Neural Network Toolbox & Matconvnet

期末報告

Neural Network toolbox

Try to use your mathwork account to install MATLAB_R2017b

Use Neural Network Toolbox of MATLAB_R2017b installed on PC of computer room, AM, NDHU
Give a talk in 15-20 minutes

Install toolbox MatConvNet
Give a talk in 15-20 minutes

MatconvNet

- Download toolbox from **http://www.vlfeat.org/matconvnet/**

Installing and compiling the library

In order to install the library, follows these steps:

1. Download and unpack the library source code into a directory of your choice. Call the path to this directory <MatConvNet>.
2. Compile the library.
3. Start MATLAB and type:

```
> run <MatConvNet>/matlab/vl_setupnn
```

```
mex -setup
```
2

```
mex -setup C++
```
3

```
> cd <MatConvNet>
> addpath matlab
> vl_compilenn
```
4

MatconvNet

MatConvNet: CNNs for MATLAB

Obtaining MatConvNet

Documentation

Extensions

Getting started

Use cases

Other information

- Manual (PDF)
- MATLAB functions
- FAQ
- Discussion group

Extensions

- Third party contributions and extensions, also accessible using `vl_contrib`, third-party co several modern object detectors.

Getting started

- Quick start guide
- Installation instructions
- Using pre-trained models: VGG-VD, GoogLeNet, FCN, ...
- Training your own models
- CNN wrappers: linear chains or DAGs
- Working with GPU accelerated code
- Tutorial (classification), tutorial (regression), slides

download
Tutorial

Getting started

Read and understand the requirements and installation instructions. The s practical are:

- Code and data: practical–cnn–2017a.tar.gz
- Code only: practical–cnn–2017a–code–only.tar.gz
- Data only: practical–cnn–2017a–data–only.tar.gz
- Git repository (for lab setters and developers)

download
ex1
ex2
ex3
ex4
ex5

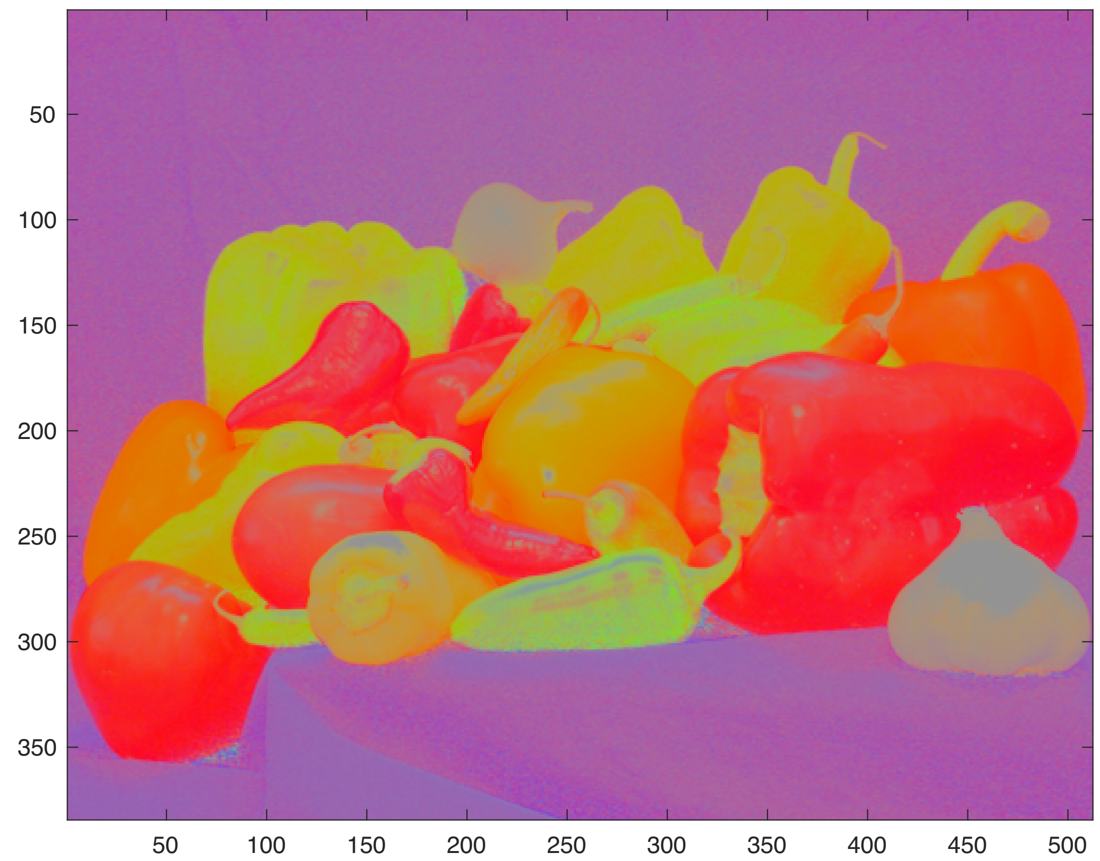
After the installation is complete, open and edit the script `exercise1.m` in the MATLAB editor. The script contains commented code and a description for all steps of this exercise, for Part I of this document. You can cut and paste this code into the MATLAB window to run it, and will need to modify it as you go through the session. Other files `exercise2.m`, `exercise3.m`, and `exercise4.m` are given for Part II, III, and IV.

