CS6501: Deep Learning for Visual Recognition

Convolutional Neural Networks

Today's Class

Automatic Differentiation (AutoGrad)

Convolutional Neural Networks

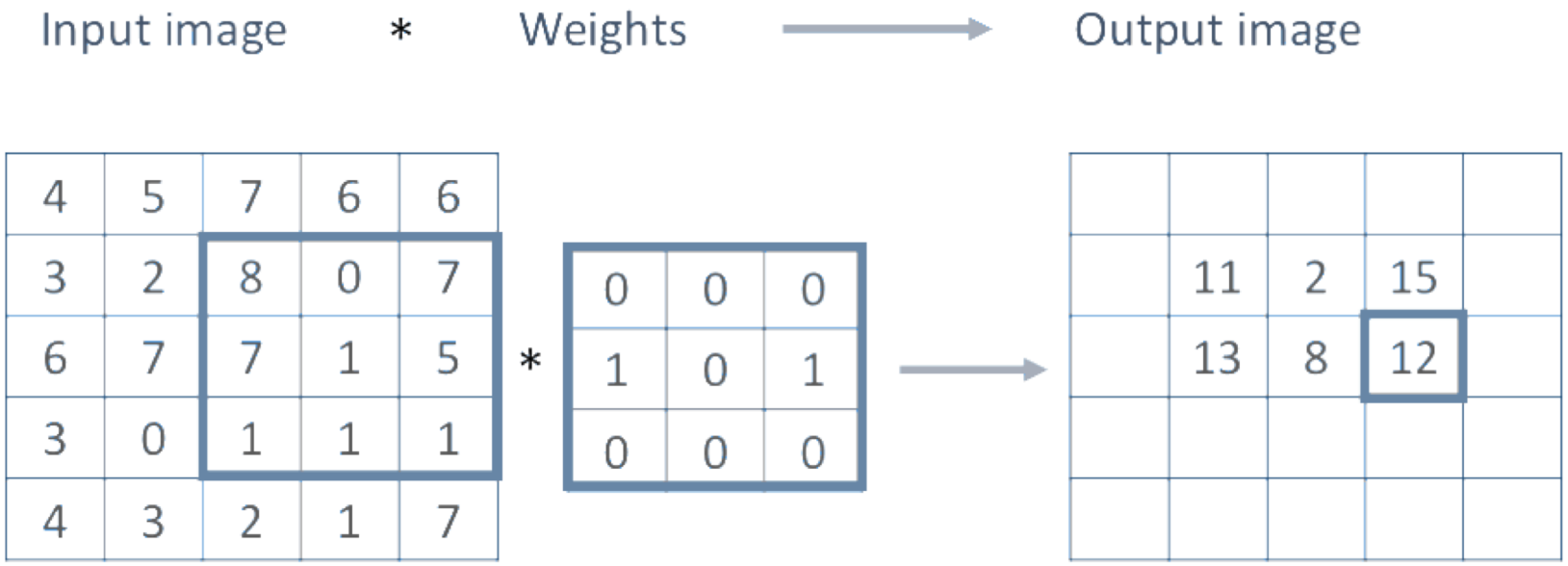
- Revisiting Convolutions
- The Convolutional Layer
- Strided Convolutions / Grouped Convolutions / Dilated Convolutions
- Spatial Pooling Operations

Automatic Differentiation

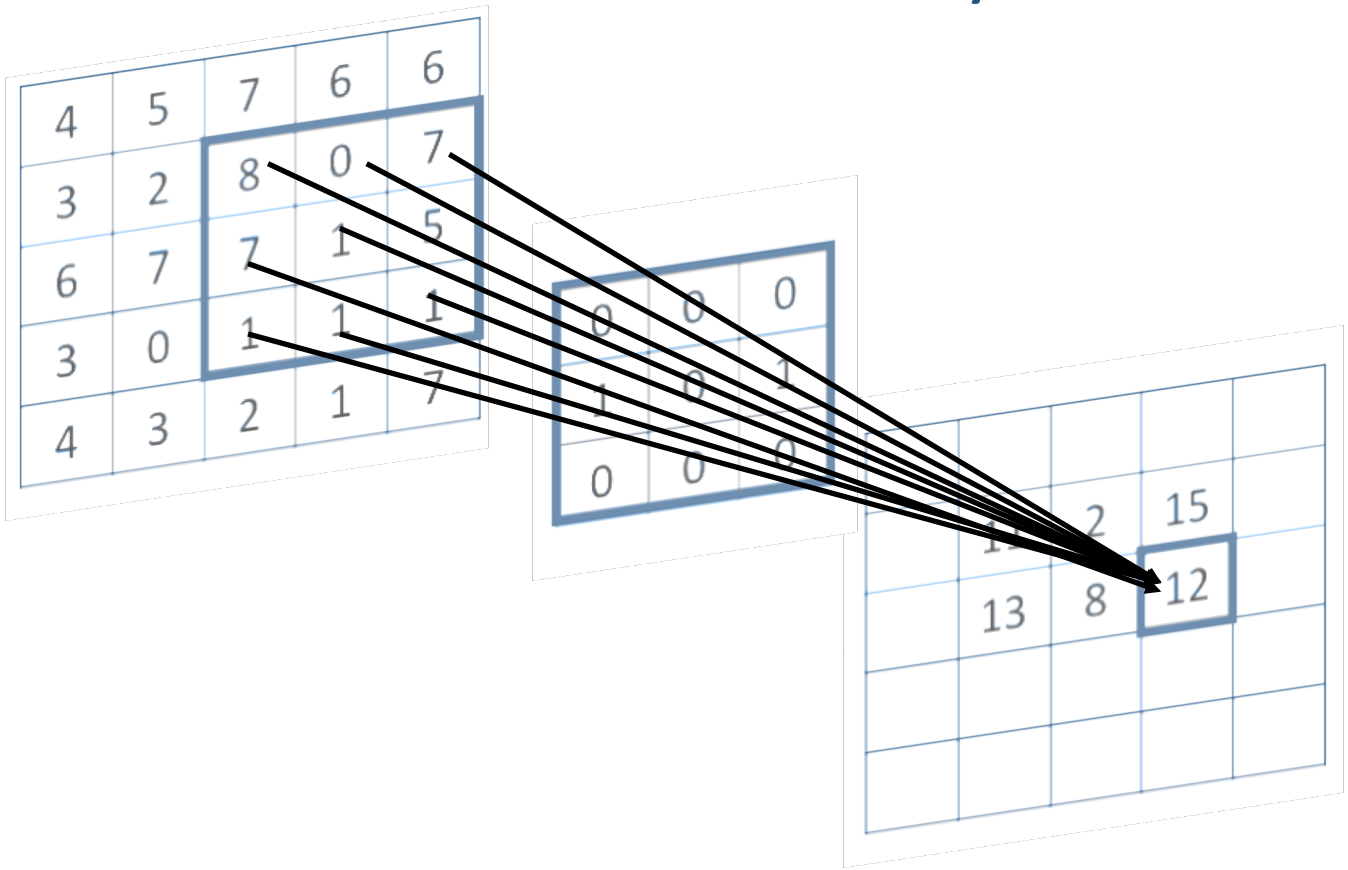
You only need to write code for the forward pass,
backward pass is computed automatically.

