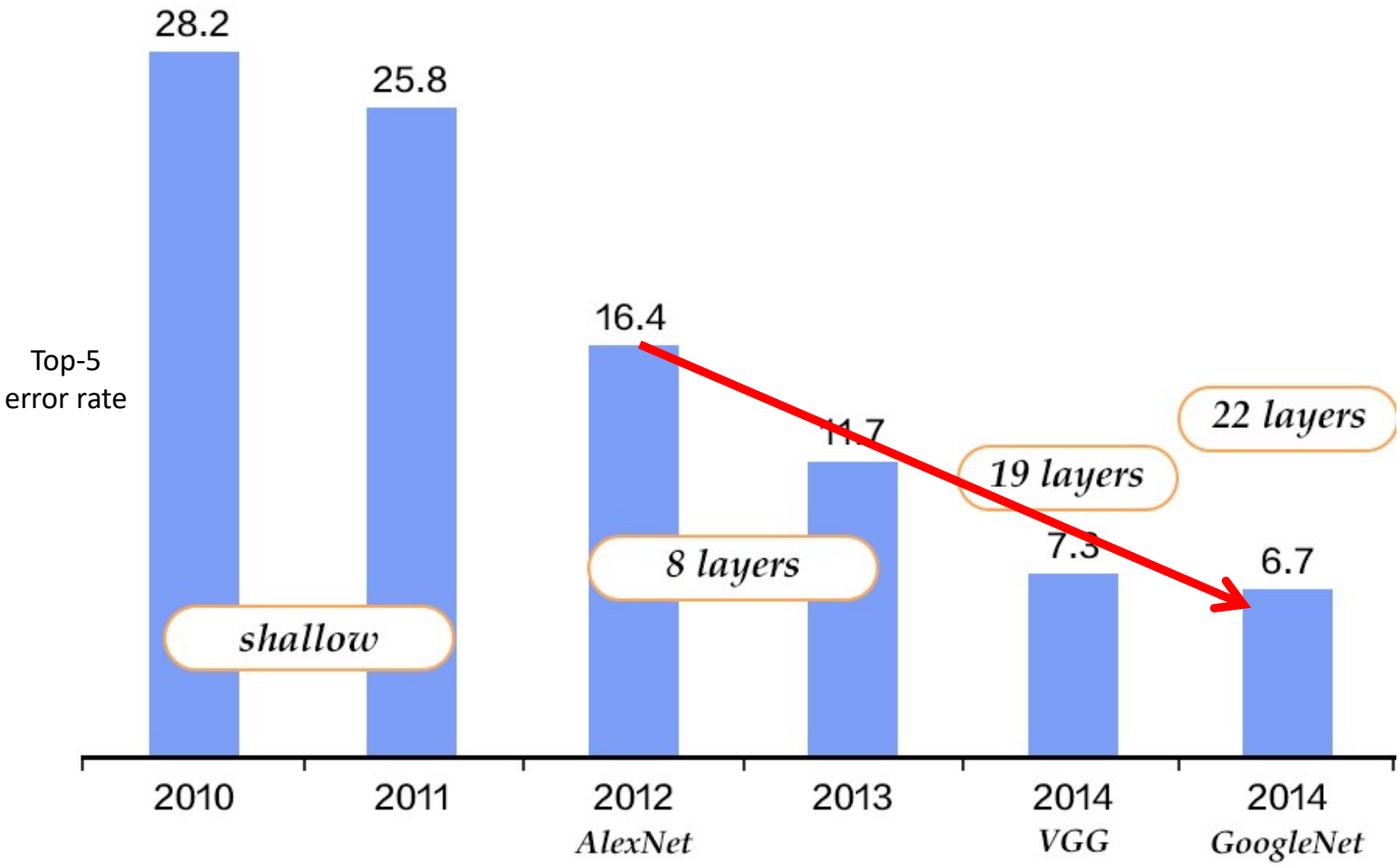


- Two auxiliary classifiers (The purple boxes in the image) introduced.
- Softmax is applied to the outputs of two of the inception modules, and auxiliary loss computed over the same labels.
- The total loss function is a weighted sum of the auxiliary loss and the real loss.

Summary

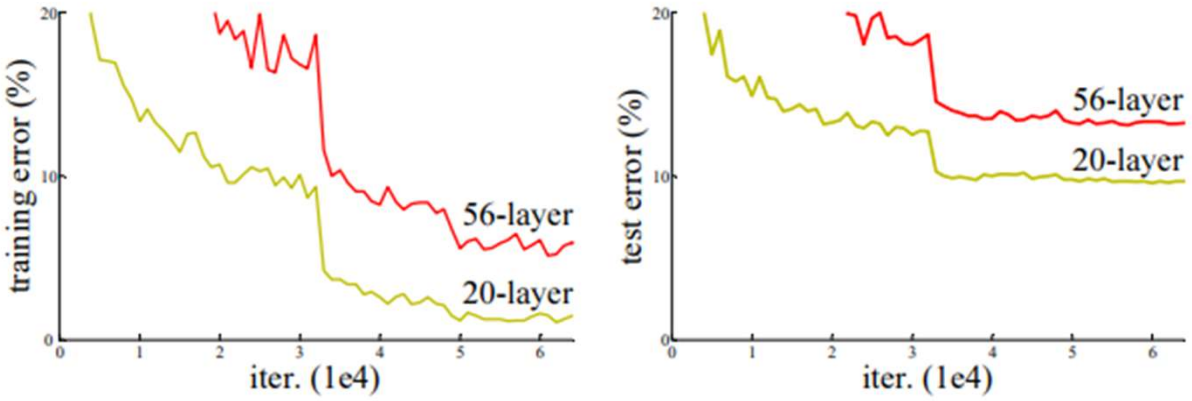
- 1x1 convolutions were used for reducing the dimensionality of the layers while making the model deeper.
- Average global pooling is performed before the fully connected layers in order to reduce the number of feature maps.
- Two auxiliary losses were introduced at the earlier layers of the model for the sake of propagating good gradients to the initial layers.
- The Inception module combines the outputs of differently sized filters. The parallel structure of multiple scales enables the module to capture both smaller and larger motifs in the pixel-data.

