Andrey Ustyuzhanin

Convolutional Neural Networks

Deep Learning building blocks for images

2021

MLHEP 2021

NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS

LAMBDA • HSE

Yandex

SCHOOL OF DATA ANALYSIS

EPFL

SIT Schaffhausen Institute of Technology

Image Recognition

Banksy

- ”Kid”
- ”Nurse”
- “Superhero”
- “Banksy”
- “Game”
- “Painting”
- ?

Image Recognition

Banksy

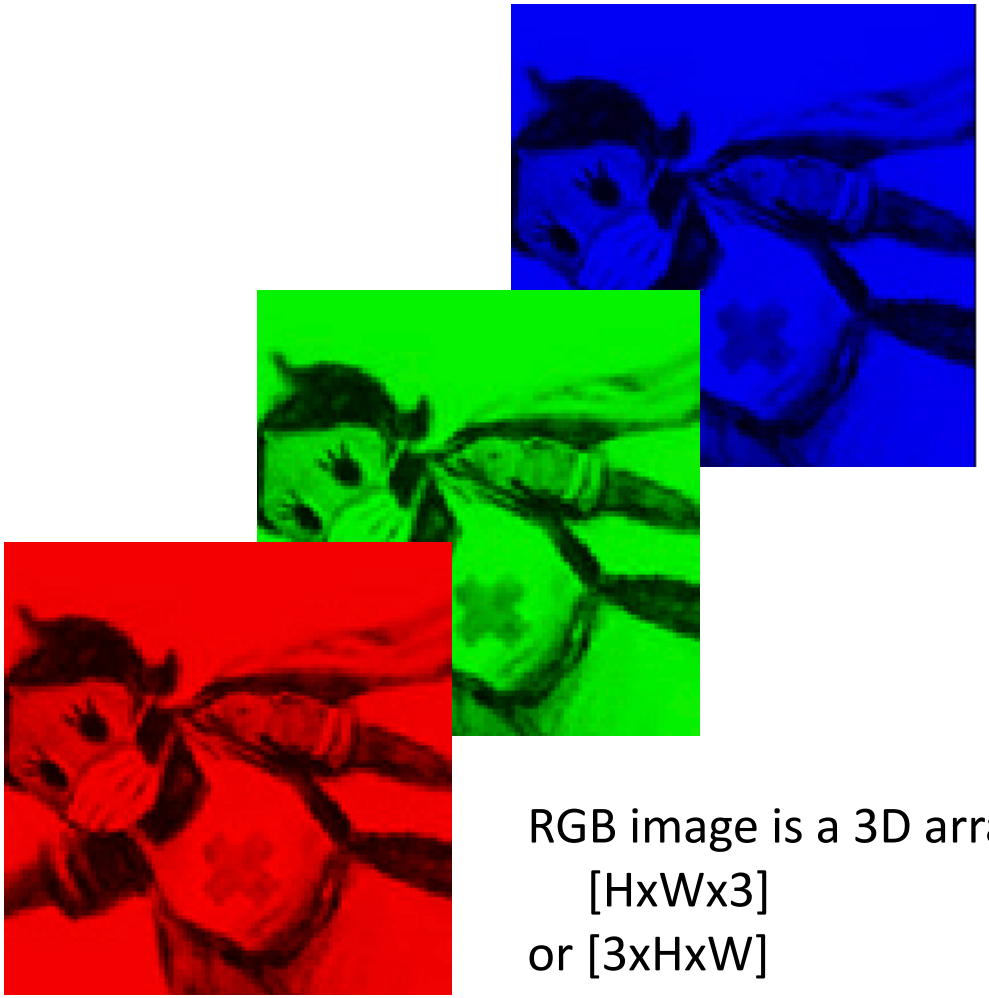
Grayscale image is a matrix of pixels [H x W]

Pixel = **pic**ture **el**ements

Each pixel stores number [0,255] for brightness

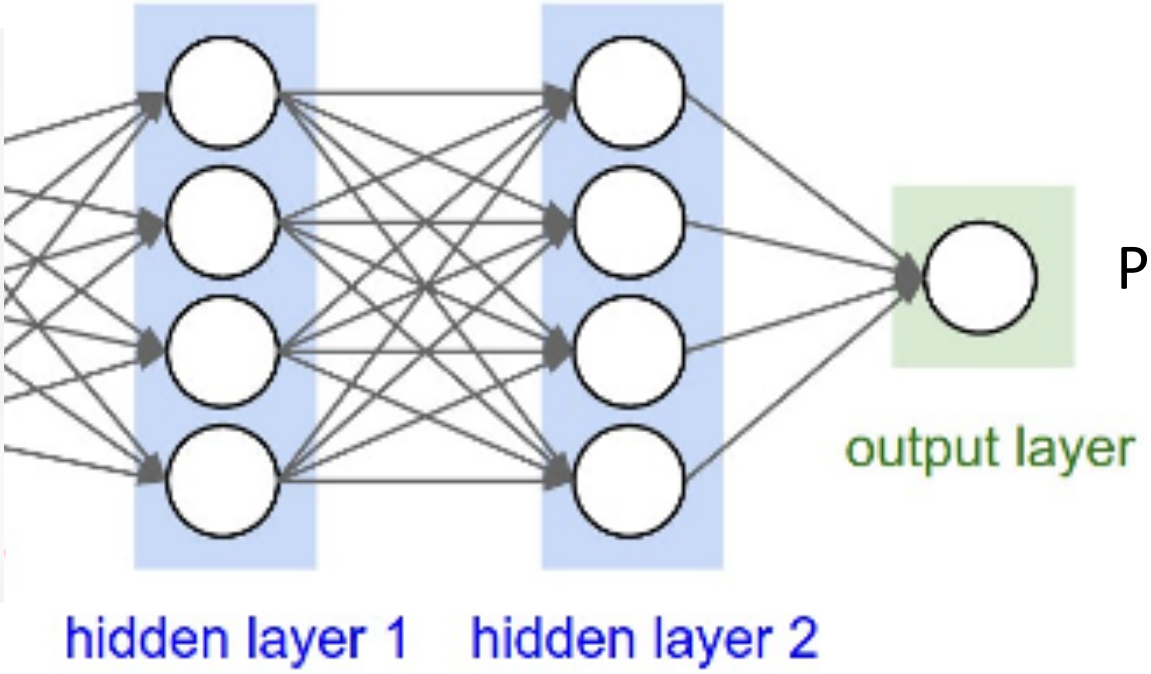
Image Recognition

Banksy

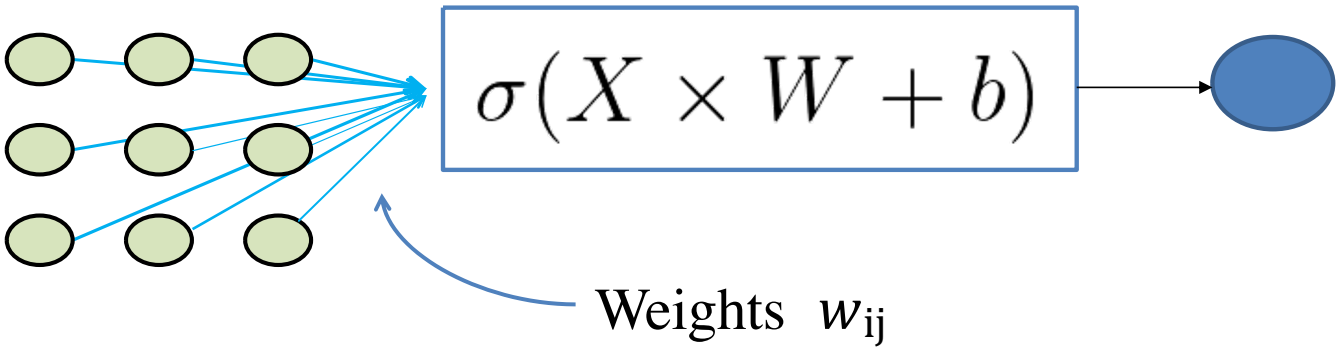


RGB image is a 3D array
[HxWx3]
or [3xHxW]