- Code and data: practical-cnn-2017a.tar.gz

- from http://www.robots.ox.ac.uk/~vgg/practicals/cnn/index.html

Part 1: CNN building blocks

- ex1



Part 2: back-propagation and derivatives

Part 3: learning a tiny CNN

Part 4: learning a character CNN

Part 5: using pretrained models

MatconvNet/examples/cnn_vgg_faces

cnn_vgg_faces

Aamir_Khan (3), score 0.519

Problem
Architecture
Learning
Data
Execution
Results
your comments
.
.
.
Reference

MatconvNet/examples/mnist
cnn_mnist_experiments

Problem
Architecture
Learning
Data
Execution
Results
your comments
.
.
.
Reference

MatconvNet/examples/fast_rcnn

fast_rcnn_demo

Detections for class 'car'

Problem
Architecture
Learning
Data
Execution
Results
your comments
.
.
.
Reference

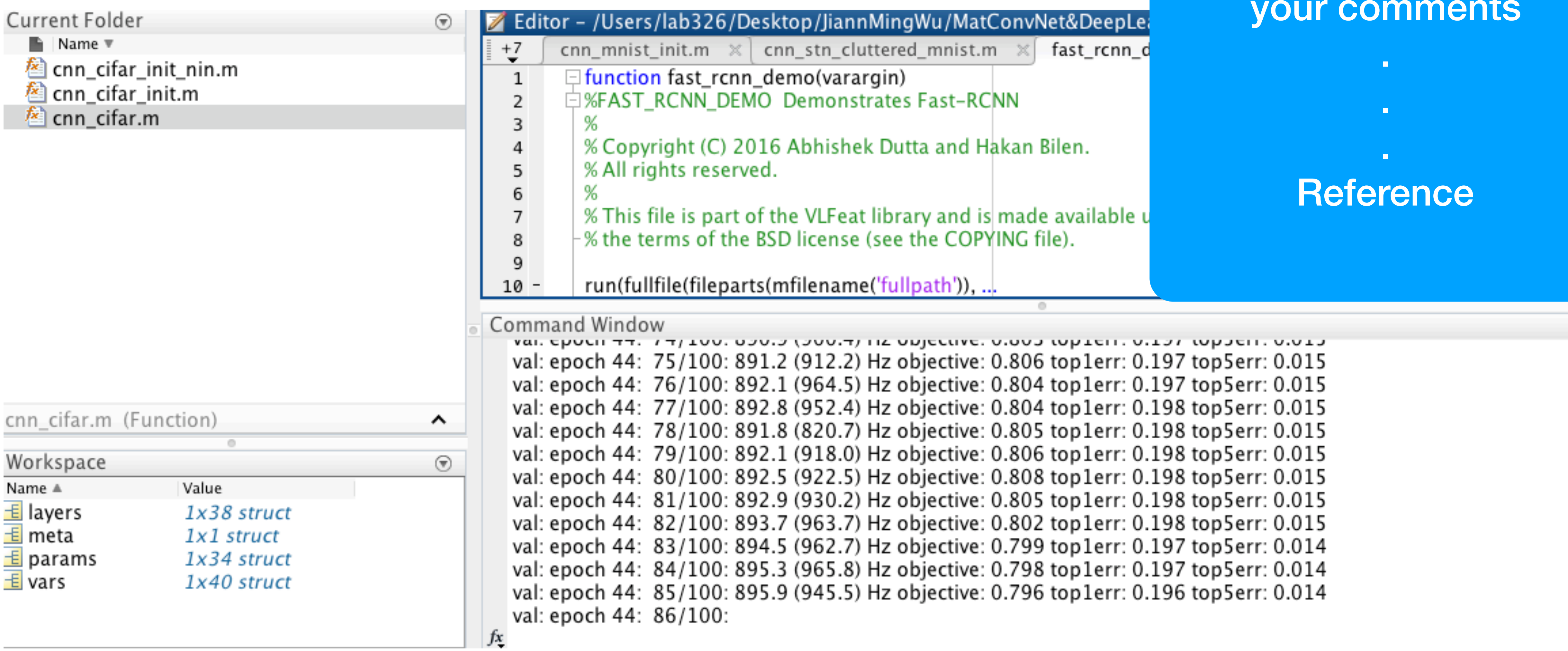
Please download again fast-rcnn-vgg16-pascal07-dagnn

http://www.vlfeat.org/matconvnet/pretrained/

MatconvNet/examples/cifar

cnn_cifar

Problem
Architecture
Learning
Data
Execution
Results
your comments
.
.
.
Reference

