Deep Learning for Vision & Language

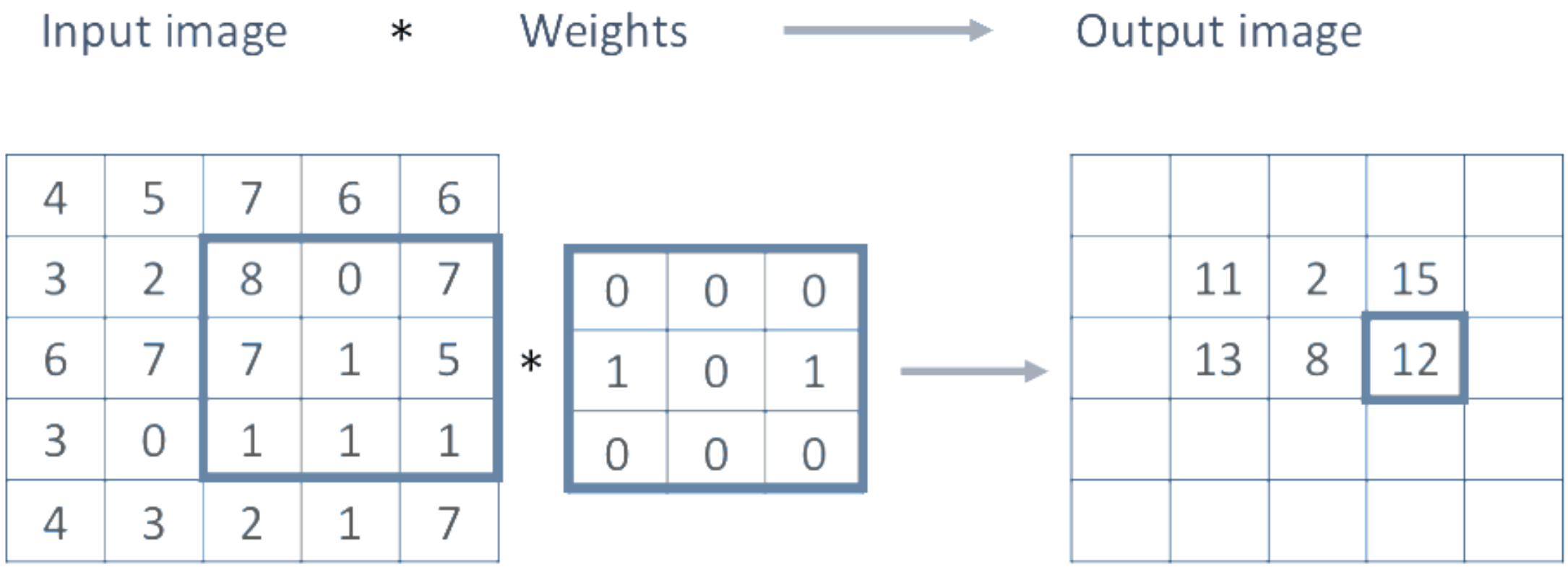
Computer Vision II: Convolutional Neural Network Architectures

Assignment 2

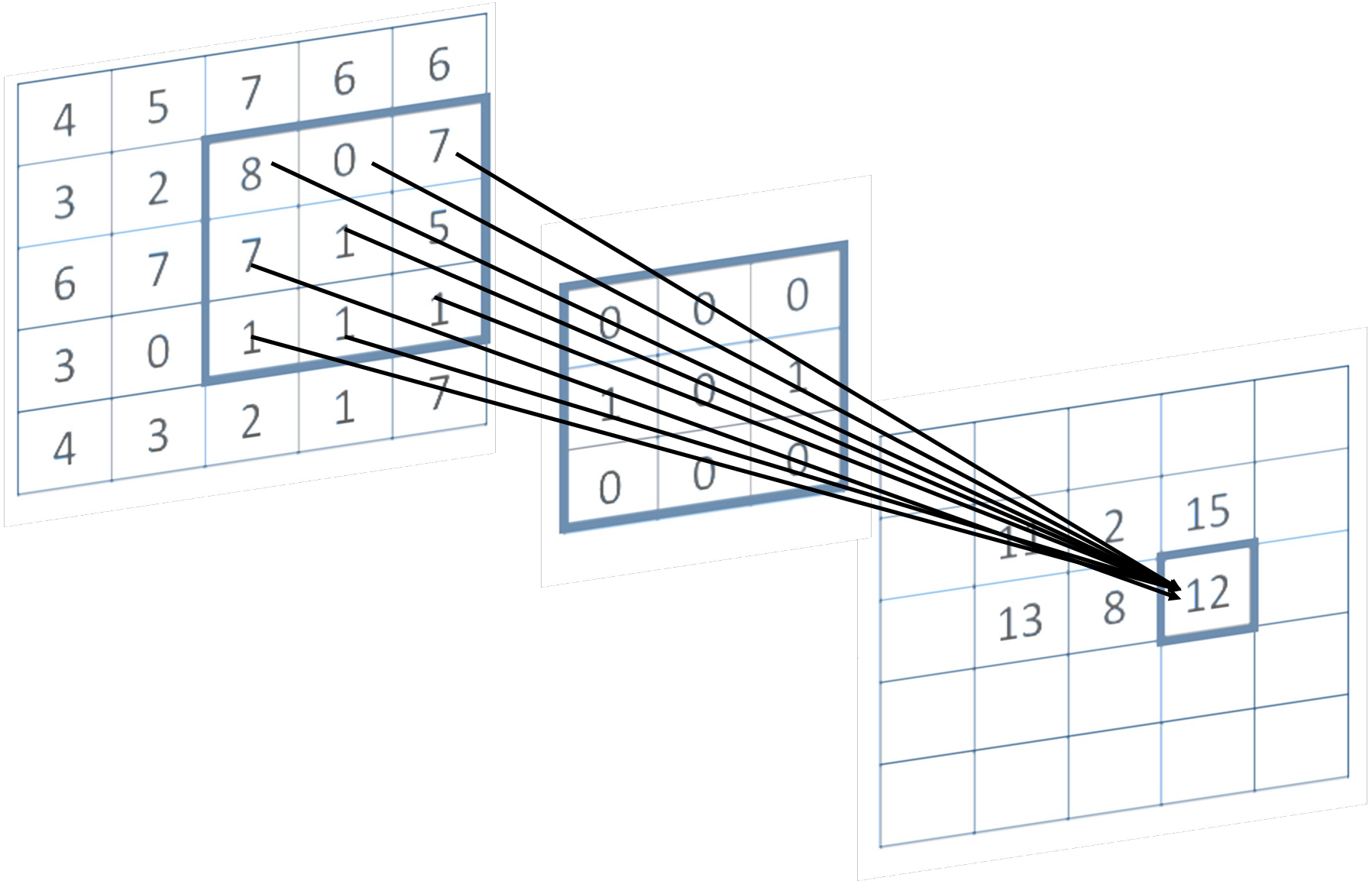
- Assignment 2 is released and is available on the class website and to be submitted via Canvas.

- Due: Monday February 12th.

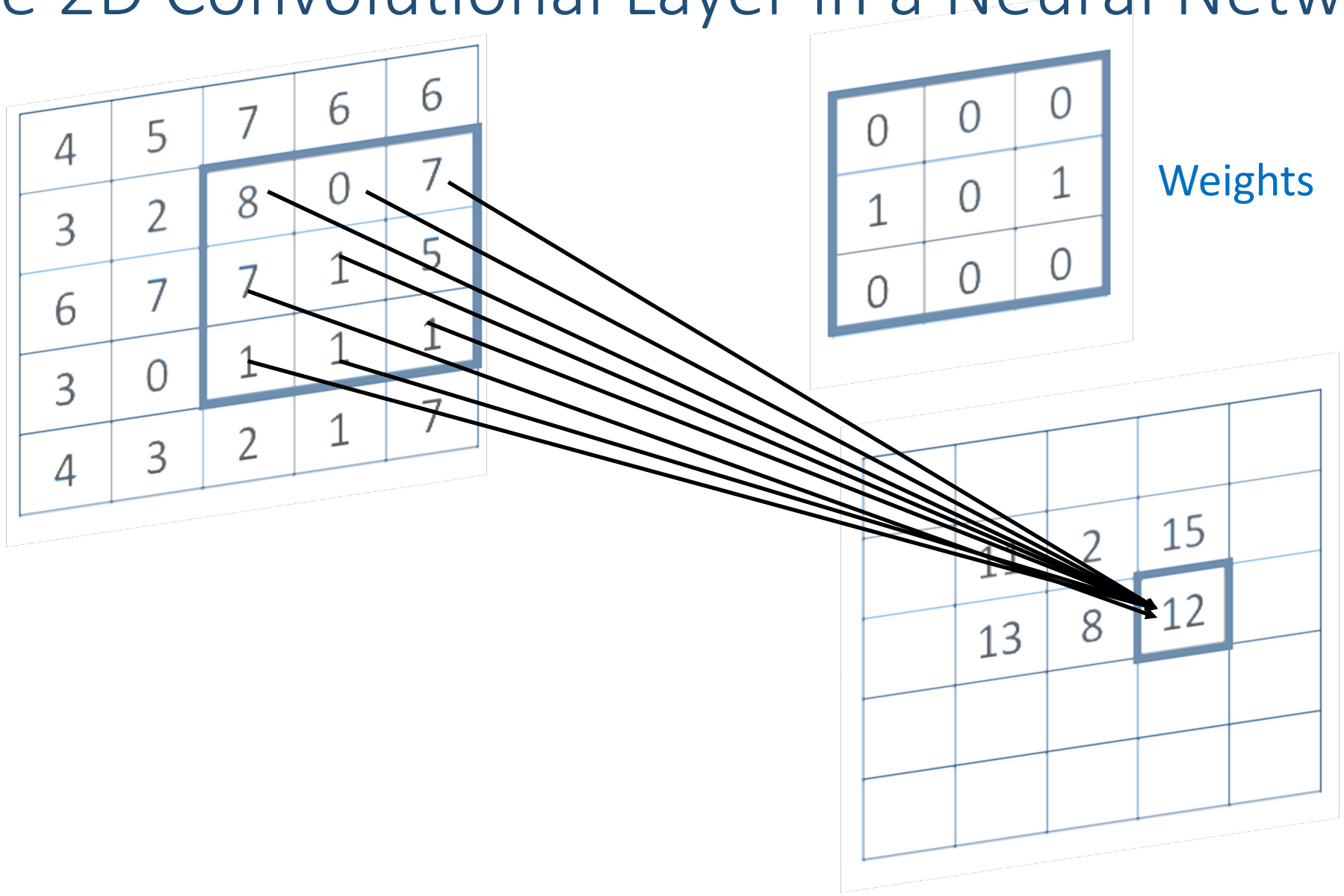
The 2D Convolutional Layer in a Neural Network



