Course: Deep Learning

Unit 2: Computer Vision

Convolutional Neural Networks (I): Fundamentals

Luis Baumela

Universidad Politécnica de Madrid

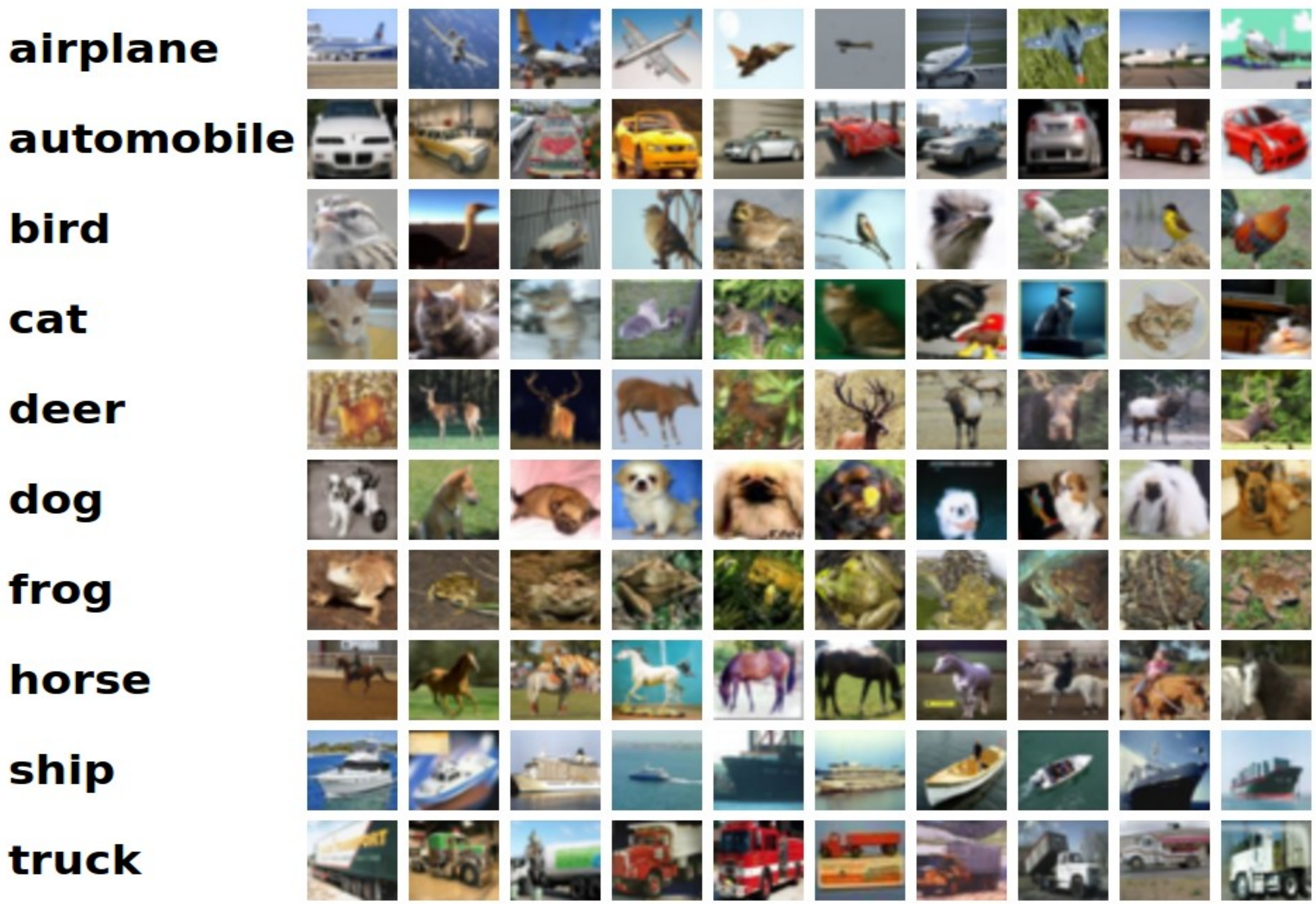
Convolutional Neural Networks

1. Introduction
2. CNN fundamental ideas
   - Local connectivity
   - Parameter sharing
   - Pooling and subsampling
   - Biological interpretation
3. CNN construction
   - Convolutional layer
   - Pooling layer
4. Other Convolutional layers
   - Dilated (atrous) convolutions
   - Grouped convolutions
5. Limitations

Introduction

- Goal: Object recognition (image classification)

  Identify the foreground object in an image
  For example (Cifar 10)



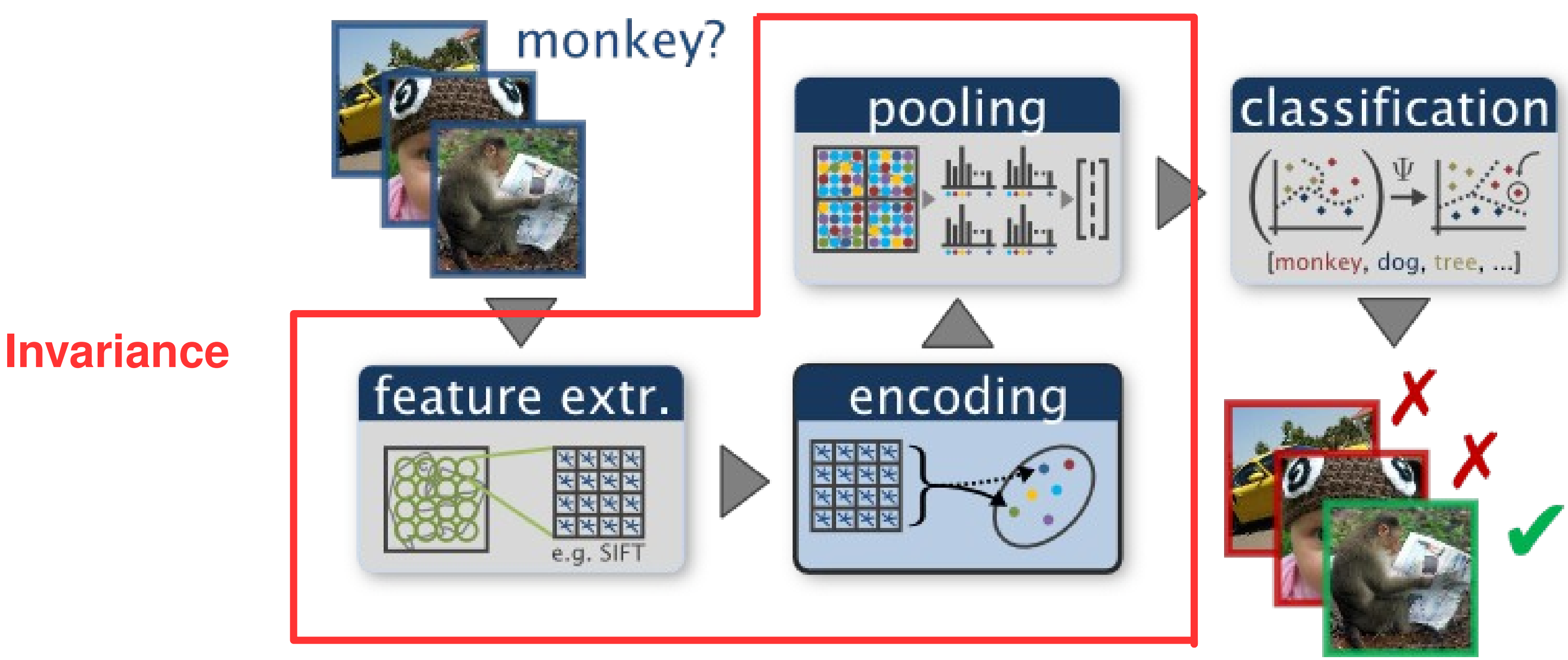
Introduction

- Object recognition difficulties ...
  1. Illumination (contrast, shadows, …)
  2. Geometric variability (scale, rotations, …)
  3. Deformation
  4. Clutter, occlussion
  5. High intra-class variability

Introduction

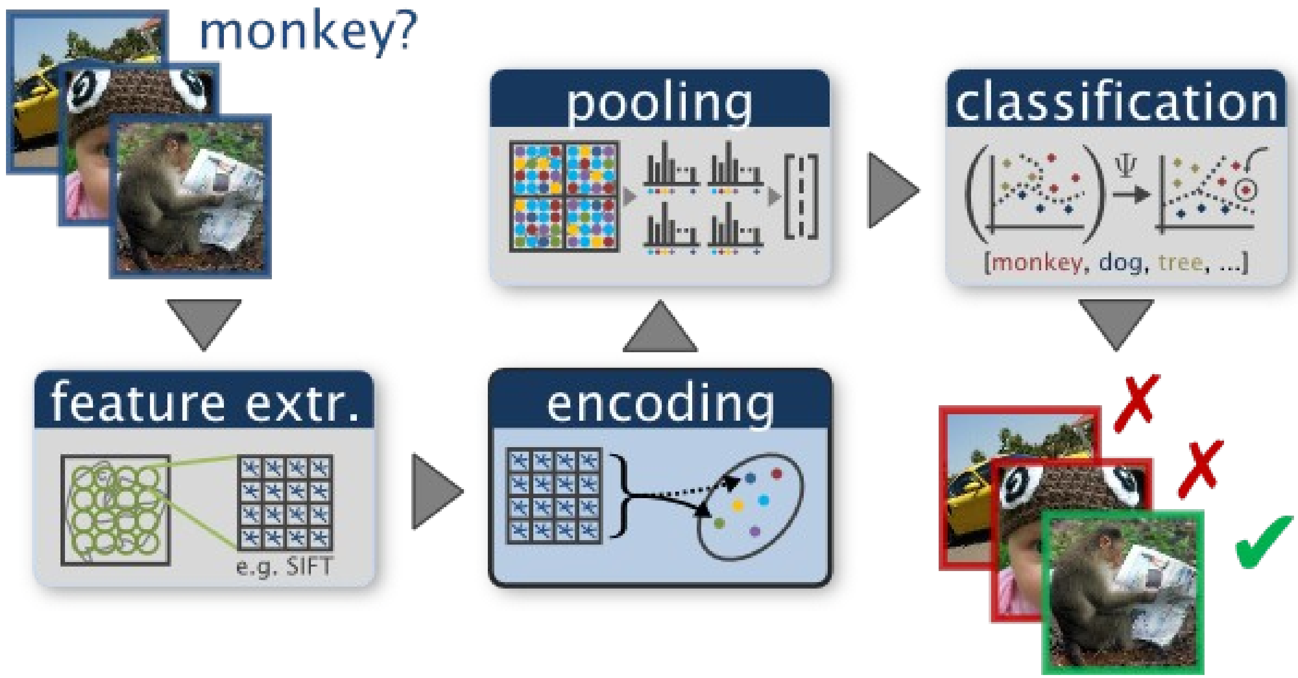
- Object recognition difficulties ...
  1. Illumination (contrast, shadows, …)
  2. Geometric variability (scale, rotations, …)
  3. Deformation
  4. Clutter, occlussion
  5. High intra-class variability