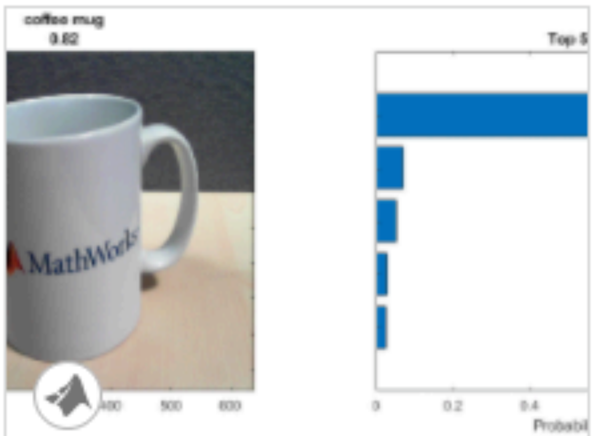


https://www.mathworks.com/help/nnet/examples.html#bvljehw

神經網絡工具箱示例

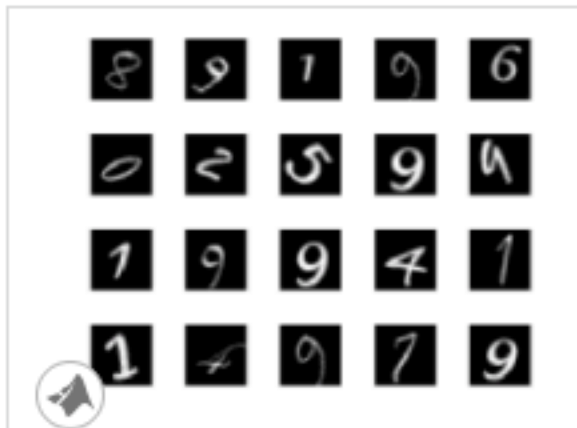
R2

深度學習



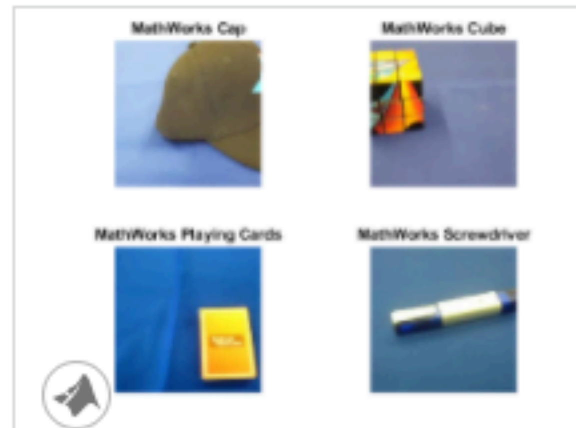
使用深度學習分類攝像頭圖像

使用預訓練的深度卷積神經網絡AlexNet實時分類來自攝像頭的圖像。



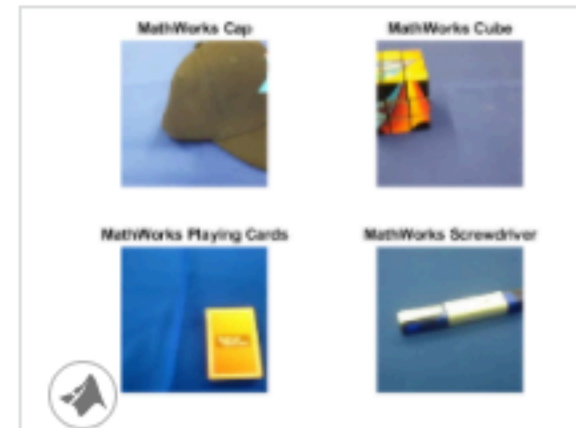
創建簡單的深度學習網絡進行分類

創建和訓練一個簡單的捲積神經網絡深度學習分類。卷積神經網絡是深度學習的基本工具，特別適合圖像識別。



轉移學習使用AlexNet

對預訓練的AlexNet卷積神經網絡進行微調，以對新的圖像集進行分類。



使用AlexNet進行特徵提取

從預訓練的捲積神經網絡中提取學習的特徵，並使用這些特徵來訓練圖像分類器。特徵提取是使用預訓練深層網絡的表示能力的最簡單和最快的方

examples

Classify Webcam Images Using Deep Learning
Create Simple Deep Learning Network for Classification
Transfer Learning Using AlexNet
Feature Extraction Using AlexNet
Deep Dream Images Using AlexNet

