Course: Deep Learning

Unit 3: 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
    - CNN layers
    - Convolutional layer
    - Pooling layer
    - Sample CNN

Convolutional Neural Networks

- **Acknowledgement**

Most slides taken from:

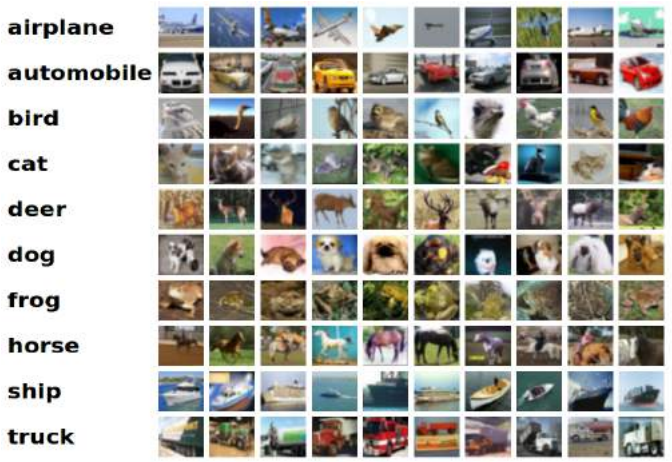
- Geoffrey Hinton
- Yann Lecun
- Hugo Larrochelle
- Andrej Karpathy
- Justin Johnson

Introduction

- Goal: Object recognition (image classification)

  Identify the foreground object in an image

  For example (Cifar 10)



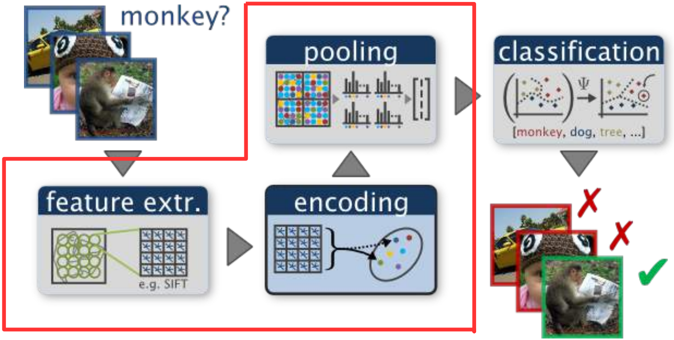
Introduction

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

Introduction

- Standard (shallow) object recognition approach

Invariance



- Handcrafted/engineered features

  Invariant to occlusions, changes illumination, position, orientation

  (are they optimal at all for the task?)
- Shallow representation: only one mid-level representation.

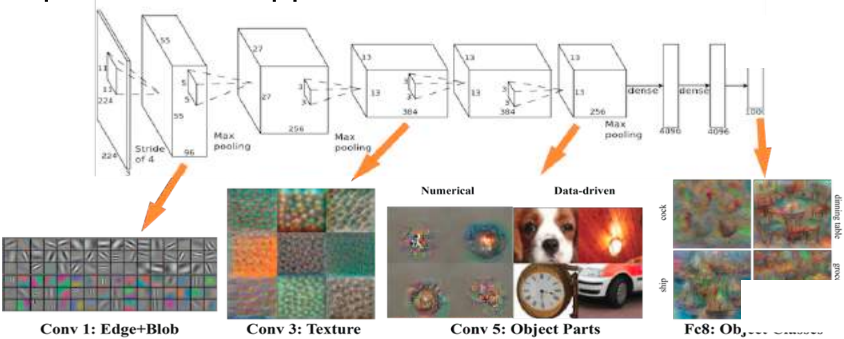
  Difficult to generalize

  Low representation capabilities (poor abstraction)
- Lots of priors in form of design solutions

  Requires few training data

Introduction

- Deep neural net approach



- Learned features

  Requires no difficult tuning/engineering tasks.

  "Optimal" for the task at hand.

- Hierarchical representation.

