Deep Learning for Vision & Language

Computer Vision III: Convolutional Neural Network Architectures

RICE UNIVERSITY

Assignment 2

- Released on course website and canvas submission available
  - Due February 13th Monday midnight Central Time (CT)
  - Same advice: Start this week even for topics we have not covered yet
  - Assume: Google Colab will likely fail the last two days before deadline
  - First two points are for free: just basic programming
  - Next two points almost free if you completed the previous assignment (Requires model training so get this out of the way soon)
  - Next three points require some thinking but no training of models (However it requires using a heavy model Google's FLAN T5)
  - Next three points require some thinking and tinkering with models and also no training (Also requires using OpenAI's CLIP)
  - Assume: You will have to restart the cloud instance many times because you will run out of memory – try not to re-run cells that are loading models in the same session.

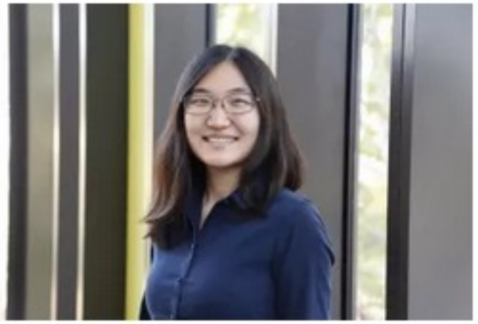
Happening Tomorrow

Happening Thursday

https://riceuniversity.zoom.us/j/97815558341?pwd=aW9YcExUUitIZnNIZzd2VXJldDNwdz09

CS Colloquium:AI for Scientists: Accelerating Discovery through Knowledge, Data & Learning

12:00pm - 1:00pm CST



Jennifer Sun

Contact Info:

Annepha Hurlock,
annepha@rice.edu

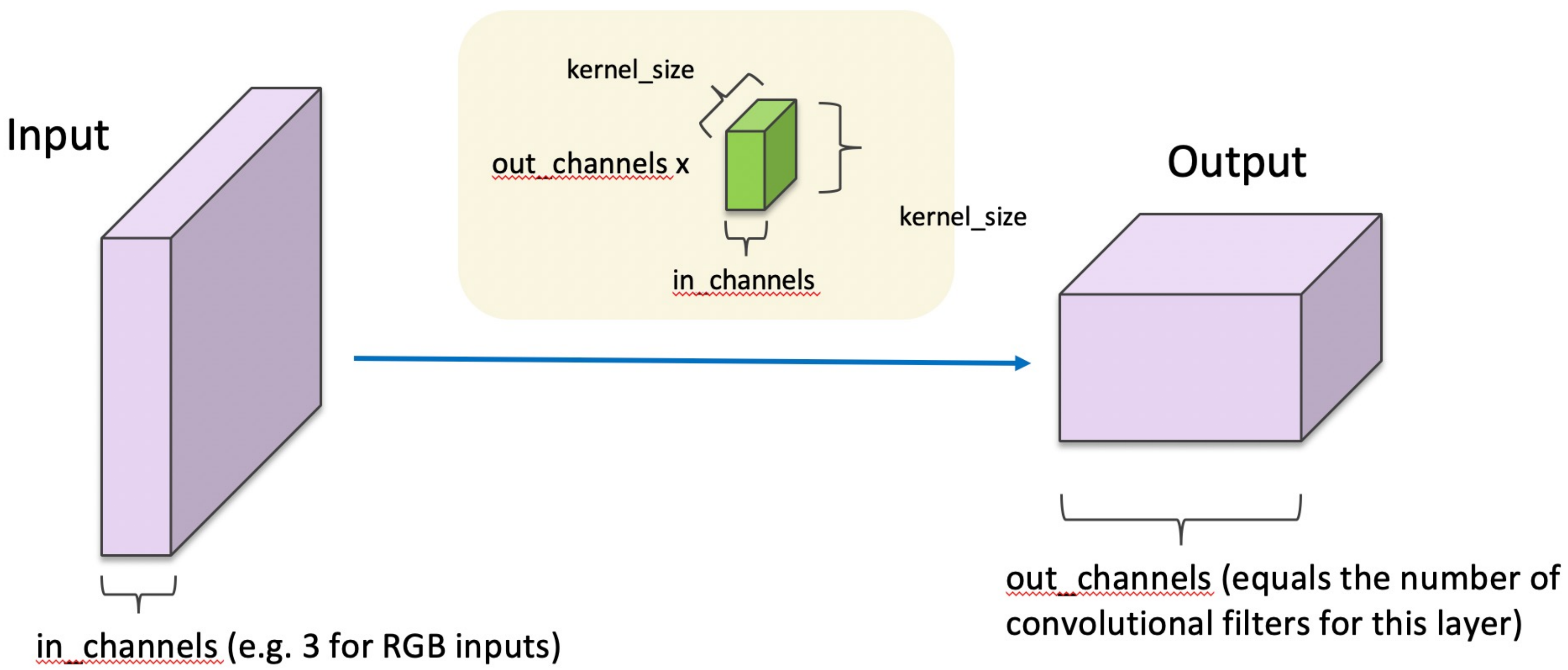
Join on Zoom:

https://riceuniversity.zoom.us
/j/97815558341?pwd=aW9YcExUUitIZnNIZzd2

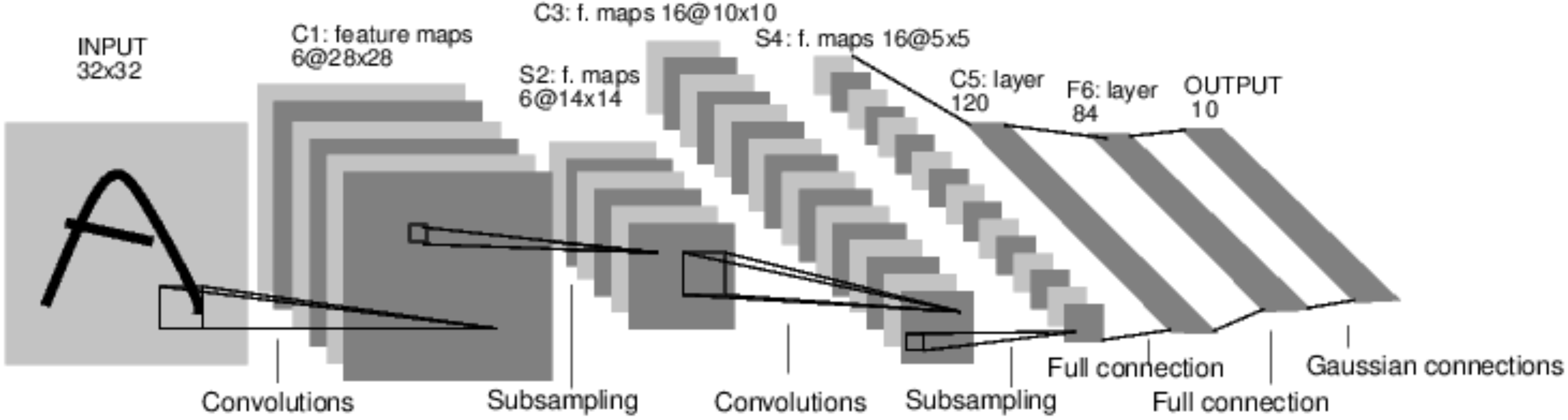
Abstract: With rapidly growing amounts of experimental data, machine learning is increasingly crucial for automating scientific data analysis. However, many real-world workflows demand expert-in-the-loop attention and require models that not only interface with data, but also with experts and domain knowledge. My research develops full stack solutions that enable scientists to scalably extract insights from diverse and messy experimental data with minimal supervision. My approaches learn from both data and expert knowledge, while exploiting the right level of domain knowledge for generalization. In this talk, I will present progress towards developing automated scientist-in-the-loop solutions, including methods that automatically discover meaningful structure from data such as self-supervised keypoints from videos of diverse behaving organisms. I will also present methods that use these interpretable structures to inject