Pytorch (Facebook -- mostly):	https://pytorch.org/
Tensorflow (Google -- mostly):	https://www.tensorflow.org/
DyNet (team includes UVA Prof. Yangfeng Ji):	http://dynet.io/

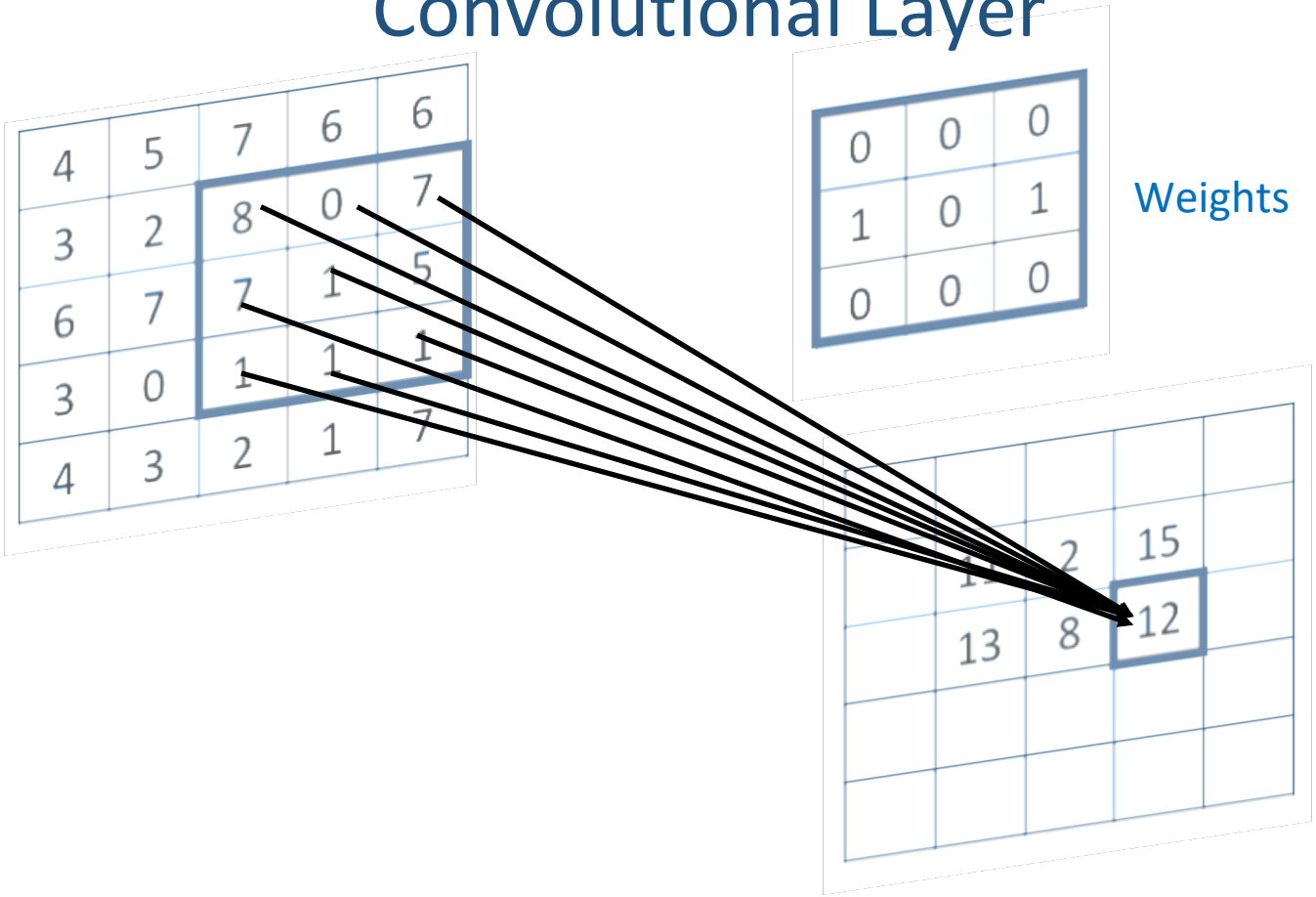
Convolutional Layer



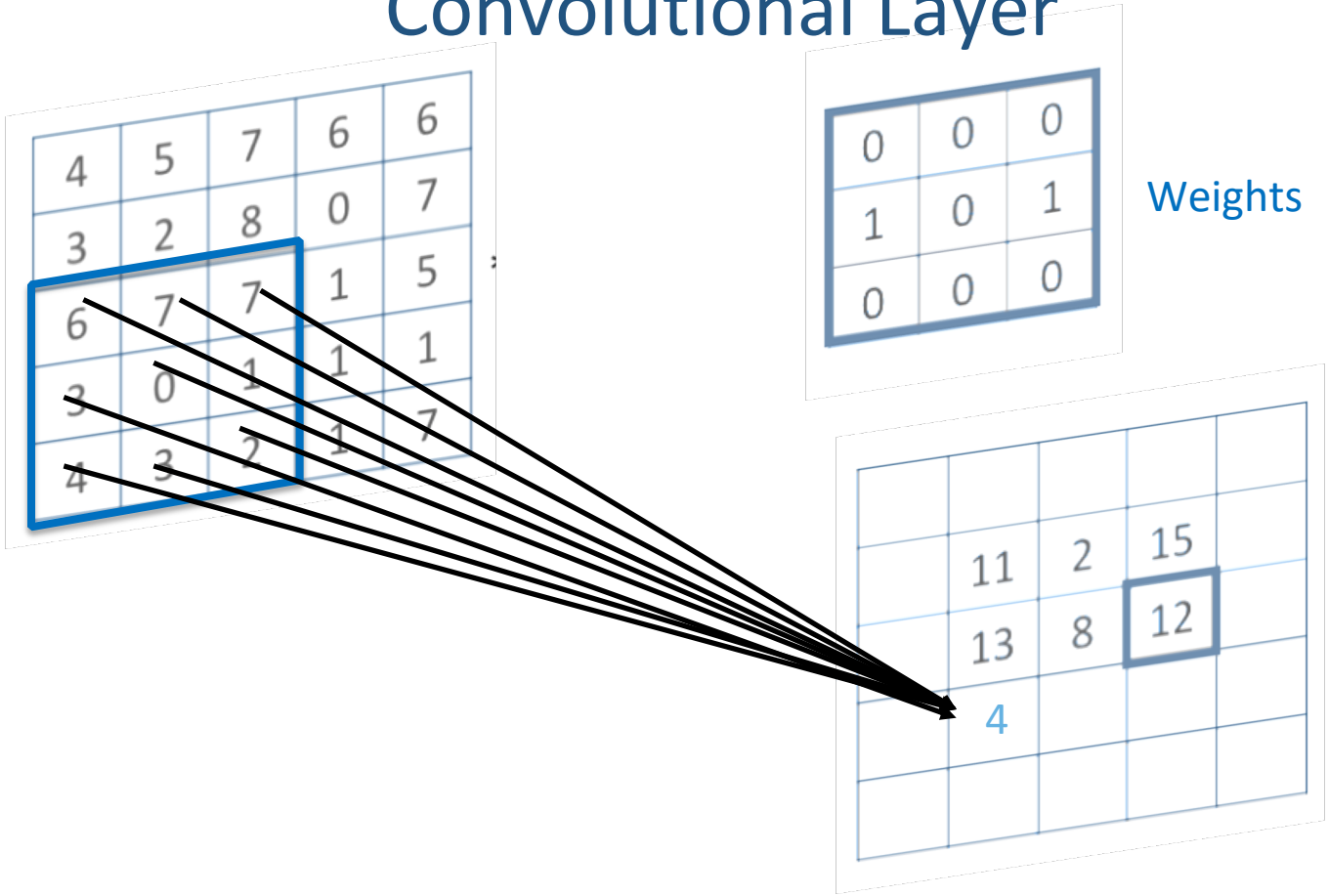
Convolutional Layer



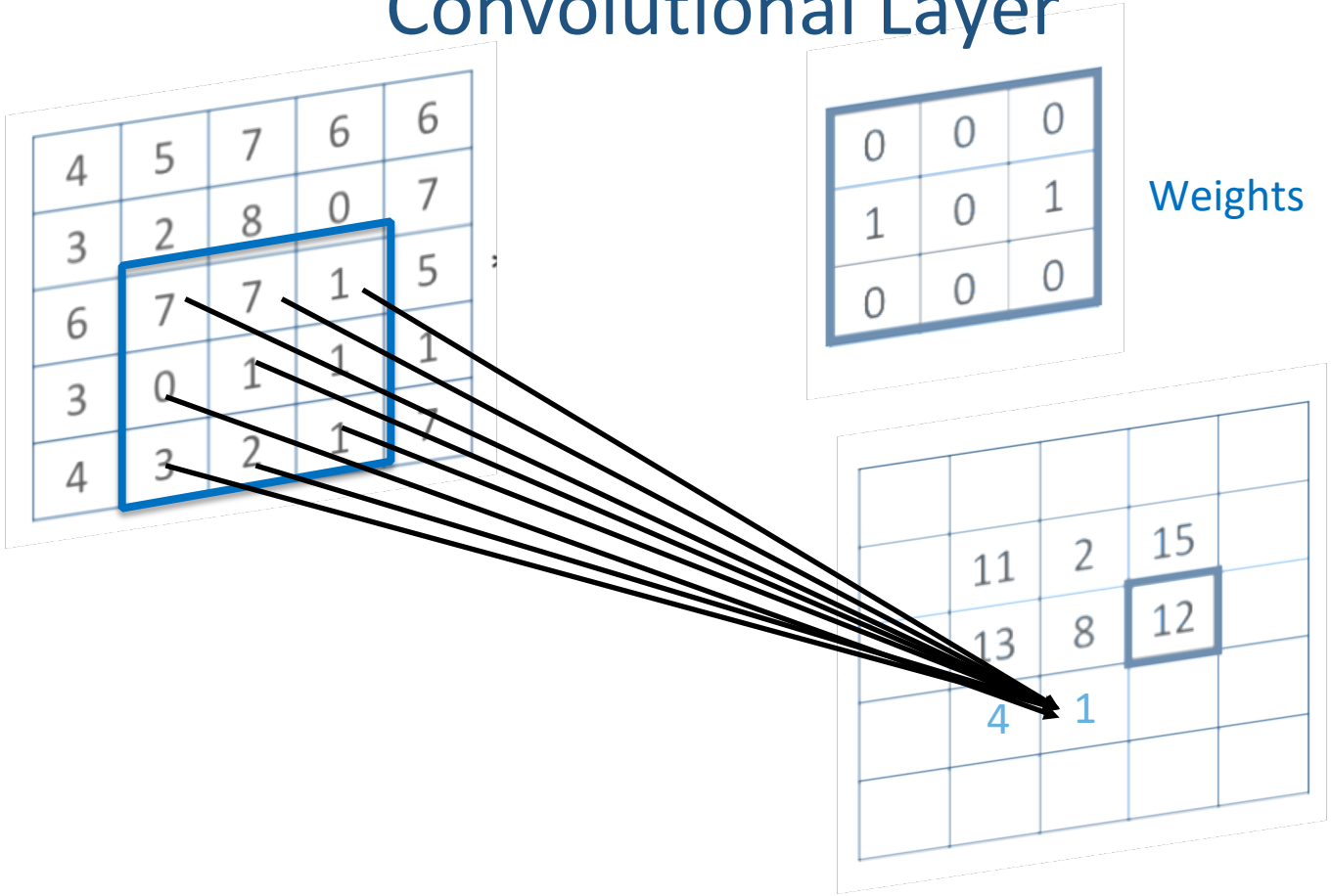
Convolutional Layer