General idea

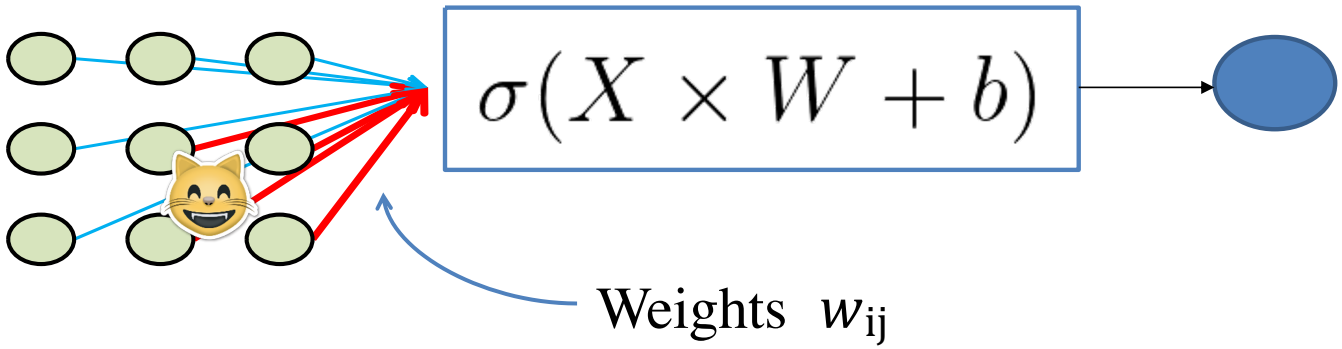


Problem with images



Slide from Andrey Zimovnov's “Deep learning for computer vision”

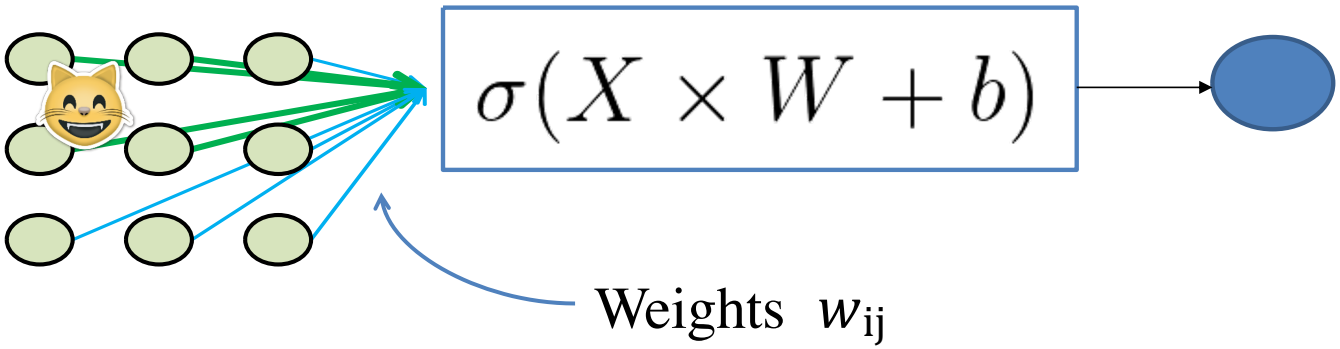
Problem with images



On this object, you will train red weights to react on cat face

Slide from Andrey Zimovnov's “Deep learning for computer vision”

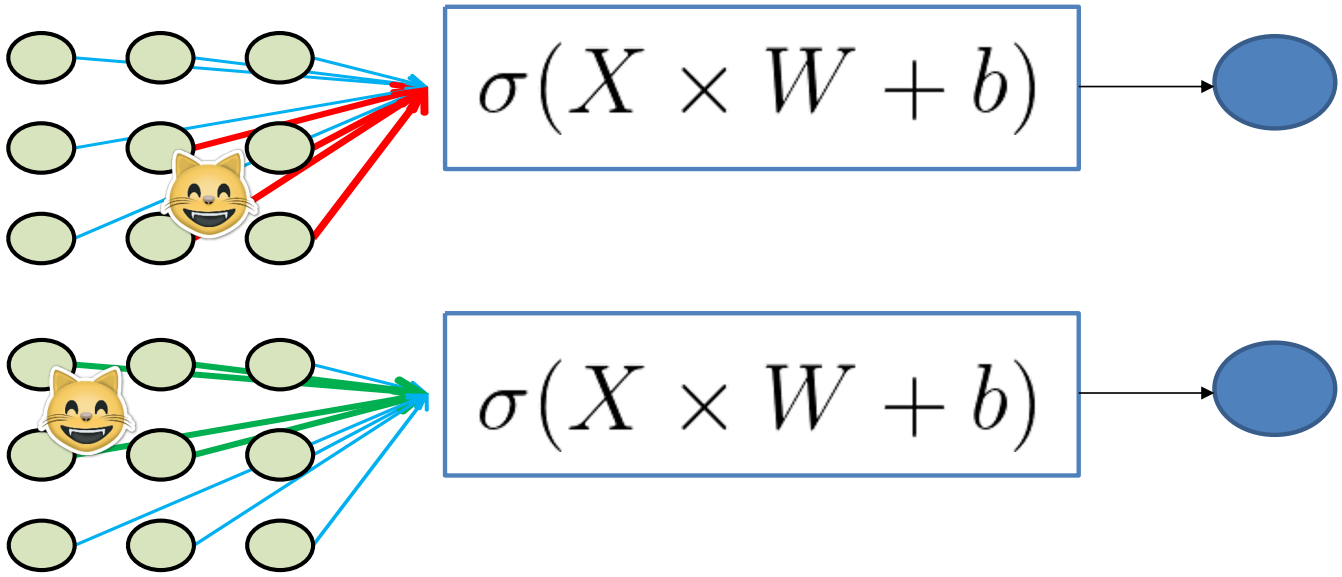
Problem with images



On this object, you will train green weights to react on cat face

Slide from Andrey Zimovnov's “Deep learning for computer vision”

Problem with images



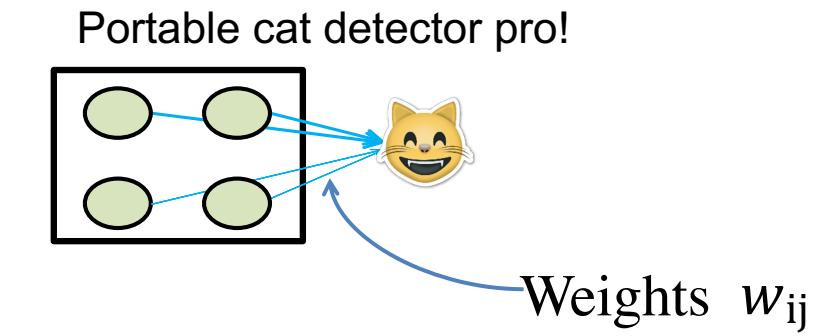
You network will have to learn those two cases separately!

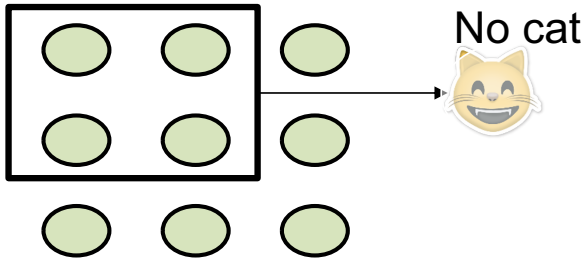
Worst case: one neuron per position.

Slide from Andrey Zimovnov's “Deep learning for computer vision”

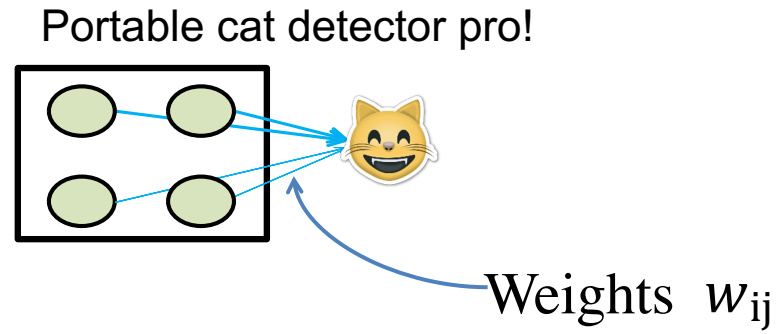
Main idea: let’s encode ”cat face” by weight tensor that we can shift across the image.