CNN layers didn't always have to be stacked up sequentially. A creative structuring of layers can lead to improved performance and computationally efficiency


Top-5
error rate
28.2
25.8
16.4
11.7
7.3
6.7
shallow
8 layers
19 layers
22 layers
2010
2011
2012
AlexNet
2013
2014
VGG
2014
GoogleNet

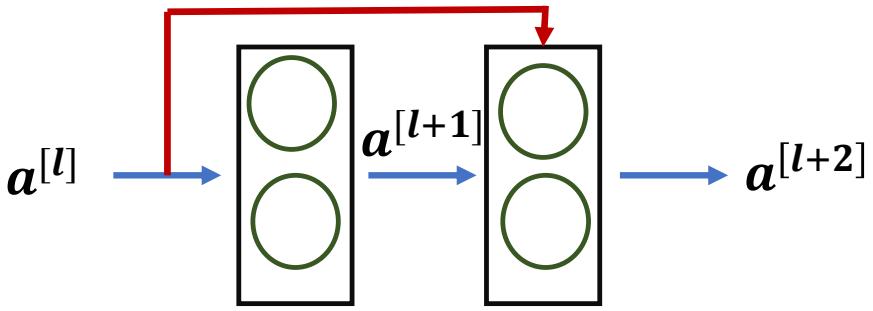
How deep?

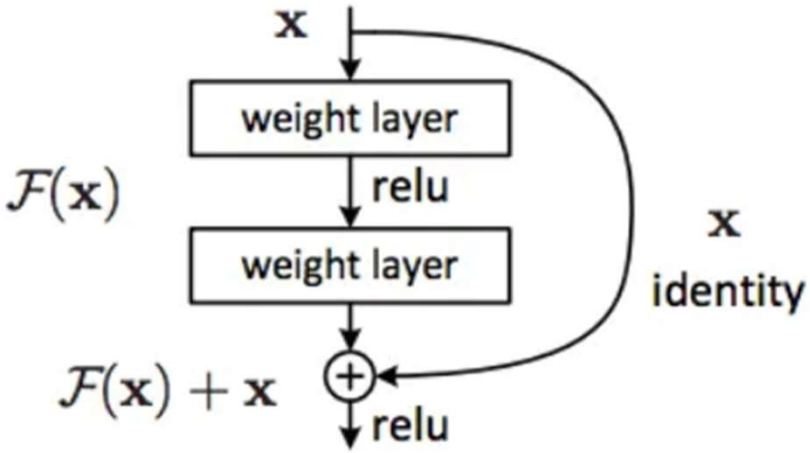
- There is a maximum threshold for depth with the traditional CNN model.



- Evidence shows that the best ImageNet models using convolutional and fully-connected layers typically contain between 16 and 30 layers.

Residual Block





Short-cut/Skip Connection

$a^{[l]} \rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ $a^{[l+1]} \rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ $a^{[l+2]}$

Main path

$$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$$

$$a^{[l+1]} = g(z^{[l+1]})$$

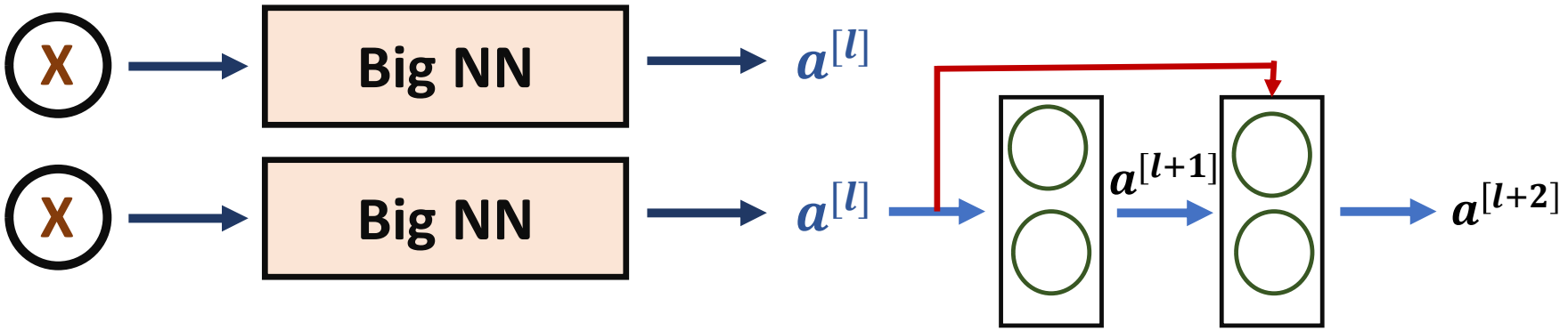
$$z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$$

$$a^{[l+2]} = g(z^{[l+2]})$$

$$a^{[l+2]} = g(z^{[l+2]} + a^{l})$$

Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun

How Residual Network helps?



$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right) = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

If $W^{[l+2]} \to 0,\ b^{[l+2]} \to 0,\ a^{[l+2]} = g\left(a^{[l]}\right) = a^{[l]}$

$$a^{[l+2]} = a^{[l]}$$

Identity function is easy for Residual block to learn, ensuring that the higher layers of the model do not perform any worse than the lower layers.