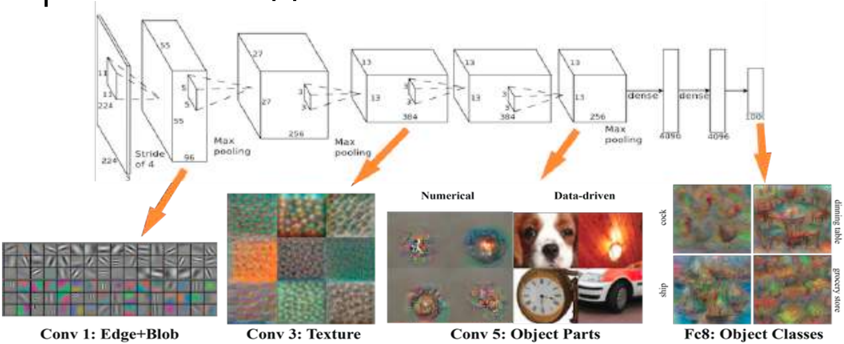
  Lets us better represent the data using higher abstractions.

- 100% data driven.

  Requires a lot of data <u>to be trained from scratch.</u>

Introduction

- Deep 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.
- Solution
  - Use prior information to reduce number of parameters (CNNs)
  - Regularize the solution

Introduction

- CNNs: a Neural Net approach to object recognition

  Neural nets specifically adapted for computer vision problems:

  - Can deal with very high-dimensional inputs

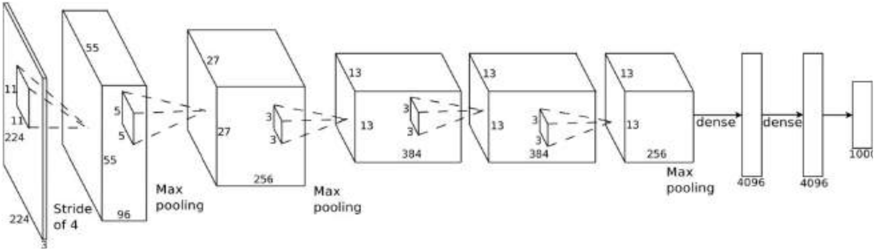
    256 x 256 x 3 images = 196 608 input data

  - Exploit the topology of pixels

    multi channeled images, video,

  - Certain degree of invariance

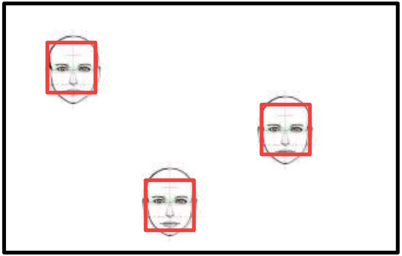
    translation, illumination changes, ...



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 2D space

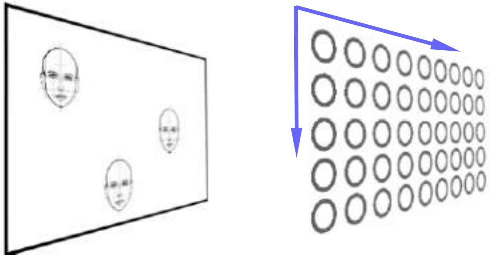


Face detection

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.