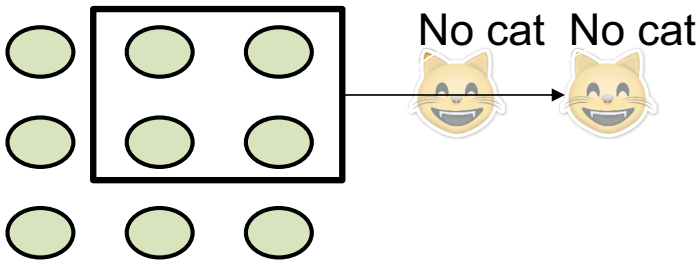
Same features for each spot



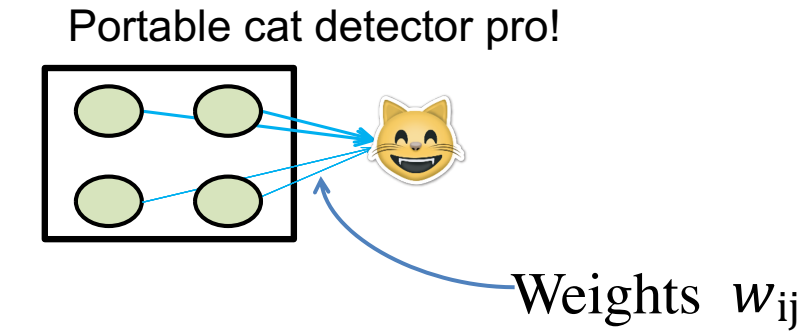


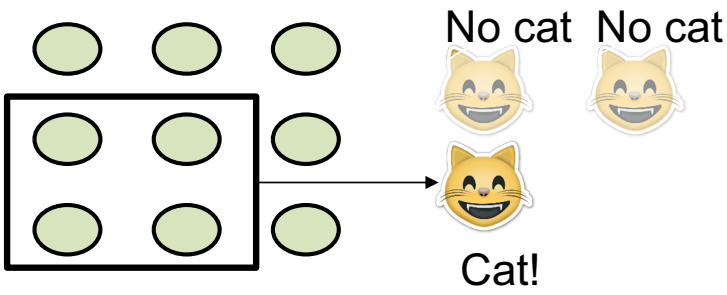
Same features for each spot



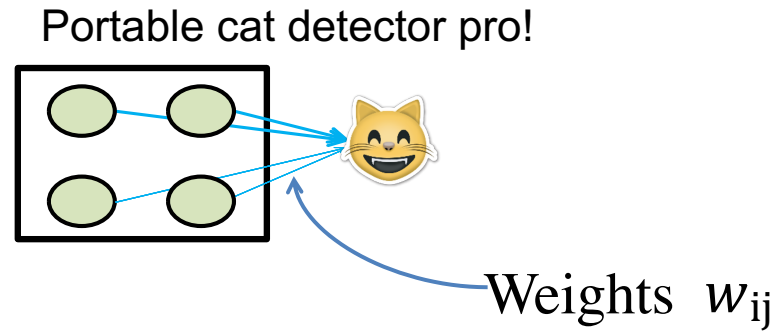


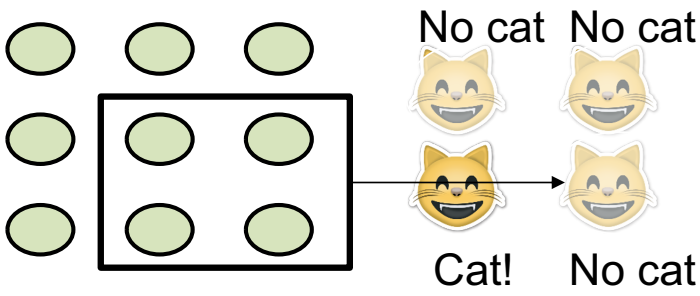
Same features for each spot



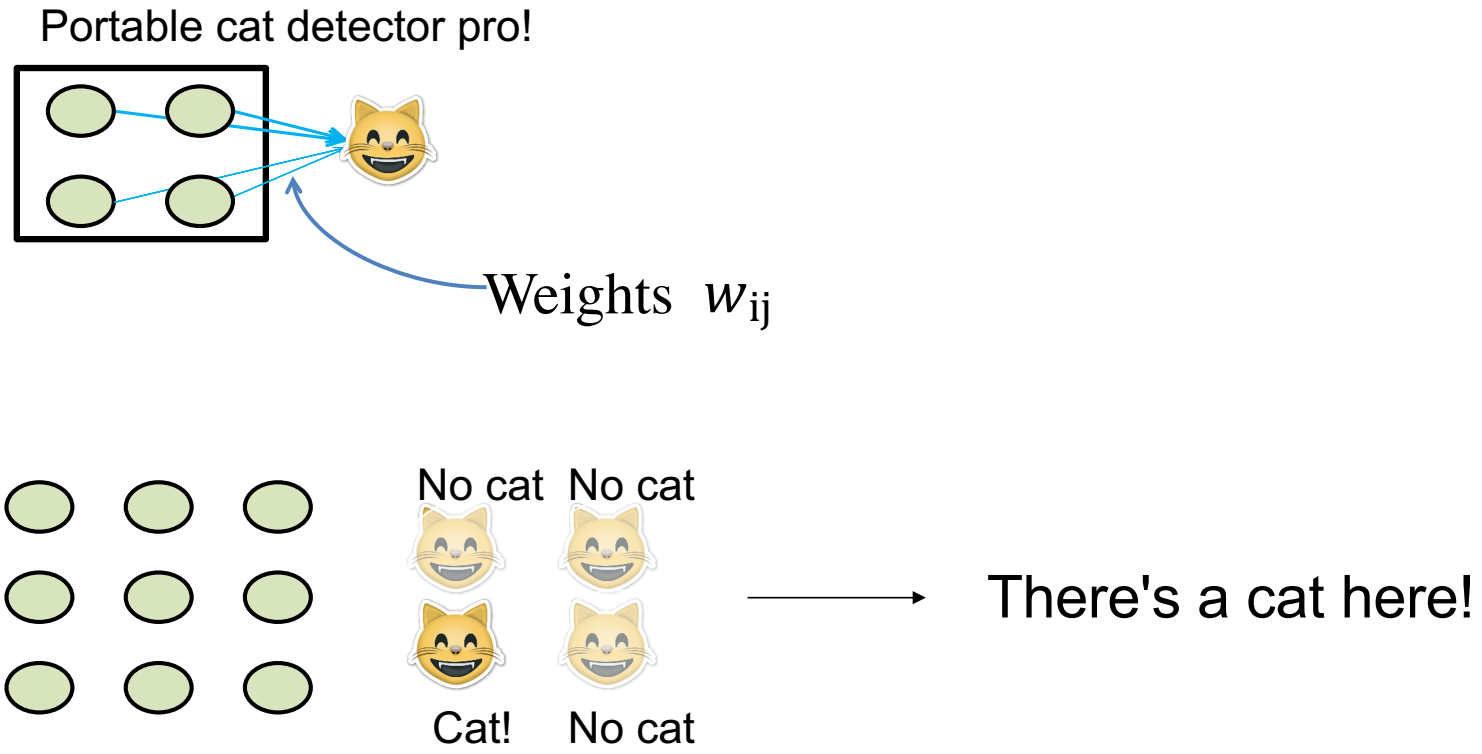


Same features for each spot



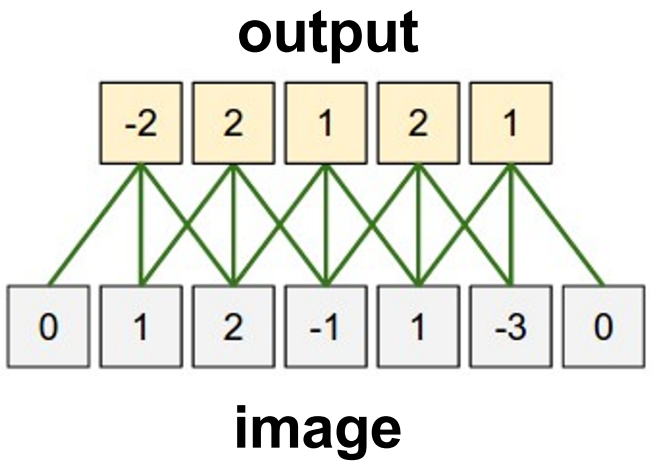


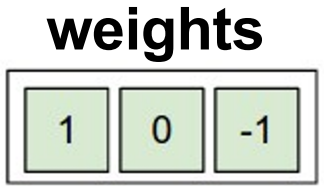
Same features for each spot