Convolutional Layer in pytorch

class `torch.nn.Conv2d`(*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True*) [source]



Convolutional Network: LeNet



Yann LeCun

TITLE	CITED BY	YEAR
Gradient-based learning applied to document recognition Y LeCun, L Bottou, Y Bengio, P Haffner Proceedings of the IEEE 86 (11), 2278-2324	11736	1998

LeNet in Pytorch

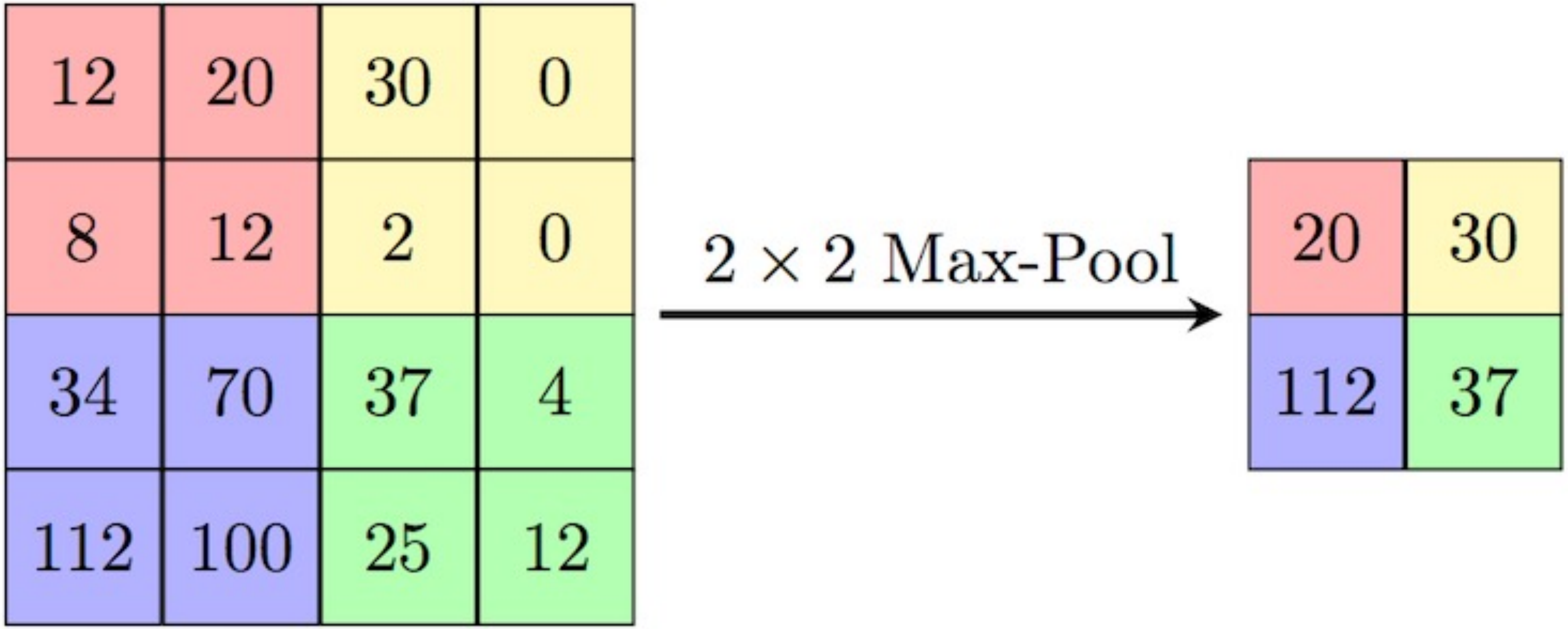
```python
# LeNet is French for The Network, and is taken from Yann Lecun's 98 paper
# on digit classification http://yann.lecun.com/exdb/lenet/
# This was also a network with just two convolutional layers.
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        # Convolutional layers.
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)

        # Linear layers.
        self.fc1 = nn.Linear(16*5*5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        out = F.relu(self.conv1(x))
        out = F.max_pool2d(out, 2)
        out = F.relu(self.conv2(out))
        out = F.max_pool2d(out, 2)

        # This flattens the output of the previous layer into a vector.
        out = out.view(out.size(0), -1)
        out = F.relu(self.fc1(out))
        out = F.relu(self.fc2(out))
        out = self.fc3(out)
        return out
```