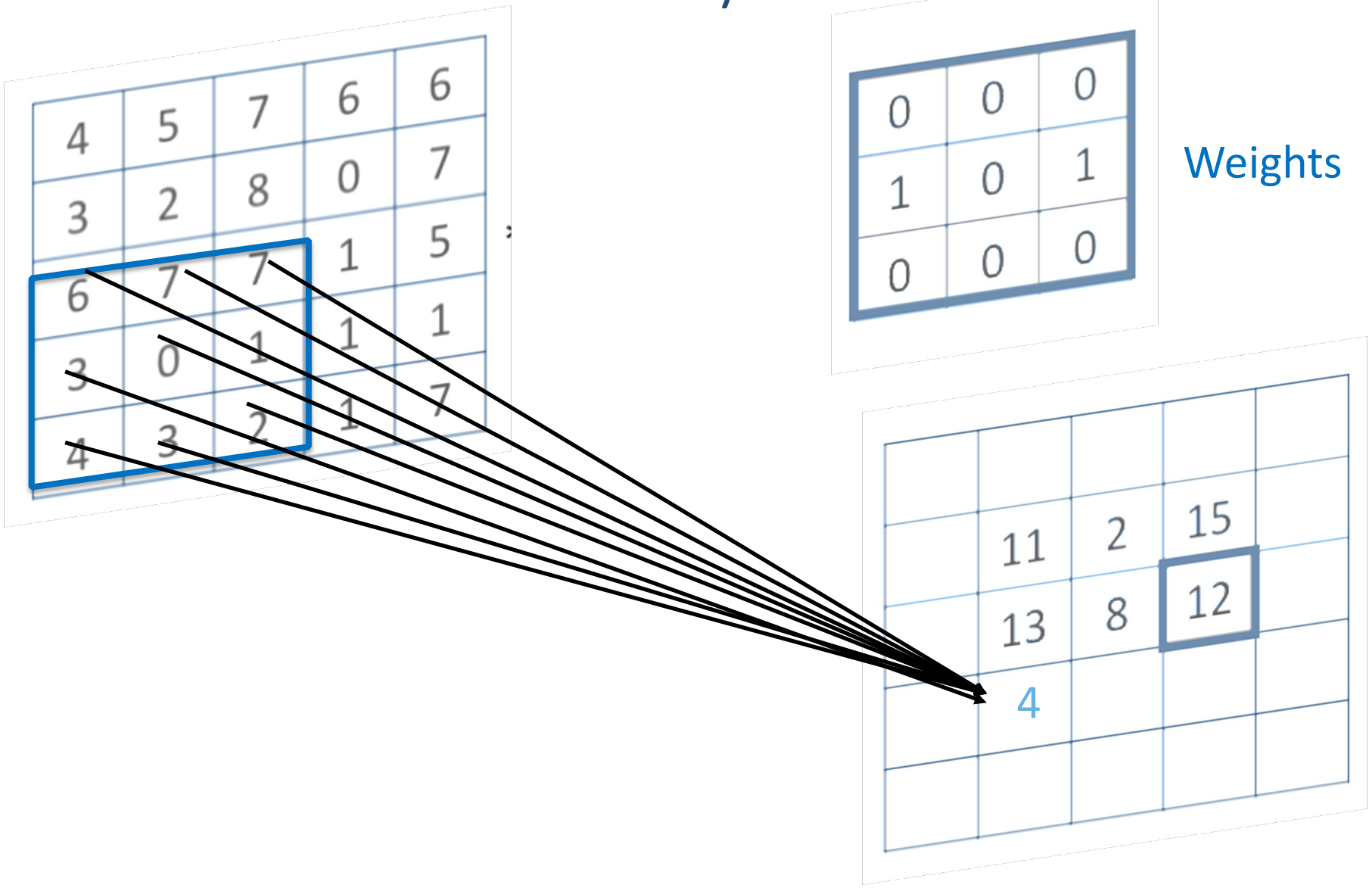
The 2D Convolutional Layer in a Neural Network



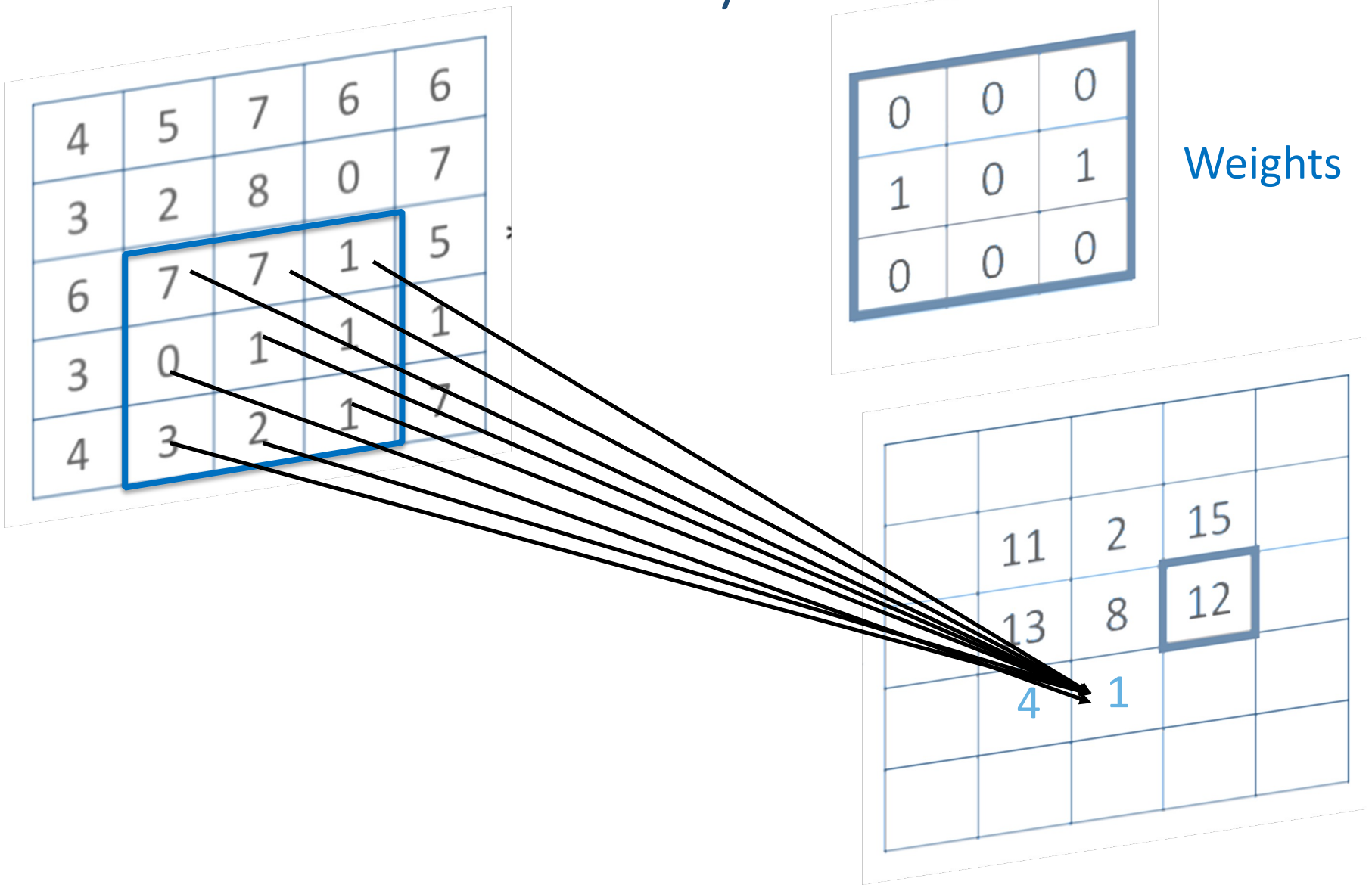
The 2D Convolutional Layer in a Neural Network



The 2D Convolutional Layer in a Neural Network



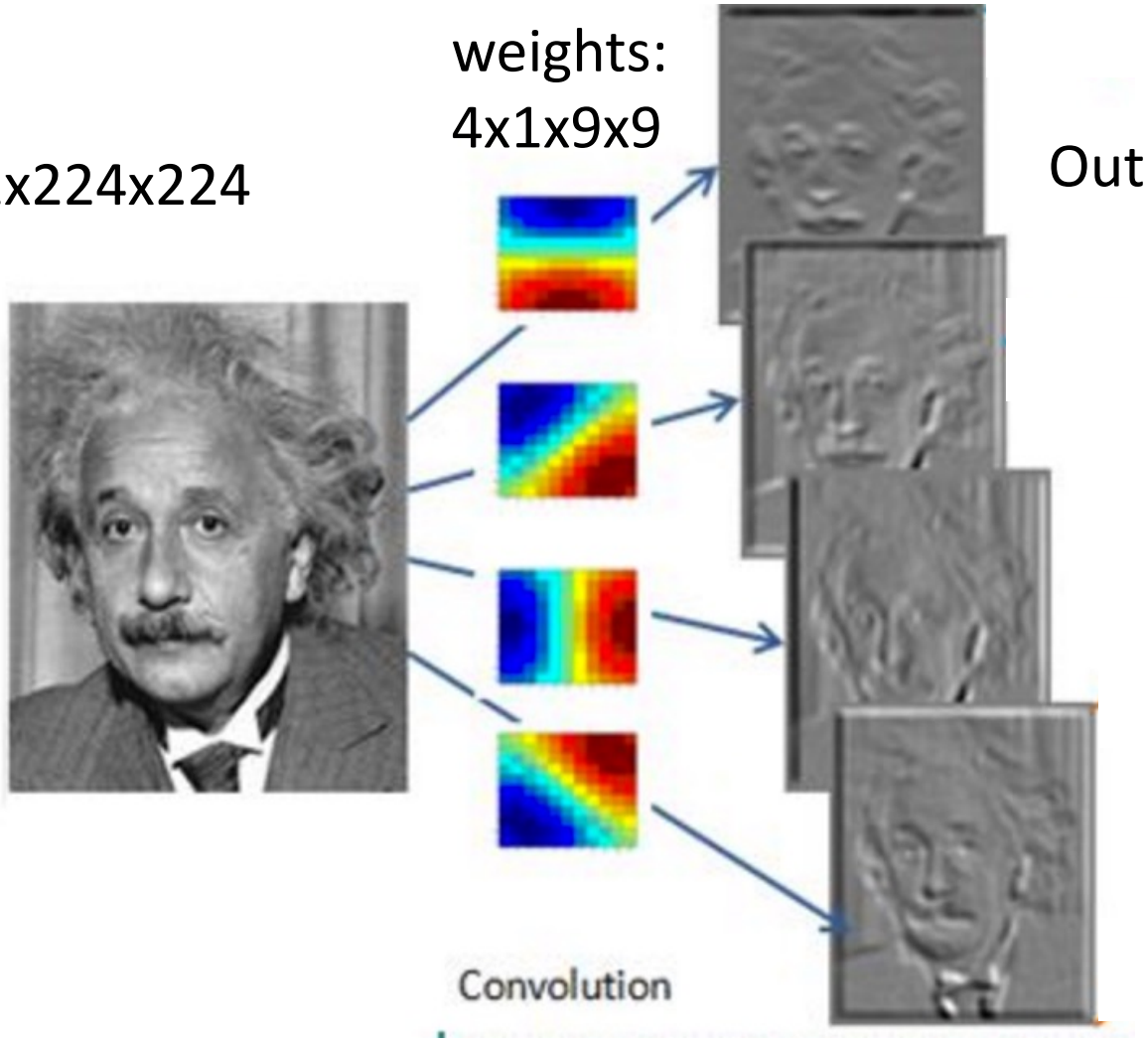
The 2D Convolutional Layer in a Neural Network