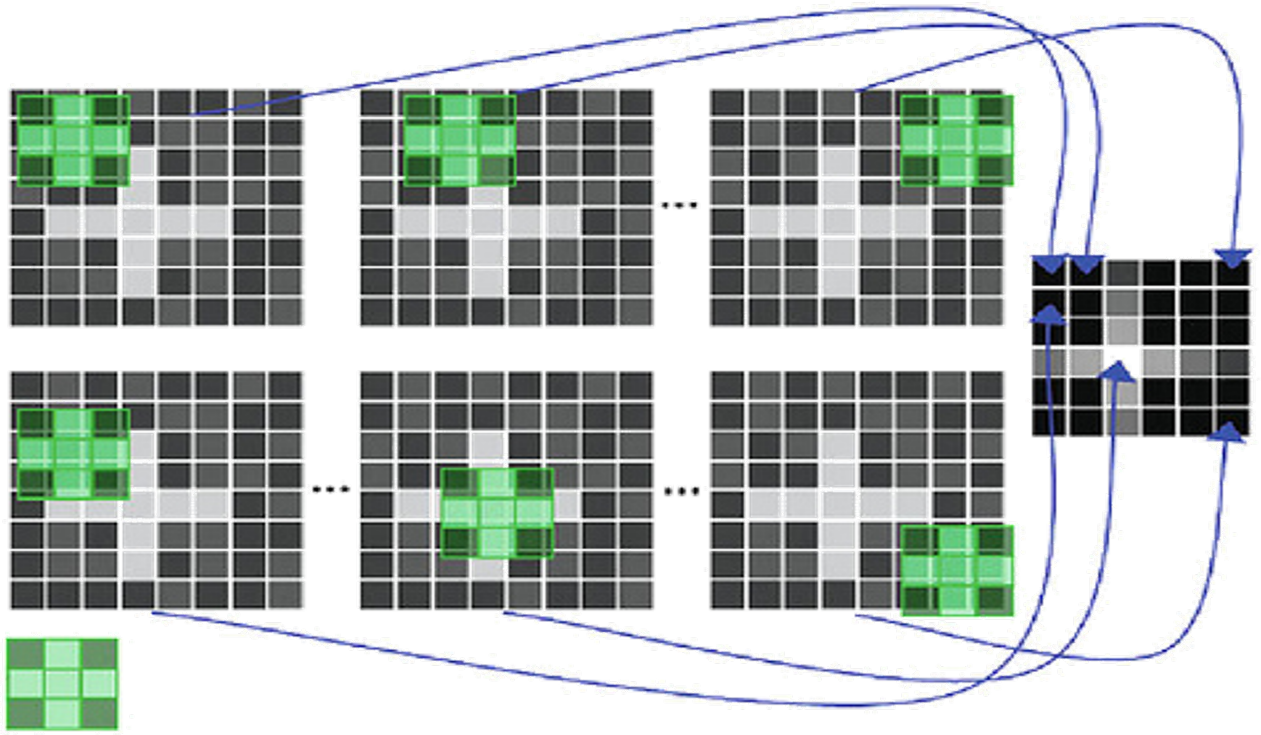
Convolution

- Apply same weights to all patches



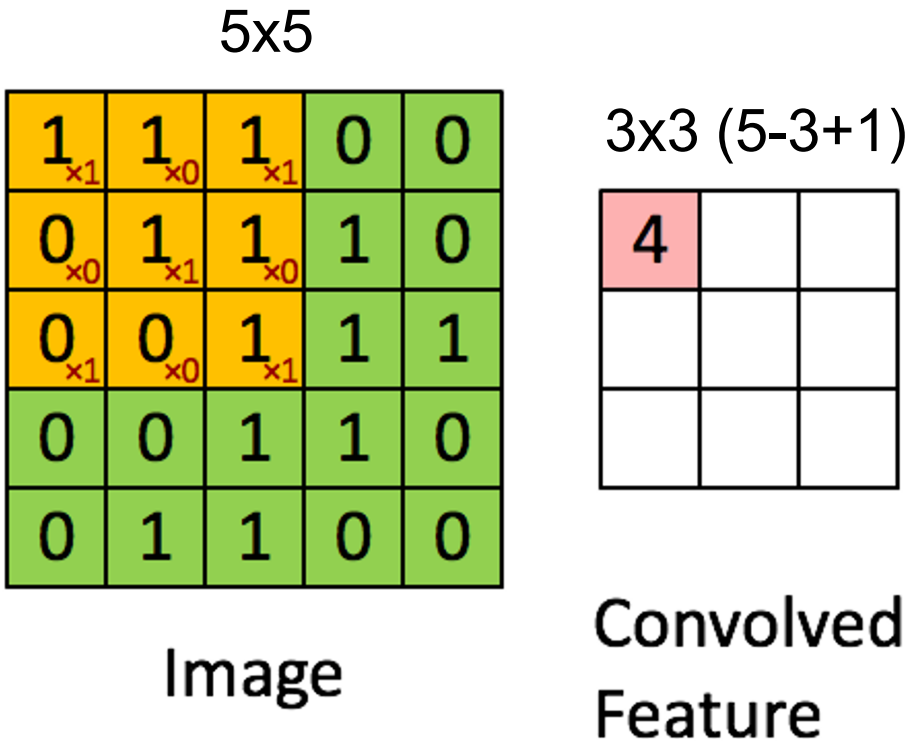


Convolution



apply one “filter” to all patches

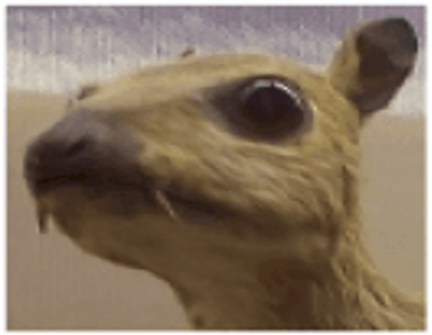
Convolution



Intuition: how cat-like is this square?

Convolution

Input image



Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

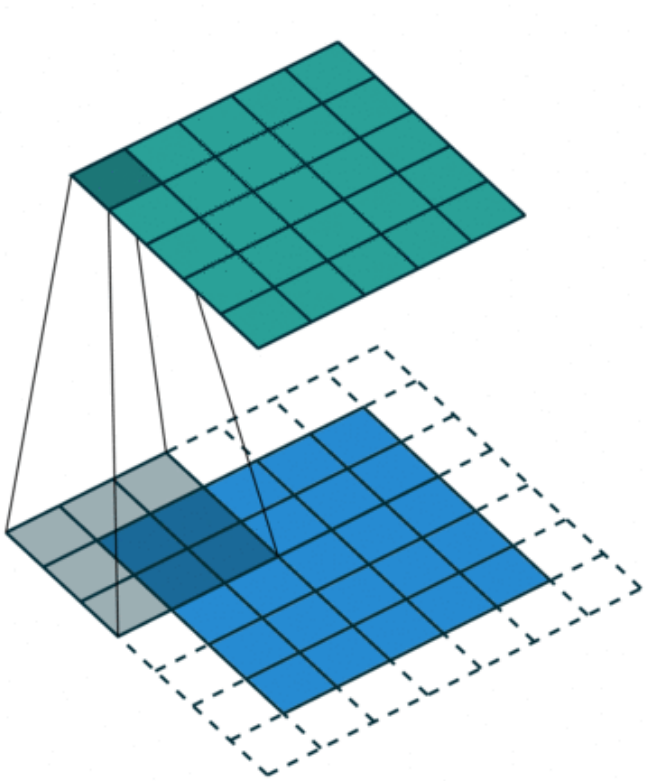
Feature map

Intuition: how **edge-like** is this square?