AIMS Big Data

Lecture 3: Deep Learning

Andrea Vedaldi

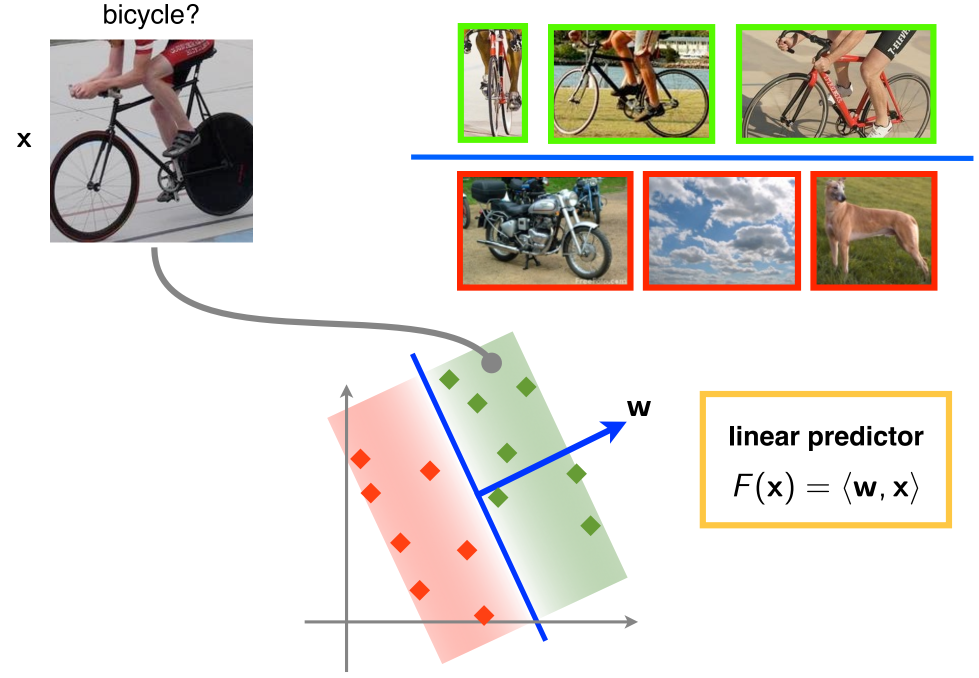
For slides and up-to-date information:

`http://www.robots.ox.ac.uk/~vedaldi/teach.html`

UNIVERSITY OF OXFORD

Linear predictor

2



Data representations

3

Using linear predictors on non-vectorial data

Meaningful representation

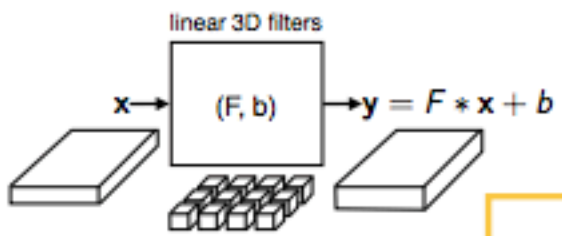
4

representation

http://www.robots.ox.ac.uk/~vedaldi/assets/teach/2015/vedaldi15aims-bigdata-lecture-4-deep-learning-handout.pdf