SpatialMaxPooling Layer



https://paperswithcode.com/method/max-pooling

LeNet Summary

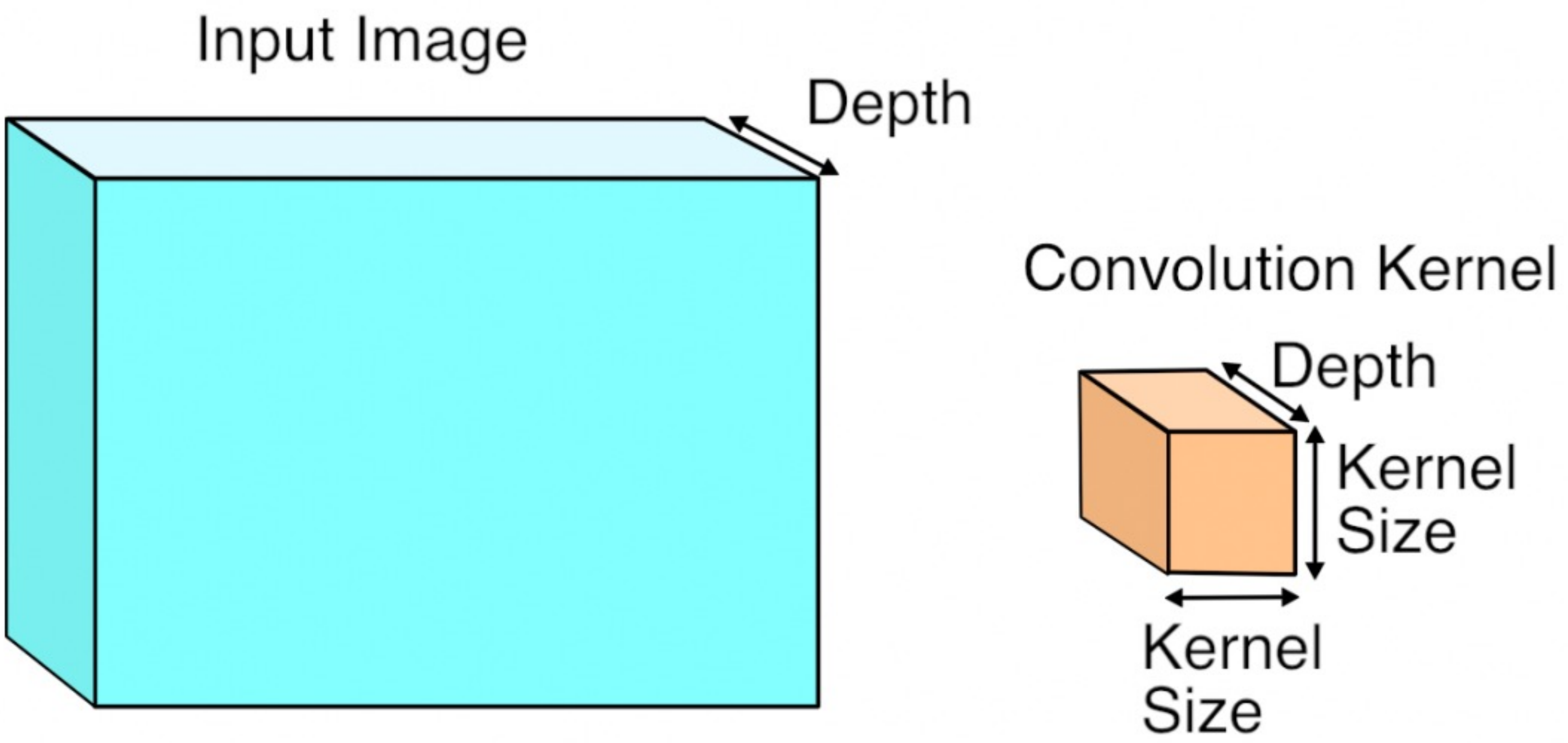
- 2 Convolutional Layers + 3 Linear Layers

- + Non-linear functions: ReLUs or Sigmoids
  + Max-pooling operations

New Architectures Proposed

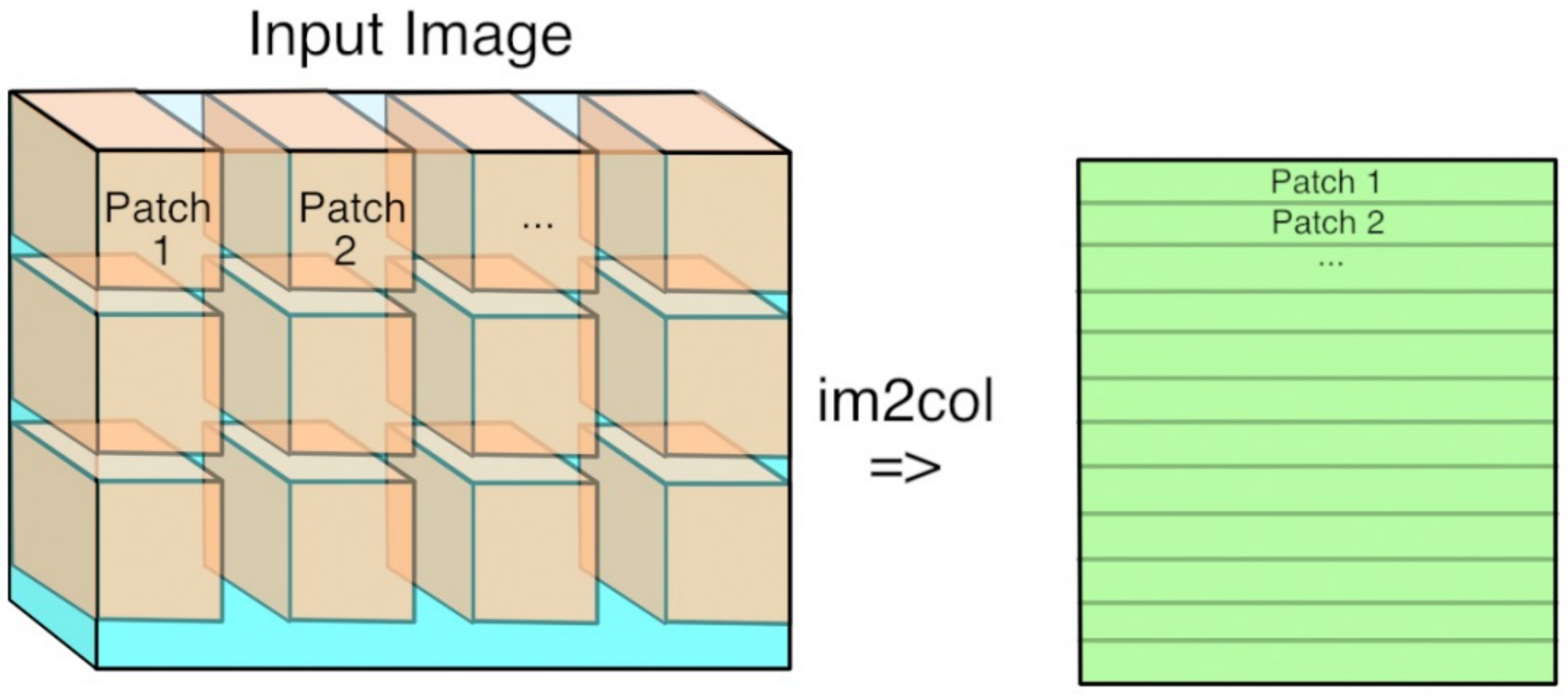
- Alexnet (Kriszhevsky et al NIPS 2012) **[Required Reading]**
- VGG (Simonyan and Zisserman 2014)
- GoogLeNet (Szegedy et al CVPR 2015)
- ResNet (He et al CVPR 2016)
- DenseNet (Huang et al CVPR 2017)