Convolutional Layer (with 4 filters)

Input: 1x224x224

weights:
4x1x9x9



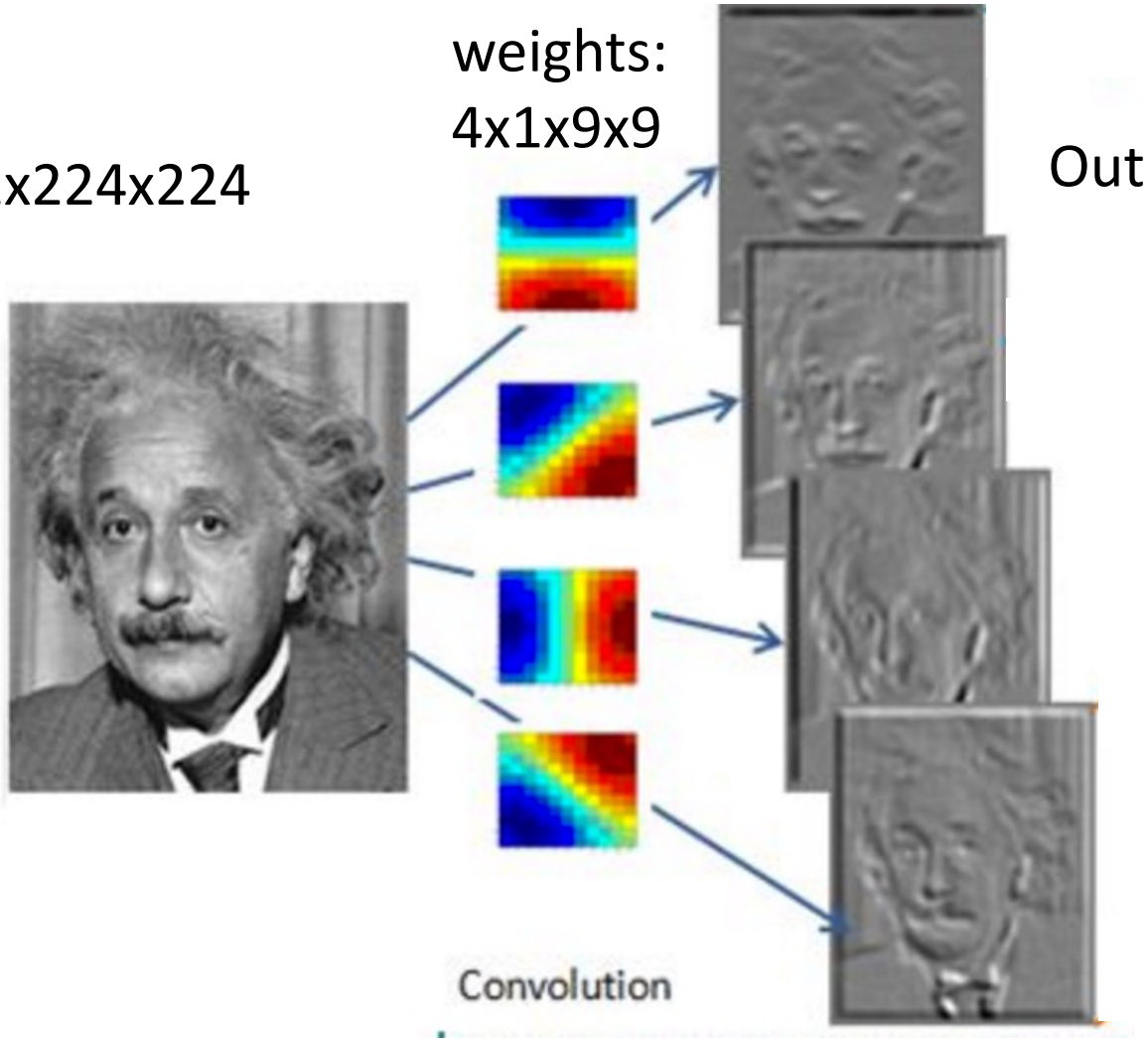
Output: 4x224x224

if zero padding,
and stride = 1

Convolutional Layer (with 4 filters)