ResNet Architecture

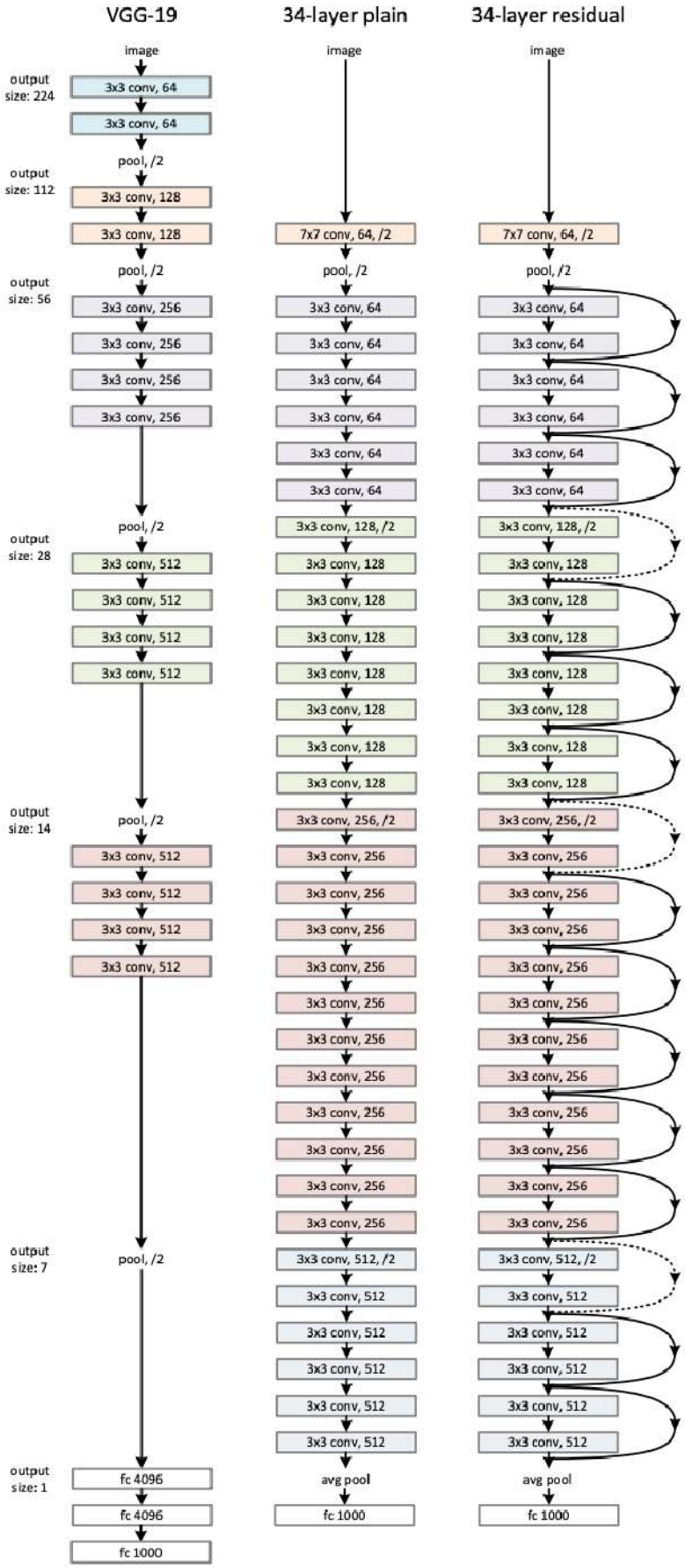
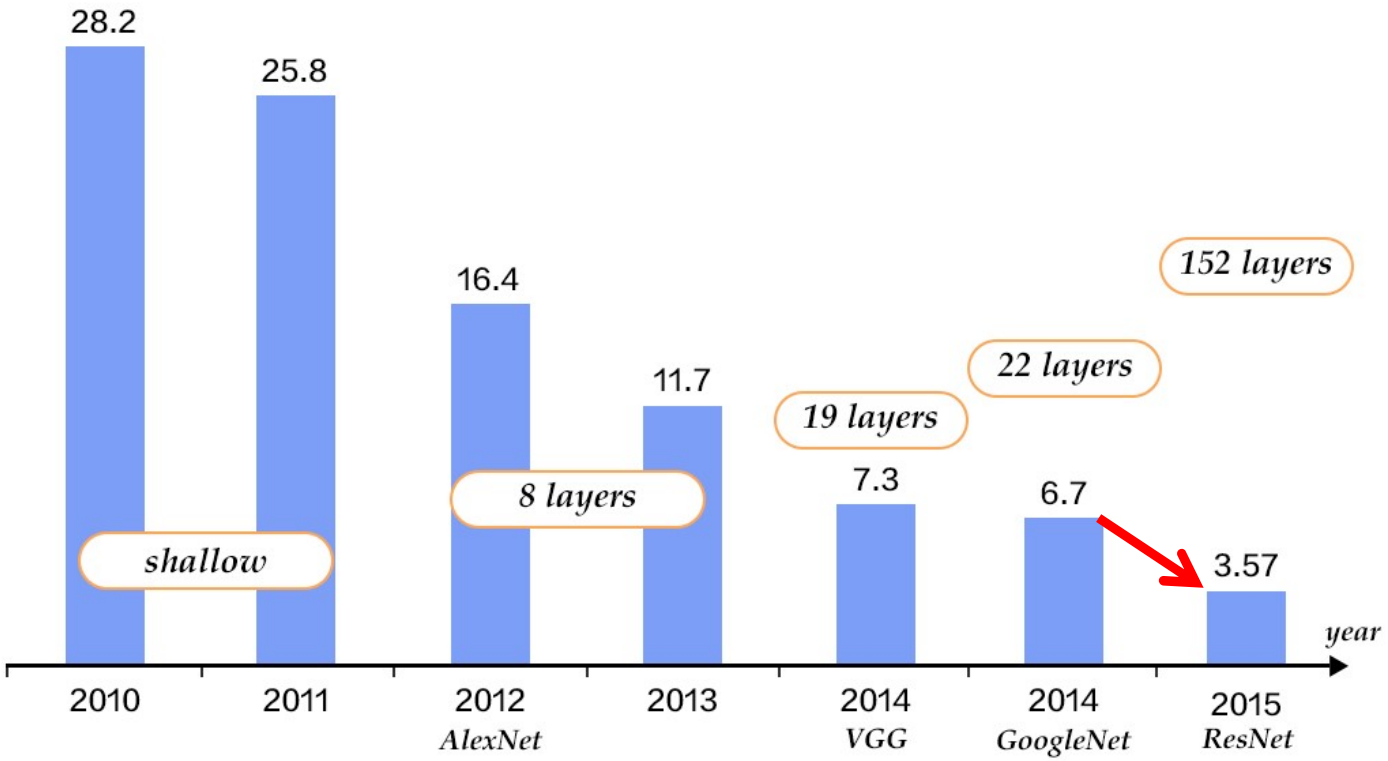


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.