- Standard (shallow) object recognition approach



Introduction

- Standard (shallow) object recognition approach

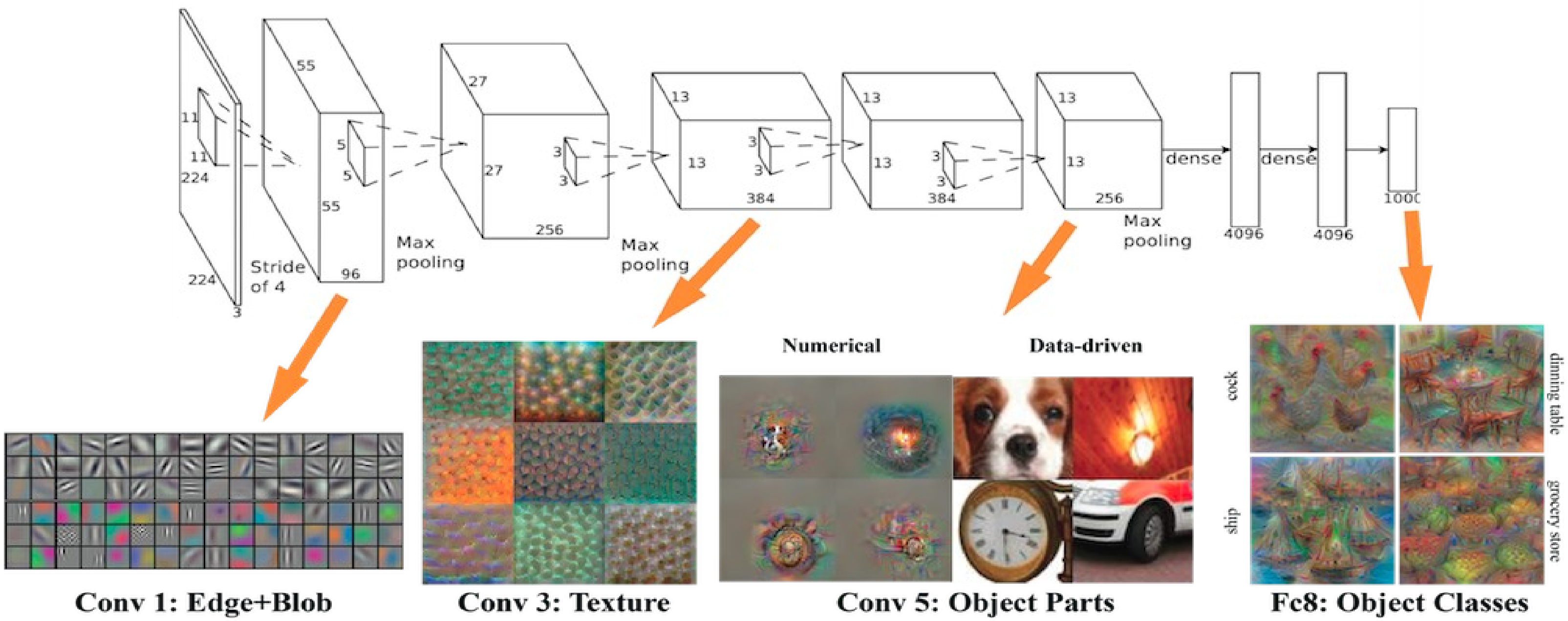


– Handcrafted/engineered features
  - Invariant to occlusions, changes illumination, position, orientation.
  - Not optimal for a certain task, but quite general.
  - Cannot be trained end-to-end
– Shallow representation: only one mid-level representation.
  - Difficult to generalize, low representation capabilities (poor abstraction)

\+ Lots of priors in form of design solutions, quite general
  - Requires less (than DL!) training data

Introduction

- Deep neural net approach



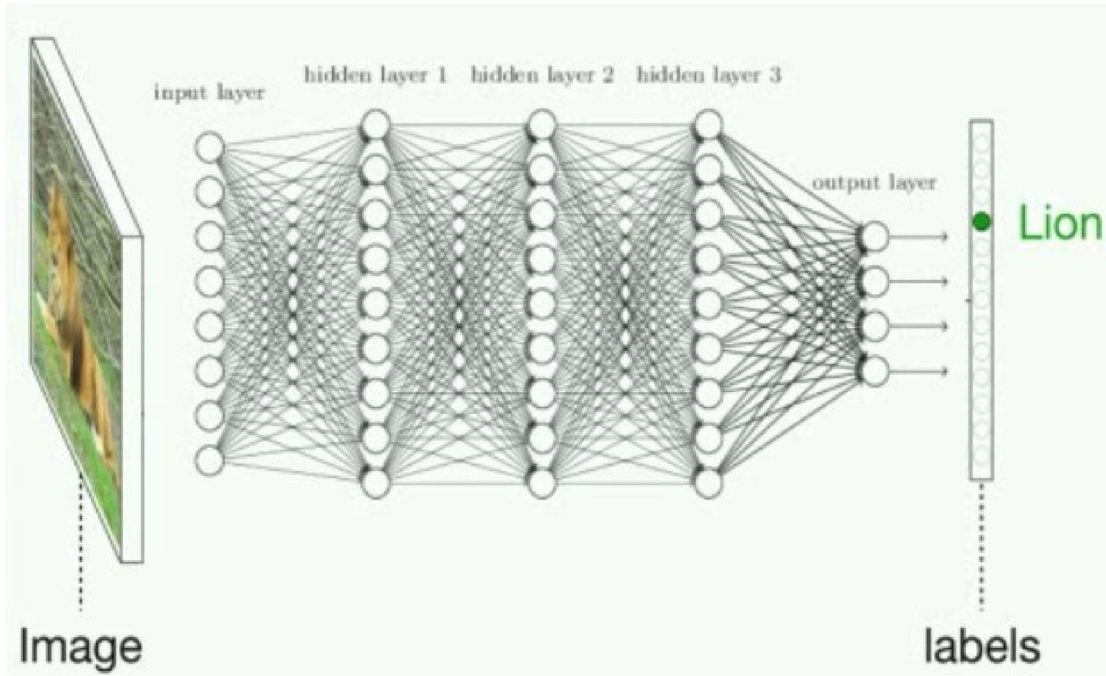
\+ Learned features
  - Requires no difficult tuning/engineering tasks.
  - End-to-end trainable: "optimal" for the task at hand.

\+ Hierarchical representation
  - Lets us better represent the data using higher abstractions.

– 100% data driven
  - Requires a lot of data <u>to be trained from scratch.</u>

Introduction

- Deep feed forward neural net approach



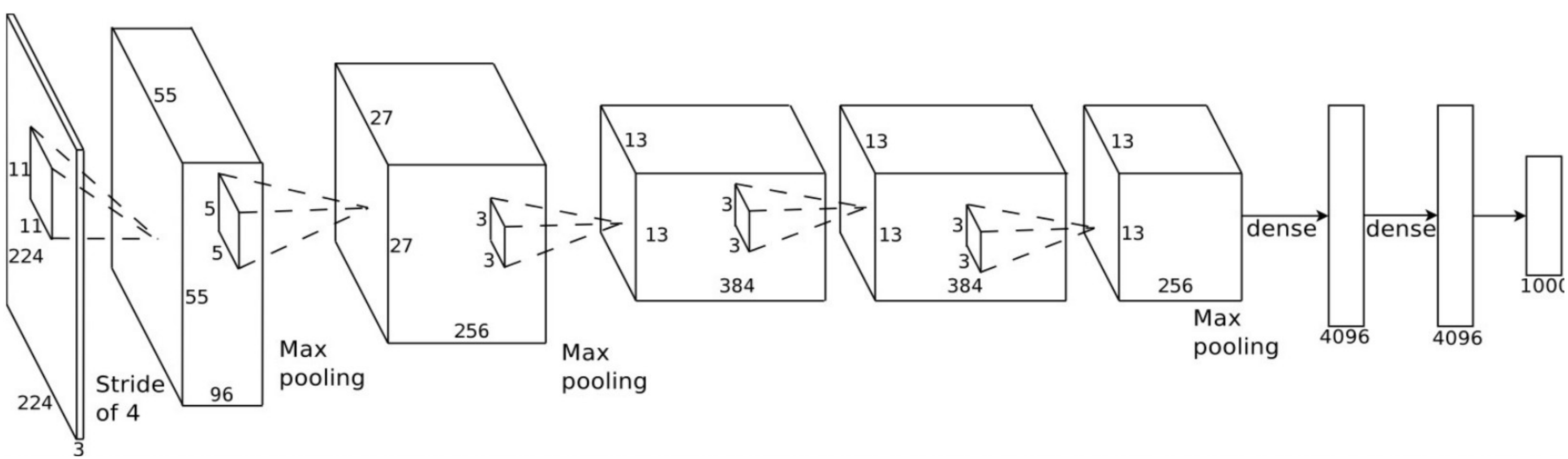
For computer vision problems

- Very high dimensional inputs (e.g. 256 x 256 x 3 = 196 k)
  Intractable in terms of data and computational requirements.
- Difficult to learn the spatial distribution of image information
- Solution
  - ➔ Use prior information to design an architecture with fewer parameters (introduce inductive biases).
  - ➔ Regularize the solution

Introduction

- Convolutional Neural Nets (CNNs) a Neural Net approach for image analysis

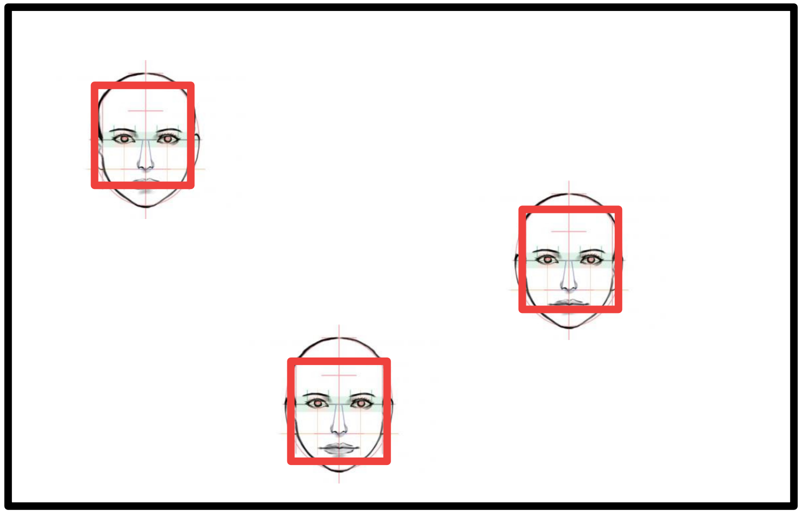
  Neural nets specifically adapted for computer vision problems:
  - Can deal with very high-dimensional inputs, for example:
    256 x 256 x 3 images = 196 608 input data
  - Exploit the spatial topology of pixels
    multi channeled images, video, sound
  - Certain degree of invariance
    translation, illumination changes, ...



CNNs fundamental ideas

- 2D / 3D spatial distribution
  - Can deal with very high-dimensional inputs
  - Exploit the 2D/3D topology of pixels
  - Certain degree of invariance

Information in images is distributed in space

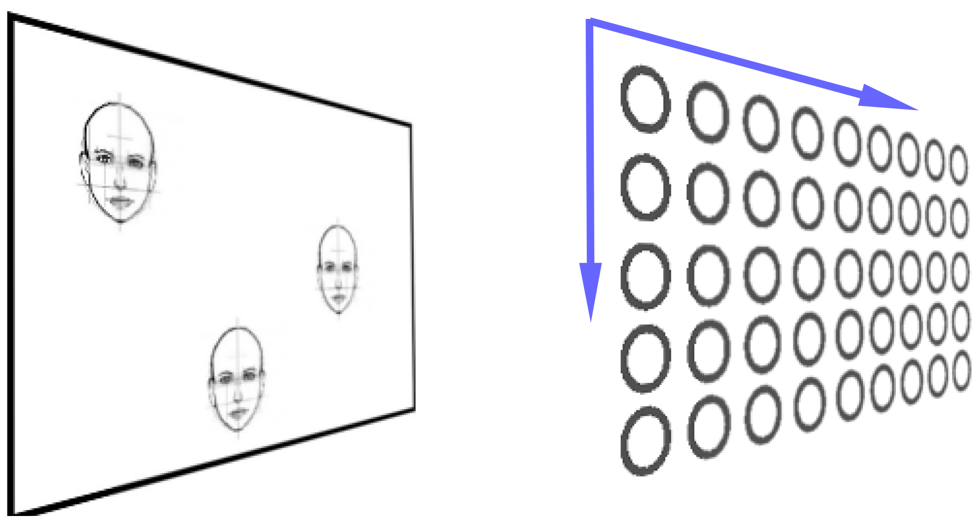


Face detection

CNNs fundamental ideas

- 2D / 3D spatial distribution
  - Can deal with very high-dimensional inputs
  - **Exploit the 2D/3D topology of pixels**
  - Certain degree of invariance

The input image and the net layers are arranged in a 2D / 3D layout.