Linear convolution

As a filter bank



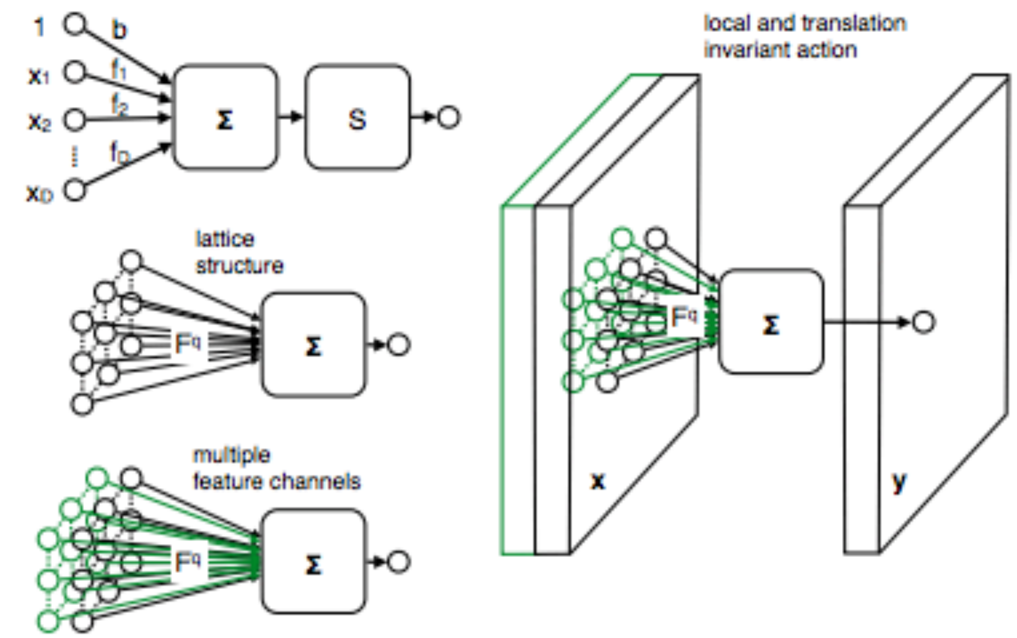
$$y_{ijq} = b_q + \sum_{u=0}^{H-1} \sum_{v=0}^{W-1} \sum_{k=1}^{K} x_{u+i,v+j,k} f_{u,v,k,q}$$

Linear, translation invariant, local:

- Input **x** = H × W × K array
- Filter bank F = H' × W' × K × Q array
- Output **y** = (H - H' + 1) × (W - W' + 1) × Q array

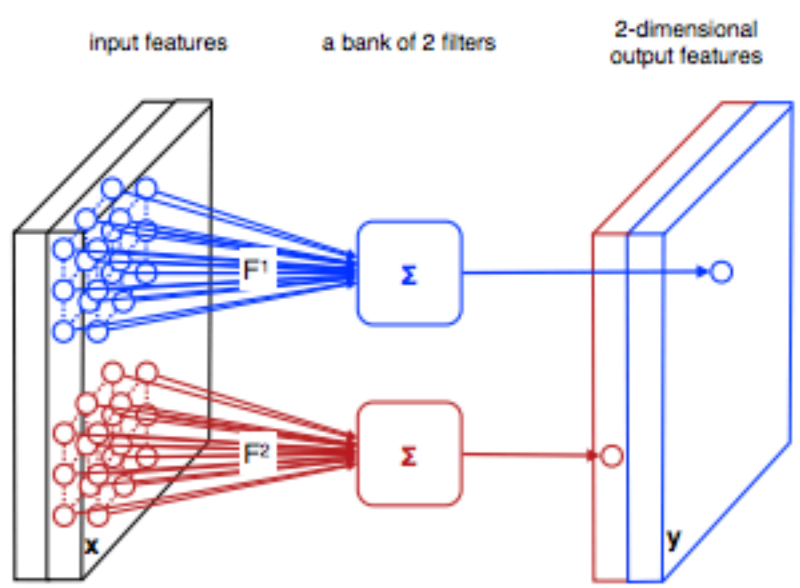
Linear convolution

As a neural network



Linear convolution

As a neural network



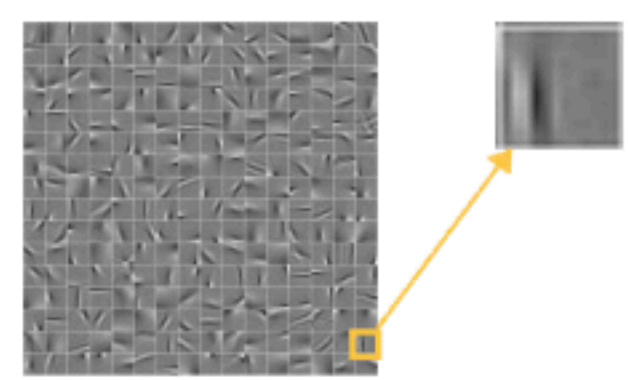
Linear convolution

Filter bank example

A bank of 256 filters (learned from data)

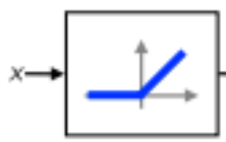
Each filter is 1D (it applies to a grayscale image)