28.2
25.8
16.4
11.7
7.3
6.7
3.57
shallow
8 layers
19 layers
22 layers
152 layers
year
2010
2011
2012
AlexNet
2013
2014
VGG
2014
GoogleNet
2015
ResNet

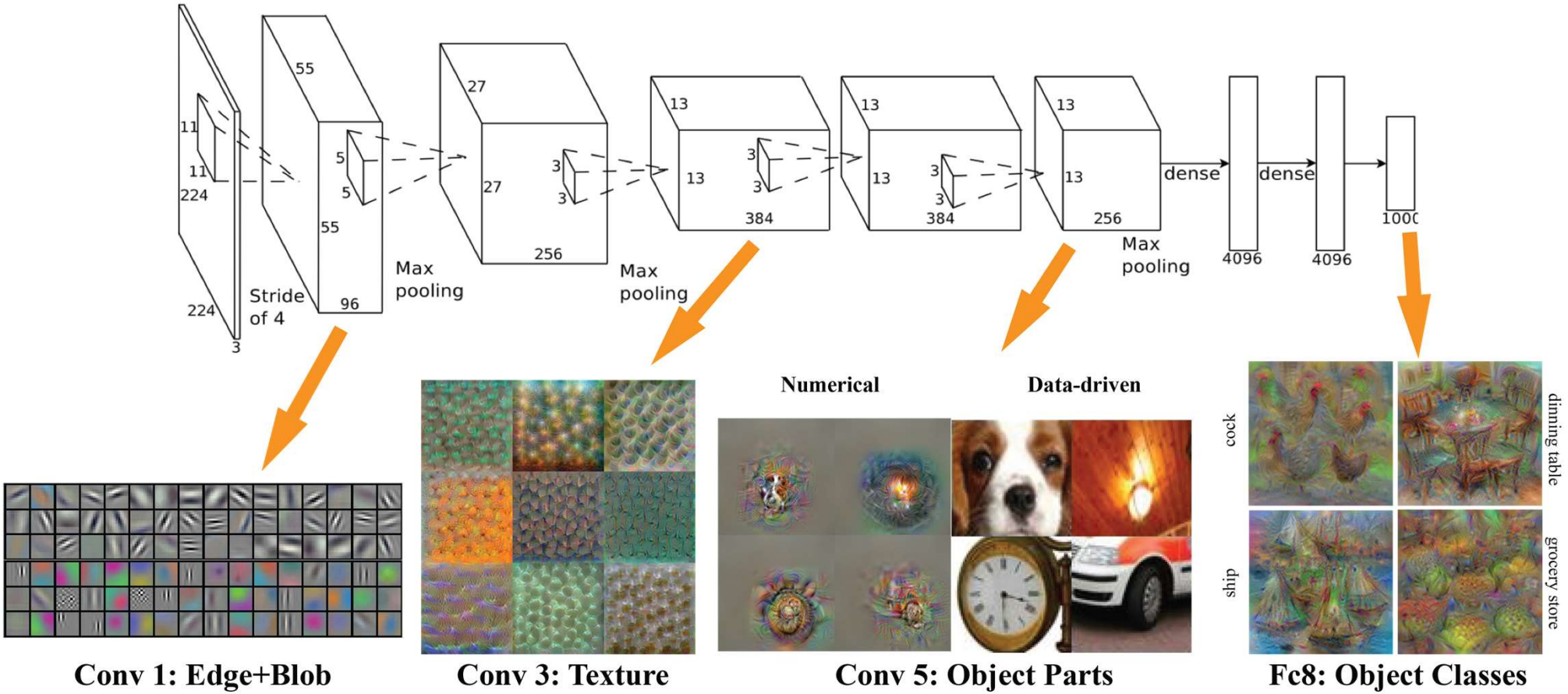
- Deep learning is based on large, high-quality datasets. What to do when you don’t have a large dataset?

- Lost in the wilderness of neural network’s hyperparameters with no idea where to start and what to configure????