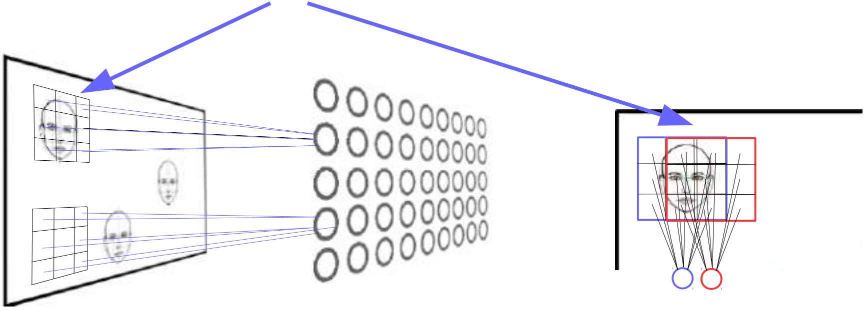
CNNs fundamental elements

- 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**

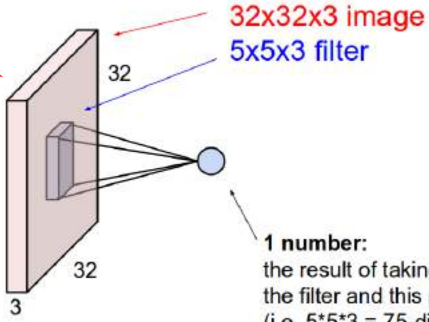


CNNs fundamental ideas

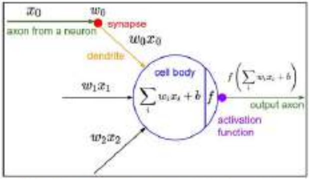
- Each unit is a standard NN cell
  - **Can deal with very high-dimensional inputs**
  - **Exploit the 2D/3D topology of pixels**
  - Certain degree of invariance

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



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)



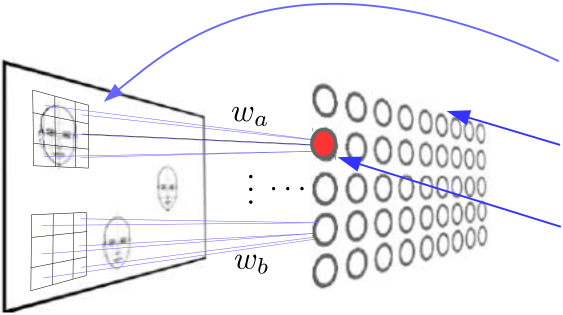
It's just a neuron with local connectivity...

http://cs231n.stanford.edu/

CNNs fundamental ideas

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

All cells in the same slice share their weights



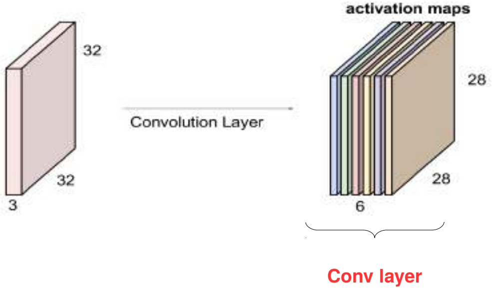
$$\boxed{w_a = w_b = \dots}$$

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

CNNs fundamental ideas

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

A layer is a set of feature/activation maps

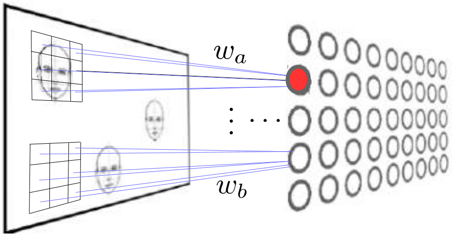


- Each cell covers an input region, the receptive field.
- The pre-activation in the feature map of a slice is a convolution
- Each slice detects a certain feature: feature / activation map
- Each cell in a feature map fires whenever the feature is detected in its receptive field.
- A **convolutional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

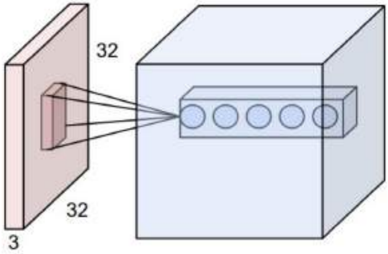
http://cs231n.stanford.edu/

CNNs fundamental ideas

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



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



http://cs231n.stanford.edu/

CNNs fundamental ideas

- Backpropagation with weight constraints

We assume we initialize all parameters in a feature map such that

$$w_a = w_b = \dots$$

We compute

$$\frac{\partial \mathcal{L}(W, \mathcal{D})}{\partial w_a}, \qquad \frac{\partial \mathcal{L}(W, \mathcal{D})}{\partial w_b}, \quad \dots$$

und update w_a and w_b

$$\{w_a = w_b = \dots\} \Leftarrow \frac{\partial \mathcal{L}(W, \mathcal{D})}{\partial w_a} + \frac{\partial \mathcal{L}(W, \mathcal{D})}{\partial w_b} + \dots$$

A relief for vanishing gradients!