Convolutional Layers as Matrix Multiplication



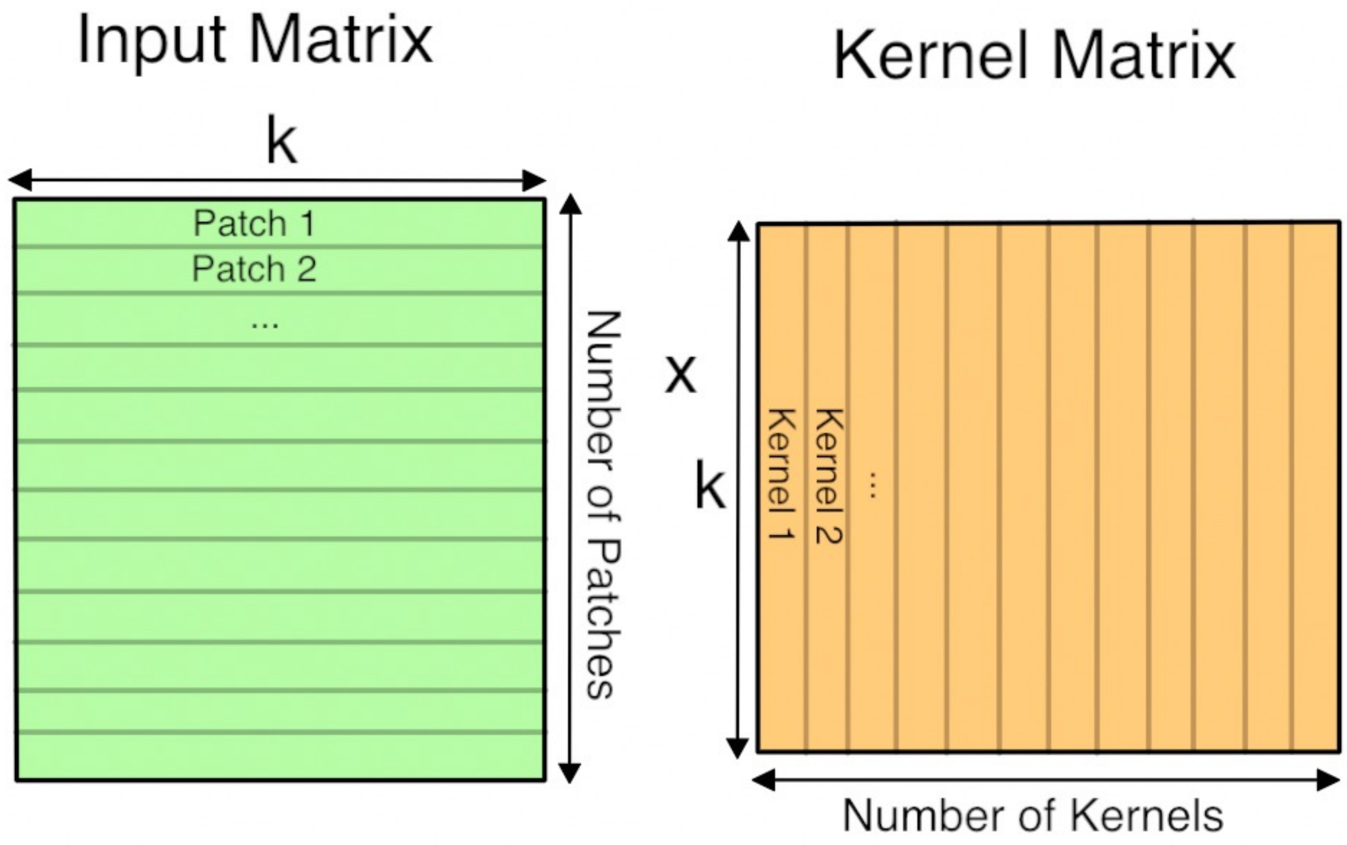
https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Convolutional Layers as Matrix Multiplication



https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Convolutional Layers as Matrix Multiplication



Pros?
Cons?

https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

CNN Computations are Computationally Expensive

- However highly parallelizable
- GPU Computing is used in practice
- CPU Computing in fact is prohibitive for training these models

The Alexnet network (Krizhevsky et al NIPS 2012)

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

The Problem: Classification

Classify an image into 1000 possible classes:
e.g. Abyssinian cat, Bulldog, French Terrier, Cormorant, Chickadee, red fox, banjo, barbell, hourglass, knot, maze, viaduct, etc.

cat, tabby cat (0.71)
Egyptian cat (0.22)
red fox (0.11)
.....

The Data: ILSVRC

Imagenet Large Scale Visual Recognition Challenge (ILSVRC): Annual Competition

1000 Categories

~1000 training images per Category

~1 million images in total for training

~50k images for validation

Only images released for the test set but no annotations,
evaluation is performed centrally by the organizers (max 2 per week)