Each filter is 16 × 16 pixels



Gating functions

Component-wise non-linearity



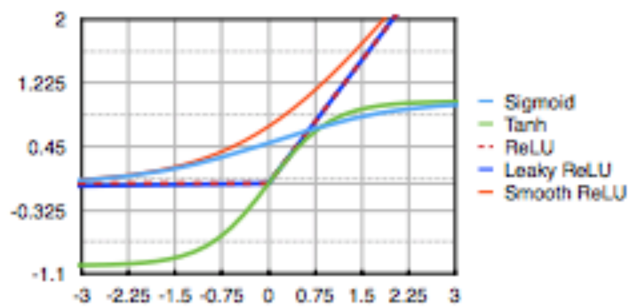
$y = \frac{1}{1+e^{-x}}$ sigmoid

$y = \tanh(x)$ hyperb. tan

$y = \max\{0, x\}$ ReLU

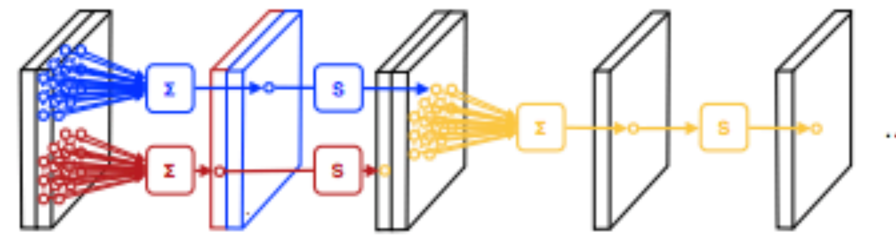
$y = \log(1 + e^{x})$ Soft ReLU

$y = \epsilon x + (1-\epsilon)\max\{0, x\}$ Leaky ReLU



Multiple layers

Convolution, gating, convolution, …

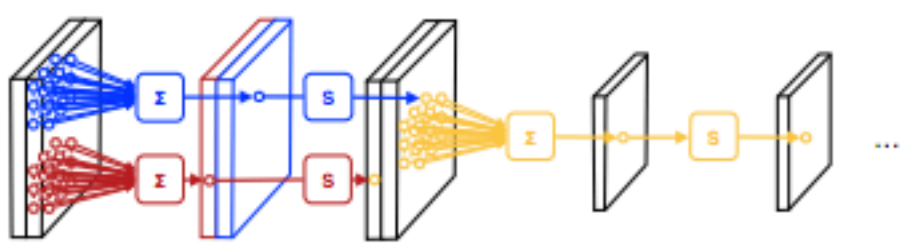


Filters are followed by non-linear operators (e.g. gating, but see later)

Multiple such layers are chained together

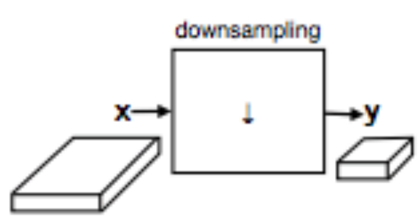
Multiple layers

Downsampling



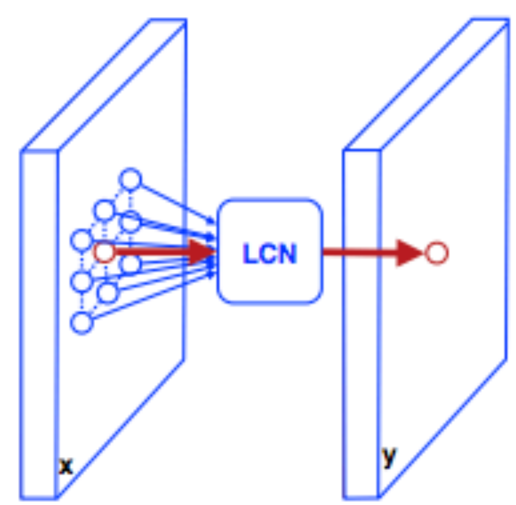
Filters are often followed (or incorporate) downsampling

This is often compensated by an increase in the number of feature channels (not shown)



Local contrast normalisation

Normalise image/feature patches



$$y_{ijq} = \frac{x_{ijq} - \mu_{ijq}}{\sigma_{ijq}}$$

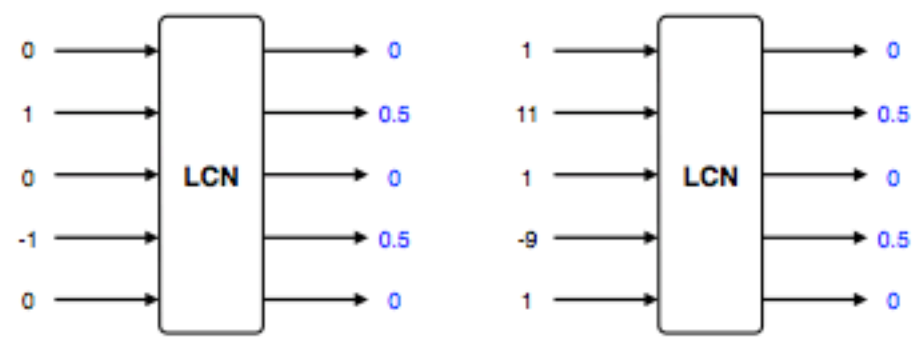
$$\mu_{ijq} = \frac{1}{|\mathcal{N}(i,j)|} \sum_{(u,v)\in\mathcal{N}(i,j)} y_{uvq}$$

$$\sigma^2_{ijq} = \frac{1}{|\mathcal{N}(i,j)|} \sum_{(u,v)\in\mathcal{N}(i,j)} (y_{uvq} - \mu_{ijq})^2$$