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

  - Introduces invariance to local translations
  - Reduces the number of hidden units
  - Discards information about the location of features in the image

Typically:
- Non-overlapping
- Small windows

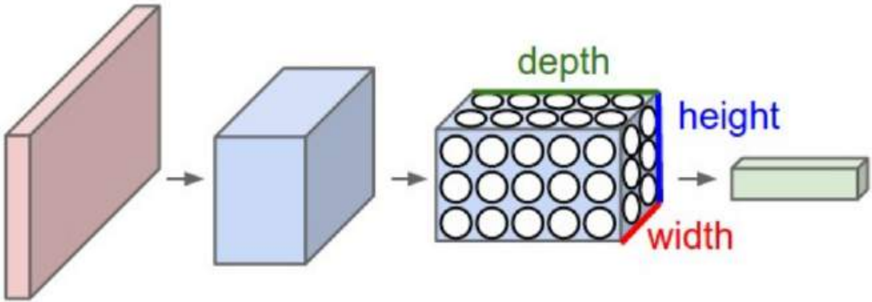
Types
- `max`
- `average`
- `weighted aver`

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.

  It is a three-dimensional neural net.



http://cs231n.stanford.edu/

CNNs fundamental ideas

- CNNs have a biological interpretation

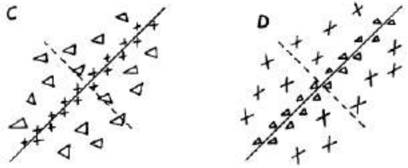
  [Hubel & Wiesel J. Physiology, 1962]

  Hierarchical model composed of

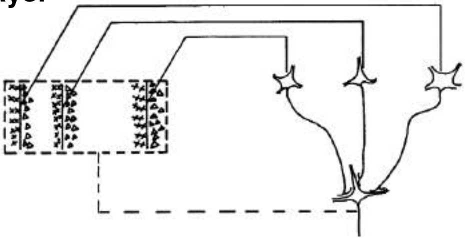
  - S (simple) cells activate when they detect basic shapes like lines
  - C (complex) cells combine the activation of the simple cells activating to the same shapes but with less sensibility to position

  S cells would behave like a convolutional layer, and

  C cells would act as a pooling layer



Arrangement of receptive fields

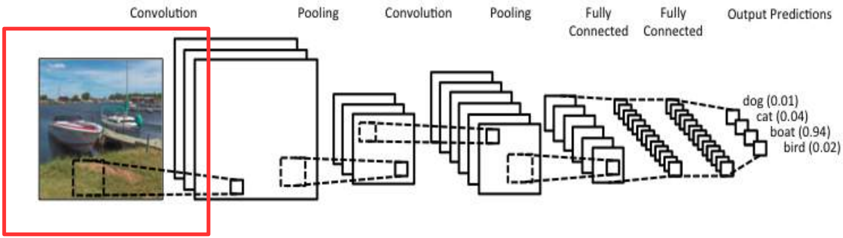


Organization of complex receptive fields

Conv Net construction

- Typical layers
  - **Input**

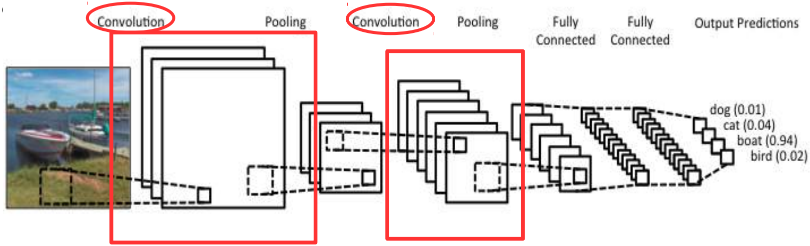
    holds the raw pixel values (RGB, Grey, ….)
  - Conv
  - ReLU
  - Pool
  - FC



Conv Net construction

- Typical layers
  - Input
  - **Conv**

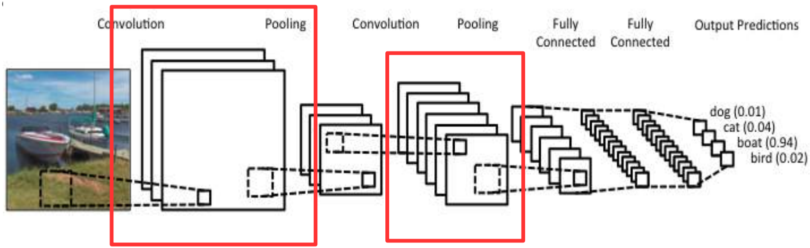
    holds all the feature activation maps computed from convolutions on the input data (an image or another layer).
  - ReLU
  - Pool
  - FC