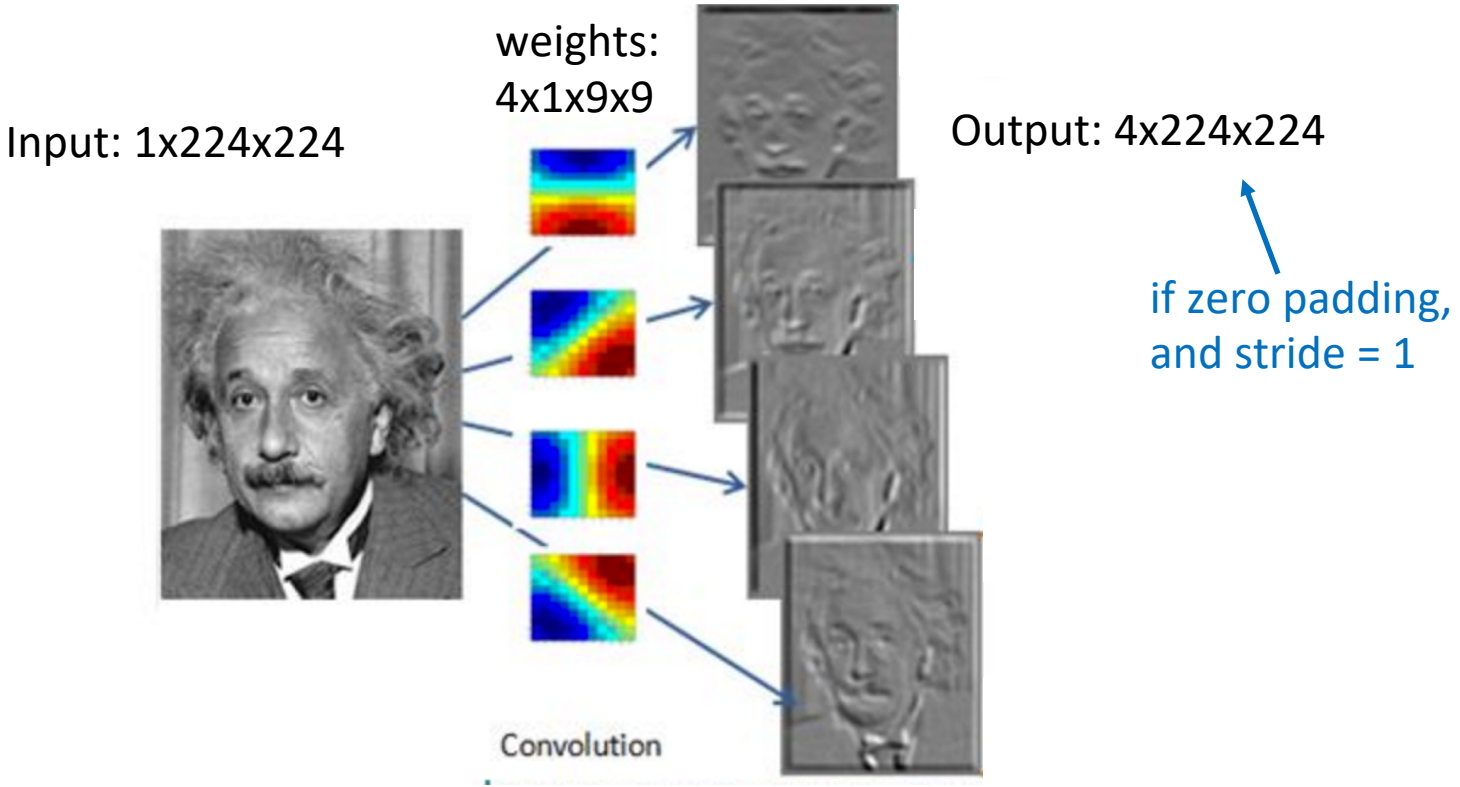
Convolutional Layer



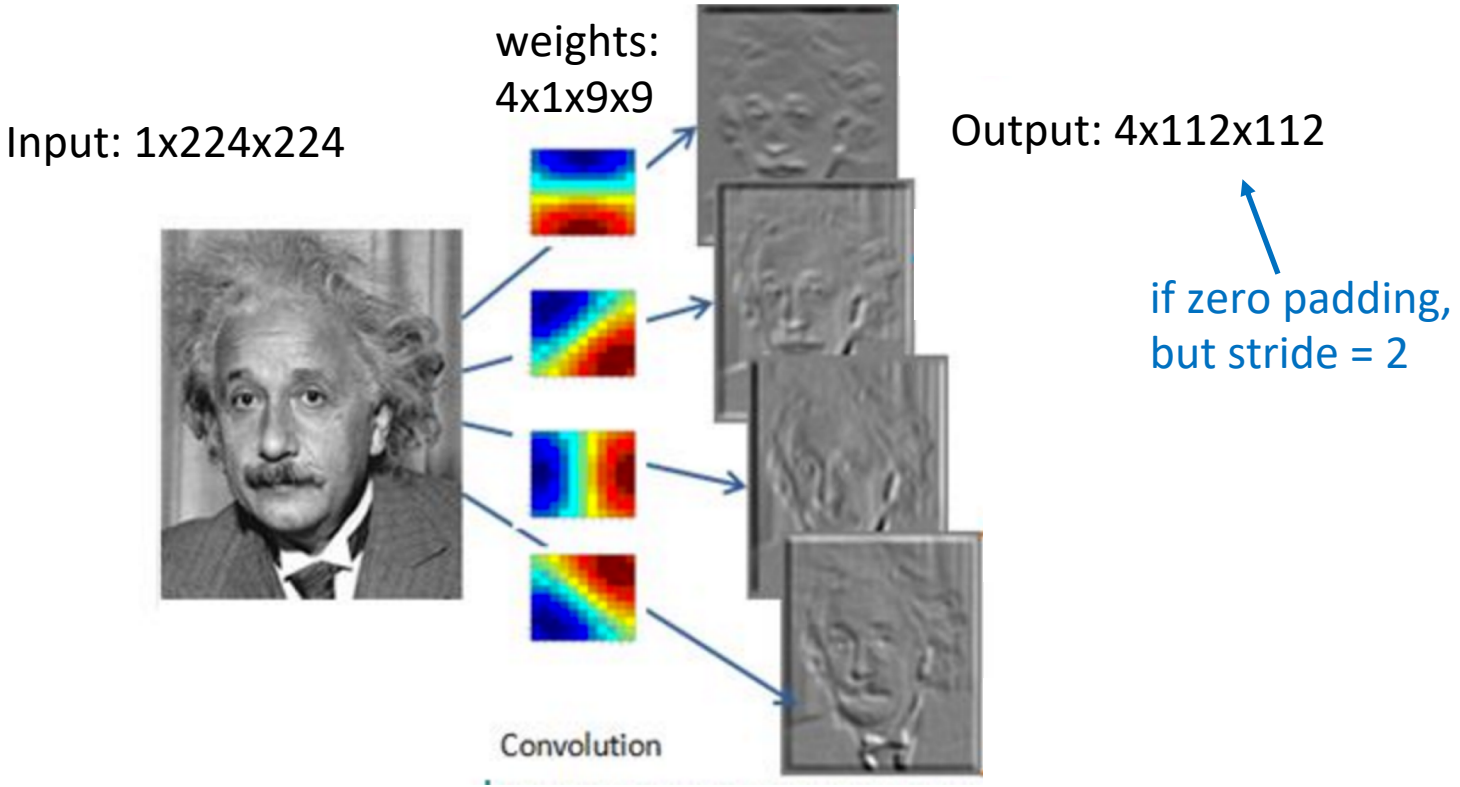
Convolutional Layer



Convolutional Layer (with 4 filters)