Transfer Learning

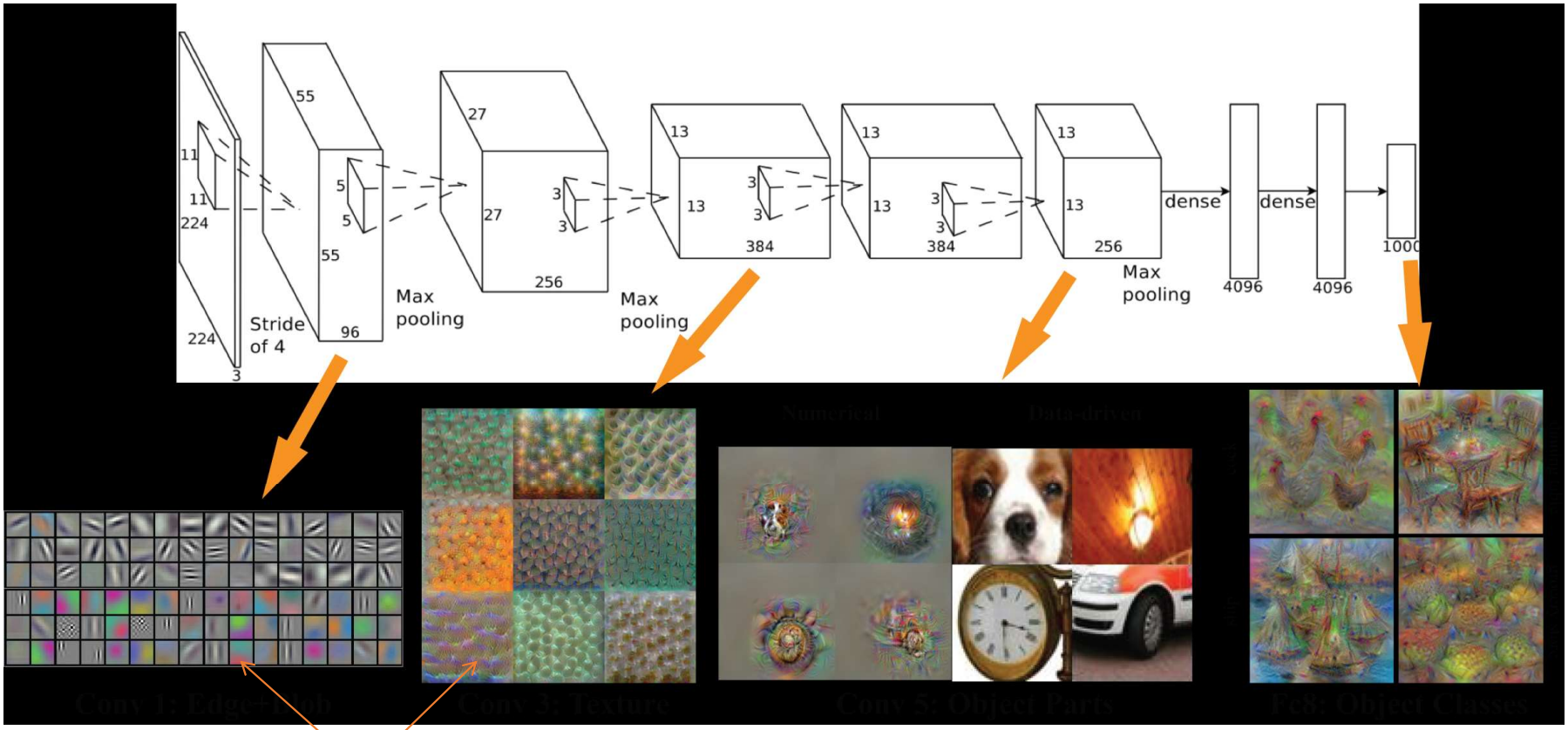
Fine-tuning or Freezing

Filters learnt by AlexNet



https://www.cc.gatech.edu/~hays/compvision/proj6/deepNetVis.png

Filters learnt by AlexNet



These features seem generic enough. Can we re-use them ?

https://www.cc.gatech.edu/~hays/compvision/proj6/deepNetVis.png

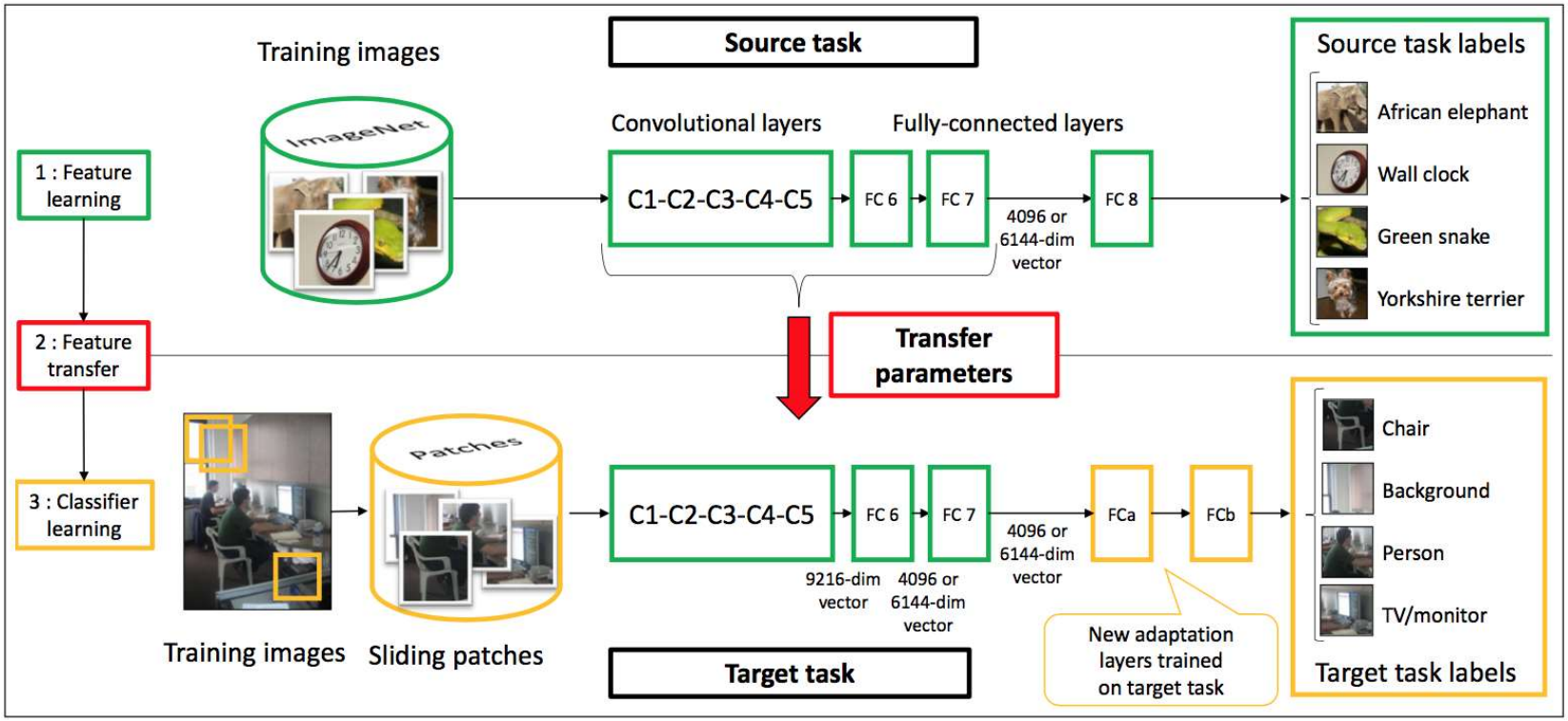
Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.