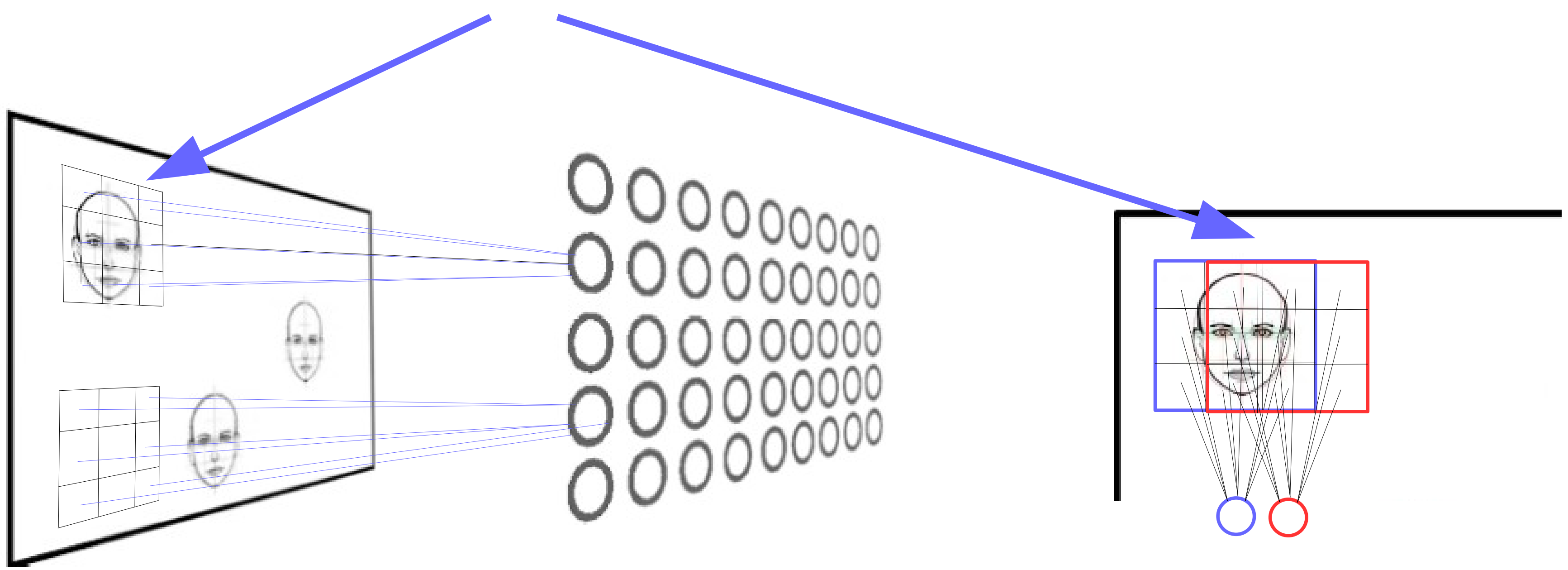


CNNs fundamental elements

- Local / sparse interactions

Neural nets specifically adapted for computer vision problems:
- **Can deal with very high-dimensional inputs**
- **Exploit the 2D/3D topology of pixels**
- Certain degree of invariance

Each cell only “looks at” a small portion of the previous layer/image: the **receptive field**



CNNs fundamental ideas

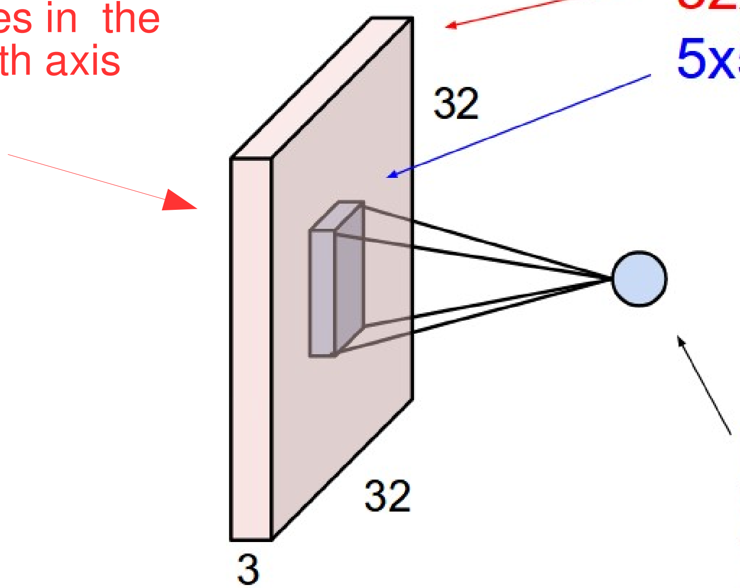
- Each unit is a standard NN cell
  - **Can deal with very high-dimensional inputs**
  - **Exploit the 2D/3D topology of pixels**
  - Certain degree of invariance

  Each cell only “looks at” a small spatial portion of the previous layer/image: the **receptive field**

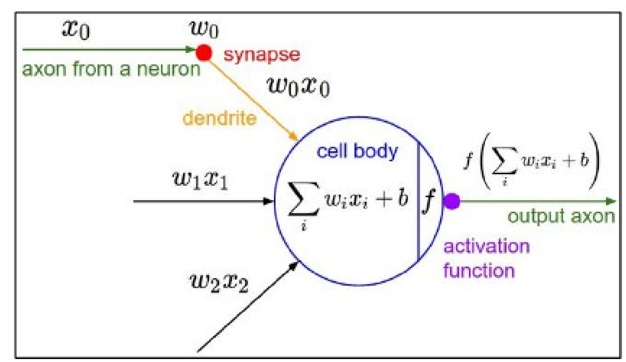
“looks at” all slices in the depth axis

32x32x3 image

5x5x3 filter



1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)



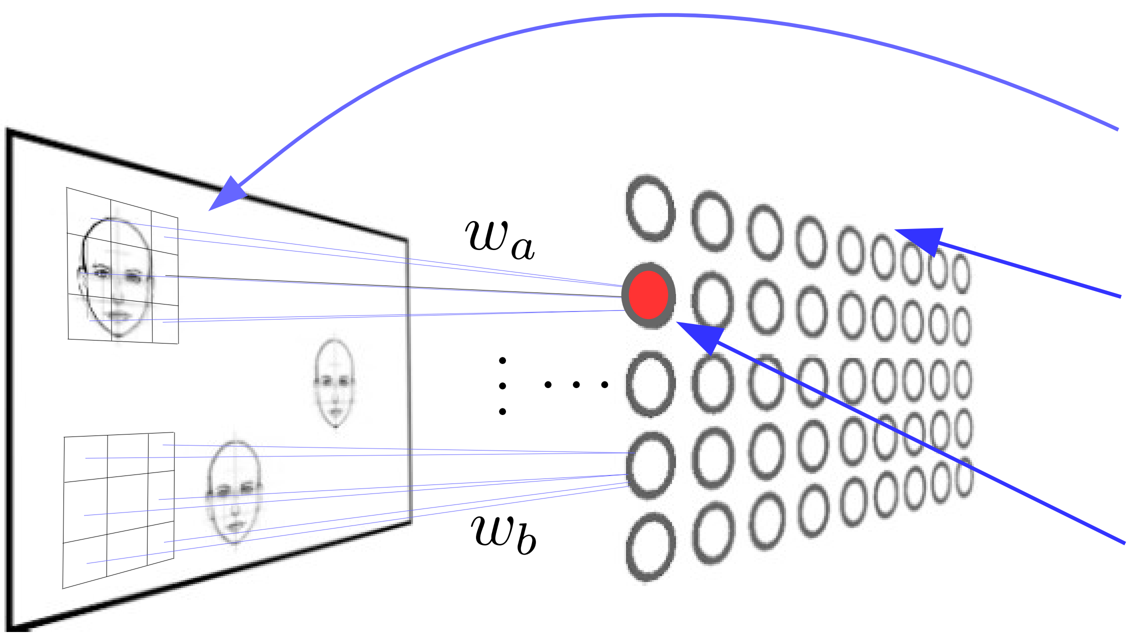
It’s just a neuron with local connectivity...

http://cs231n.stanford.edu/

CNNs fundamental ideas

- Parameter sharing
  - Can deal with very high-dimensional inputs
  - Exploit the 2D/3D topology of pixels
  - **Translation covariance**

  All cells in the same slice share their weights



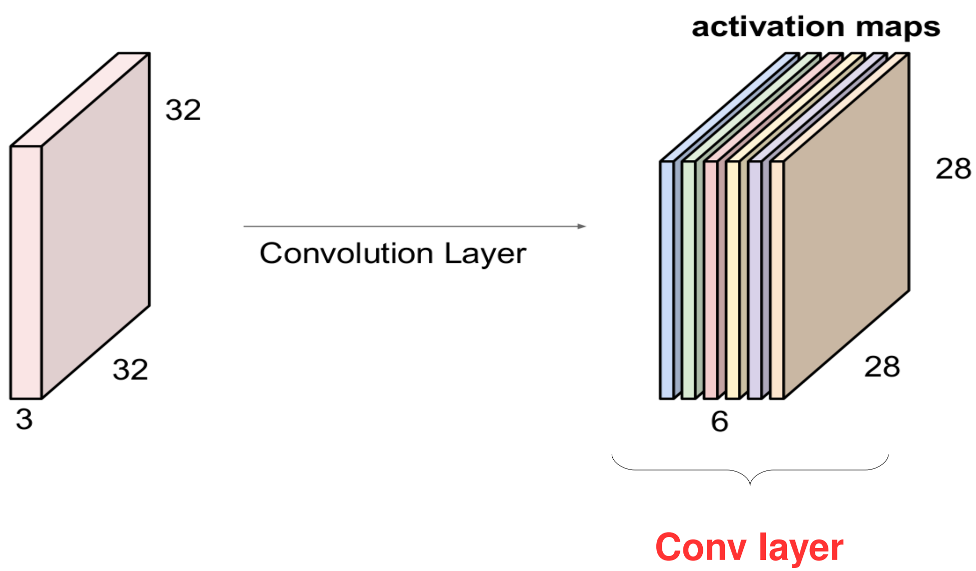
- Each cell covers an input region, the **receptive field**.
- Each cell fires whenever the feature is detected in its receptive field.
- Each slice detects a certain feature: **feature / activation map**

$$\boxed{w_a = w_b = \dots}$$

CNNs fundamental ideas

- CNN layer
  - **Can deal with very high-dimensional inputs**
  - **Exploit the 2D/3D topology of pixels**
  - **Translation covariance**