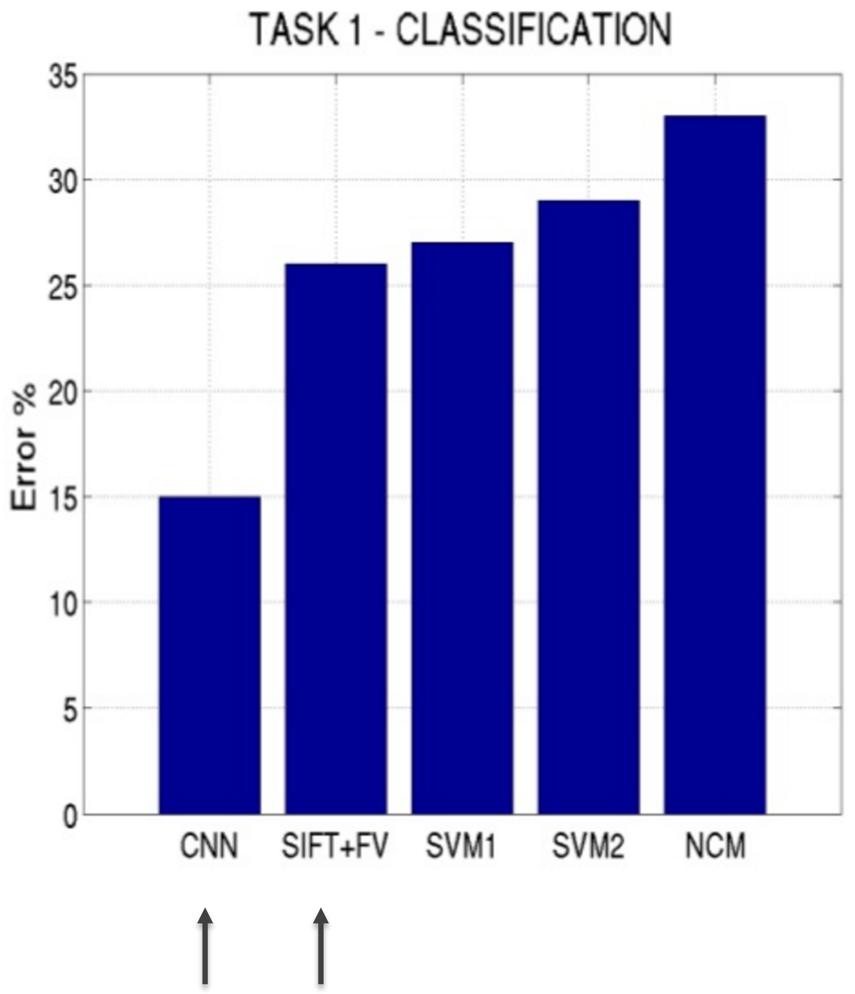
The Evaluation Metric: Top K-error

True label: Abyssinian cat

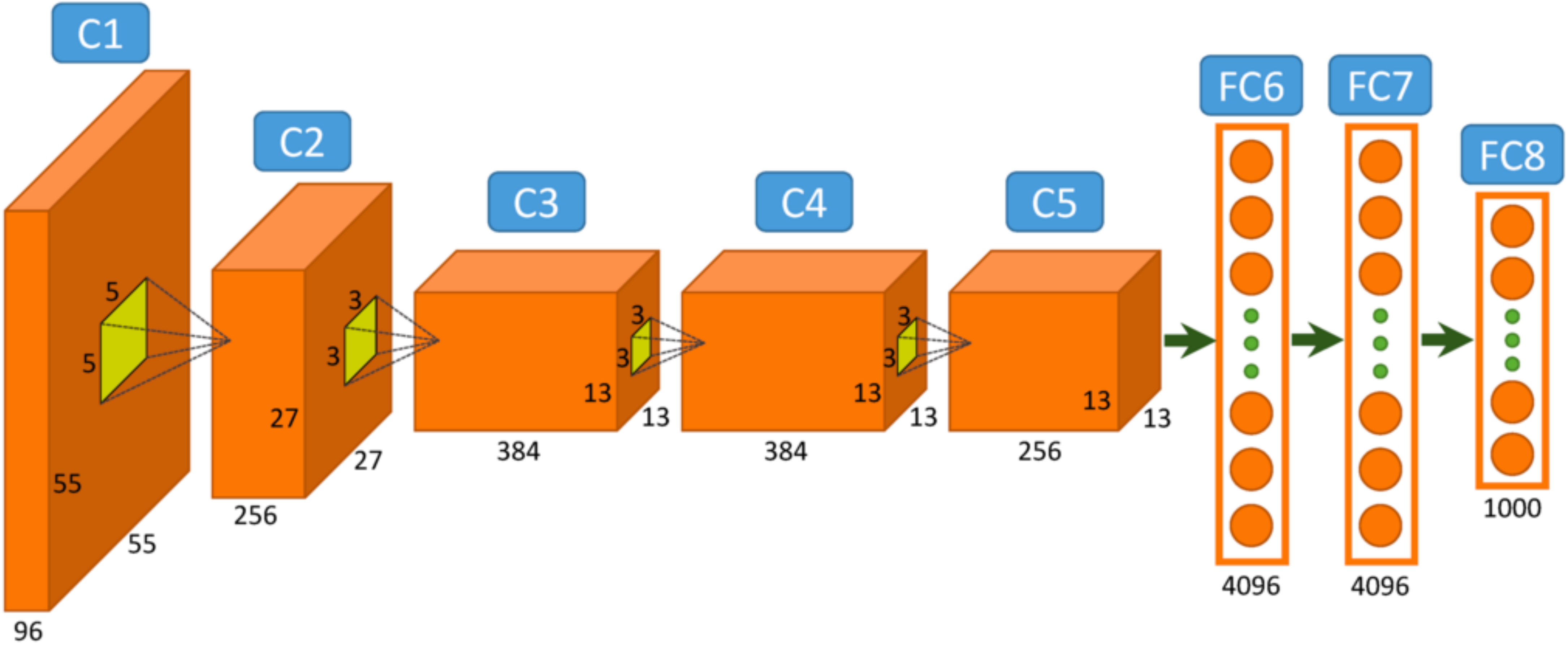
Top-1 error: 1.0	Top-1 accuracy: 0.0
Top-2 error: 1.0	Top-2 accuracy: 0.0
Top-3 error: 1.0	Top-3 accuracy: 0.0
Top-4 error: 0.0	Top-4 accuracy: 1.0
Top-5 error: 0.0	Top-5 accuracy: 1.0

cat, tabby cat (0.61)
Egyptian cat (0.22)
red fox (0.11)
Abyssinian cat (0.10)
French terrier (0.03)
.....

Top-5 error on this competition (2012)

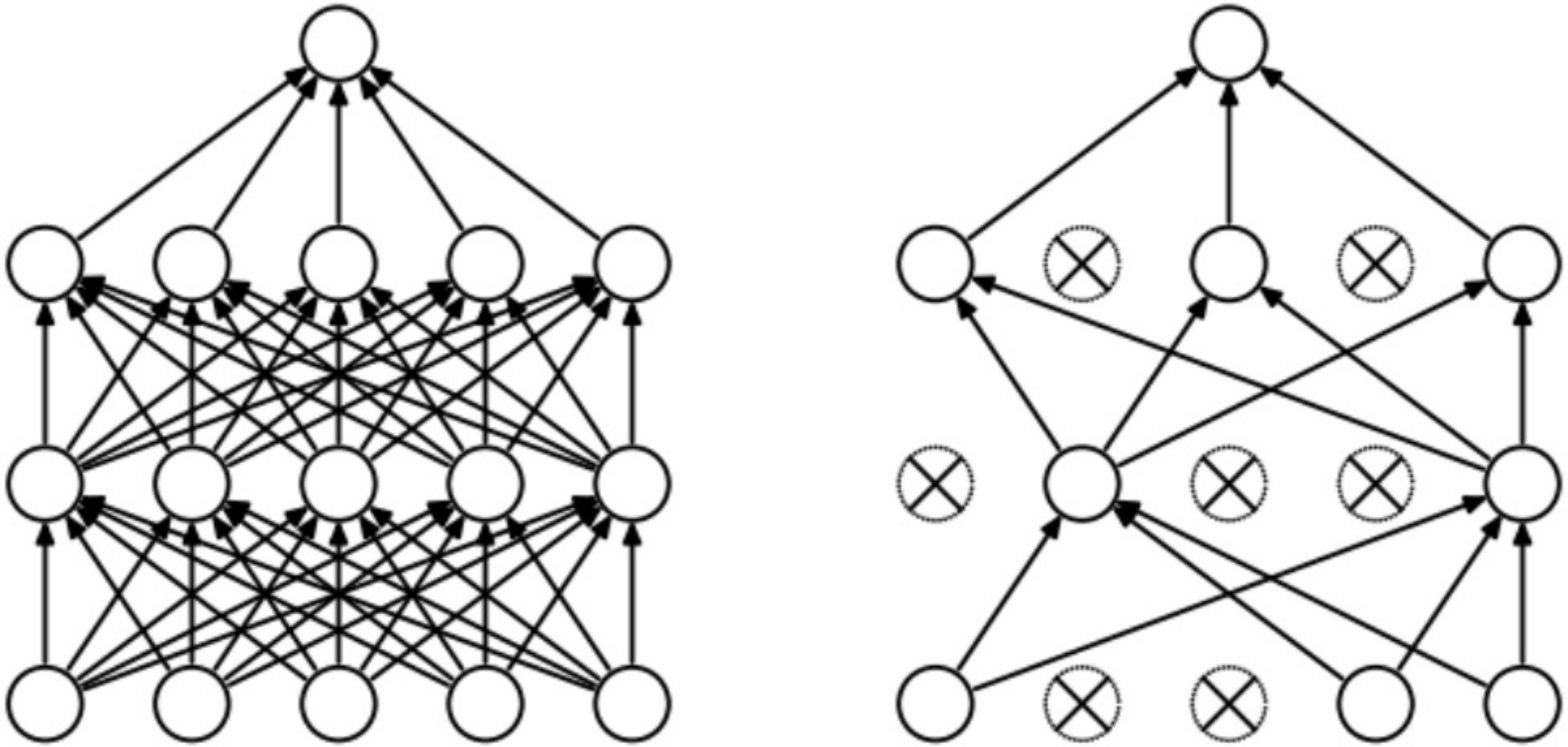


Alexnet



https://www.saagie.com/fr/blog/object-detection-part1

Dropout Layer



(a) Standard Neural Net

(b) After applying dropout.