Input: 1x224x224

weights:
4x1x9x9

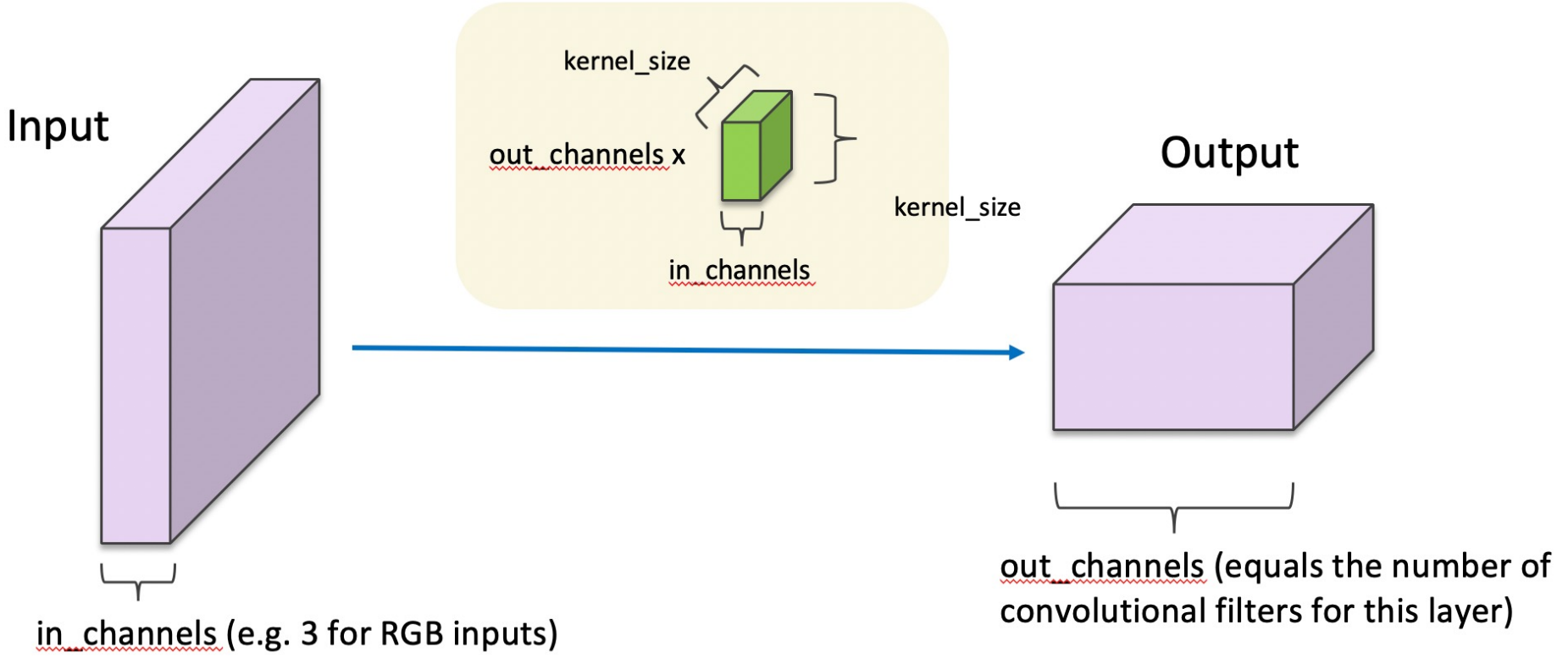


Output: 4x112x112

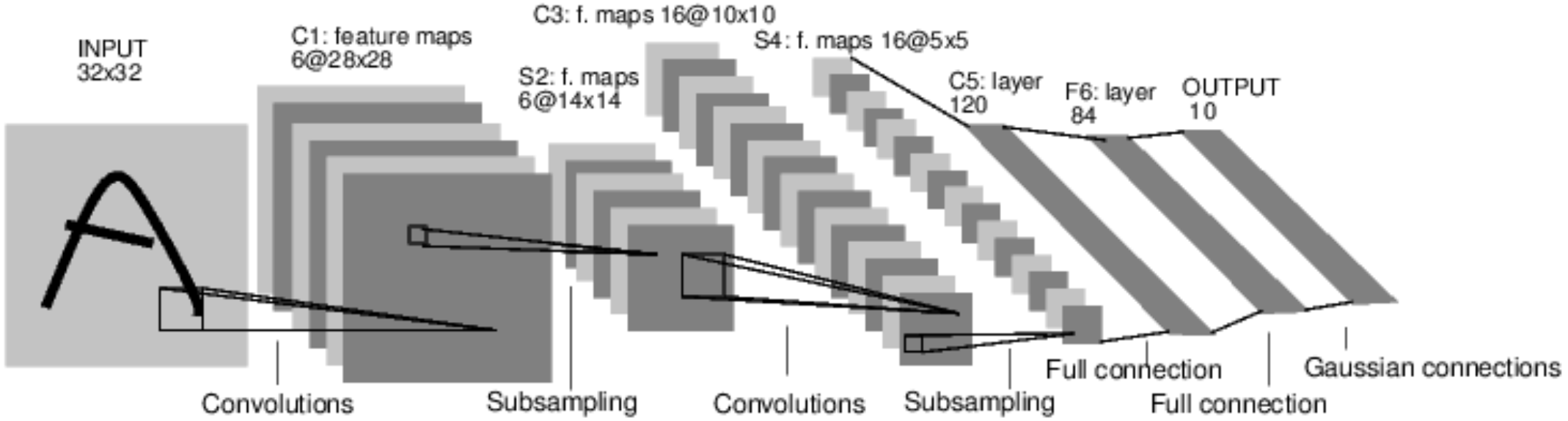
if zero padding,
but stride = 2

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



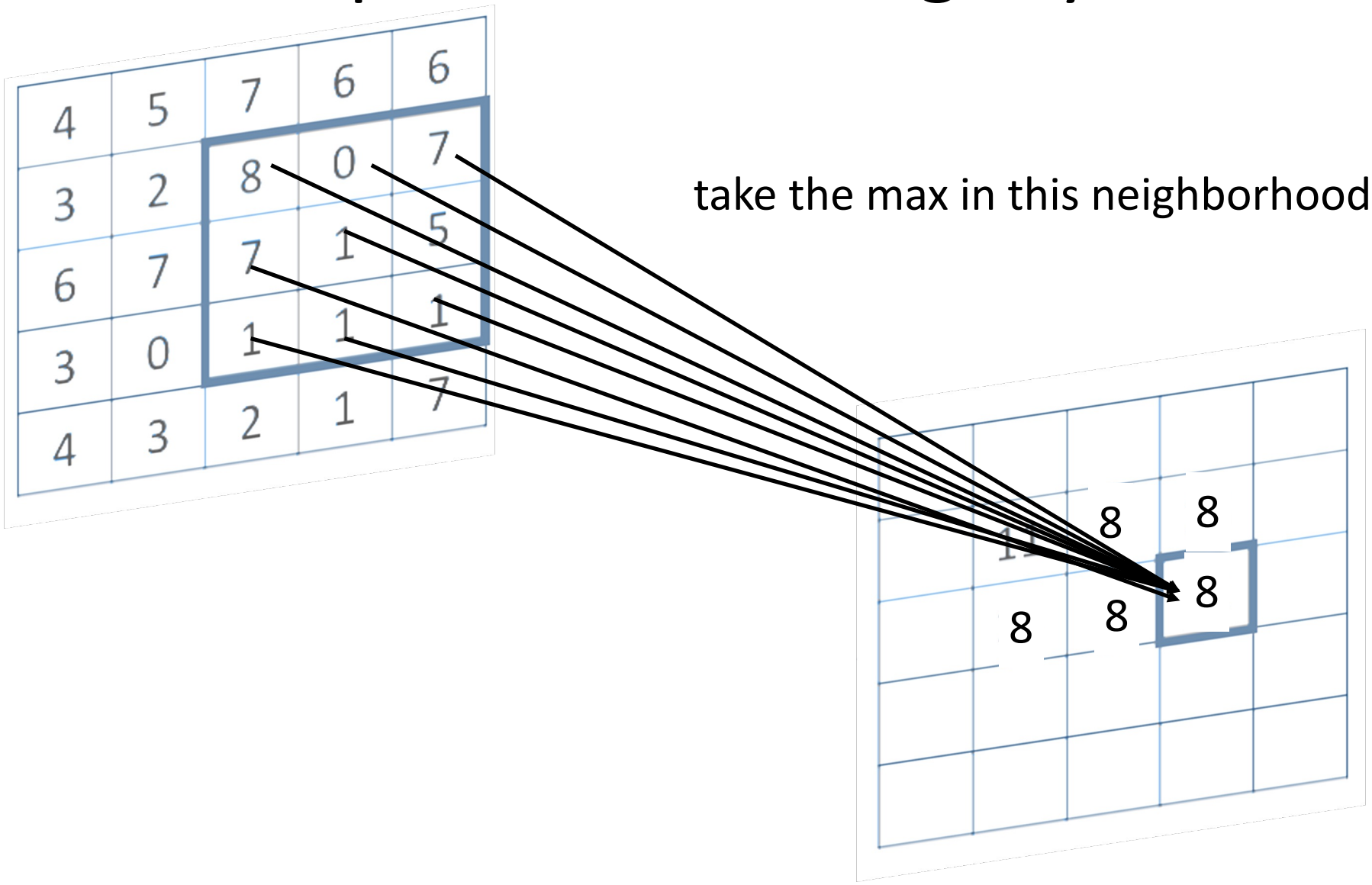
Yann LeCun

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	61889	1998

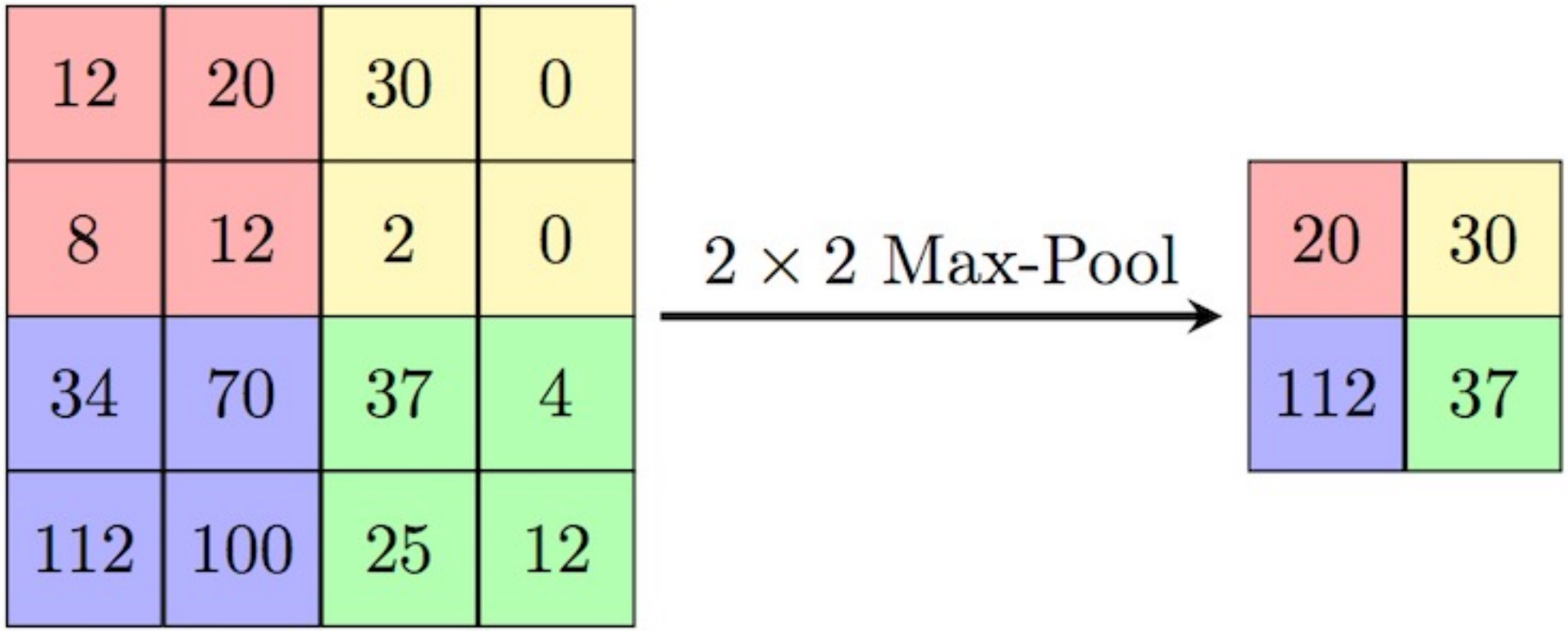
LeNet in Pytorch

```
# 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



SpatialMaxPooling Layer



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

LeNet Summary

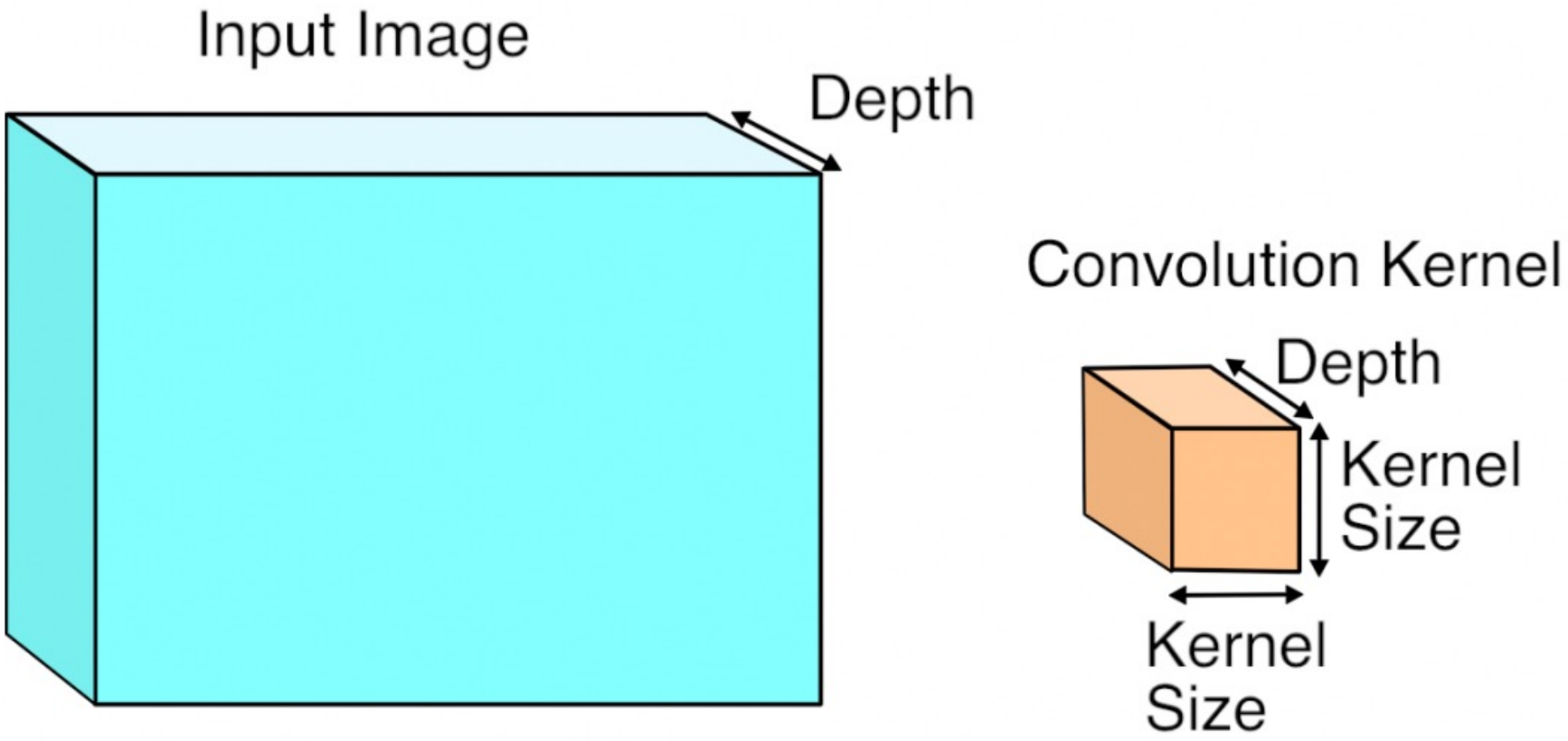
- 2 Convolutional Layers + 3 Linear Layers

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

New Architectures Proposed

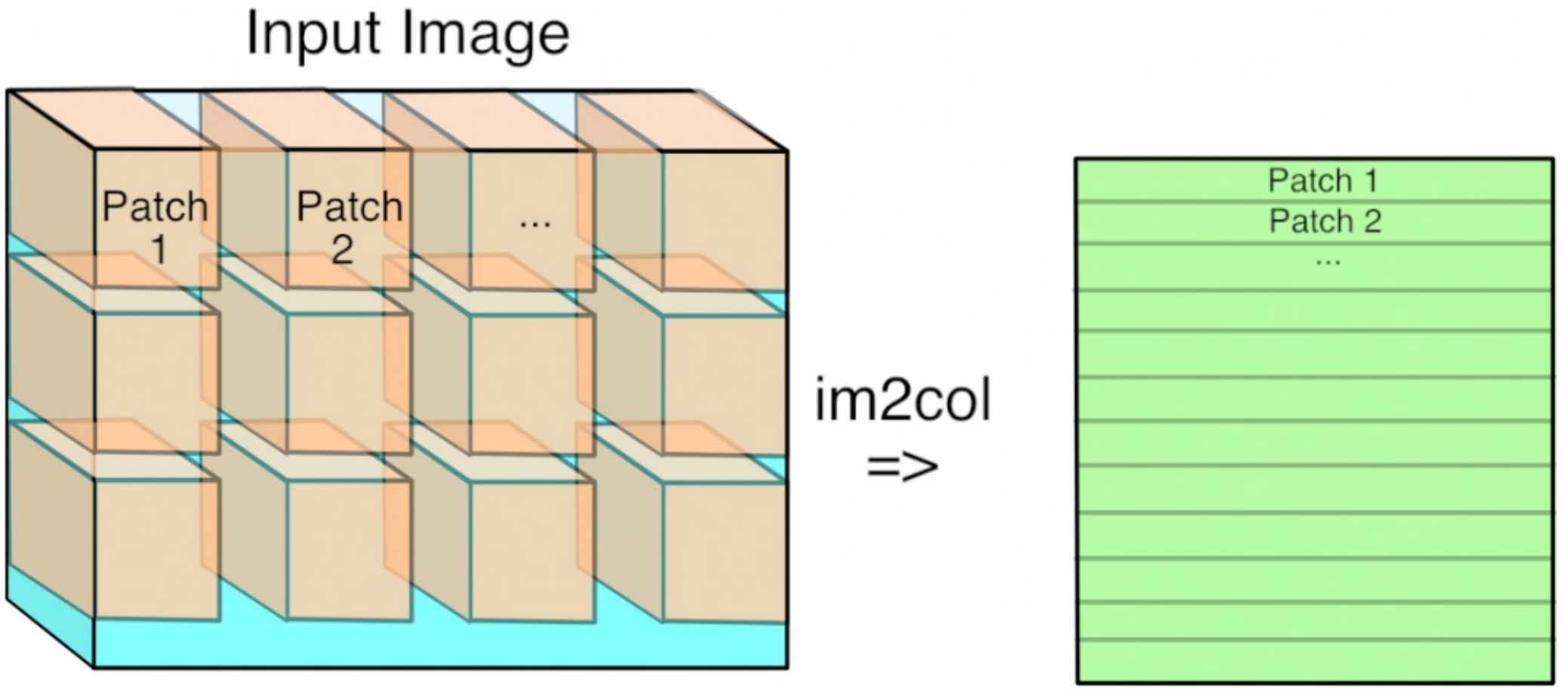
- Alexnet (Kriszhevsky et al NIPS 2012) **[Required Reading]**
- VGG (Simonyan and Zisserman 2014)
- GoogLeNet (Szegedy et al CVPR 2015)
- ResNet (He et al CVPR 2016)
- DenseNet (Huang et al CVPR 2017)