  A convolutional layer is composed by set of feature/activation maps

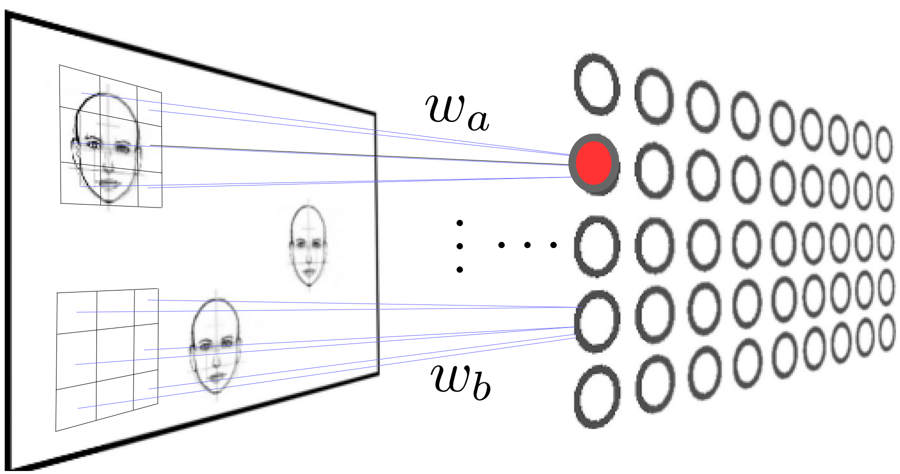


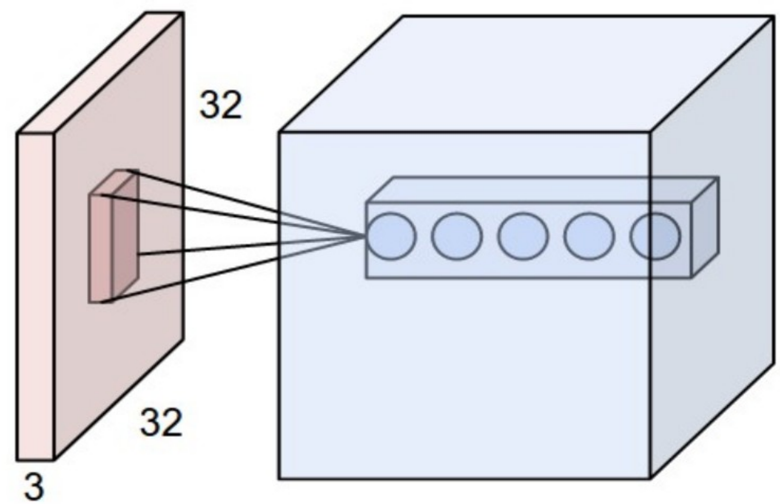
- Each cell covers an input region, the receptive field.
- Each cell fires whenever the feature is detected in its receptive field.
- Each slice detects a certain feature: feature / activation map.
- The pre-activation in the feature map of a slice is a **convolution**
- A **convolutional layer** is a group of activation maps

http://cs231n.stanford.edu/

CNNs fundamental ideas

- The activations of cells in a spatial location represent the features detected in that location





The number of parameters in a feature map depends on the size of the receptive field.

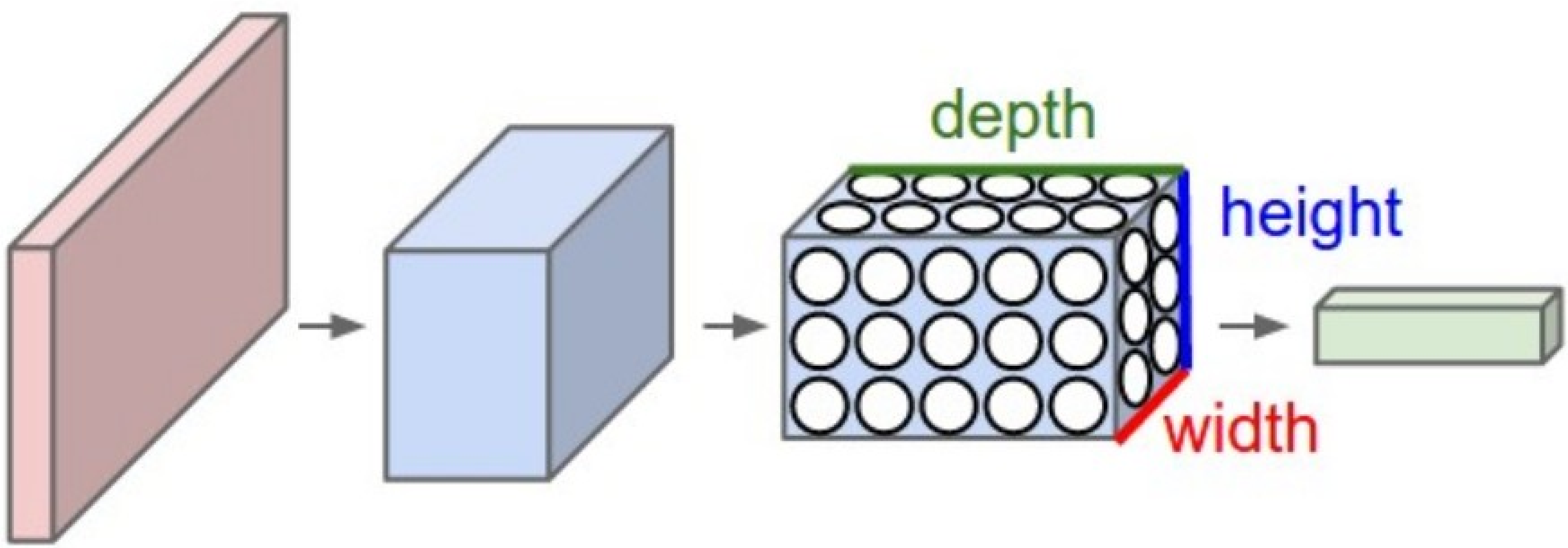
http://cs231n.stanford.edu/

CNNs fundamental ideas

- A ConvNet arranges its neurons in three dimensions

  Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations.

  The number of parameters in a conv layer does not depend on the spatial dimension of the input tensor.



http://cs231n.stanford.edu/

CNNs fundamental ideas

- Spatial pooling
  Summarizes statistics of neigbouring activations

0,01	0,15	0,14	0,10
0,02	0,15	0,15	0,02
0,40	0,60	0,02	0,03
0,70	0,40	0,03	0,02

0,15	0,15
0,70	0,03

Typical operators:
- Non-overlapping
- Small windows

Types
- `max`
- `average`
- `weighted aver`

$\equiv$ strided conv

Improves the peformance of image classification models:
- Introduces invariance to small local translations
- Reduces the spatial extent of the net

However, it
- discards information about the location of features in the image

CNNs fundamental ideas

- Spatial pooling

It has important opponents

"The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well (for image classification) *is a disaster."*

Geoffrey Hinton

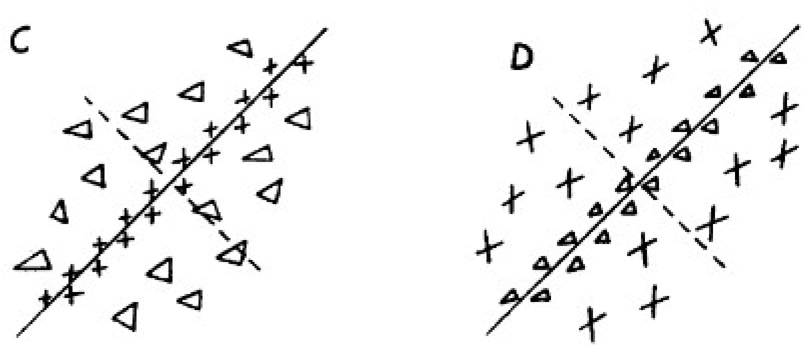
Non overlaping pools lose valuable information about where things are.

https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/

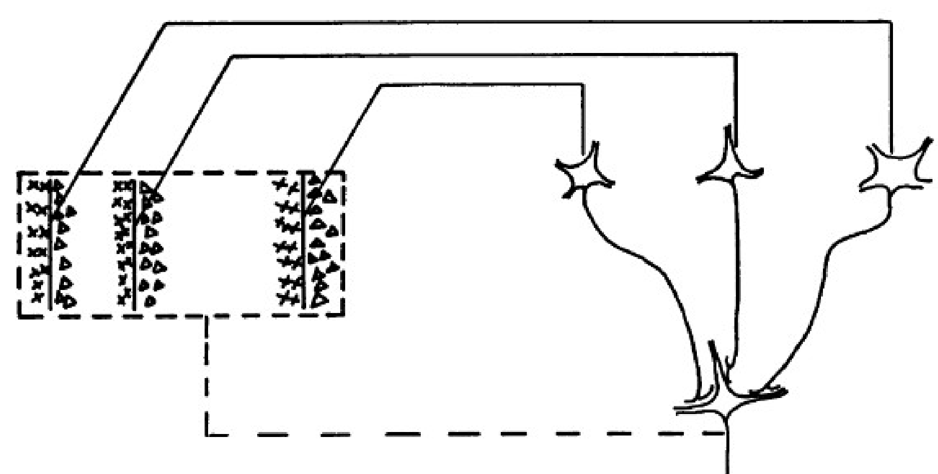
CNNs fundamental ideas

- CNNs have a biological interpretation
  [Hubel & Wiesel J. Physiology, 1962]
  Hierarchical model composed of
  - S (simple) cells activate when they detect basic shapes like lines
  - C (complex) cells combine the activation of the simple cells activating to the same shapes but with less sensibility to position

S cells would behave like a convolution layer
C cells like a pooling layer



Arrangement of receptive fields

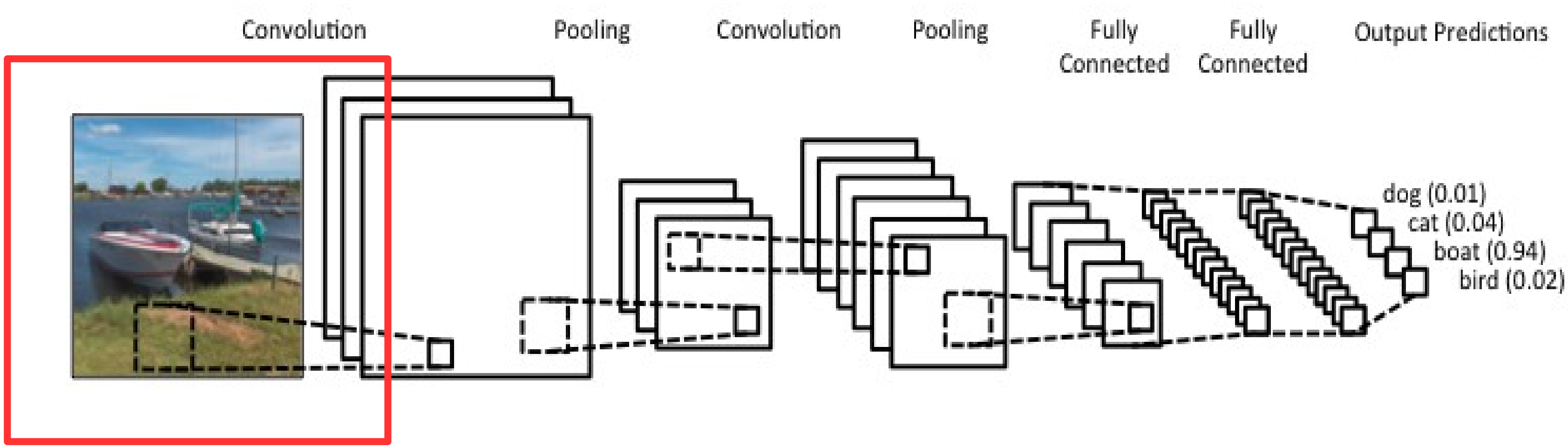


Organization of complex receptive fields

Hubel, Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, The Journal of Physiology, 1962