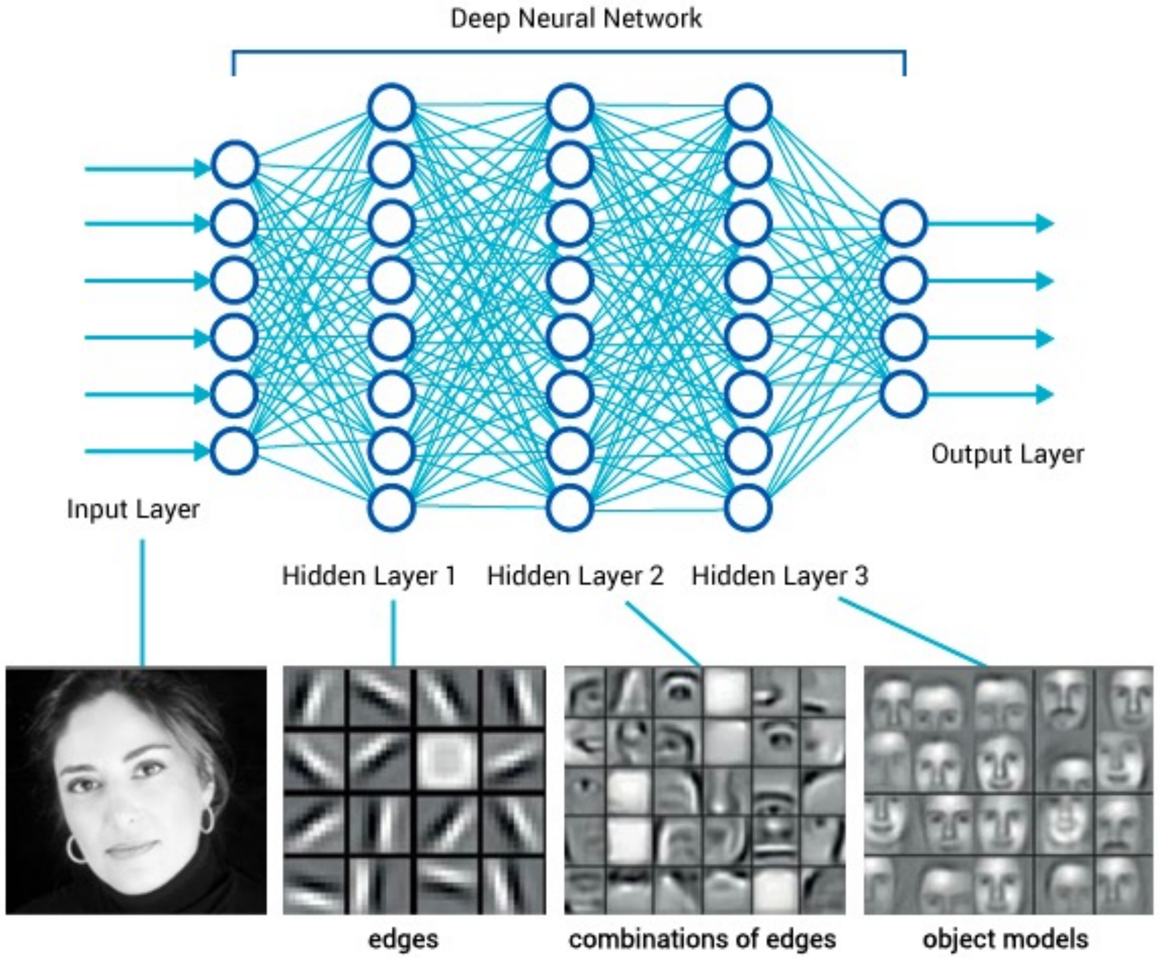
Srivastava et al 2014

Pytorch Code for Alexnet

- In-class analysis

  https://github.com/pytorch/vision/blob/master/torchvision/models/alexnet.py

What is happening?



https://www.saagie.com/fr/blog/object-detection-part1

SIFT + FV + SVM (or softmax)

→ Feature extraction (SIFT) → Feature encoding (Bag of words) → Classification (SVM or softmax) →