- No need to train from scratch – Leverage past powerful deep learning models and transfer their knowledge to the specific domain

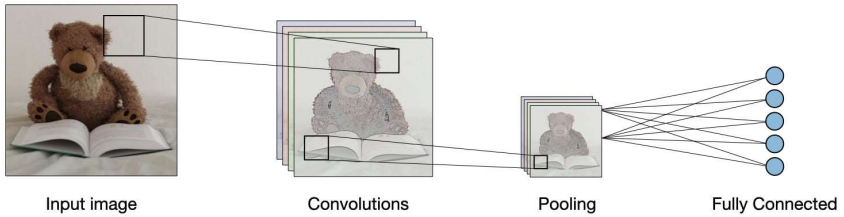
- Two major strategies:
  - ConvNet as fixed feature extractor: Use a pre-trained net, remove the output layer, and use the rest of the network as a feature extractor for a related dataset
  - Fine-tuning the ConvNet : Use a pre-trained net, use its weights as initialization to train a deep net for a new but related task (useful when we don't have much training data for the new task)

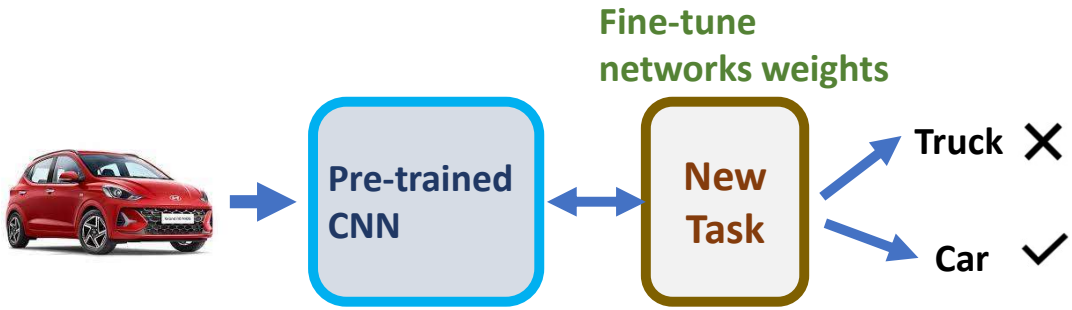
Transfer Learning



Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks [Oquab et al. CVPR 2014] https://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Oquab_Learning_and_Transferring_2014_CVPR_paper.html

CNN vs Transfer Learning





Training Data	1000s to millions of labelled images
Computation	Compute intensive
Training Time	Days to Weeks for real problems
Model Accuracy	High (can be overfitting to small datasets)

Training Data	100s to 1000s of labelled images
Computation	Moderate Computation
Training Time	Seconds to minutes
Model Accuracy	Good, depends on the pre-trained CNN model

Transfer Learning Strategies

- ***Which*** part of the knowledge can be transferred from the source to the target to improve the performance of the target task?
- ***When*** to transfer and when not to, so that one improves the target task performance/results and does not degrade them?
- ***How*** to transfer the knowledge gained from the source model based on our current domain/task?

When and how to fine-tune?

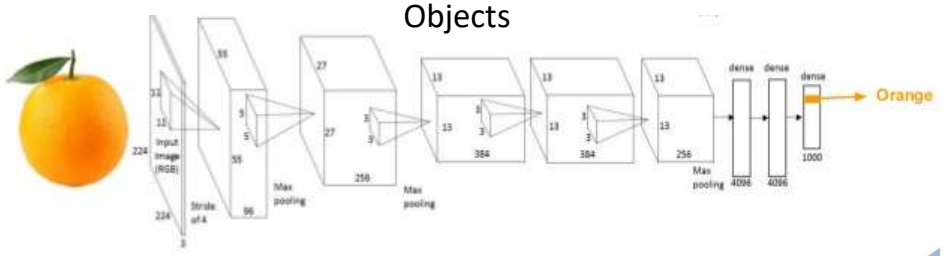
	You have little data	You have a lot of data
Datasets are similar	**Train a classifier (Linear SVM) on top of bottleneck features**	**Fine-tune several or all layers**
Datasets are different	**Train a classifier on deep features of the CNN**	**Fine-tune all layers (use pre-trained weights as an initialization for your CNN)**