<Demo>

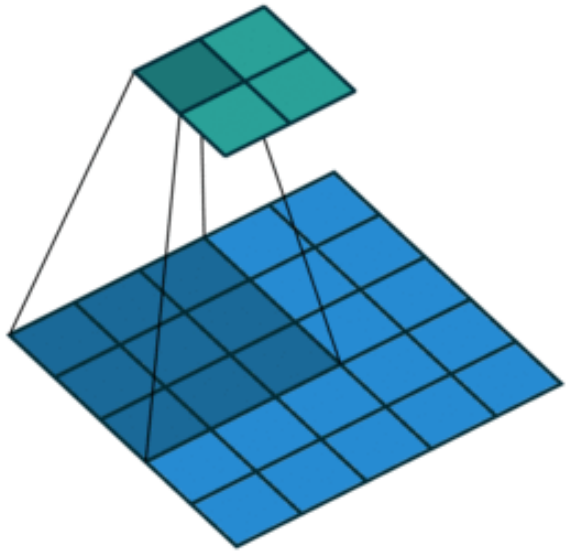
Convolution tricks: padding

- pad the edges with extra, “fake” pixels (usually of value 0, hence the oft-used term “zero padding”). This way, the kernel when sliding can allow the original edge pixels to be at its center, while extending into the fake pixels beyond the edge, producing an **output the same size as the input;**
- **padding_mode (string, optional)** – 'zeros', 'reflect', 'replicate' or 'circular'. Default: 'zeros'

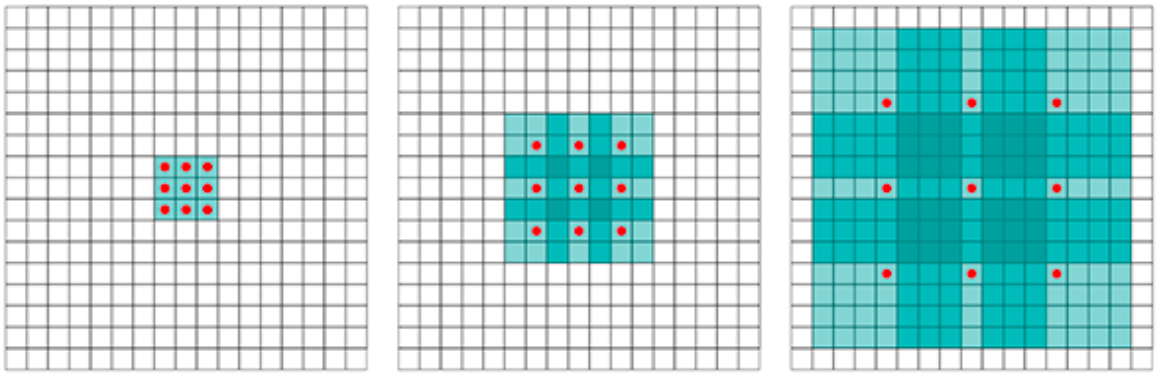


Convolution tricks: striding and dilation

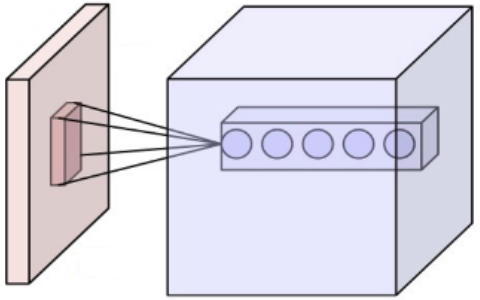
- The idea of the **stride** is to skip some of the slide locations of the kernel (see figure on the right)
- **Dilated convolution** is a basic convolution only applied to the input volume with defined gaps (see below). You may use dilated convolution when:



- You are working with higher resolution images, but fine-grained details are still important
- You are constructing a network with fewer parameters



Convolution Layer

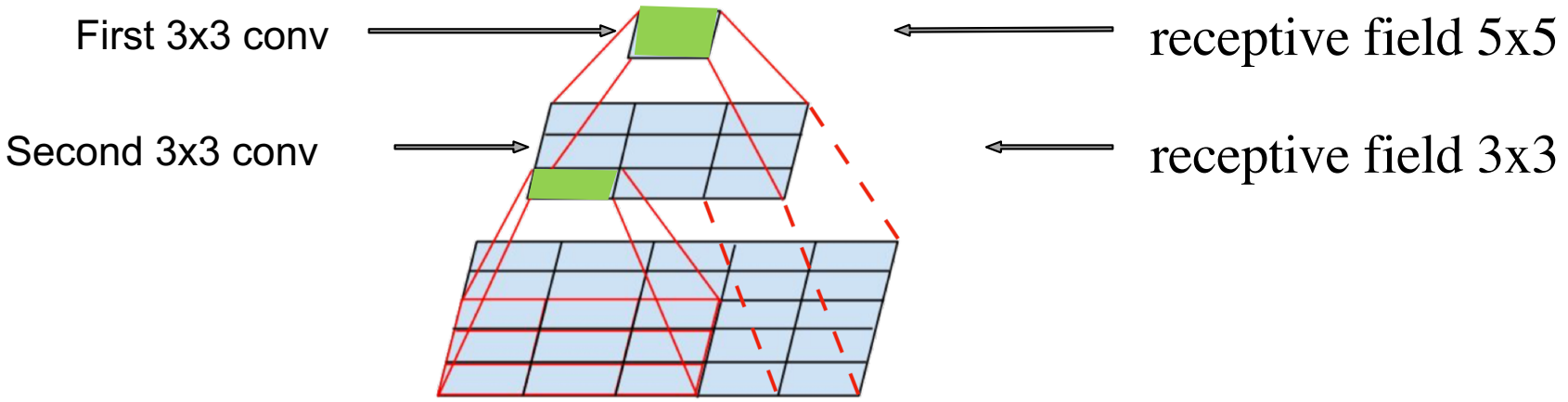


- Intuition: how cat-like is this square?
- May learn arbitrary number of kernels, one per output channel
- the output value of the layer with input size (N, C_{in}, H, W) and output (N, C_{out}, H_{out}, W_{out}) is

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

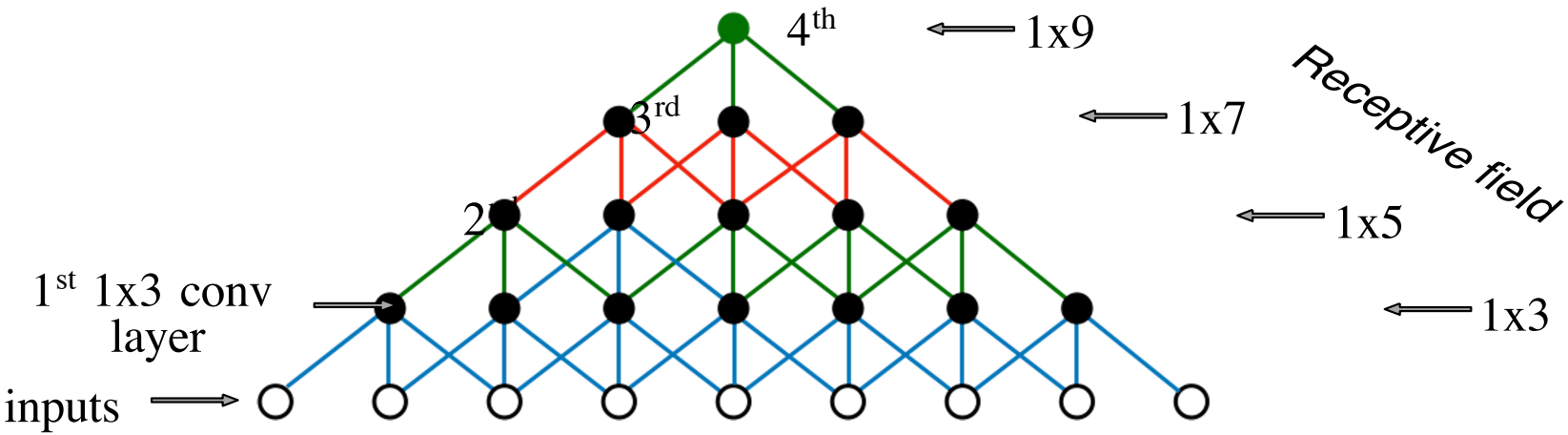
  N - batch size, C - number of channels.
- Number of learnable parameters: C_{out} * (C_{in} * size(kernel) + 1)
- https://pytorch.org/docs/master/generated/torch.nn.Conv2d.html