Local contrast normalisation

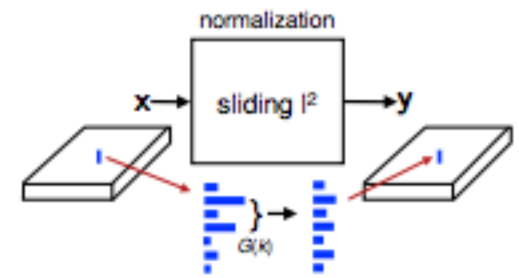
Example

It has a local equalising effect:



Local feature normalisation

Across feature channels rather than spatially



$$y_{ijk} = x_{ijk}\left(\kappa + \alpha \sum_{q\in G(k)} x_{ijq}^2\right)^{-\beta}$$

Operates at each spatial location independently

Normalise groups G(k) of feature channels

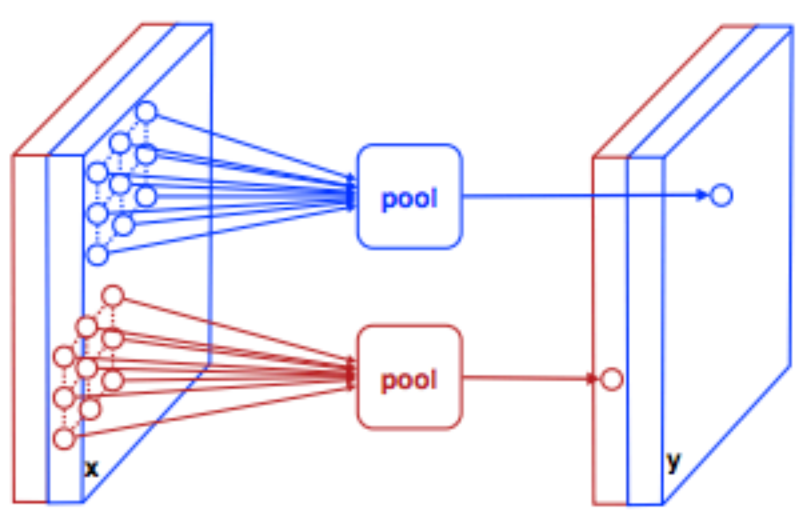
Groups are usually defined in a **sliding window manner**

Spatial pooling

Reduce dependency on precise location

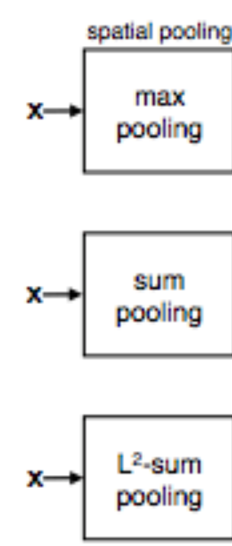
Pooling compute the average / max of the features in a neighbourhood.

It is applied channel-by-channel.



Spatial pooling

Variants



$$y_{ijk} = \max\{x_{uvk} : (u,v) \in \mathcal{N}(i,j)\}$$

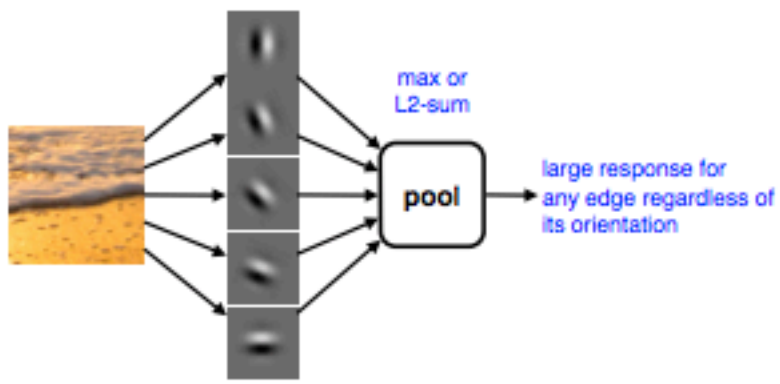
$$y_{ijk} = \sum_{(u,v)\in\mathcal{N}(i,j)} x_{uvk}$$