Convolutional Layers as Matrix Multiplication



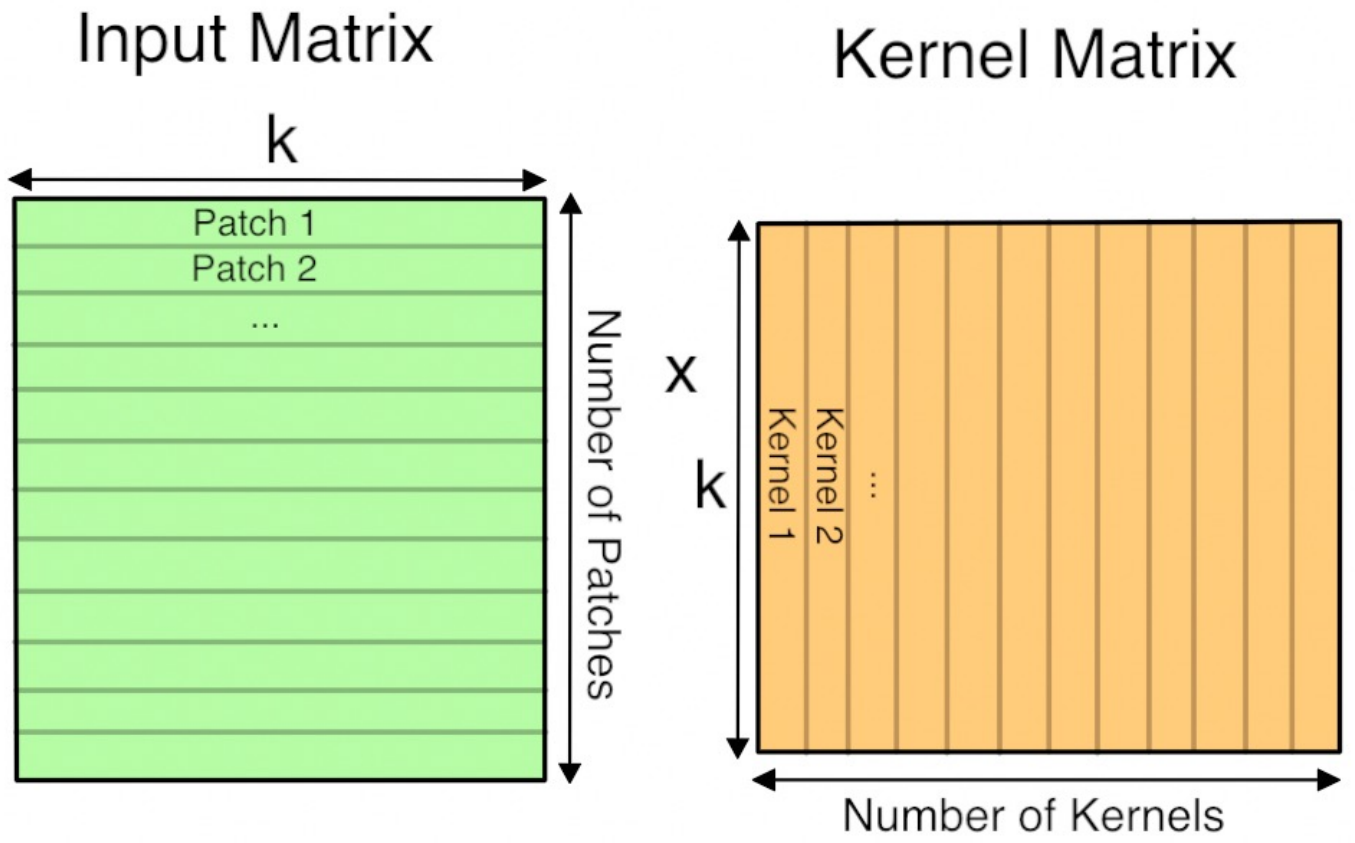
https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

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)

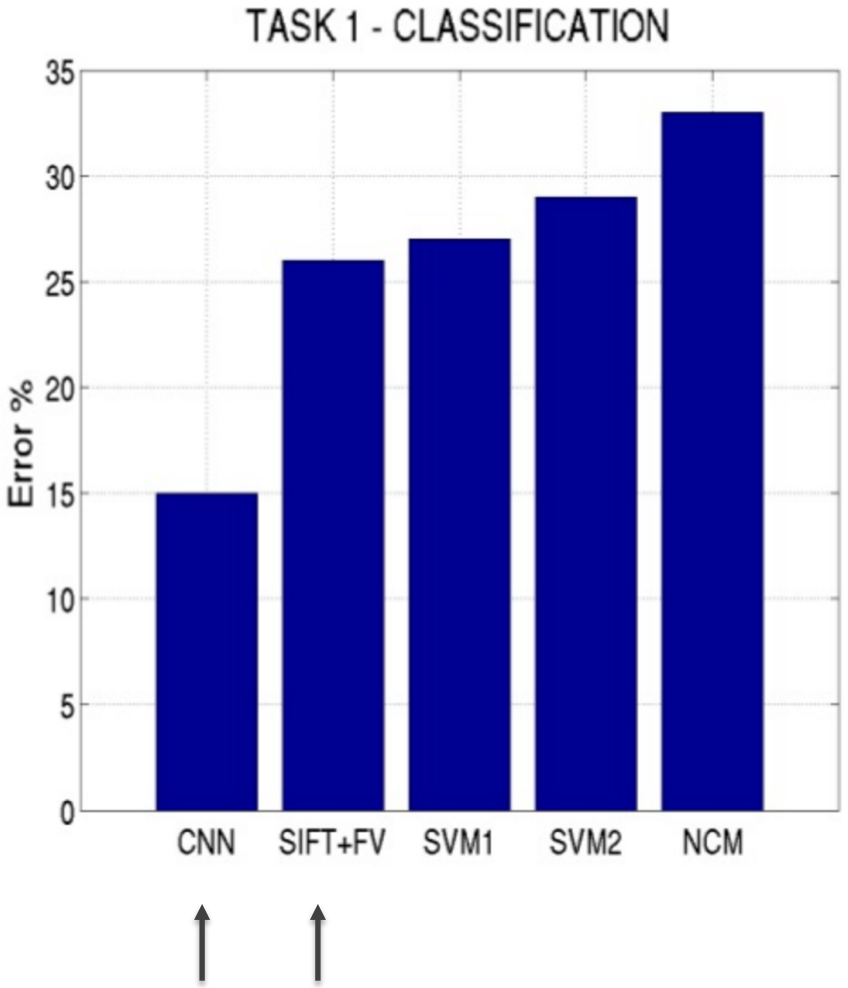
The Evaluation Metric: Top K-error

True label: Abyssinian cat

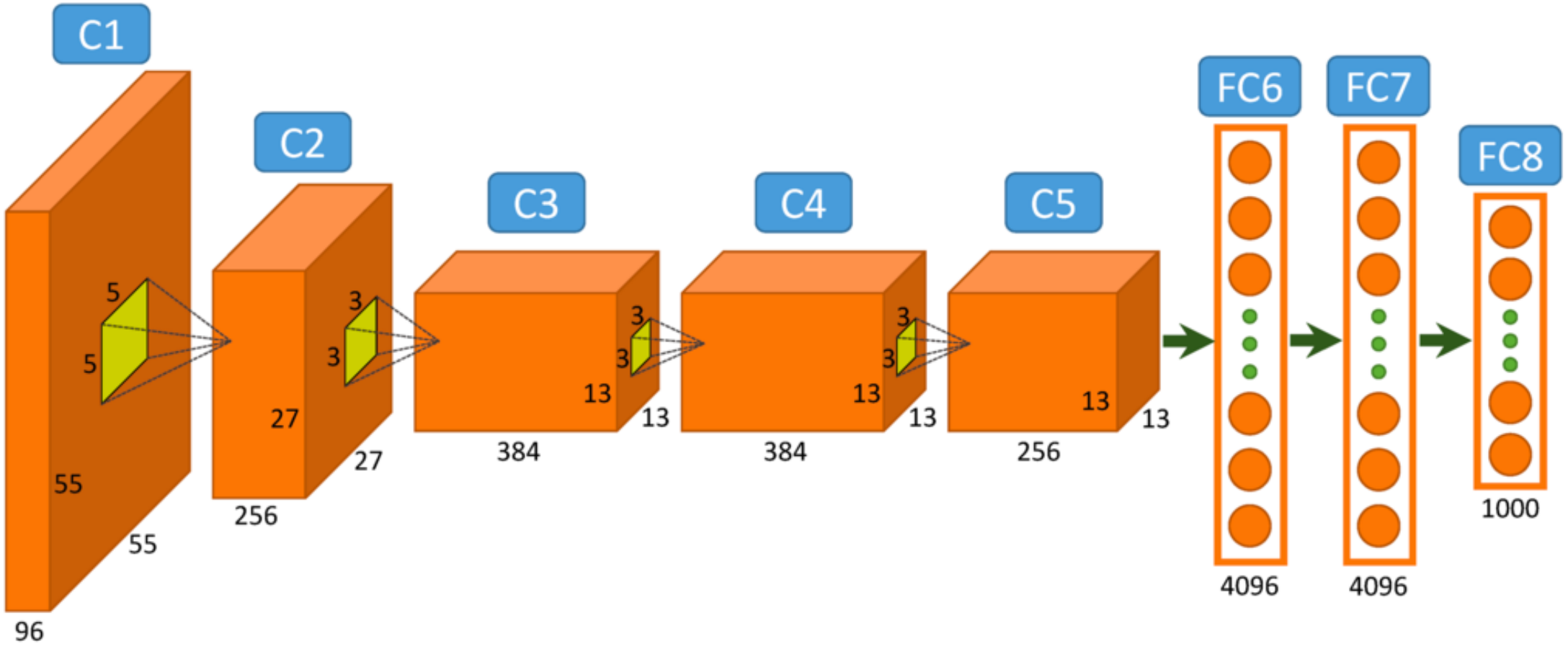
Top-1 error: 1.0	Top-1 accuracy: 0.0
Top-2 error: 1.0	Top-2 accuracy: 0.0
Top-3 error: 1.0	Top-3 accuracy: 0.0
Top-4 error: 0.0	Top-4 accuracy: 1.0
Top-5 error: 0.0	Top-5 accuracy: 1.0

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



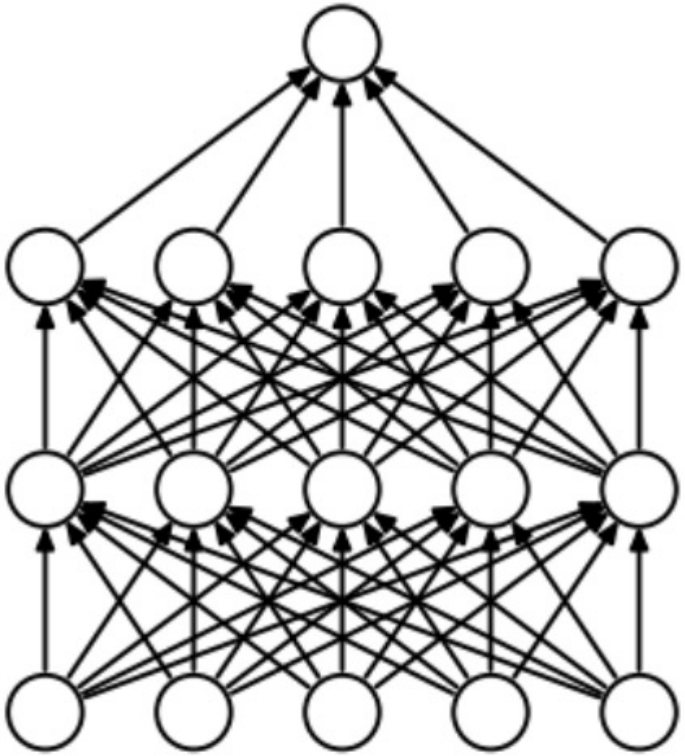
https://www.saagie.com/fr/blog/object-detection-part1

Pytorch Code for Alexnet

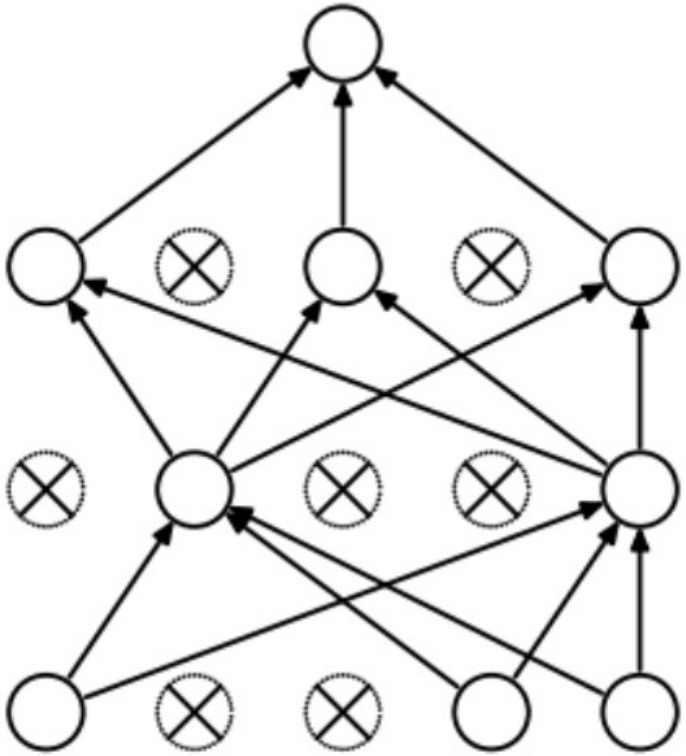
- In-class analysis

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

Dropout Layer



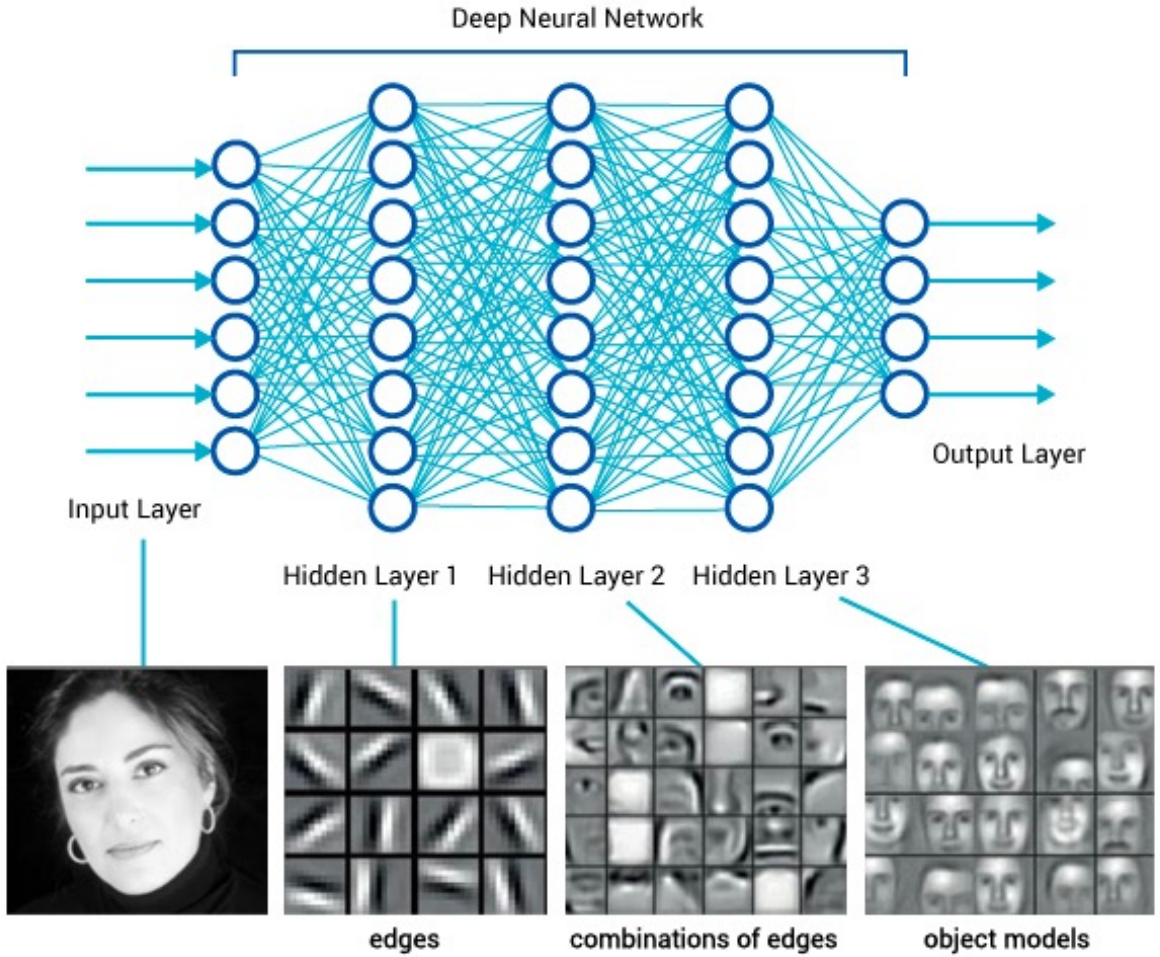
(a) Standard Neural Net



(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 (Bag of words)