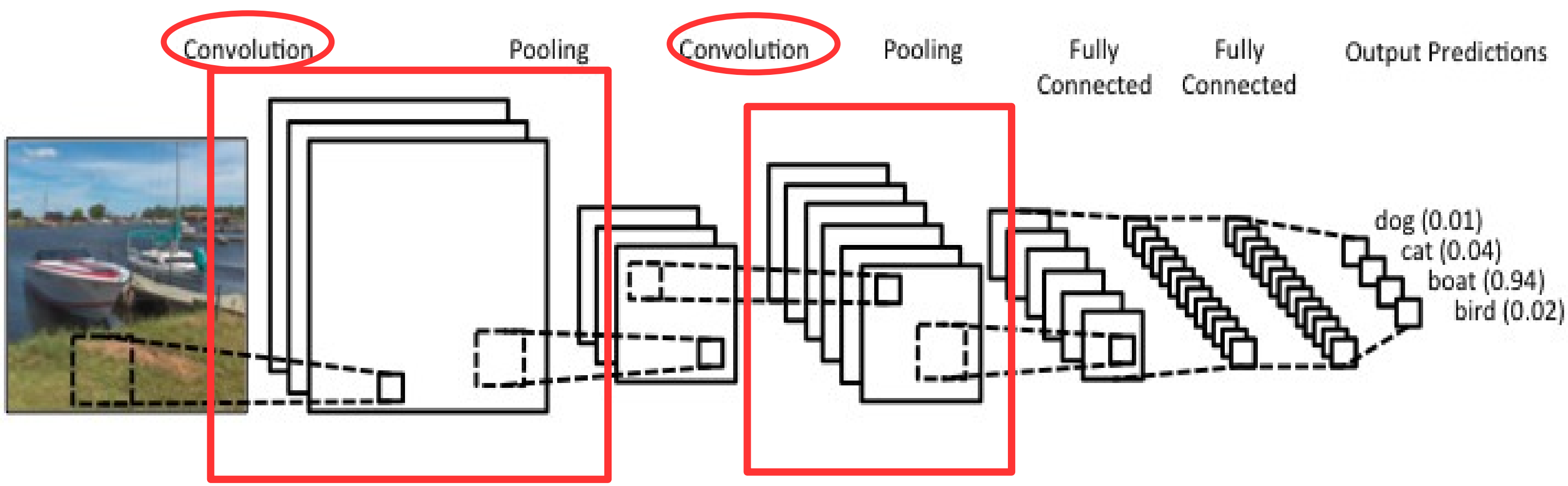
Conv Net construction

- Typical layers
  - **Input**
    holds the raw pixel values (RGB, Grey, ….)
  - Conv
  - ReLU
  - Pool
  - FC
  - SoftMax



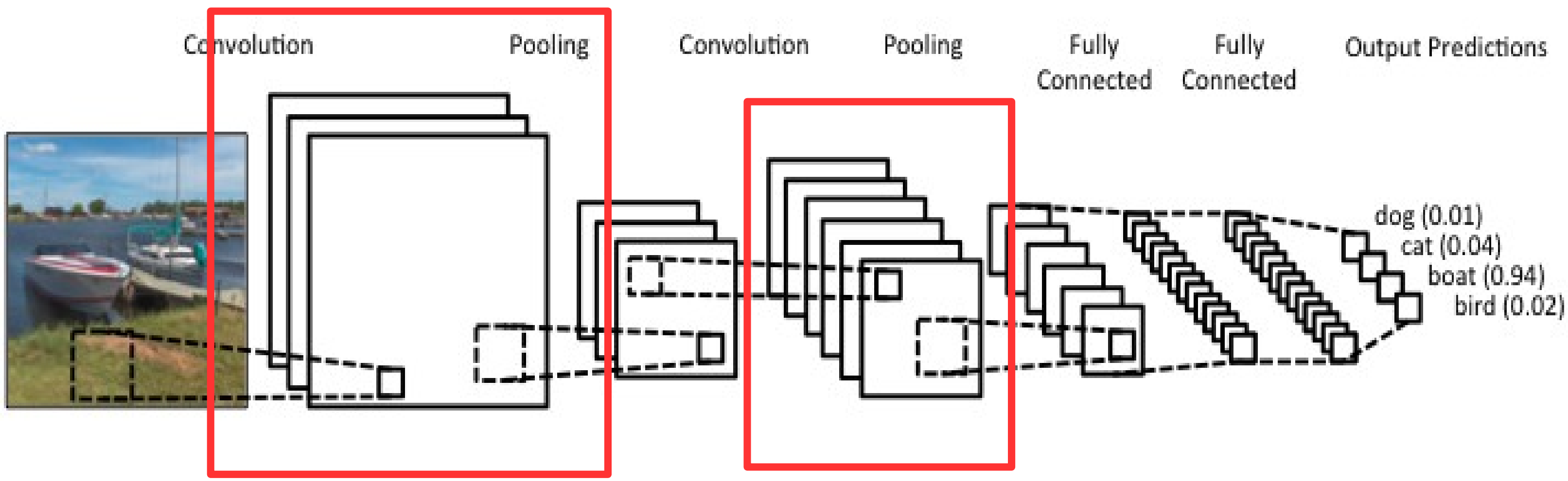
Conv Net construction

- Typical layers
  - Input
  - **Conv**
    holds all the feature activation maps computed from convolutions on the input data (an image or another layer).
  - ReLU
  - Pool
  - FC
  - SoftMax



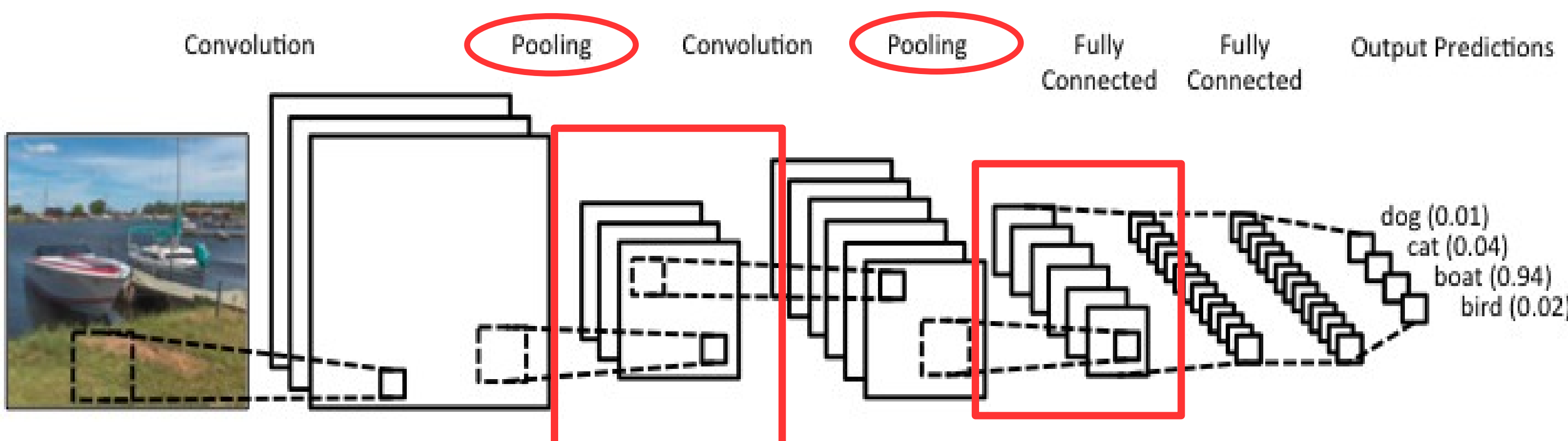
Conv Net construction

- Typical layers
  - Input
  - Conv
  - **ReLU**
    holds the result of applying the non-linear activation function to each cell in each feature map
  - Pool
  - FC
  - SoftMax



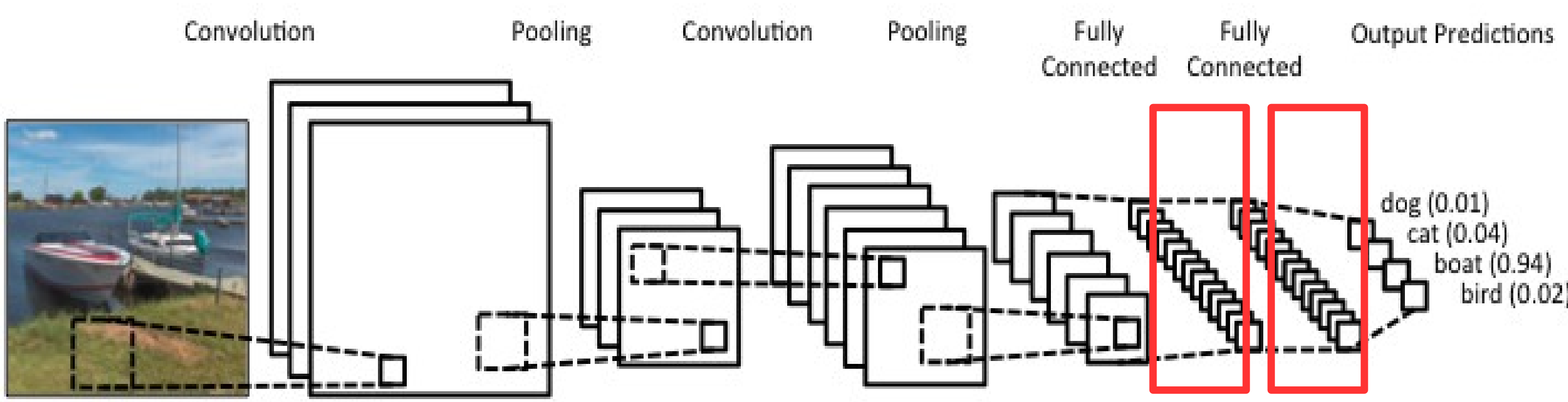
Conv Net construction

- Typical layers
  - Input
  - Conv
  - ReLU
  - **Pooling**
    performs a downsampling operation along the spatial dimension
  - FC
  - SoftMax