Receptive field



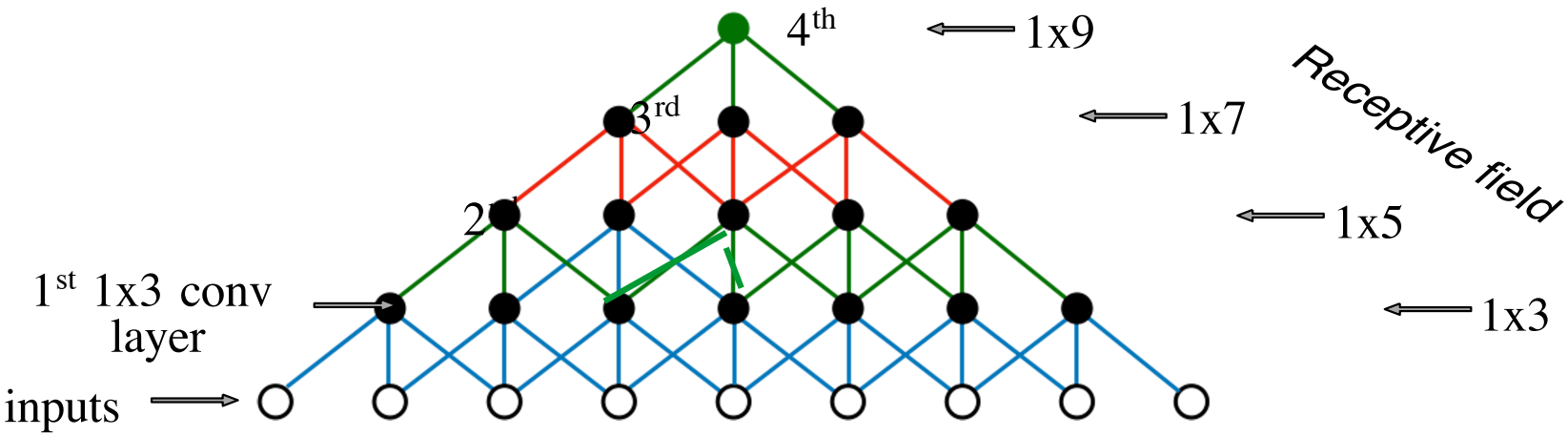
We can recognize larger objects by stacking several small convolutions!

Receptive field



Q: how many 3x3 convolutions we should use to recognize a 100x100px cat

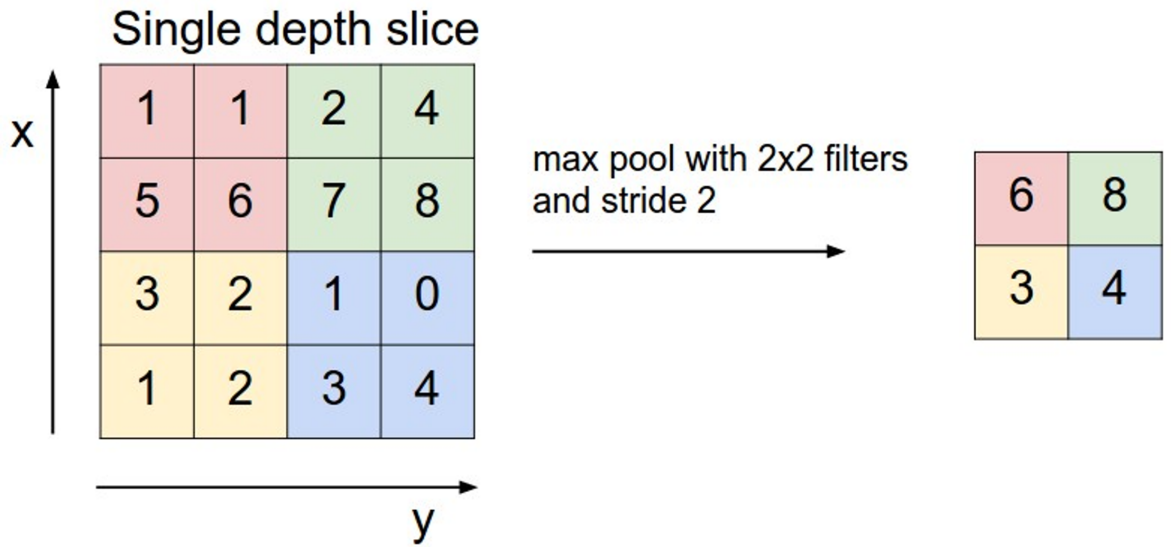
Receptive field



Q: how many 3x3 convolutions we should use to recognize a 100x100px cat

A: around 50... we need to increase receptive field faster!

Pooling



Intuition: What is the highest catness over this area?

Pooling

Motivation:

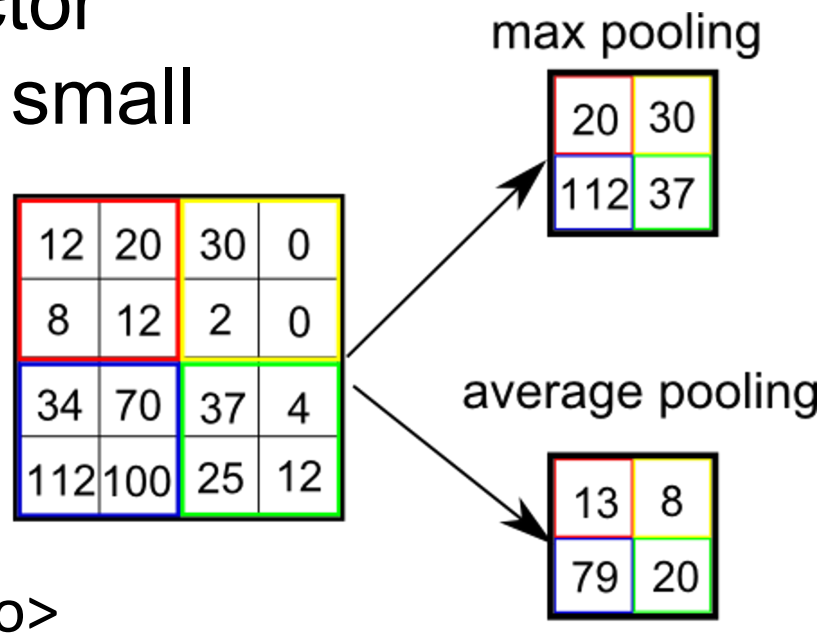
. Reduce layer size by a factor

. Make NN less sensitive to small image shifts

Popular types:

. Max

. Mean(average)