CNN Applications

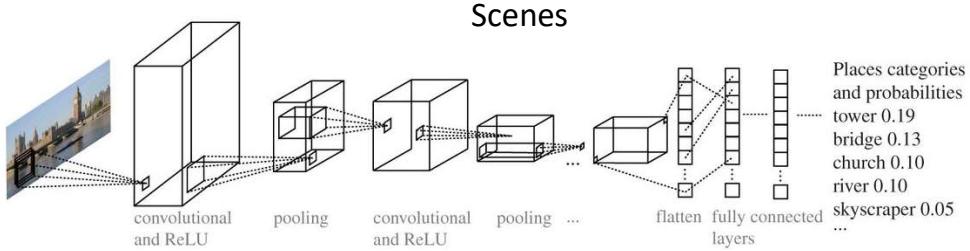
Semantic Segmentation

Image ⟶ Label

Objects

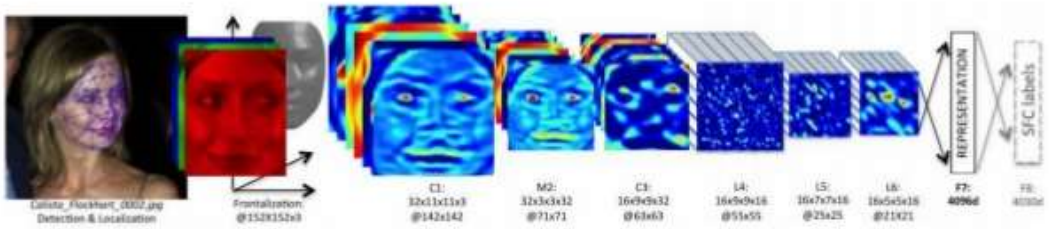


Scenes

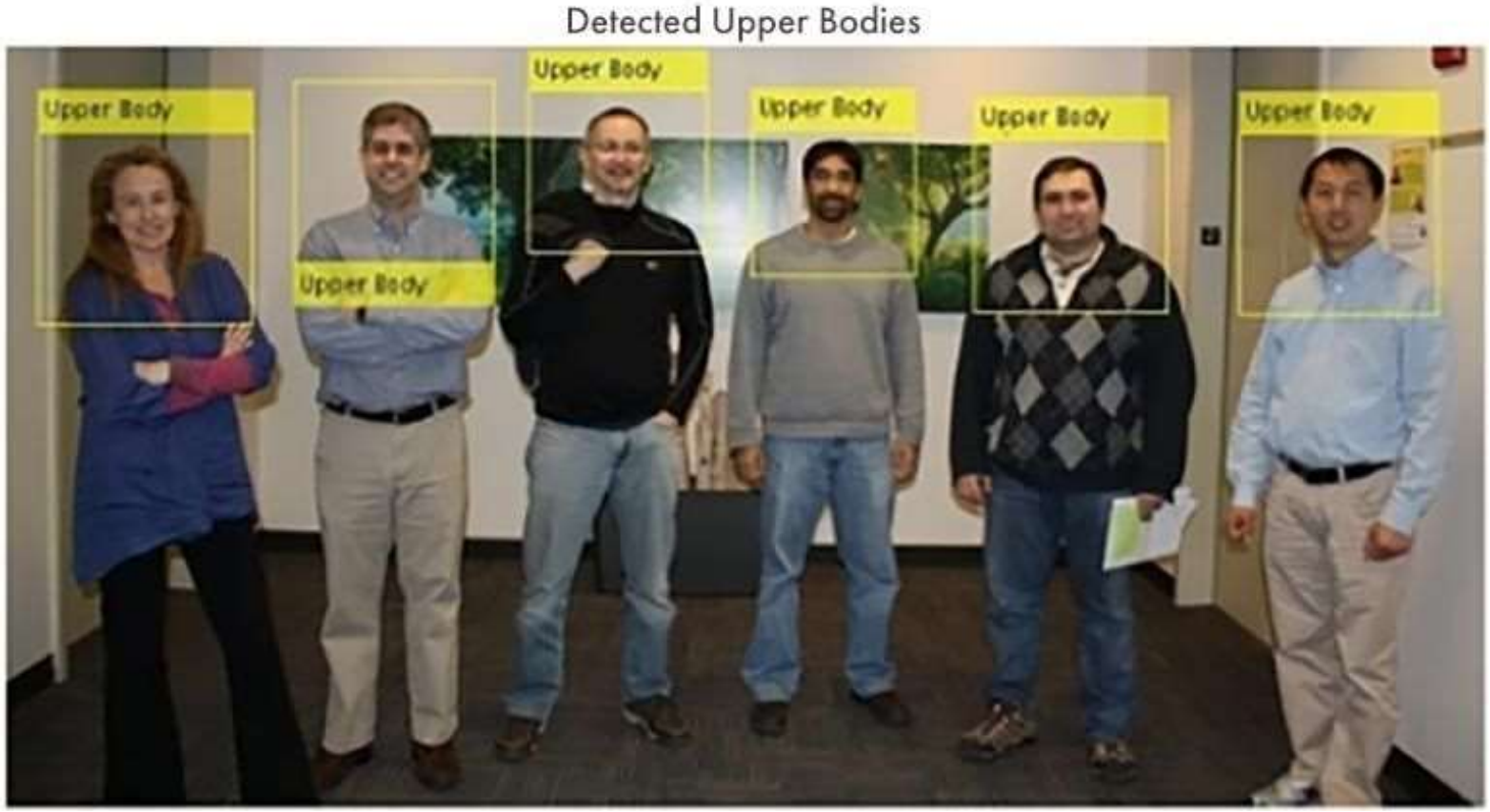


Faces

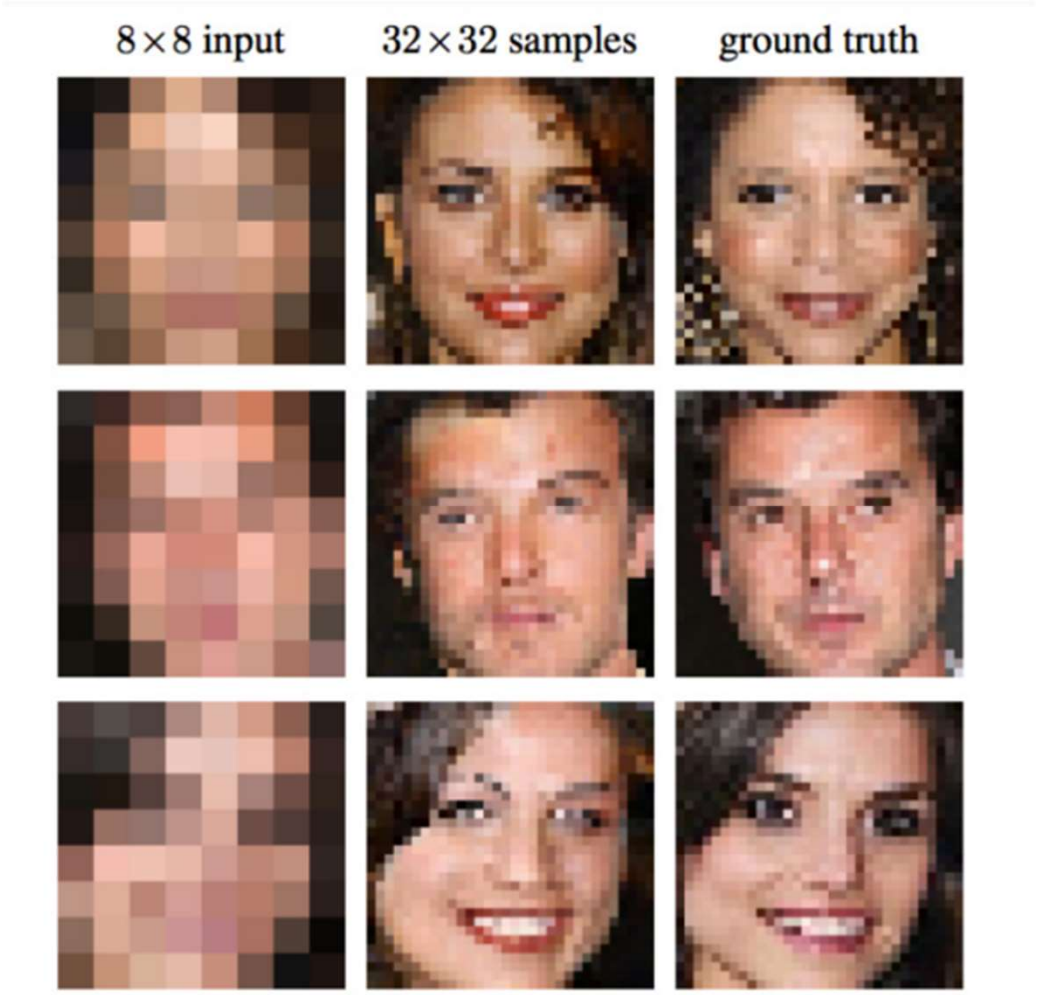
DeepFace Architecture



Object Detection

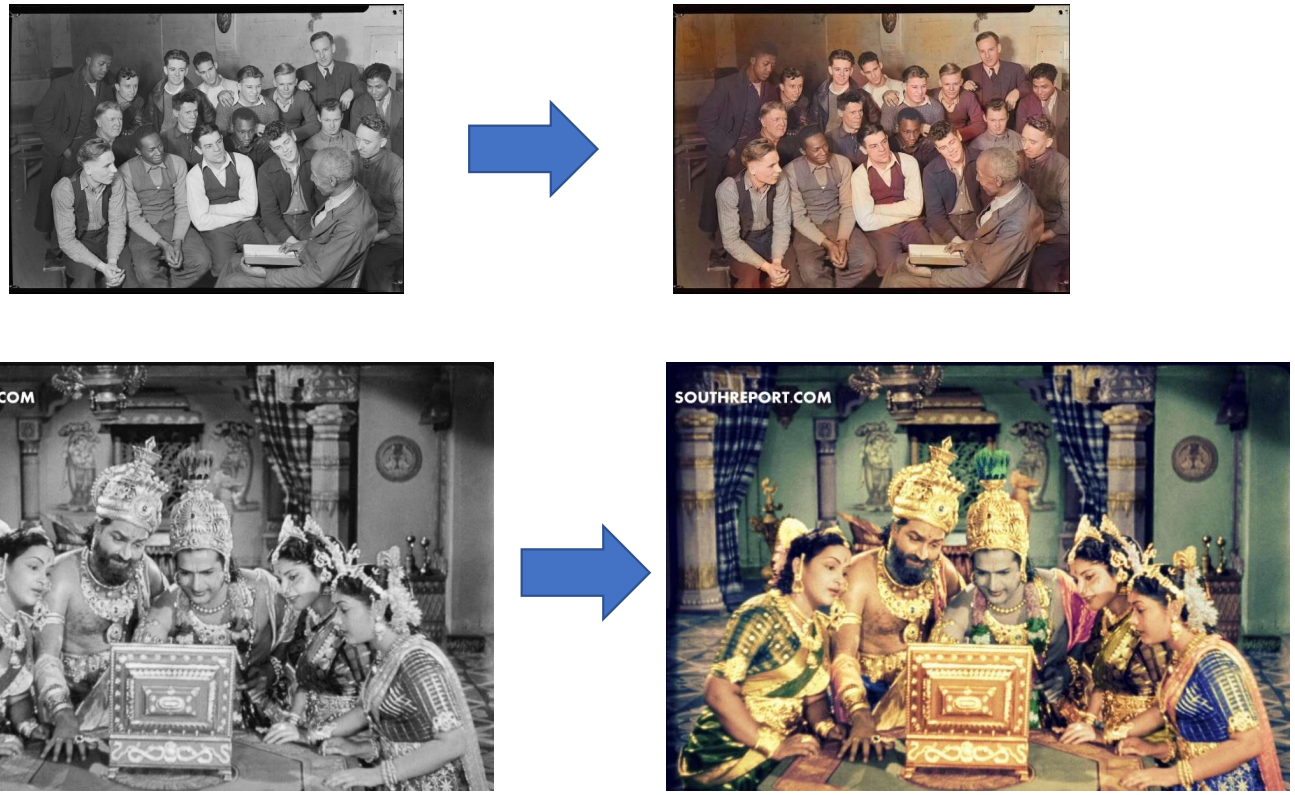


Up-scaling low-resolution images



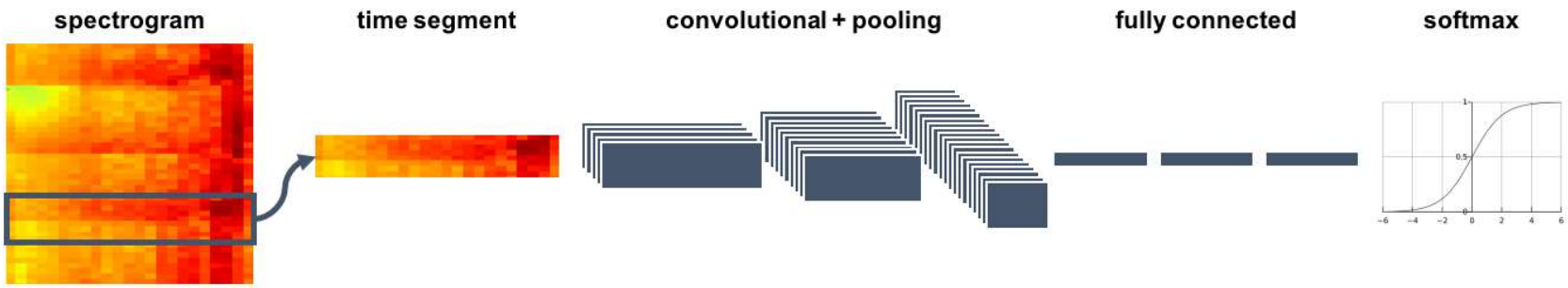
8×8 pixel photos were inputted into a Deep Learning Network which tried to guess what the original face looked like. The answer was fairly close (the correct answer is under “ground truth”

Automatic Colorization of Black and White Images



https://algorithmia.com/algorithms/deeplearning/ColorfulImageColorization

Image CNNs for non-image data



http://tommymullaney.com/projects/rhythm-games-neural-networks