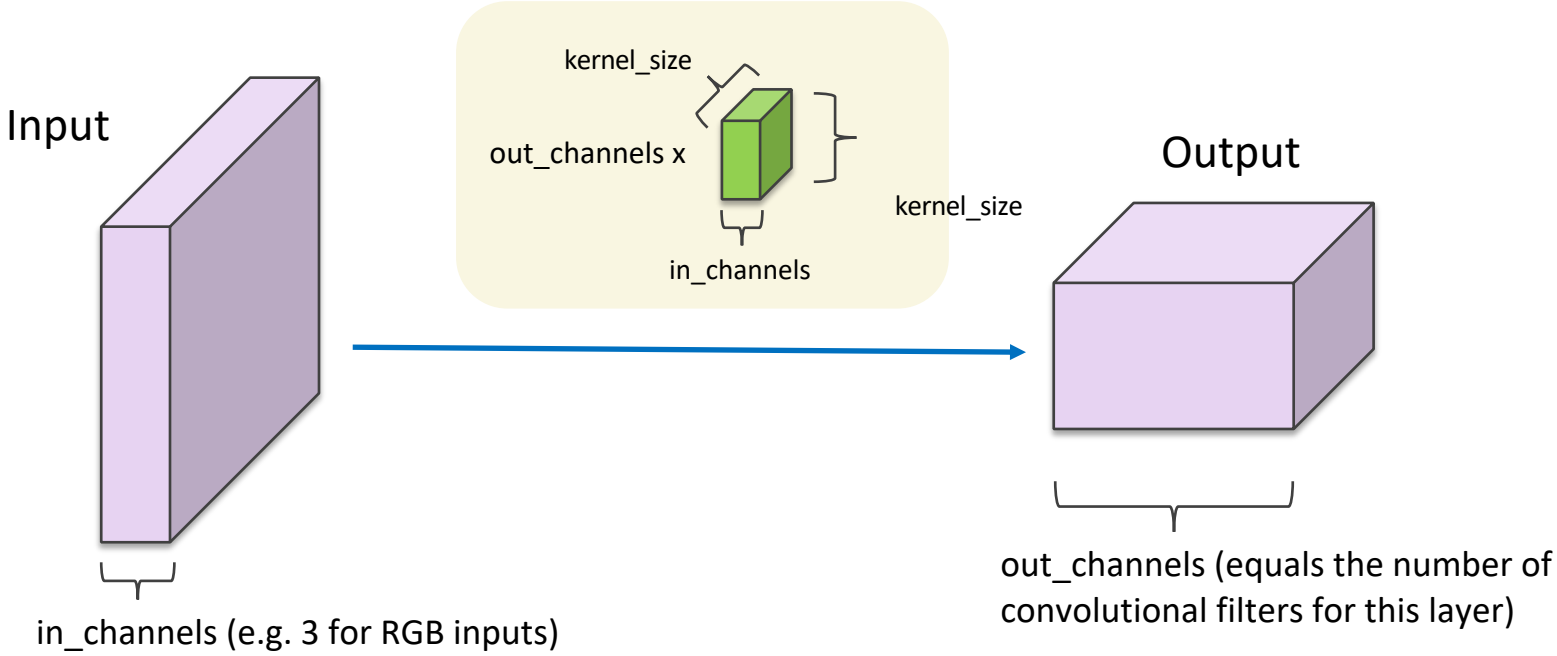


Convolutional Layer (with 4 filters)

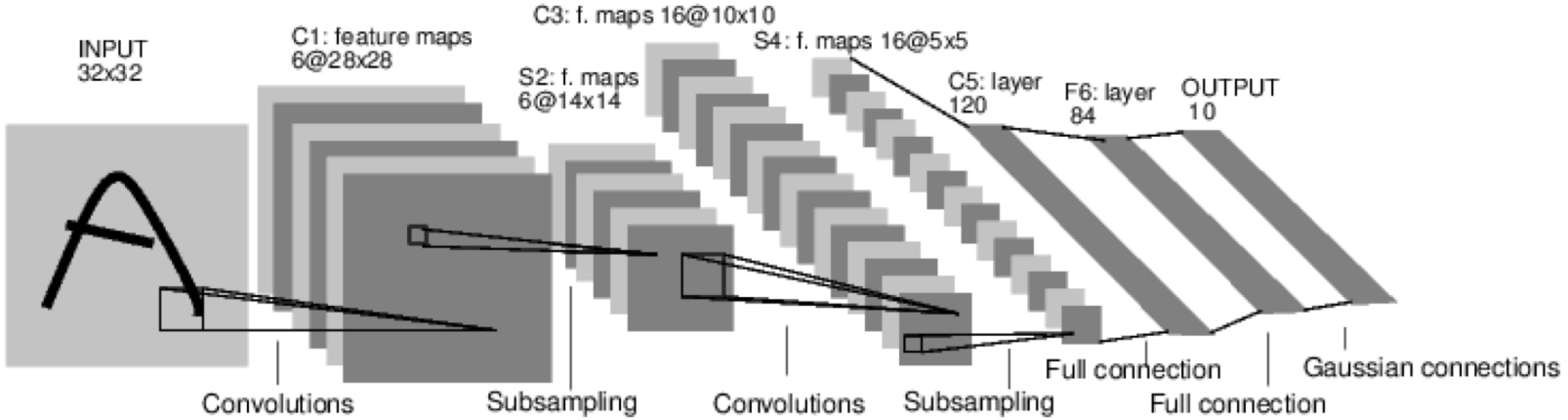


Convolutional Layer in pytorch

```
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

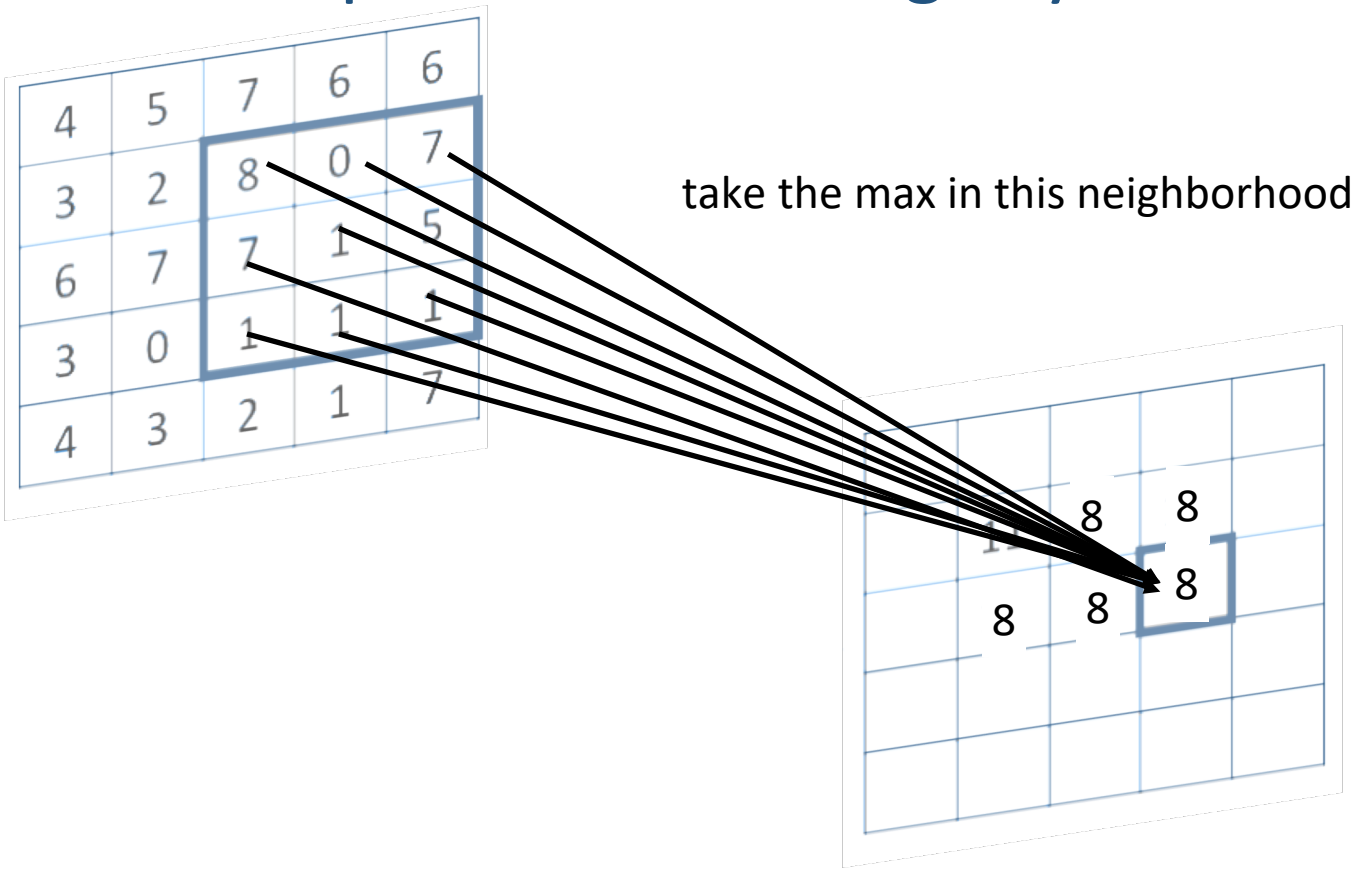
```
# LeNet is French for The Network, and is taken from Yann Lecun's 98 paper
# on digit classification http://yann.lecun.com/exdb/lenet/
# This was also a network with just two convolutional layers.
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        # Convolutional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