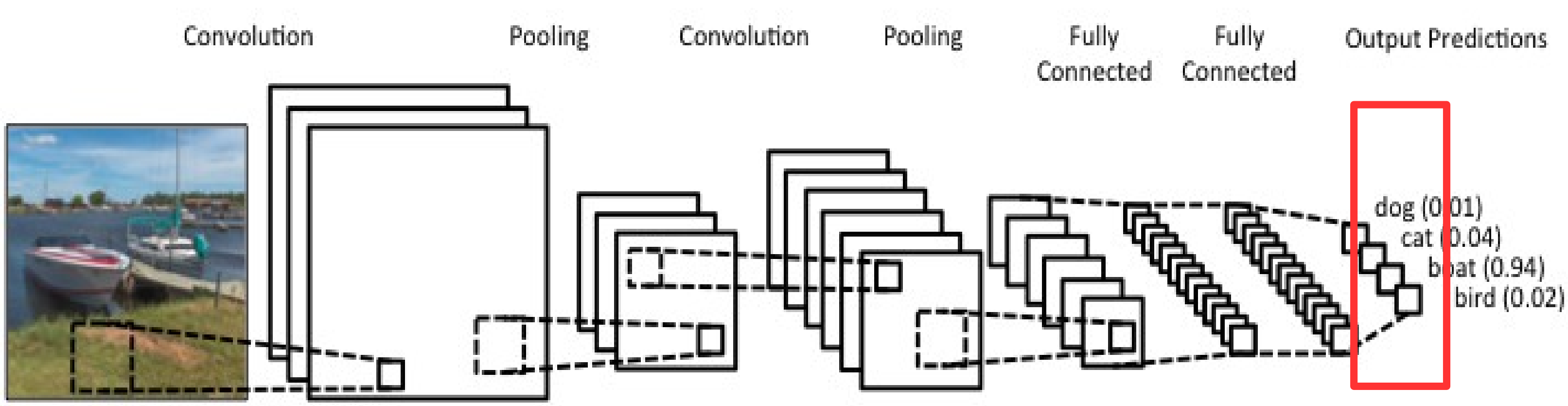
Conv Net construction

- Typical layers
  - Input
  - Conv
  - ReLU
  - Pool
  - **FC**
    fully-connected layer is the classifier
  - SoftMax



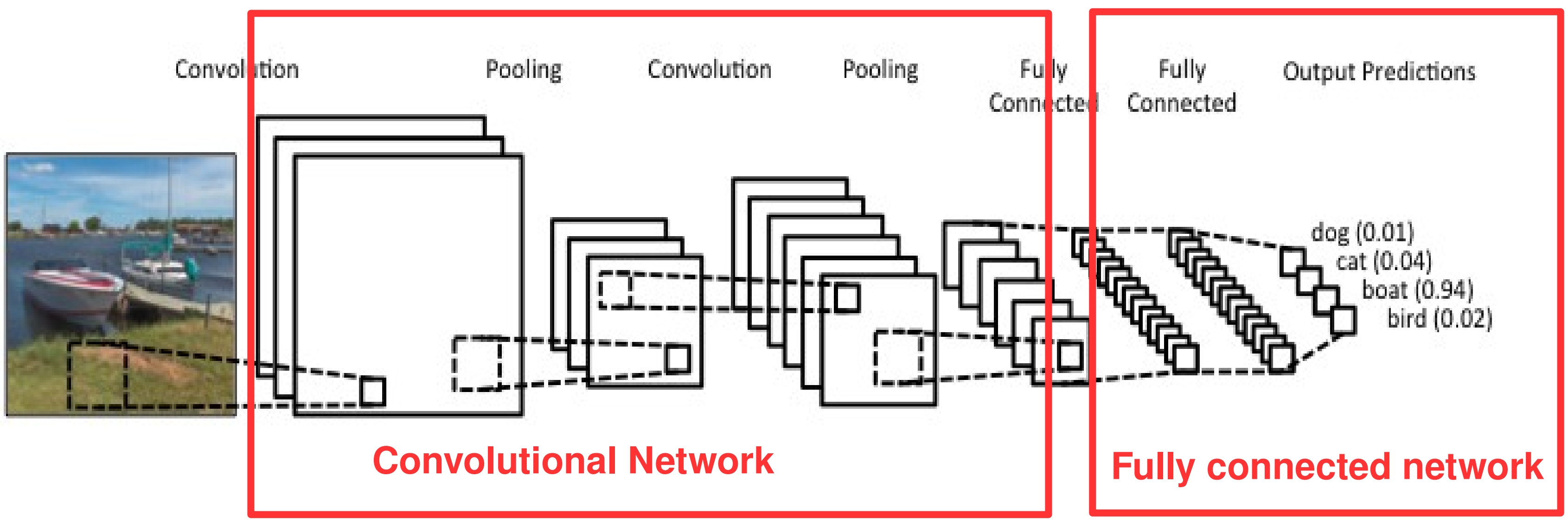
Conv Net construction

- Typical layers
  - Input
  - Conv
  - ReLU
  - Pool
  - FC
  - **SoftMax**
    final layer to compute the loss



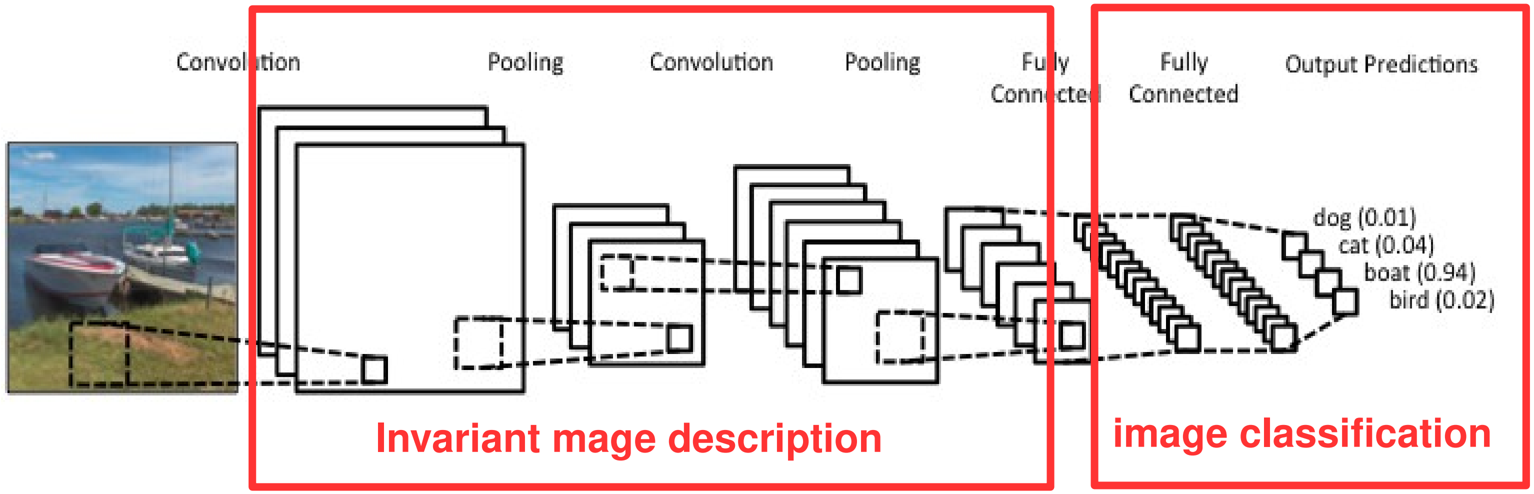
Conv Net construction

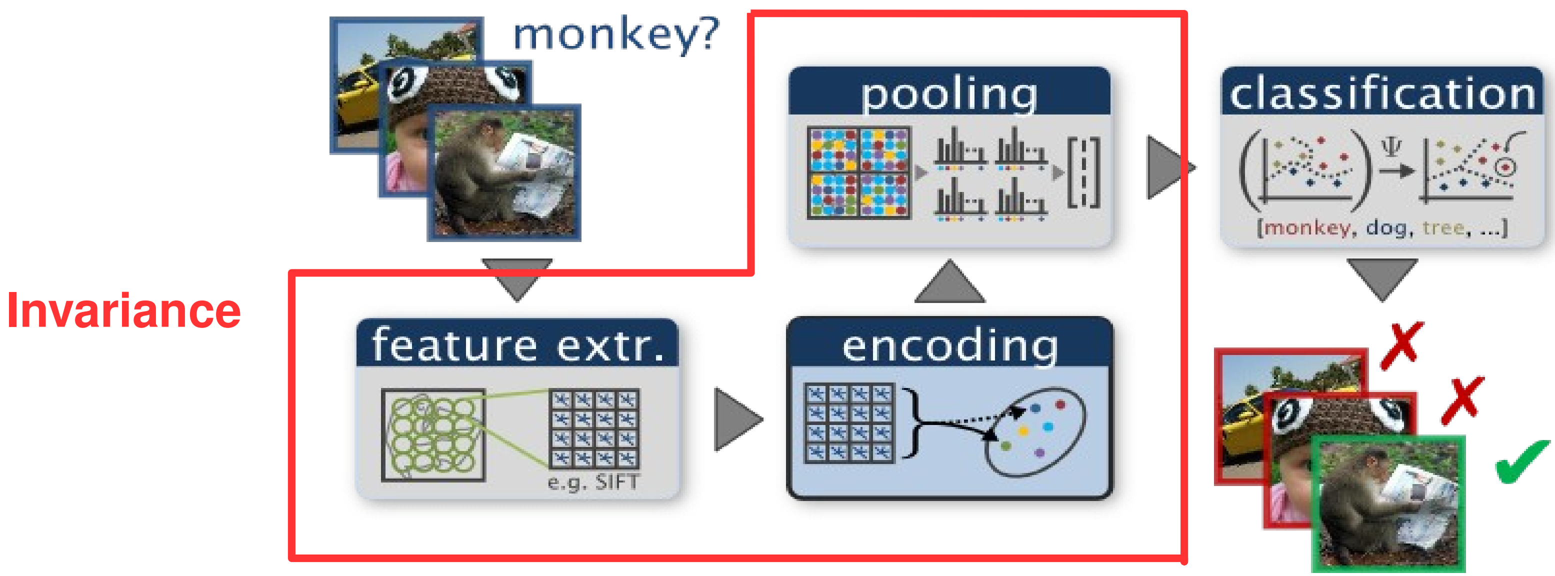
- Typical layers
  - Input
  - Conv
  - ReLU
  - Pool
  - FC
  - SoftMax
    final layer to compute the loss



Conv Net construction

- Deep vs shallow object recognition





Conv Net construction

- Convolutional layer parameters

  - Accepts a volume of size $W_1 \times H_1 \times D_1$
  - Requires four hyperparameters:
    - Number of filters K,
    - their spatial extent F,
    - the stride S,
    - the amount of zero padding P.
  - Produces a volume of size $W_2 \times H_2 \times D_2$ where:
    - $W_2 = (W_1 - F + 2P)/S + 1$
    - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
    - $D_2 = K$
  - With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
  - In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

http://cs231n.stanford.edu/

Conv Net construction

- Convolutional layer. Sample configuration in Keras

```
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', data_form
```

Arguments

K
- **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).

F
- **kernel_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.

S
- **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any `dilation_rate` value != 1.

P
- **padding**: one of `"valid"` or `"same"` (case-insensitive). Note that `"same"` is slightly inconsistent across backends with `strides` != 1, as described here
- **activation**: Activation function to use (see activations). If you don't specify anything, no activation is applied (ie. "linear" activation: `a(x) = x`).
- **use_bias**: Boolean, whether the layer uses a bias vector.
- **kernel_initializer**: Initializer for the `kernel` weights matrix (see initializers).
- **bias_initializer**: Initializer for the bias vector (see initializers).