Deep Learning

→ Convolutional Network (includes both feature extraction and classifier) →

Preprocessing and Data Augmentation

Preprocessing and Data Augmentation

256

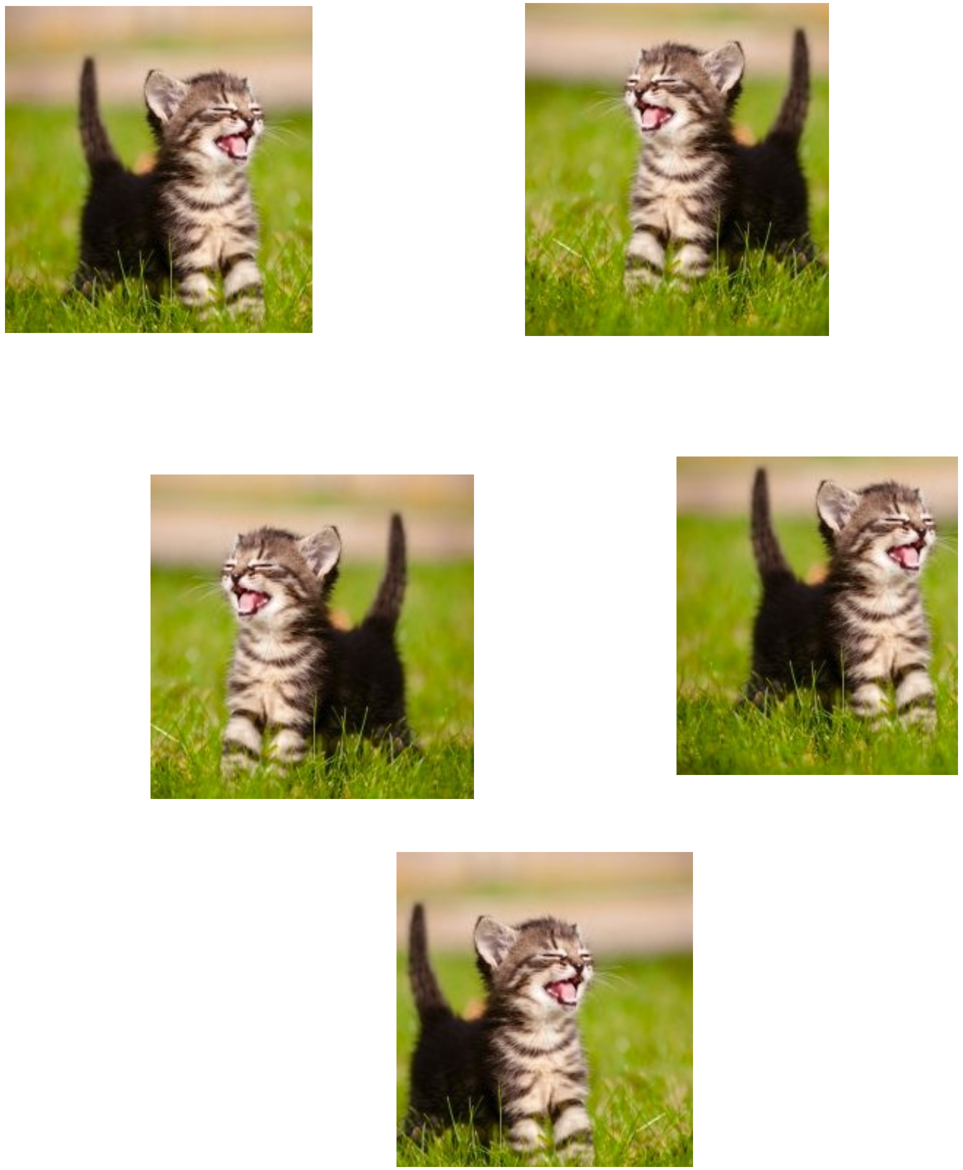
256

Preprocessing and Data Augmentation

224x224

Preprocessing and Data Augmentation

224x224



True label: Abyssinian cat

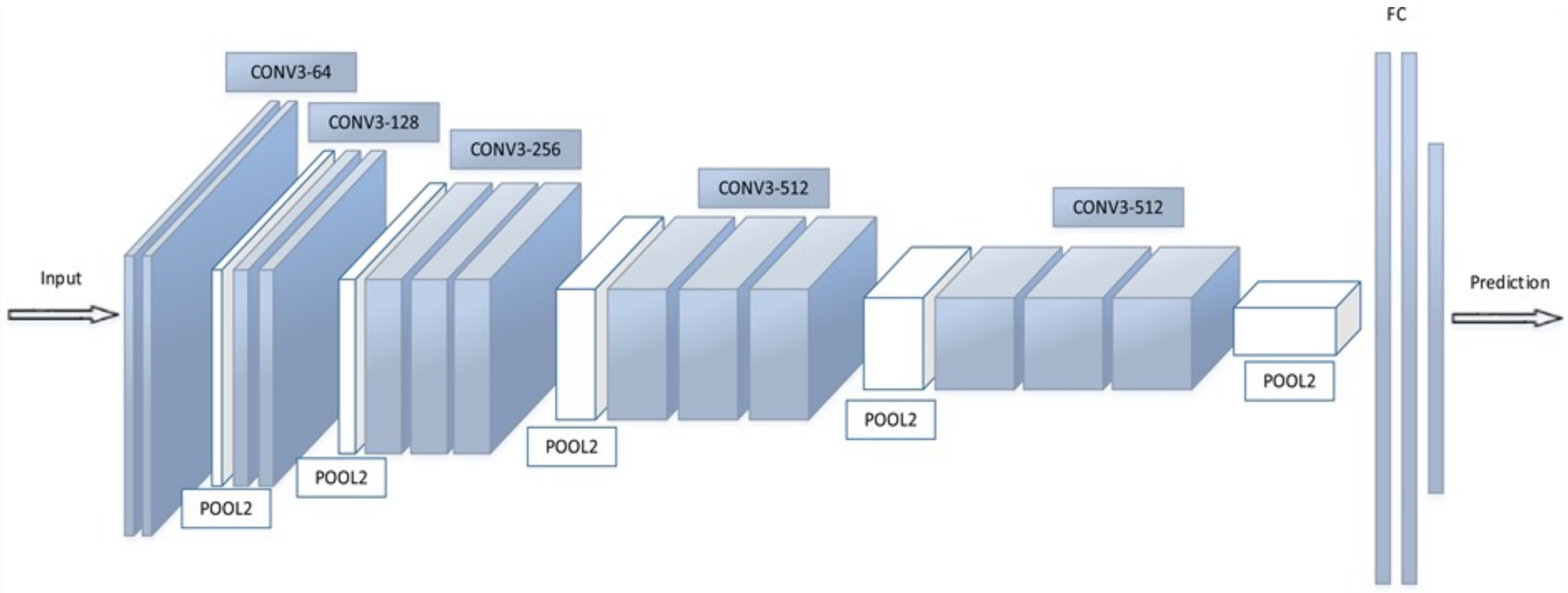
Other Important Aspects

- Using ReLUs instead of Sigmoid or Tanh
- Momentum + Weight Decay
- Dropout (Randomly sets Unit outputs to zero during training)
- GPU Computation!

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

VGG Network

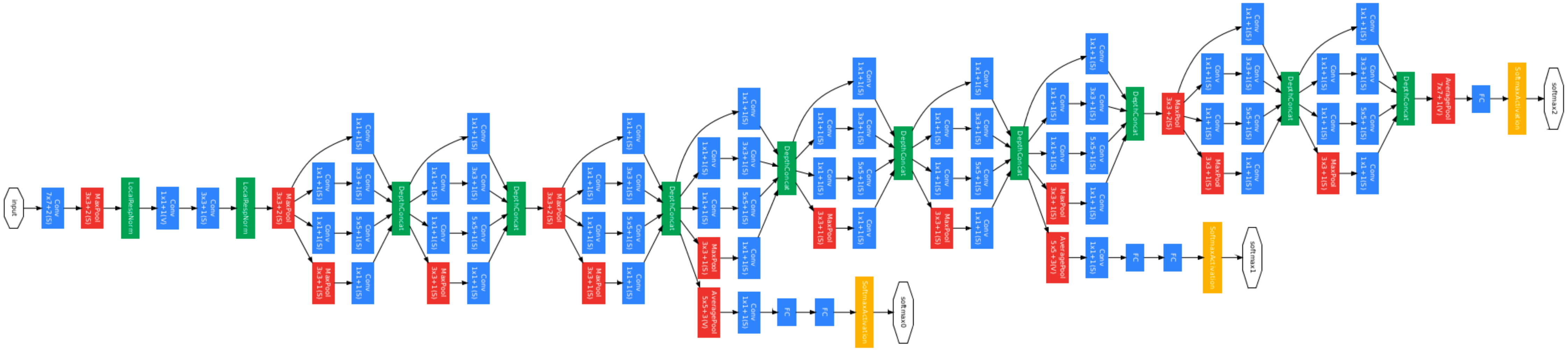
Top-5:



https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py

Simonyan and Zisserman, 2014.
https://arxiv.org/pdf/1409.1556.pdf

GoogLeNet

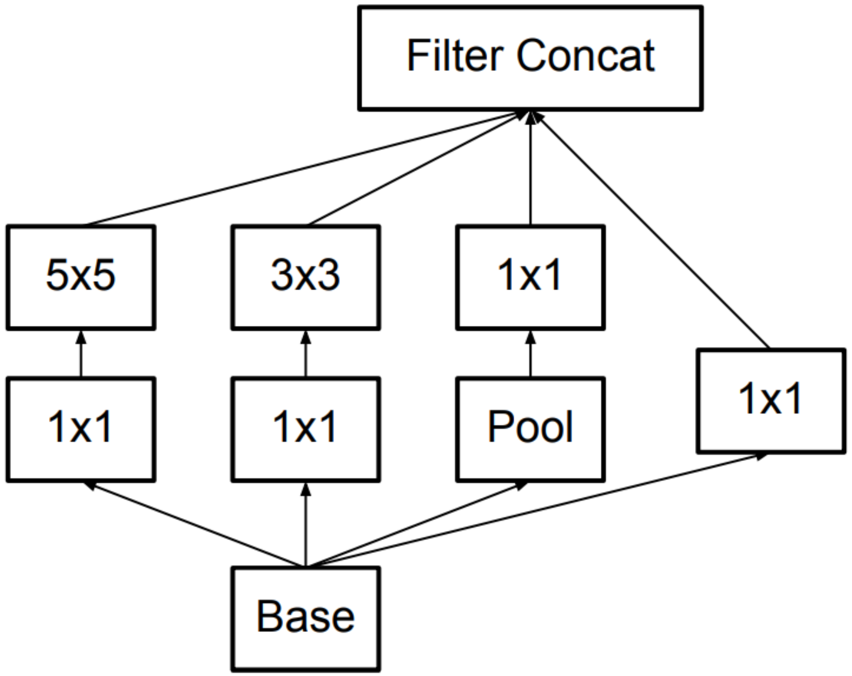


https://github.com/kuangliu/pytorch-cifar/blob/master/models/googlenet.py

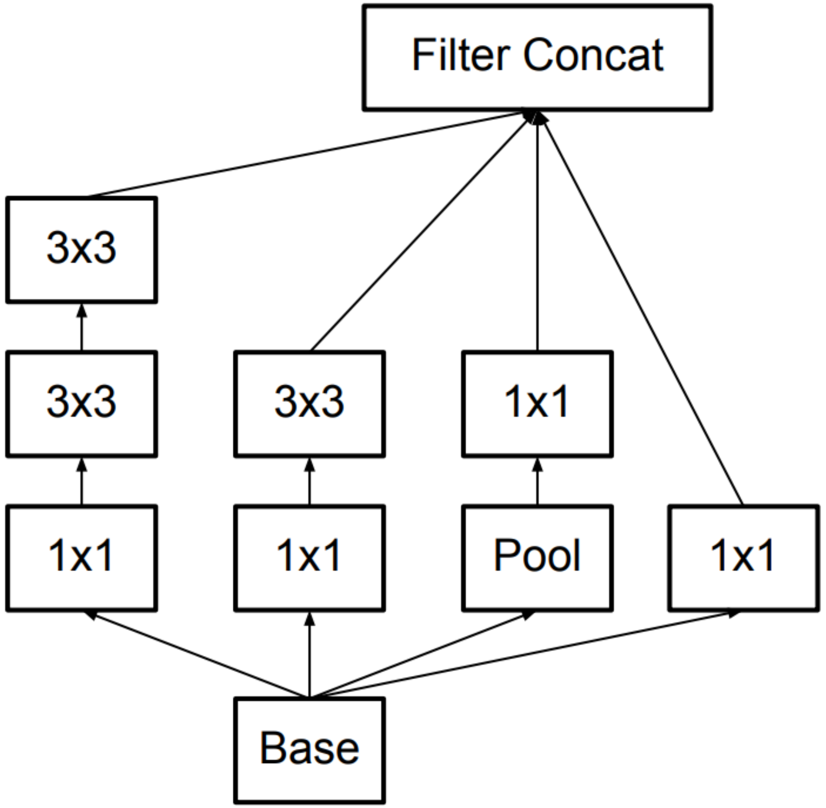
Szegedy et al. 2014

https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

Further Refinements – Inception v3, e.g.



GoogLeNet (Inceptionv1)



Inception v3

BatchNormalization Layer

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

https://arxiv.org/abs/1502.03167

ResNet (He et al CVPR 2016)

Sorry, does not fit in slide.

http://felixlaumon.github.io/assets/kaggle-right-whale/resnet.png

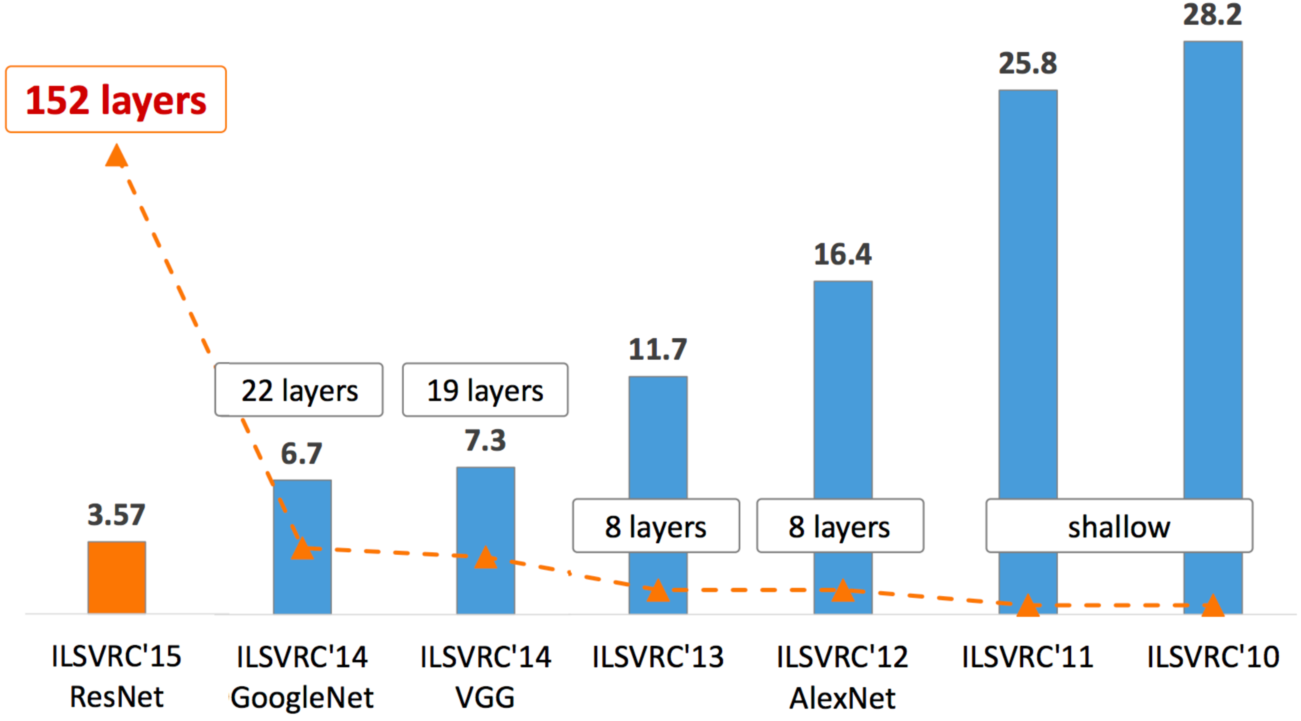
https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)



Slide by Mohammad Rastegari

Questions