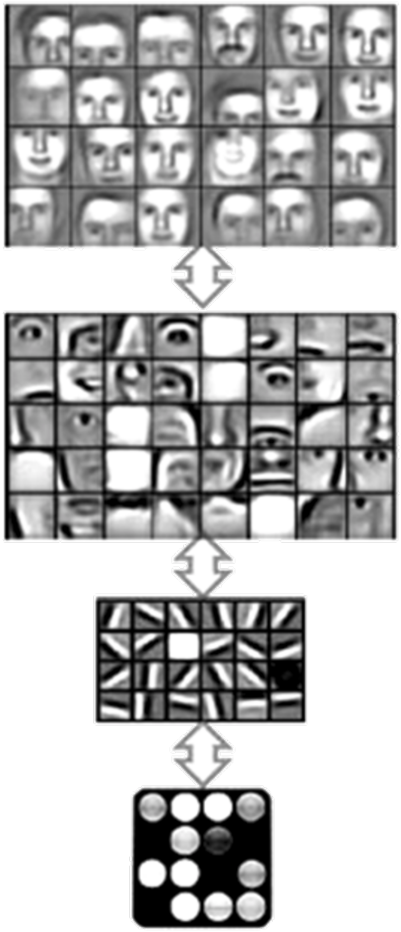


<Demo>

Self-check

- How large is the receptive field for the following architecture:
  - Conv1d(3x1)
  - MaxPool1d(3x1)
  - Conv1d(3x1)
  - MaxPool1d(3x1)
- Answer:
- How many parameters should such network learn:
  - Conv2d(3x3, 4 out channels)
  - MaxPool2d(3x3, stride=2, dilation=2)
- Answer:

Convolutional features stacking



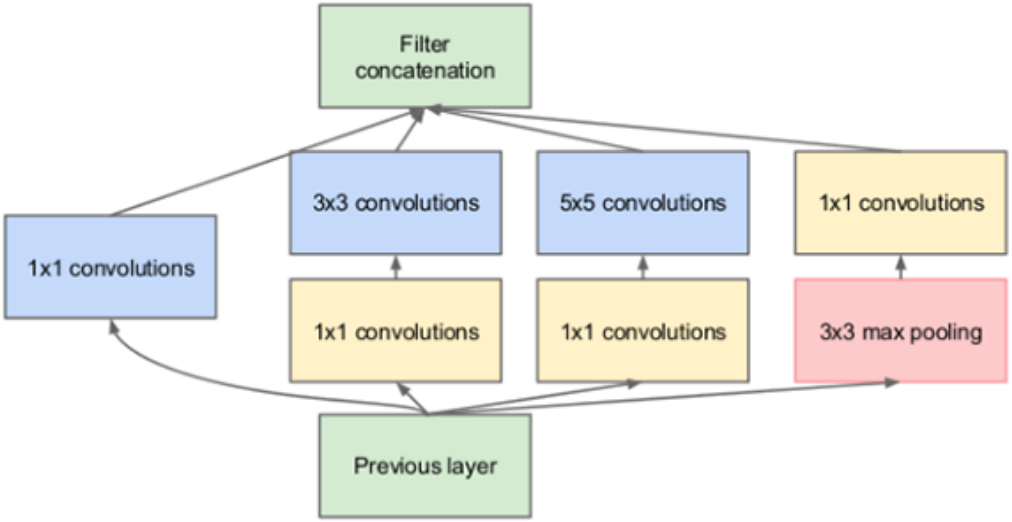
Discrete Choices

⋮

Layer 2 Features

Layer 1 Features

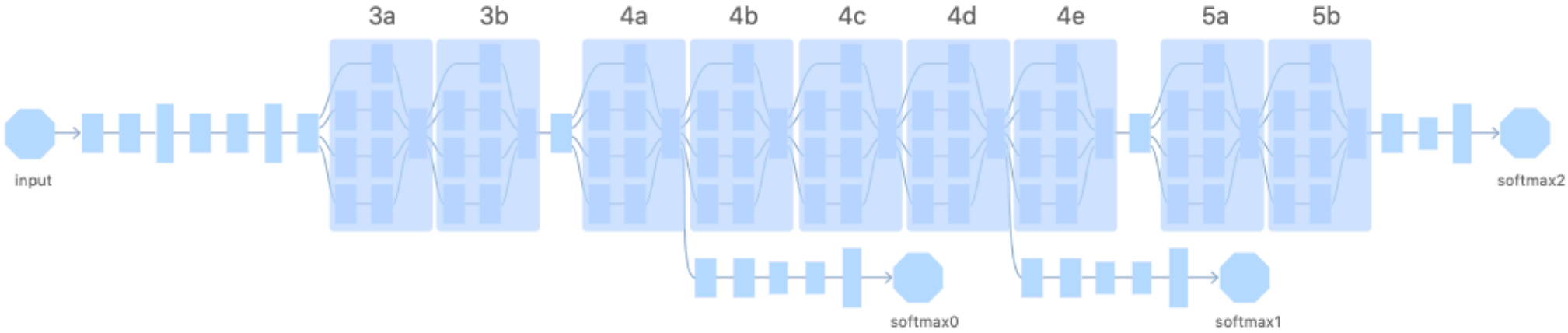
Original Data



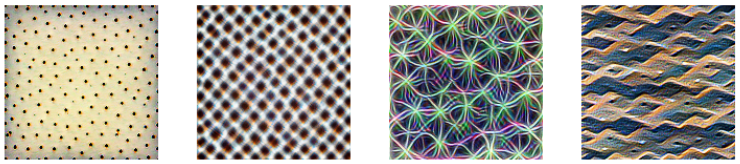
https://arxiv.org/pdf/1409.4842.pdf

State of the art computer vision: GoogleNet

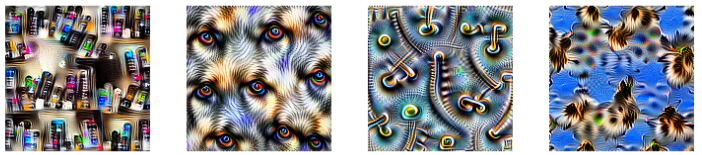
https://arxiv.org/pdf/1409.4842.pdf



Layer 3a



Layer 4a



Layer 4e

Layer 5b

https://distill.pub/2017/feature-visualization/appendix/

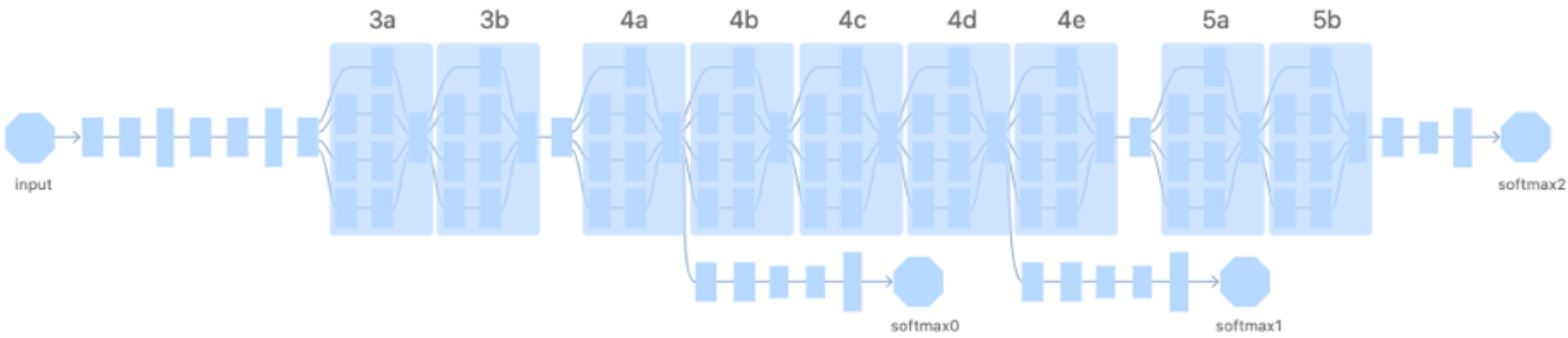
Possible problems with large networks

- MemoryError(0x...)
- Gradients can vanish
- Gradients can explode
- Activations can vanish
- Activations can explode

Possible solutions:

- Use pre-trained networks
- Different normalization techniques (wait for the next module ;-))

Combining networks




```
model = torchvision.models.resnet18(pretrained=True)
for param in model.parameters():
    param.requires_grad = False
# Replace the last fully-connected layer
# Parameters of newly constructed modules have requires_grad=True by
default
model.fc = nn.Linear(512, 100)

# Optimize only the classifier
optimizer = optim.SGD(model.fc.parameters(), lr=1e-2, momentum=0.9)
```

Other image-related tasks and libraries

General

- Image tagging, captioning, retrieval, morphing, encoding, upscaling
- Video processing, interpolation
- 3D point-clouds

HEP examples

- Jet tagging
- Particle tracking
- Calorimeter image analysis
- Cherenkov detector image analysis

- Torchvision,
- Kornia, Computer Vision Library for PyTorch
- MMF, A modular framework for vision & language multimodal research from Facebook AI Research (FAIR),
- PyTorch3D, https://pytorch3d.org/

Moar info

- Dimensionality reduction using 1x1 convolutions, https://machinelearningmastery.com/introduction-to-1x1-convolutions-to-reduce-the-complexity-of-convolutional-neural-networks/
- Sparse convolutions, https://github.com/facebookresearch/SparseConvNet
- Gauge Equivariant Convolutional Networks, https://arxiv.org/pdf/1902.04615.pdf
- Dilated convolutions, http://www.erogol.com/dilated-convolution/

Conclusion

- Images are the first-class citizens in Deep Learning world;
- Images are high-dimensional objects that demand powerful techniques for efficient processing:
  - Convolution;
  - Maxpooling;

  which are integral part of PyTorch;
- Pre-trained models and libraries are of great help.

Thank you!


austyuzhanin@hse.ru



anaderiRu


hse_lambda

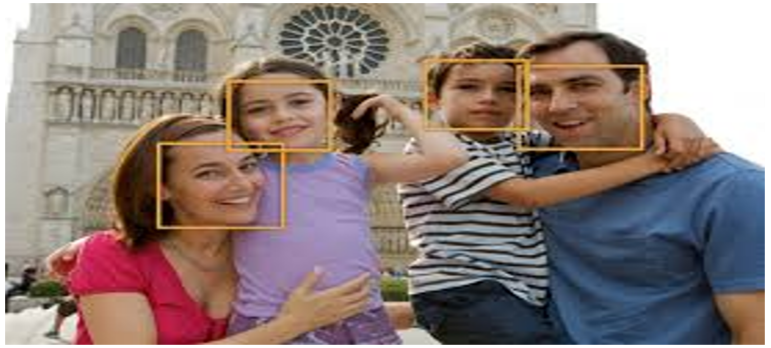
Andrey Ustyuzhanin

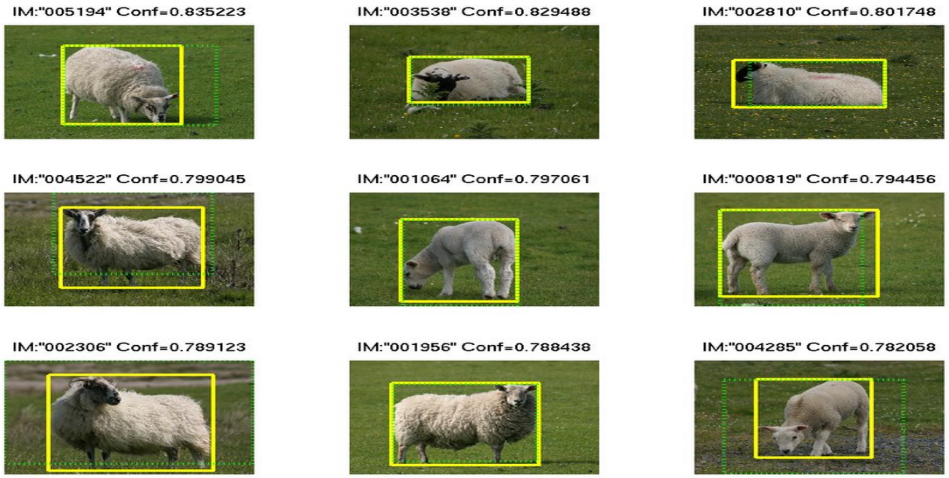
Bounding box regression

Predict object bounding box

(x0,y0,w,h)

or several bounding boxes for multiple objects.





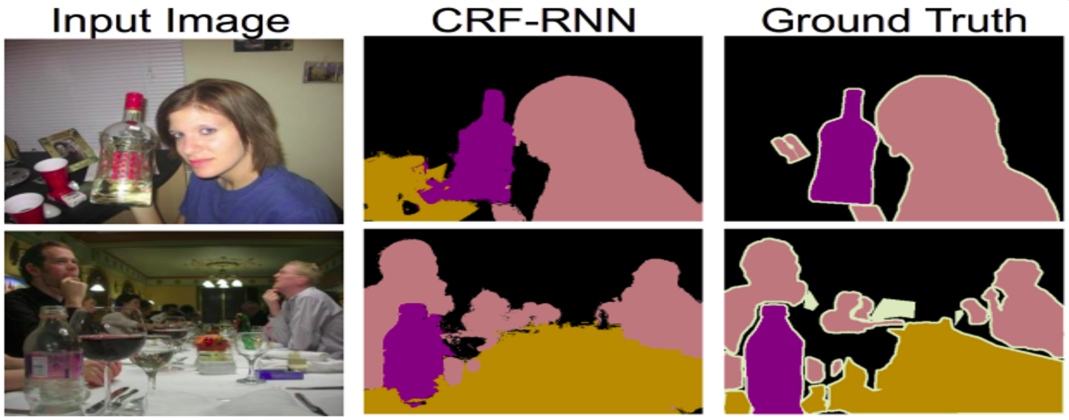
Applications examples:

- Face detection @ cameras
- Surveillance cameras
- Self-driving cars

Segmentation

Predict class for each pixel

(fully-convolutional networks)



Applications examples:

- Moar surveillance
- Brain scan labeling
- Map labeling

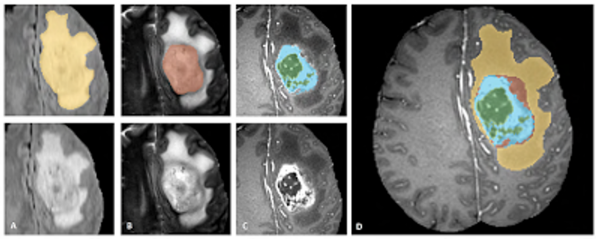
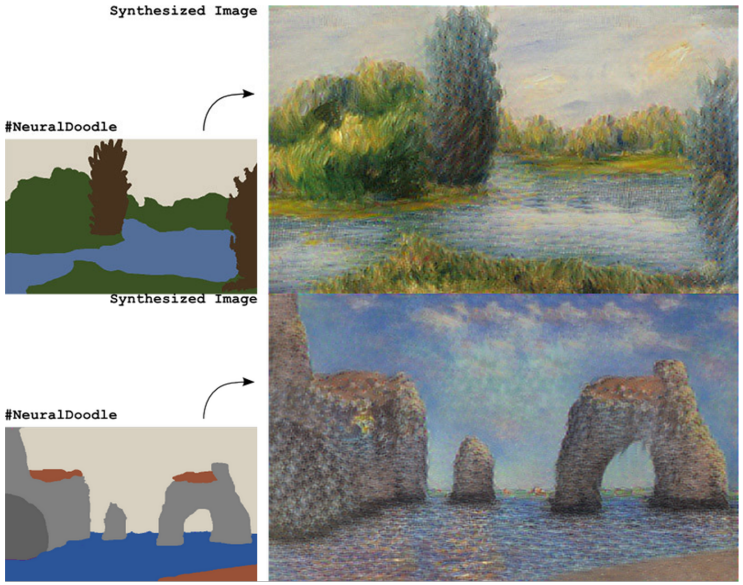


Image generation/transformation

- **Generation:** Given a set of reference images, learn to generate new images, resembling those you were given.

- **Transformation:** Given a set of reference images, learn to convert other images into ones resembling the reference set.



Neural Doodle
(D. Ulyanov et al.)