Conv Net construction

- Pooling layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires two hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input

Common settings:

F = 2, S = 2
F = 3, S = 2

Conv Net construction

- Pooling layer. Sample configuration in Keras

```
keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid', d
```

Arguments

F

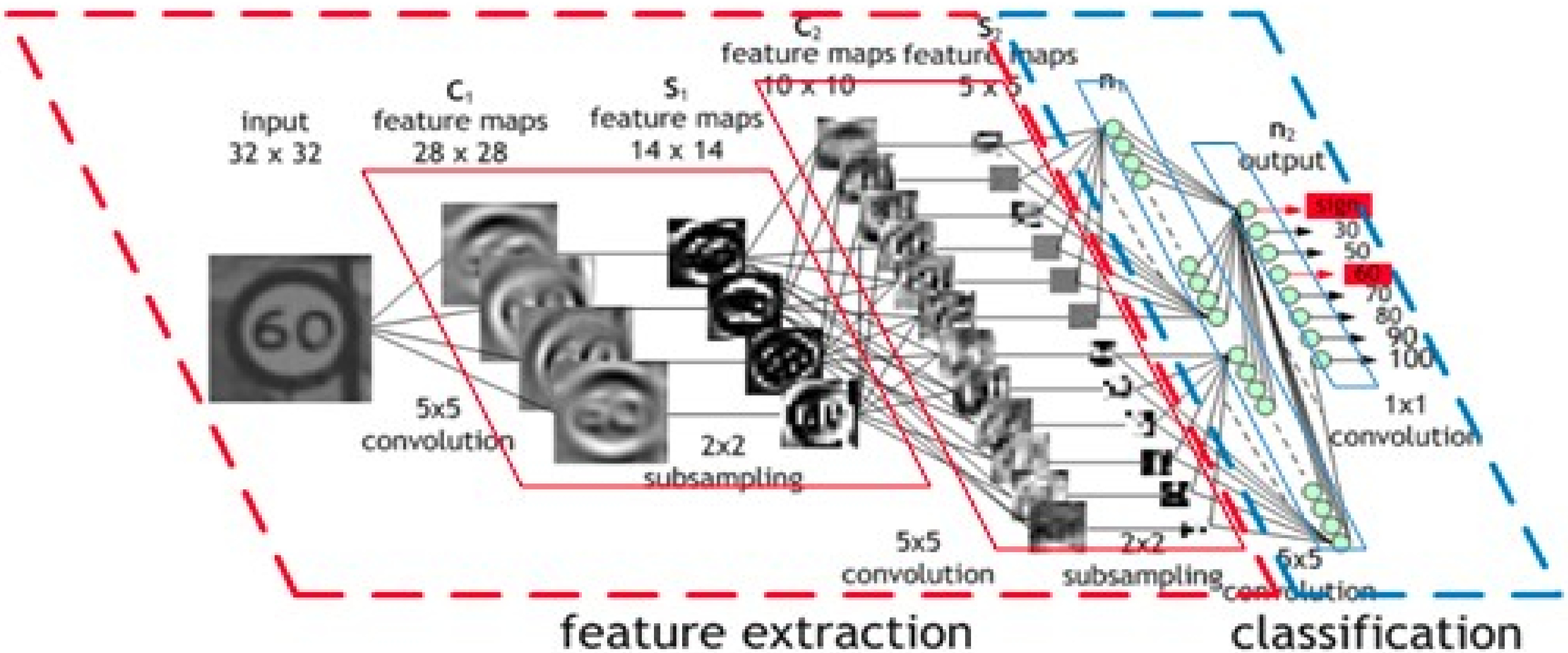
- **pool_size**: integer or tuple of 2 integers, factors by which to downscale (vertical, horizontal). (2, 2) will halve the input in both spatial dimension. If only one integer is specified, the same window length will be used for both dimensions.

S

- **strides**: Integer, tuple of 2 integers, or None. Strides values. If None, it will default to `pool_size`.
- **padding**: One of `"valid"` or `"same"` (case-insensitive).

Conv Net construction

- Sample Conv Net



Other Convolutional layers

- Dilated (atrous) convolution

  Skip d positions in the input signal when operating each kernel coefficient. Half padding dilated convolution

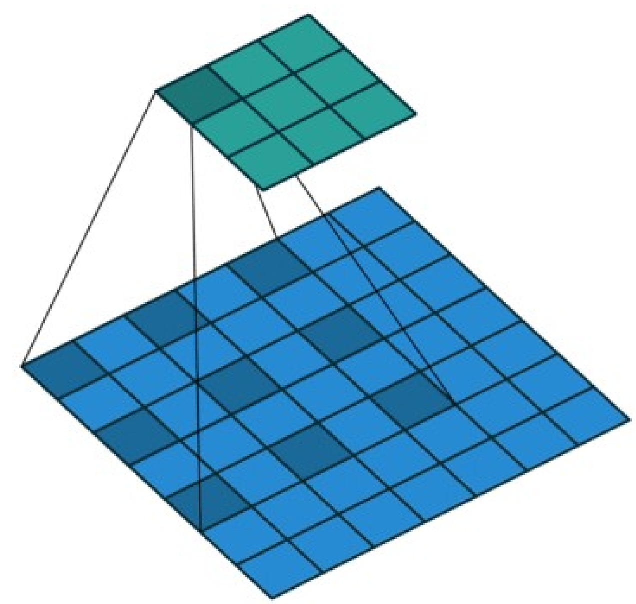
$$o(r,t) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} h(i,j)e_e[s\,r + d\,i - (m-1)/2, s\,t + d\,j - (n-1)/2]$$

  In general, the size of the output of one dim

$$\frac{M + 2p - m - (d-1)(m-1)}{s} + 1$$

  If half padded, $s = 1$ and $d = 2$ the ouput is

$$M - 2m + 2 \times N - 2n + 2$$



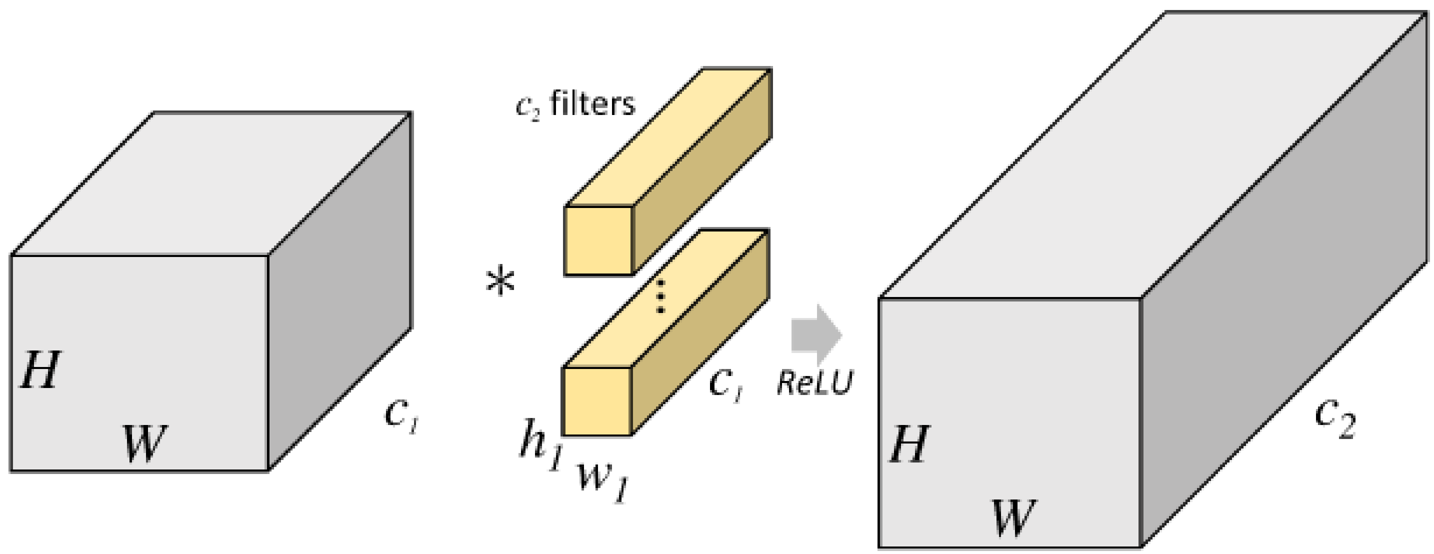
Yu, Koltun. Multiscale context agregation by dilated convolutions. ICLR 2016

https://github.com/vdumoulin/conv_arithmetic

Other Convolutional layers

- Grouped convolutions

  In a **standard convolutional layer** each filter operates on the whole third dimension of the layer/tensor below.



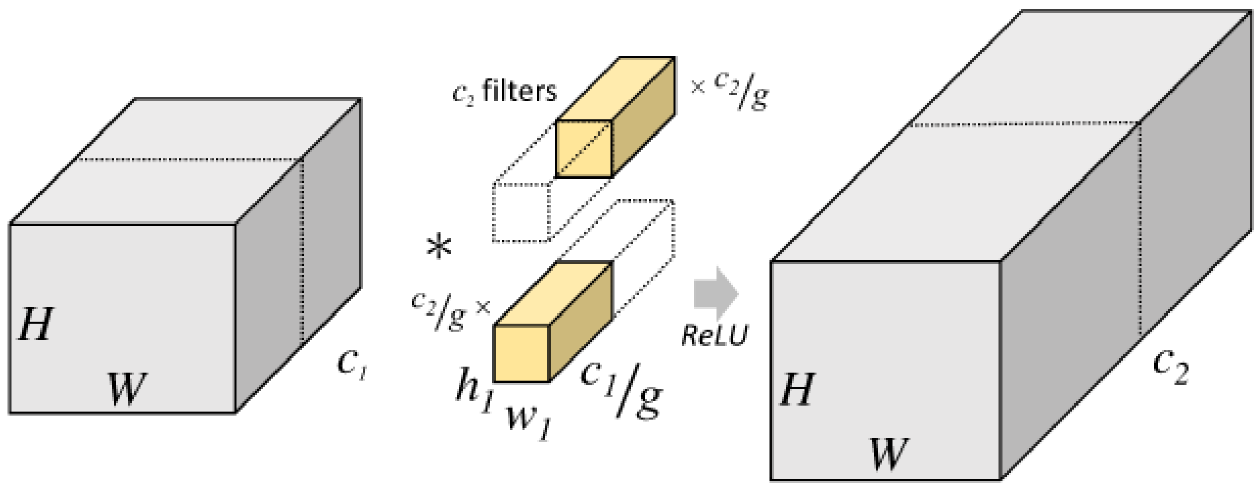
When the third dimension dominates, the number of parameters and computation increases.

https://blog.yani.ai/filter-group-tutorial/

Other Convolutional layers

- Grouped convolutions

  In a **grouped convolutional layer** each filter operates on a group of feature maps of the layer/tensor below.