Conv Net construction

- Typical layers
  - Input
  - Conv
  - **ReLU**

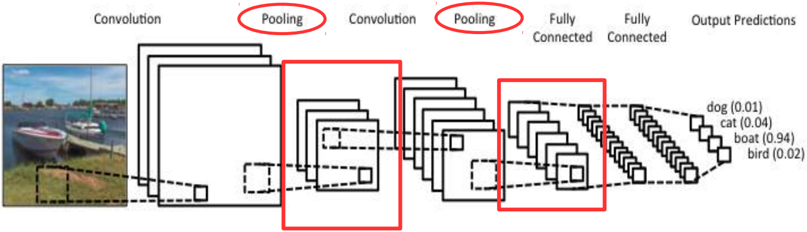
    holds the result of applying the non-linear activation function to each cell in each feature map
  - Pool
  - FC



Conv Net construction

- Typical layers
  - Input
  - Conv
  - ReLU
  - **Pooling**

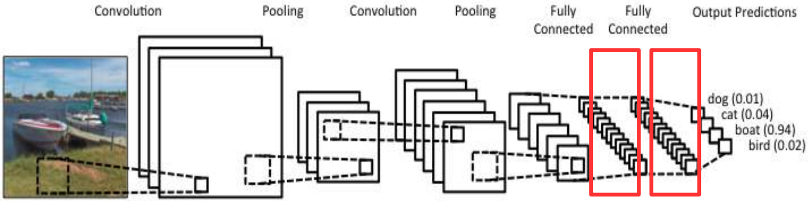
    performs a downsampling operation along the spatial dimension
  - FC
  - SoftMax



Conv Net construction

- Typical layers
  - Input
  - Conv
  - ReLU
  - Pool
  - **FC**

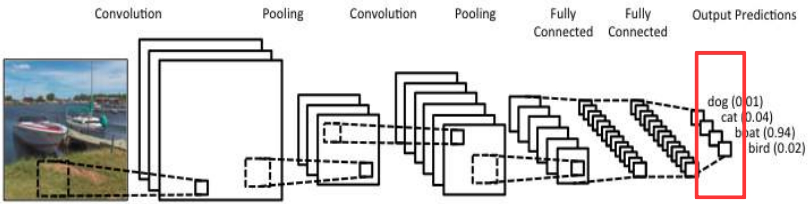
    fully-connected layer is the classifier
  - SoftMax



Conv Net construction

- Typical layers
  - Input
  - Conv
  - ReLU
  - Pool
  - FC
  - **SoftMax**

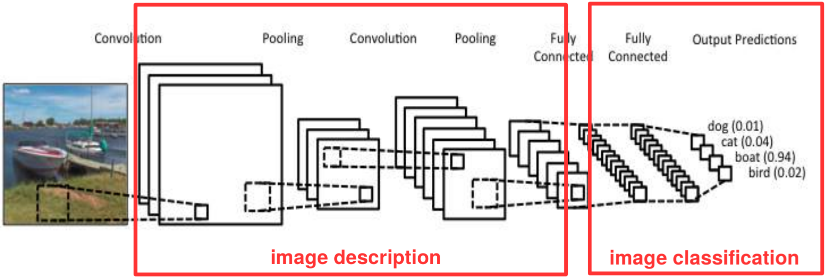
    final layer to compute the loss



Conv Net construction

- Typical layers
  - Input
  - Conv
  - ReLU
  - Pool
  - FC
  - SoftMax

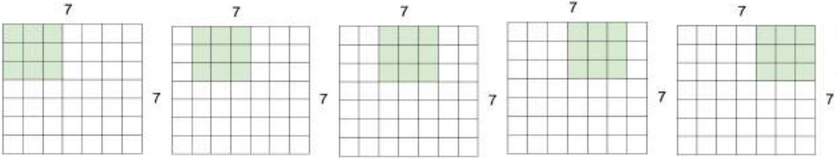
    final layer to compute the loss



Conv Net construction

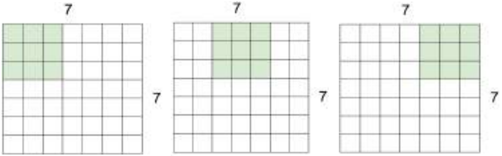
- Convolutional layer

  Spatial dimensions.