Classification (SVM or softmax)

Deep Learning

Convolutional Neural Network

Classifier (SVM or softmax)

Pre-training

Convolutional Neural Network

Classifier

1000 categories
1 million images

Fine-tuning

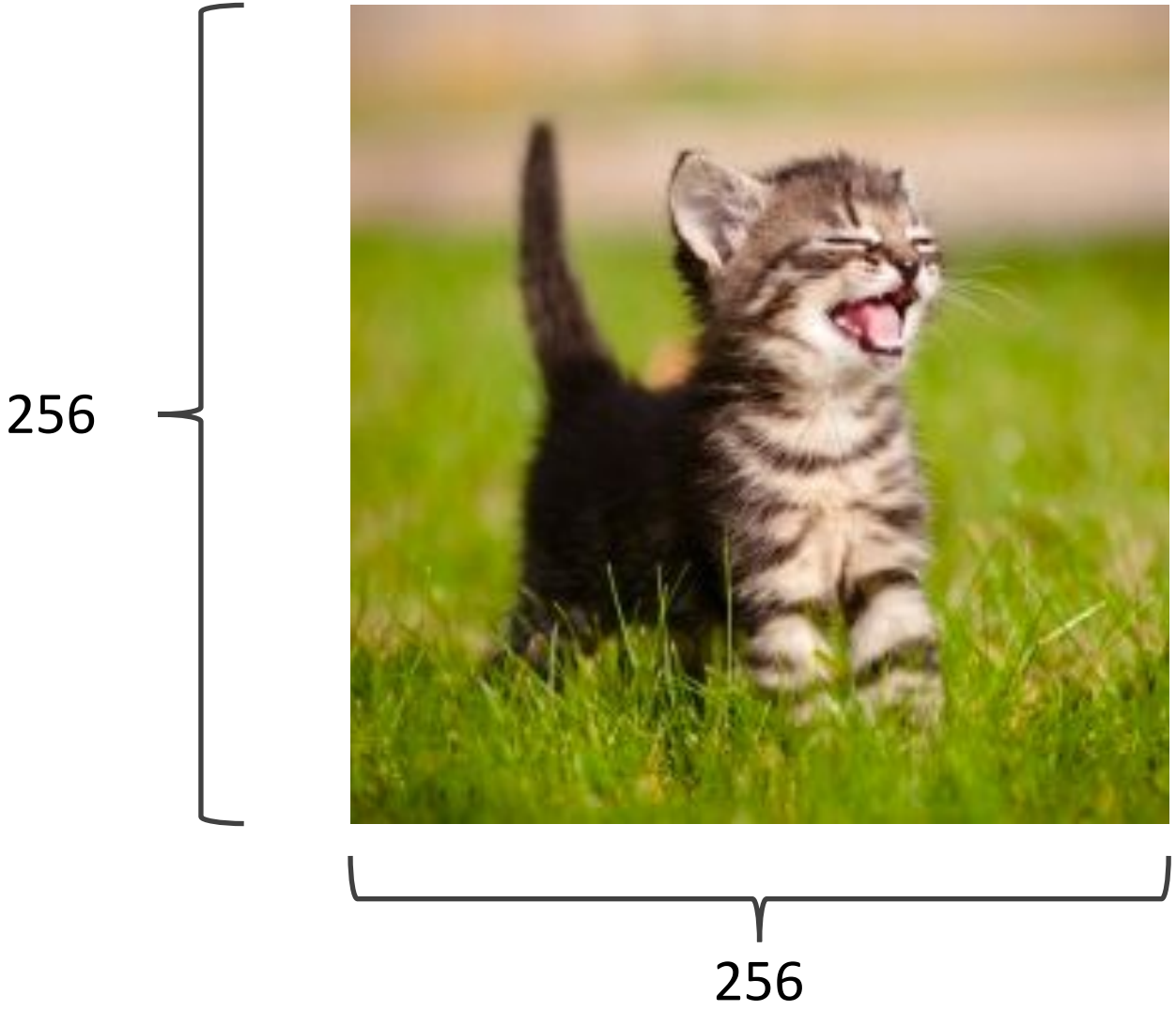
Convolutional Neural Network

My Classifier

20 categories
400 images

Preprocessing and Data Augmentation

Preprocessing and Data Augmentation



Preprocessing and Data Augmentation

224x224

Preprocessing and Data Augmentation

224x224





True label: Abyssinian cat

Other Data Augmentation options

Pytorch: https://pytorch.org/vision/stable/transforms.html

- **Basic:** RandomCrop, RandomResize, RandomHorizontalFlip
- **Geometric:** RandomRotation, RandomPerspective, RandomAffine, etc.
- **Color:** RandomAutoContrast, RandomEqualize, RandomPosterize, etc.
- **Other:** RandomErasing (https://arxiv.org/abs/1708.04896)

Advanced Data Augmentation

- MixUp – 2017 (https://arxiv.org/abs/1710.09412)
- AutoAugment - 2018 (https://arxiv.org/abs/1805.09501)
- RandAugment - 2019 (https://arxiv.org/abs/1909.13719)
- CutMix – 2019 (https://arxiv.org/abs/1905.04899)
- AugMix – 2019 (https://arxiv.org/abs/1912.02781)
- TrivialAugment - 2021 (https://arxiv.org/abs/2103.10158)

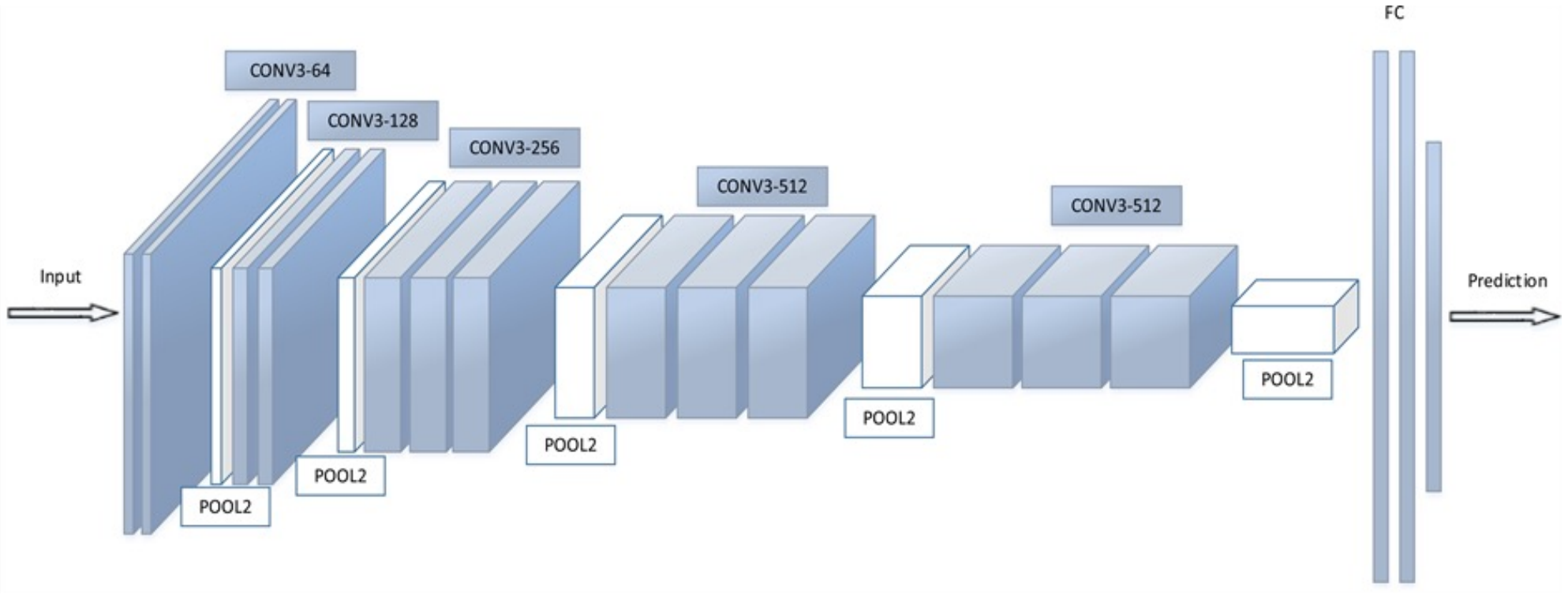
Other Important Aspects

- 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:

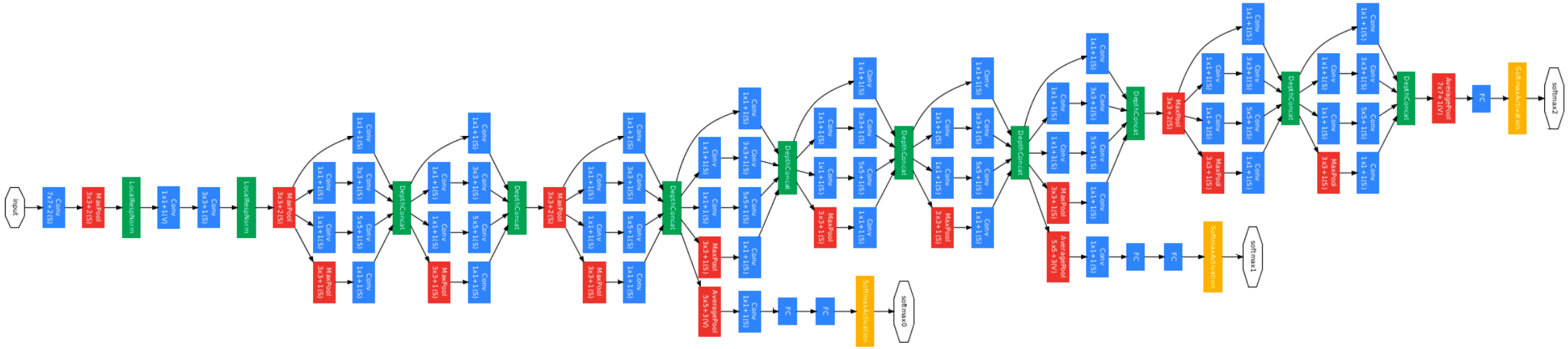


https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py

Simonyan and Zisserman, 2014.

https://arxiv.org/pdf/1409.1556.pdf

GoogLeNet

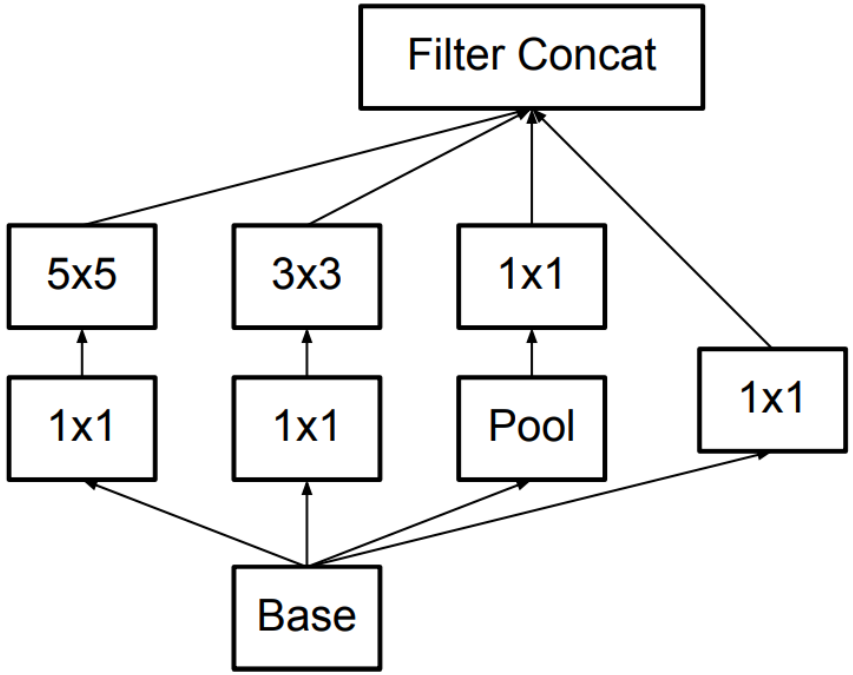


https://github.com/kuangliu/pytorch-cifar/blob/master/models/googlenet.py

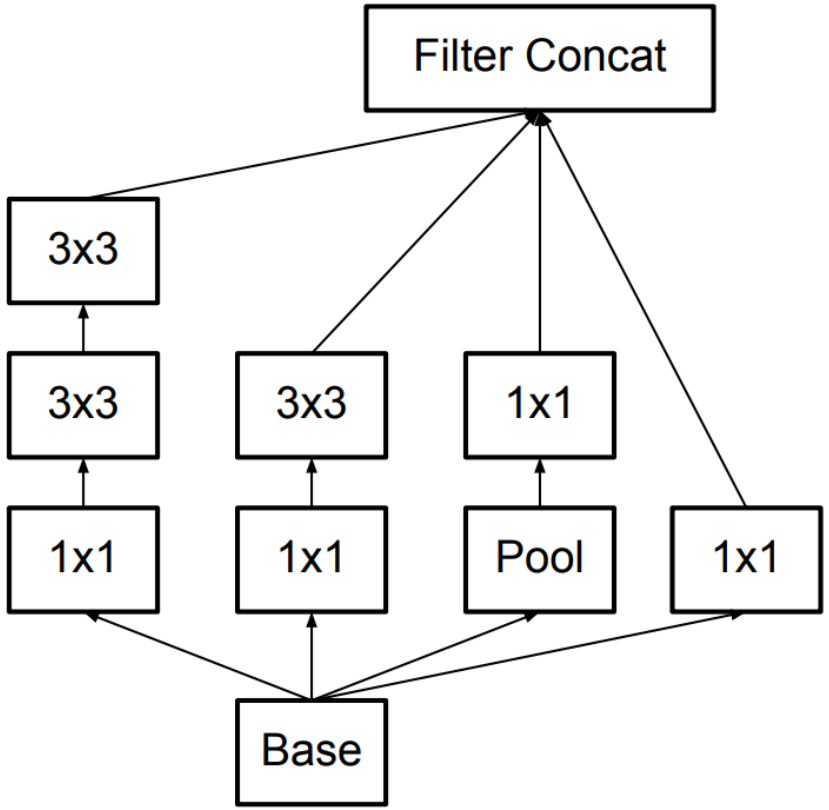
Szegedy et al. 2014

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

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

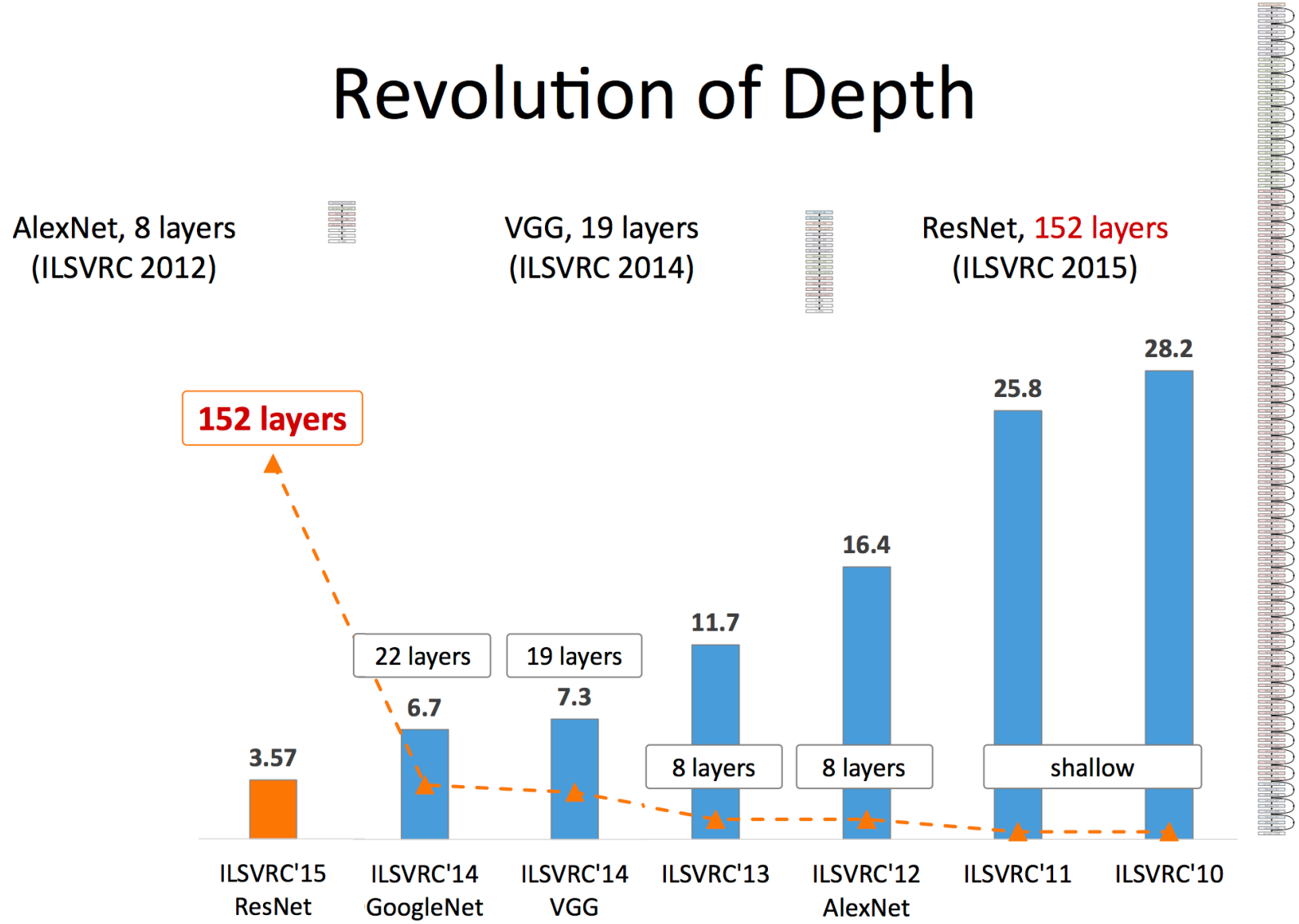
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

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

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

https://arxiv.org/abs/1502.03167

Revolution of Depth



Slide by Mohammad Rastegari

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

More Recent CNNs you can learn

- DenseNet - 2016 (Similar to ResNet but all layers connect to all next and previous layers) https://arxiv.org/abs/1608.06993
- ResNext – 2016 (Combines ideas from ResNet and InceptionNets https://arxiv.org/abs/1611.05431v2)
- NASNet – 2017 (Search the space of possible architectures) https://arxiv.org/abs/1707.07012
- EfficientNet – 2019 (Scales up both width and depth of CNNs using a more principled approach) https://arxiv.org/abs/1905.11946
- RegNet – 2020 (Paper has a more principled formula to decide on widths and depths) https://arxiv.org/abs/2003.13678
- ConvNext – 2022 (Combines many tricks both at the architecture and optimization level) https://arxiv.org/abs/2201.03545

Questions?