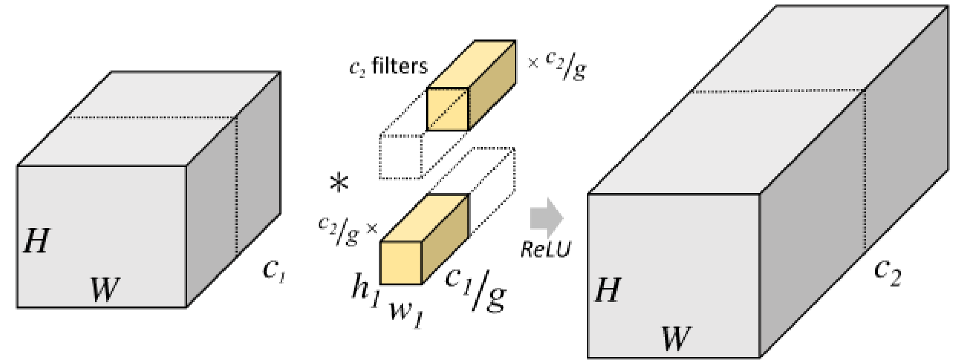
- Reduces the number of parameters and memory requirements

https://blog.yani.ai/filter-group-tutorial/

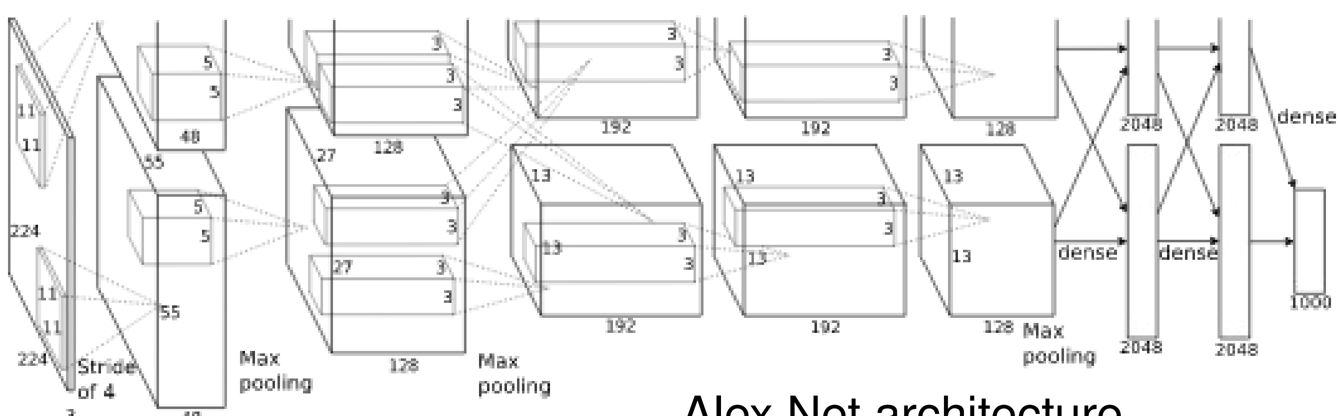
Other Convolutional layers

- Grouped convolutions

  In a grouped convolutional layer each filter operates on a group of feature maps of the layer/tensor below.



  - Reduces the number of parameters and memory requirements
  - Easy parallelization on several GPUs

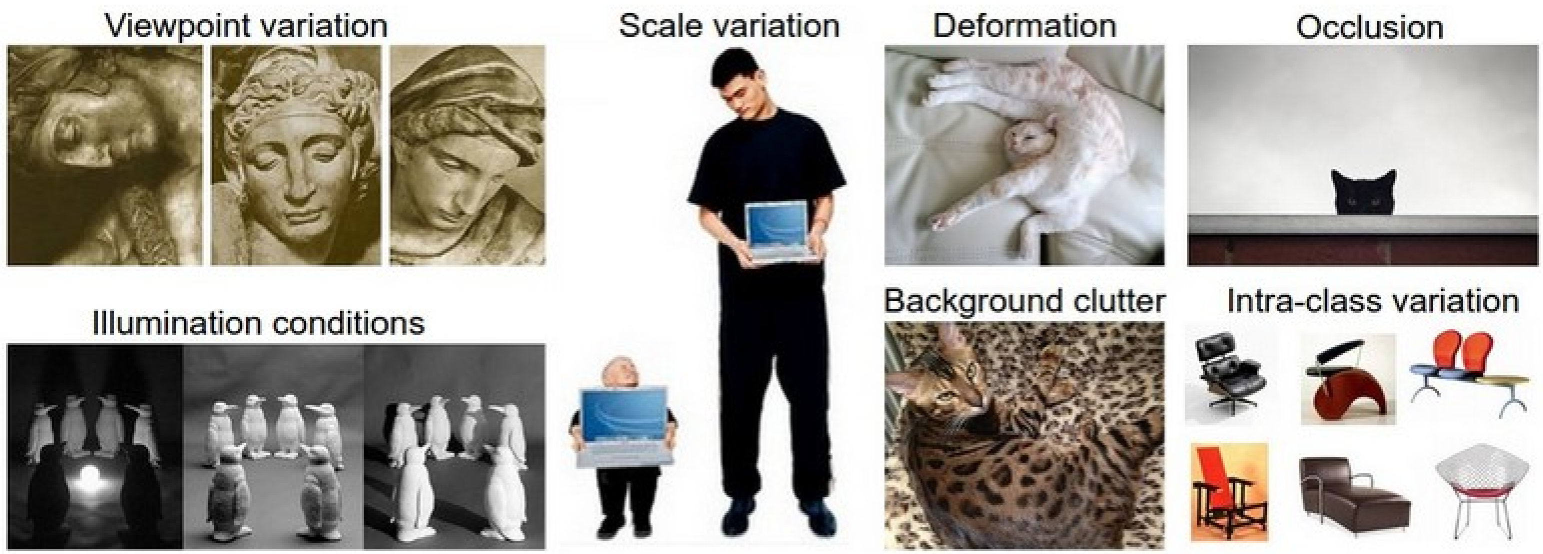


Alex Net architecture

https://blog.yani.ai/filter-group-tutorial/

Limitations

- Discussion convolutional layers
  Invariance/covariance is a key feature for classification.

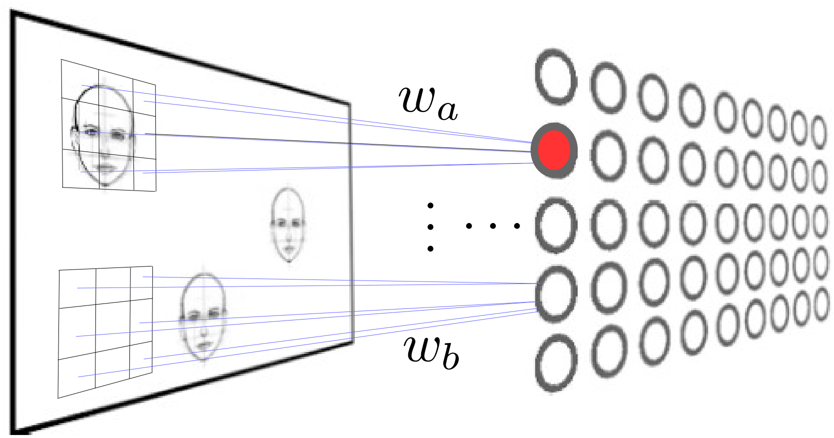


http://cs231n.stanford.edu/

Limitations

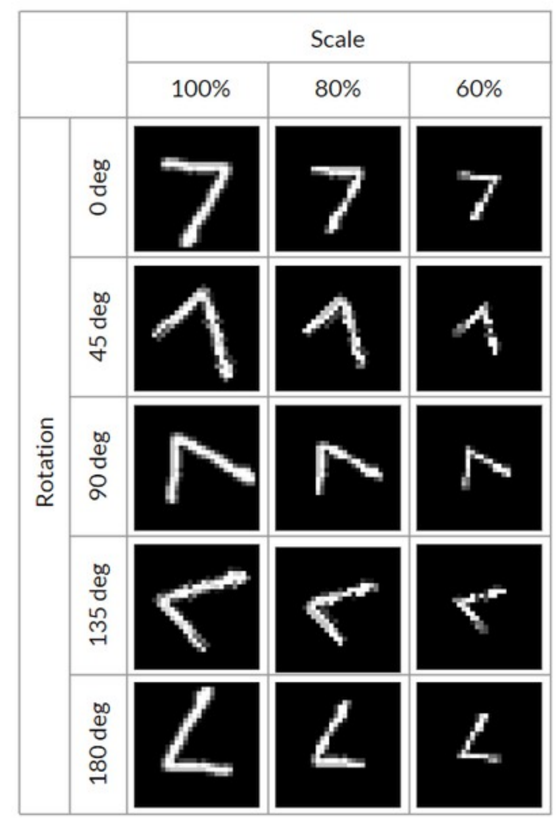
- Discussion convolutional layers

Invariance/covariance is a key feature for classification.
Convolutional layers are <u>translation covariant</u>,



but their 2D geometry is fixed, so <u>not invariant to</u>
- rotation,
- scaling,
- deformation ….

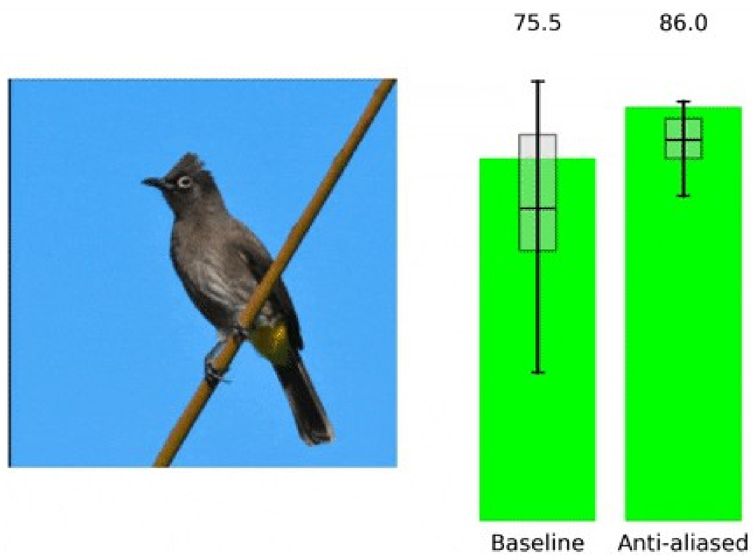
We must train the model to learn those invariances!

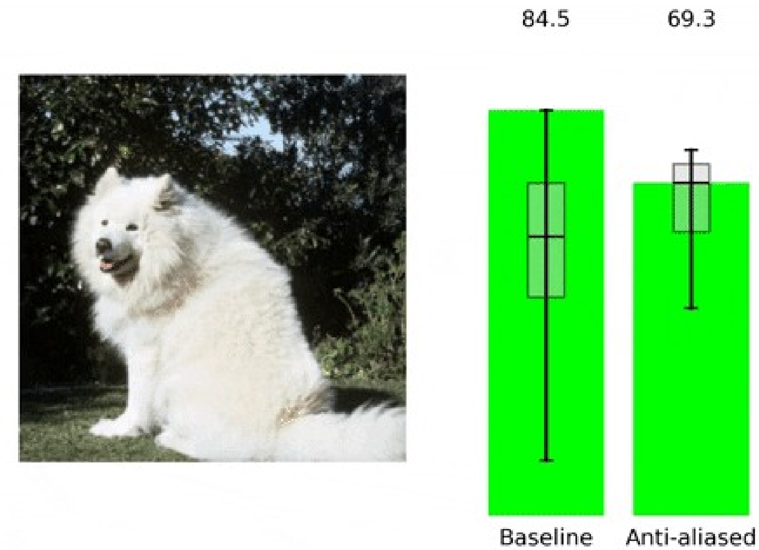


Limitations

- Discussion standard convolutional networks

  Invariance/covariance is a key feature for classification.
  Convolutional networks for classification are ~~translation invariant~~, **only partially!**





Zhang. Making Convolutional Networks Shift-Invariant Again. In ICML, 2019.

Limitations

- Discussion standard convolutional networks
  Invariance/covariance is a key feature for classification.
  Convolutional networks are ~~translation invariant~~, **only partially!**

- Why is shift invariance lost?
  Convolutions are shift-covariant
  Pooling builds up shift-invariance
  …but striding ignores Nyquist sampling theorem **and aliases!**

Zhang. Making Convolutional Networks Shift-Invariant Again. In ICML, 2019.

Key ideas

- Densely connected feed forward NNs are not adequate for processing images
  - Large # of parameters
  - # of parameters depends (grows) with image size
  - Difficult to learn spatial dependencies
- Convolutional layers introduce some inductive biases
  - 2D spatial topololgy
  - # of parameters independent of image size
  - Translation covariance
- Convolutional layers have some limitations
  - Fixed spatial structure
  - Not invariant to rotation, scaling, deformations, ...
  - Convolutional architectures not fully translation invariant