LeNet Summary

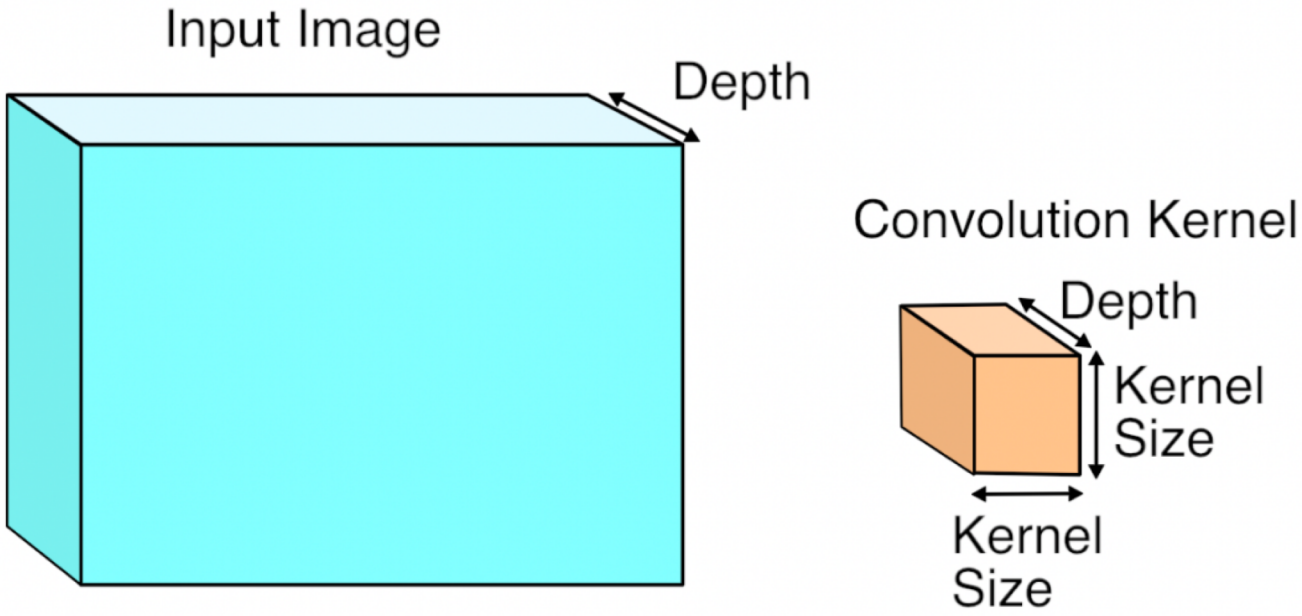
- 2 Convolutional Layers + 3 Linear Layers

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

New Architectures Proposed

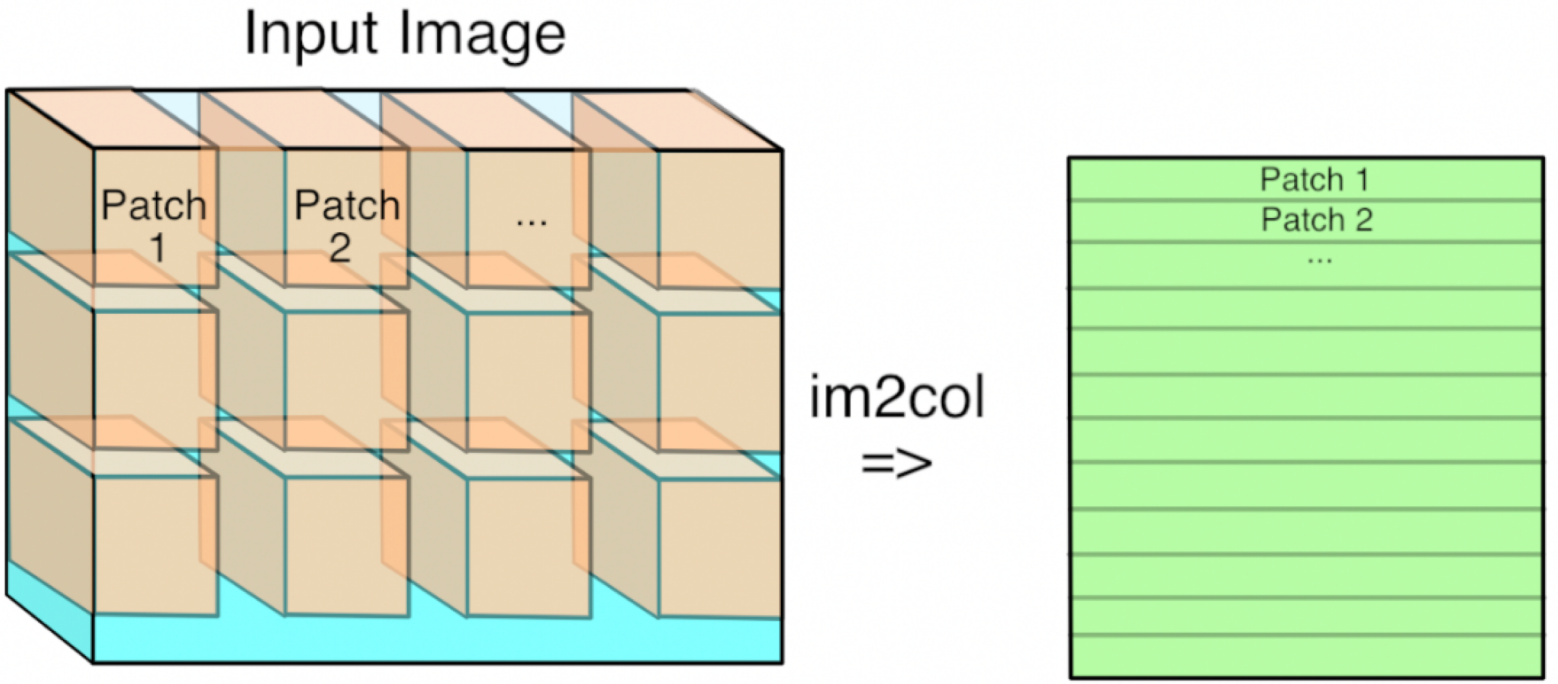
- Alexnet (Kriszhevsky et al NIPS 2012) **[Required Reading]**
- VGG (Simonyan and Zisserman 2014)
- GoogLeNet (Szegedy et al CVPR 2015)
- ResNet (He et al CVPR 2016)
- DenseNet (Huang et al CVPR 2017)

Convolutional Layers as Matrix Multiplication



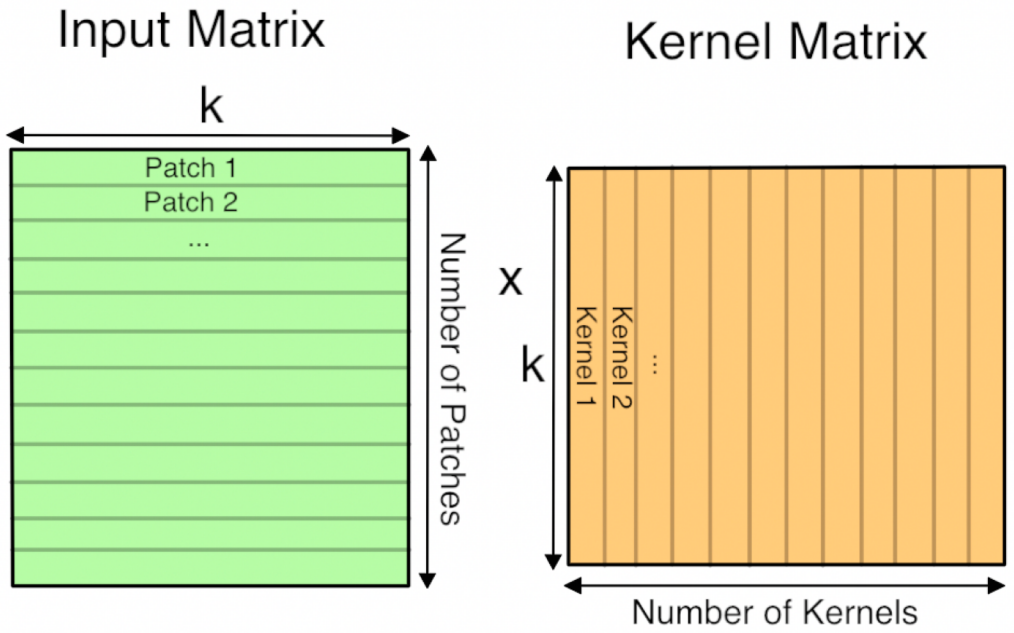
https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

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

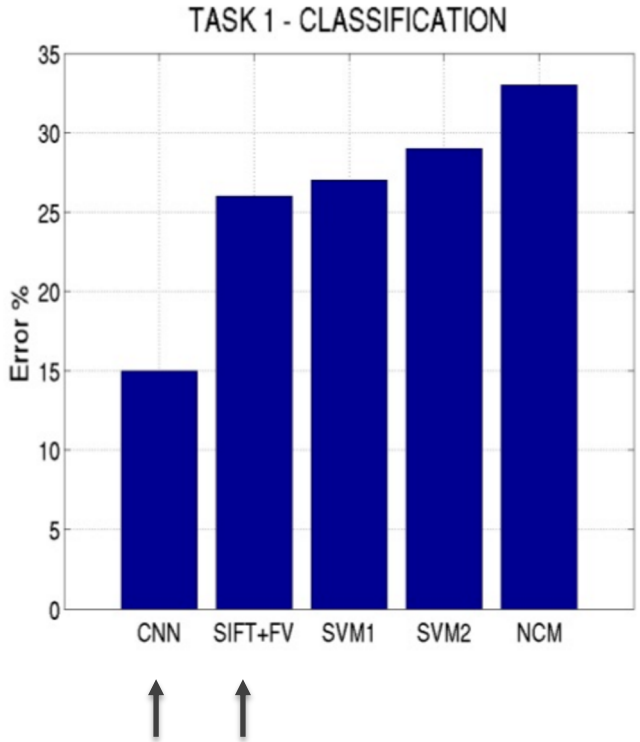
True label: Abyssinian cat

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

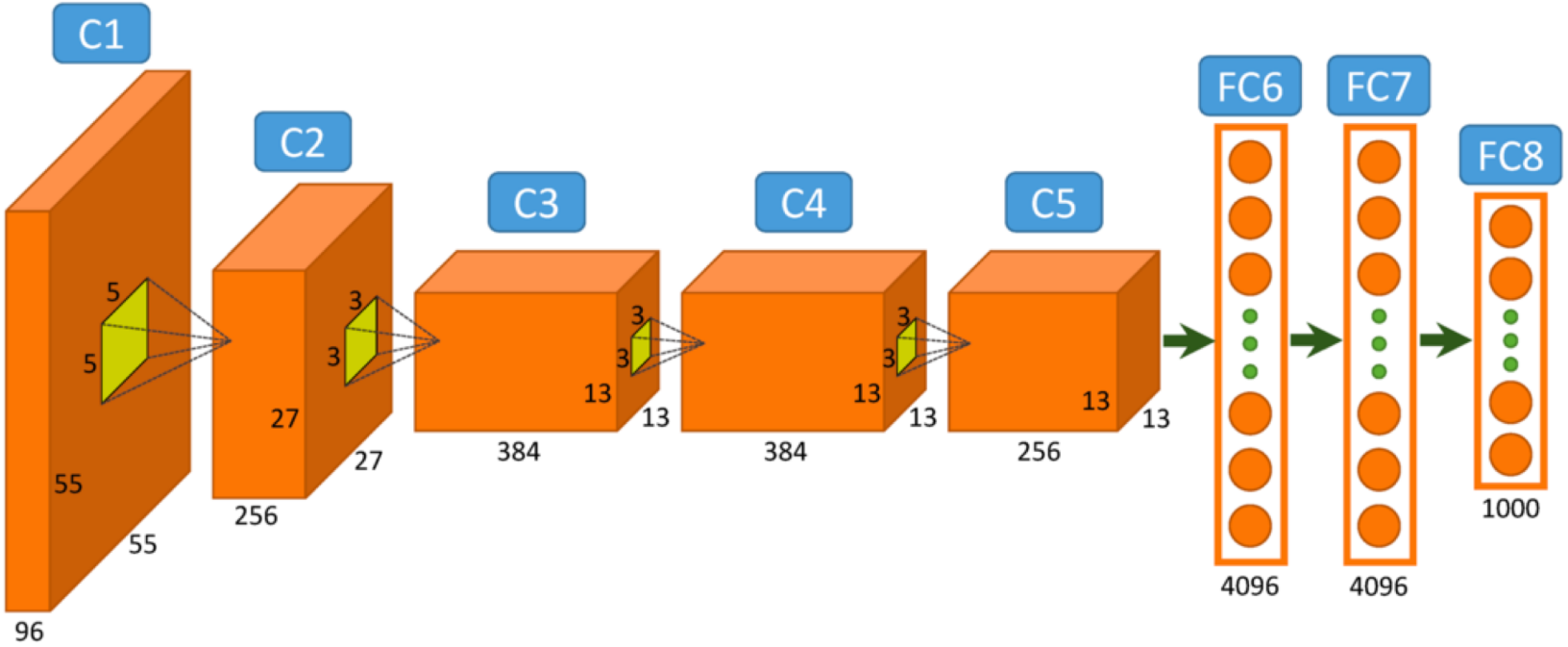
Top-1 accuracy: 0.0
Top-2 accuracy: 0.0
Top-3 accuracy: 0.0
Top-4 accuracy: 1.0
Top-5 accuracy: 1.0

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



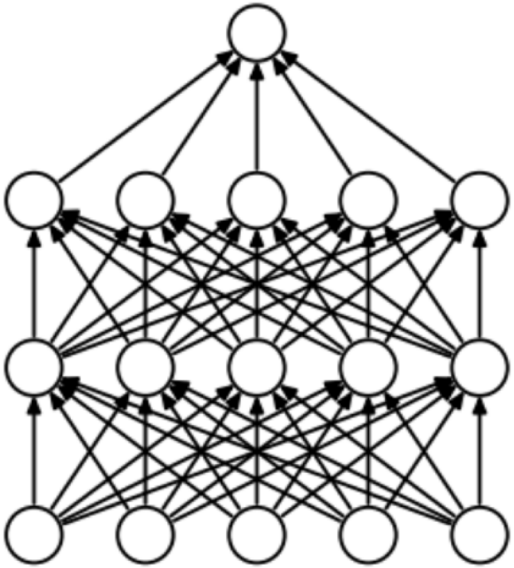
https://www.saagie.com/fr/blog/object-detection-part1

Pytorch Code for Alexnet

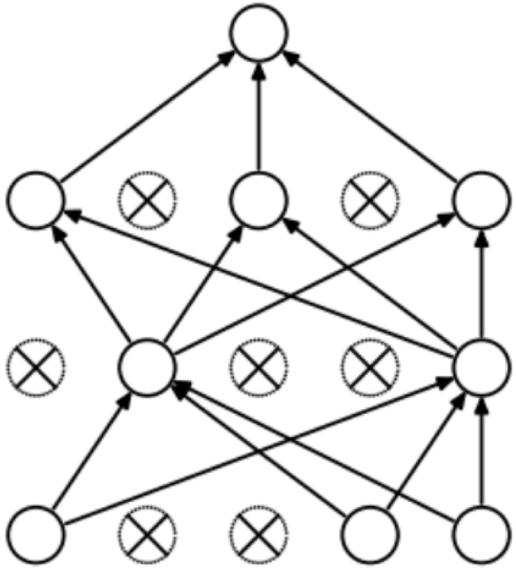
- In-class analysis

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

Dropout Layer



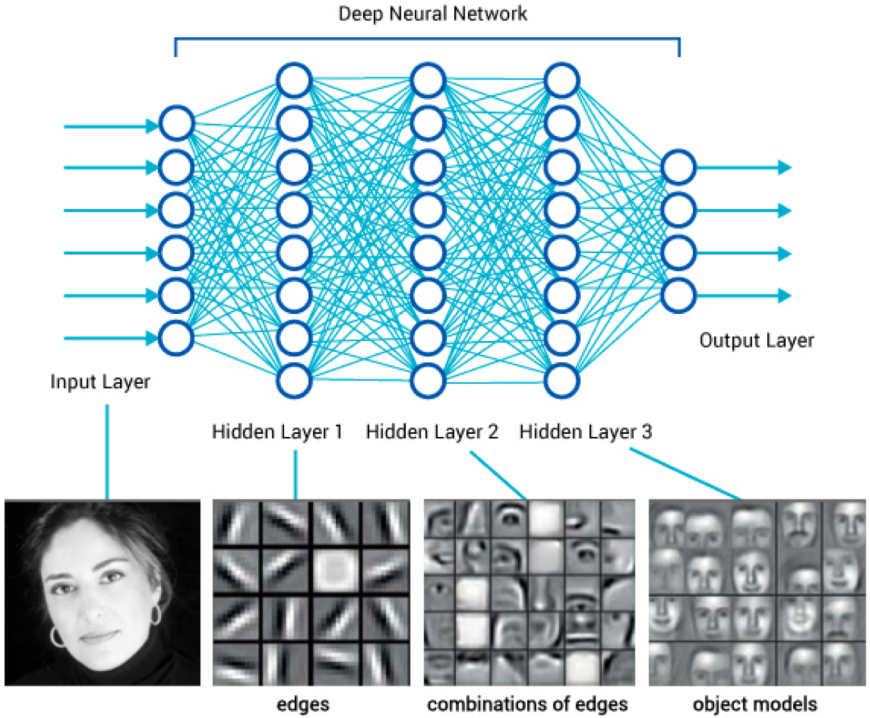
(a) Standard Neural Net



(b) After applying dropout.

Srivastava et al 2014

What is happening?



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

SIFT + FV + SVM (or softmax)

Feature extraction (SIFT)

Feature encoding (Fisher vectors)

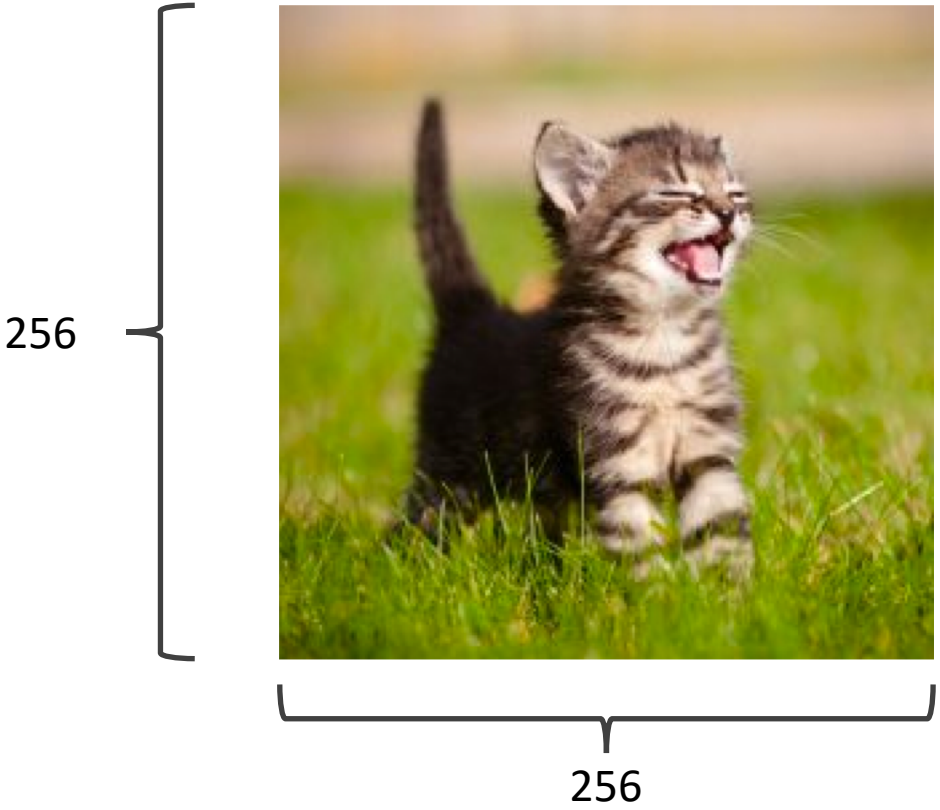
Classification (SVM or softmax)

Deep Learning

Convolutional Network
(includes both feature extraction and classifier)

Preprocessing and Data Augmentation

Preprocessing and Data Augmentation



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%**

Questions?