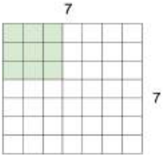
7x7 input (spatially)
assume 3x3 filter

=> 5x5 output



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

=> 3x3 output!



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

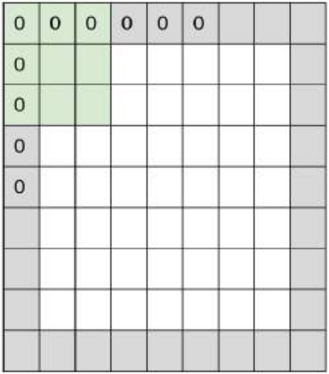
doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Can we convolve an image and get an output matrix with the same size?

Conv Net construction

- Convolutional layer

  Zero padding the image border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

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.

Must be integers, to build a proper conv layer!!

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', c
```

Arguments

- **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.

F

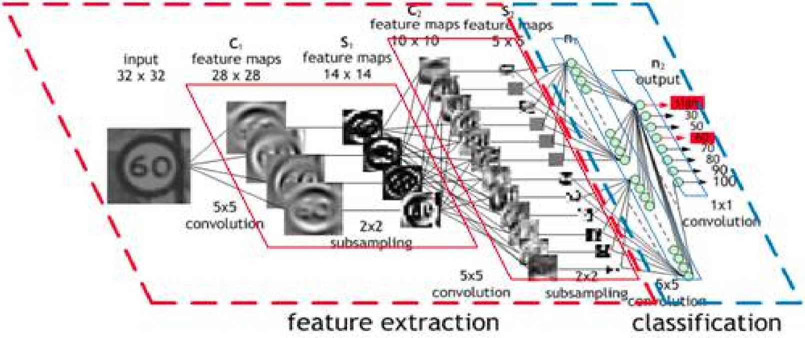
- **strides:** Integer, tuple of 2 integers, or None. Strides values. If None, it will default to `pool_size`.

S

- **padding:** One of `"valid"` or `"same"` (case-insensitive).

Conv Net construction

- Sample Conv Net



Homework for next week

Sample baseline code.

```
# Convolutional Neural Network (CNN)
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
import keras.backend as K

model = Sequential()
model.add(Conv2D(filters=48, kernel_size=(3, 3), padding='same', input_shape=(32, 32, 3)))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=96, kernel_size=(3, 3), padding='same'))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Conv2D(filters=192, kernel_size=(3, 3), padding='same'))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=32, kernel_size=(1, 1), padding='same'))
model.add(Activation('relu'))

model.add(Conv2D(filters=10, kernel_size=(4, 4), padding='valid'))
model.add(Flatten())
model.add(Activation('softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Homework for next week

Sample baseline code.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_27 (Conv2D)           (None, 32, 32, 48)        1344
_________________________________________________________________
activation_33 (Activation)   (None, 32, 32, 48)        0
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 16, 16, 48)        0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 16, 16, 96)        41568
_________________________________________________________________
activation_34 (Activation)   (None, 16, 16, 96)        0
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 8, 8, 96)          0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 8, 8, 192)         166080
_________________________________________________________________
activation_35 (Activation)   (None, 8, 8, 192)         0
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 4, 4, 192)         0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 4, 4, 32)          6176
_________________________________________________________________
activation_36 (Activation)   (None, 4, 4, 32)          0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 1, 1, 10)          5130
_________________________________________________________________
flatten_7 (Flatten)          (None, 10)                0
_________________________________________________________________
activation_37 (Activation)   (None, 10)                0
=================================================================
Total params: 220,298
Trainable params: 220,298
Non-trainable params: 0
```

Homework for next week

Goal:

- Experiment with cNNs. A special type of NN conceived for analysing images..

Steps:

- Use cNNs to beat previous results on the Cifar10.
- Do not use any regularization yet!

Next class: **regularization**!