$$y_{ijk} = \sqrt{\sum_{(u,v)\in\mathcal{N}(i,j)} x_{uvk}^2}$$

Feature pooling

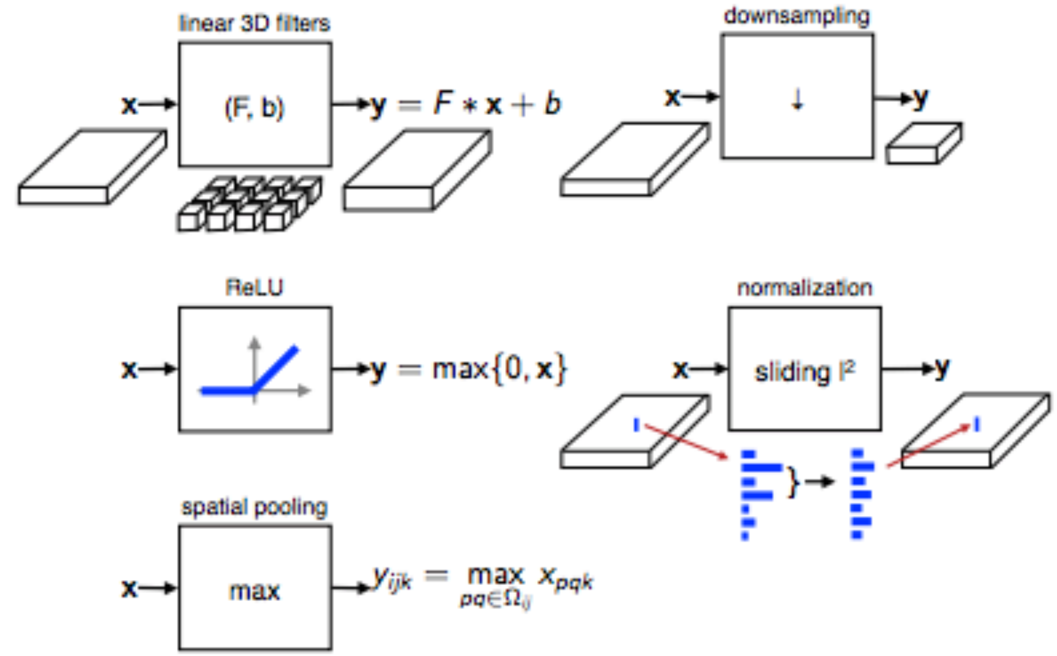
Across feature channels, not in space

Pooling across feature channels (filter outputs) can achieve invariance



L2 pooling, in particular, is invariant to the **sign of the edge filter too**

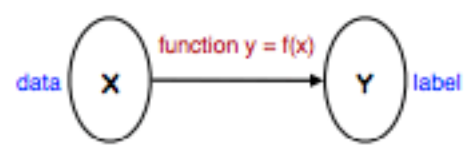
CNN components summary



Possible learning goals

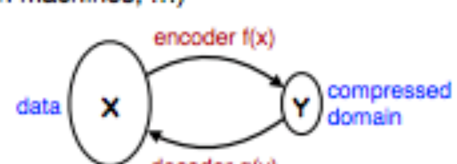
Discriminative training (neural networks)

- Classification / regression
- Solve a task (e.g. object recognition)

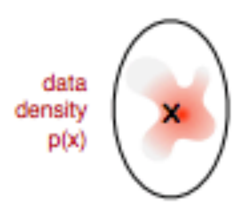


Generative training (autoencoders, Boltzman machines, …)

- Reconstruct the image from a compressed representation (autoencoder)

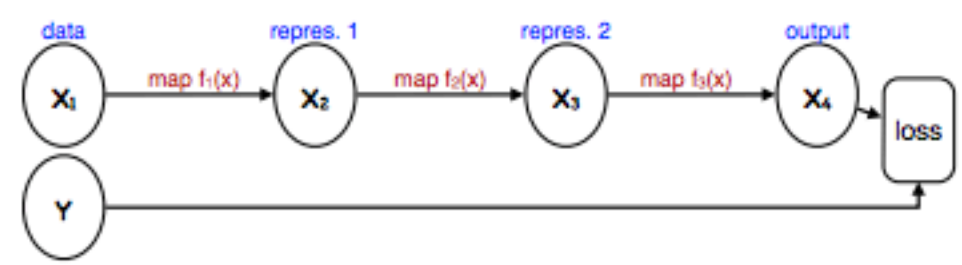


- Model the distribution of the data (Boltzman machines, …)



Stage-wise generative training

A key difficult in learning deep models is the complex interaction between all layers. Direct optimisation of a regression loss is difficult:



Generative training allows to train layer by layer using as a target the reconstruction of the layer before: