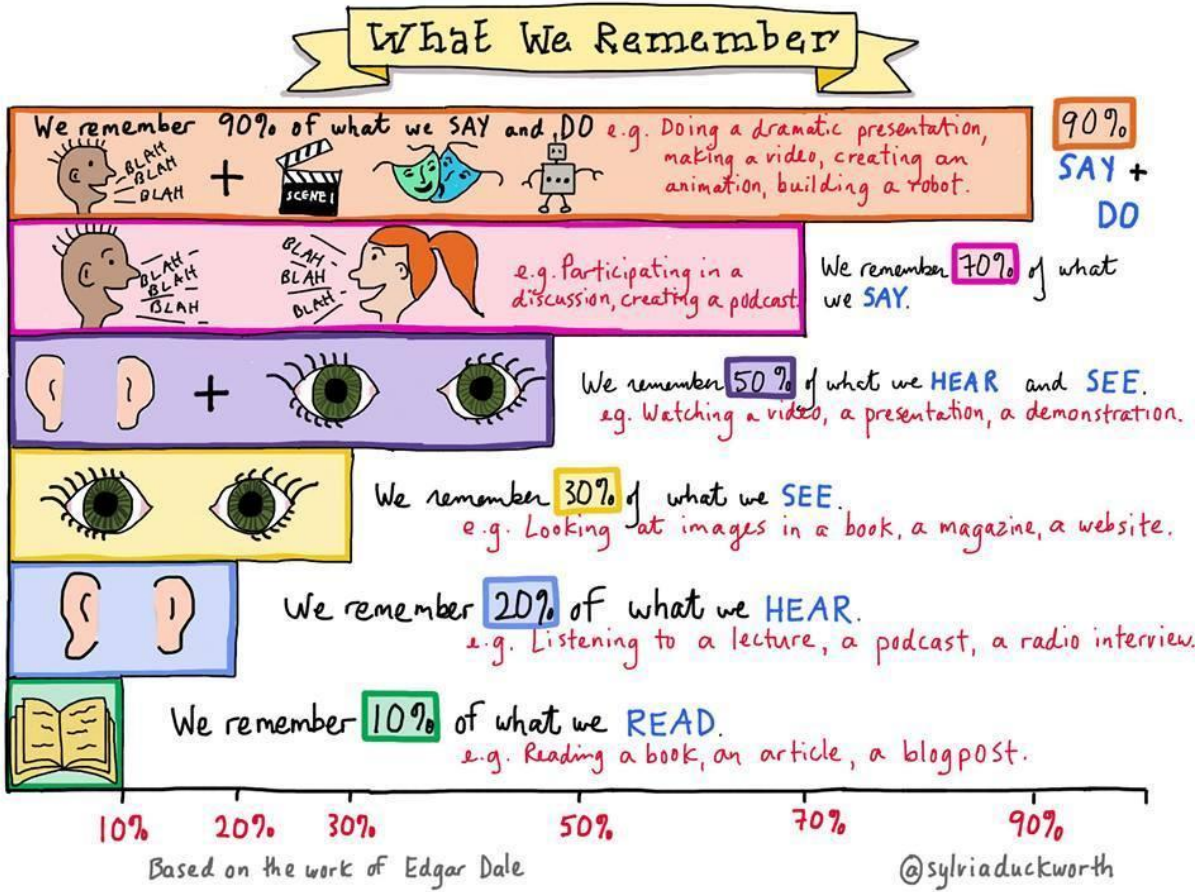
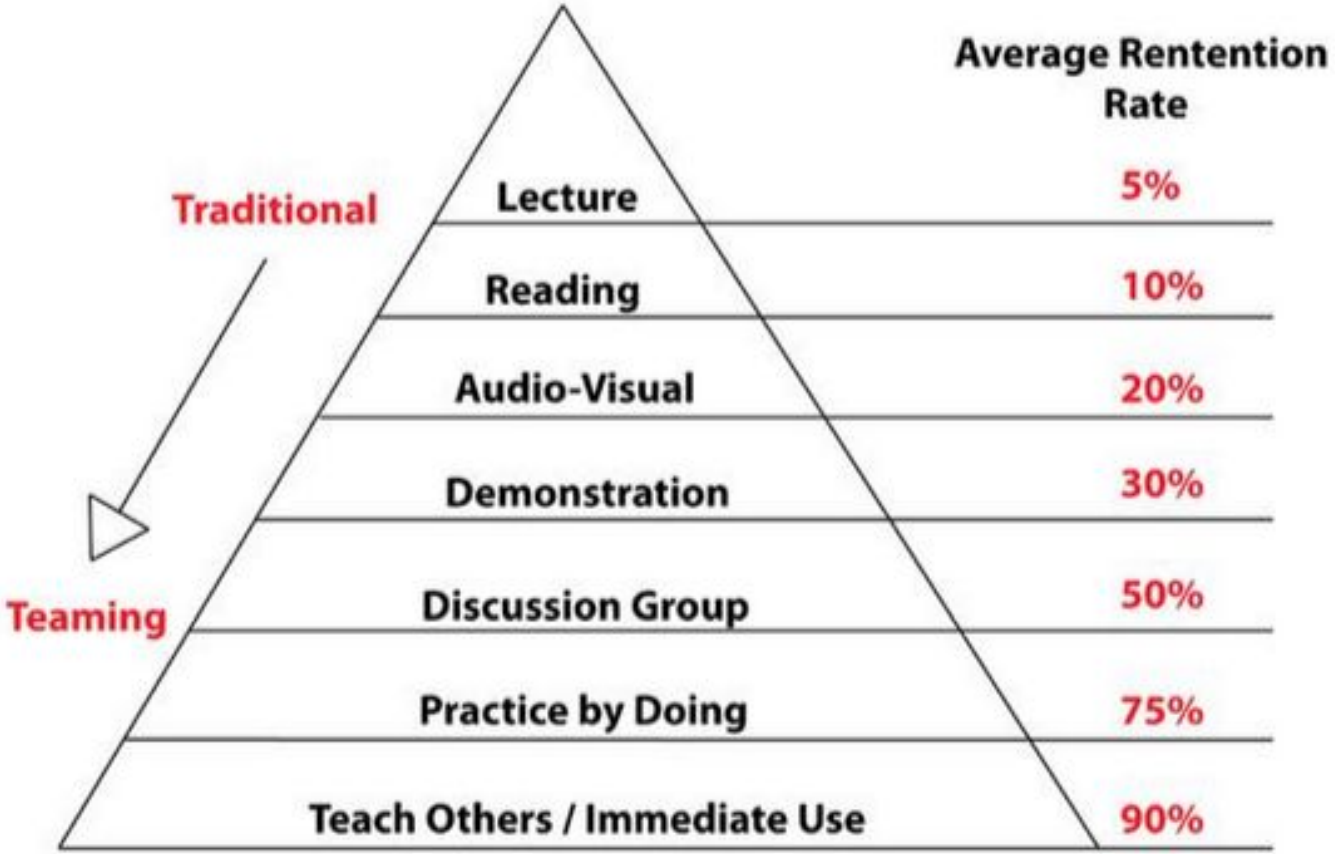
Day 1

Project kick off

Motivation


What We Remember
We remember 90% of what we SAY and DO e.g. Doing a dramatic presentation, making a video, creating an animation, building a robot.
BLAH BLAH BLAH
SCENE 1
90% SAY + DO
e.g. Participating in a discussion, creating a podcast.
We remember 70% of what we SAY.
We remember 50% of what we HEAR and SEE.
e.g. Watching a video, a presentation, a demonstration.
We remember 30% of what we SEE.
e.g. Looking at images in a book, a magazine, a website.
We remember 20% of what we HEAR.
e.g. Listening to a lecture, a podcast, a radio interview.
We remember 10% of what we READ.
e.g. Reading a book, an article, a blogpost.
10% 20% 30% 50% 70% 90%
Based on the work of Edgar Dale
@sylviaduckworth

Motivation


Average Rentention Rate
Traditional
Lecture
5%
Reading
10%
Audio-Visual
20%
Demonstration
30%
Teaming
Discussion Group
50%
Practice by Doing
75%
Teach Others / Immediate Use
90%

Development

Development

Improve your GitHub account with GitHub for Education.

Development

Show your experiments on Jupyter notebooks.

Development

Examples:

[GDSA 2015] S5: Rankings

In this week's assignment we are going to replace the random image descriptors by descriptors that are meaningful and will help us generate *smart* rankings for each one of the validation images. This notebook shows all the required steps to achieve this point, including the training of the codebook, the construction of BoW descriptors and the generation of the rankings based on feature similarity.

We start with some basic imports, including the chosen parameters:

In [1]:
```
import numpy as np
import time
# Add the root path (the path above this one) to the pythonpath.
sys.path.insert(0,'../')
from src.params import get_params

params = get_params()
```

params includes the definition of the directories where all files will be saved, as well as the parameters related to the feature extraction and ranking steps. For example:

In [2]:
```
print "Number of clusters:", params['descriptor_size']
print "Descriptor type:",params['descriptor_type']
print "Keypoint detector:", params['keypoint_type']
print "Resize dimension:", params['max_size']
print "Distance metric:", params['distance_type']
```

```
Number of clusters: 512
Descriptor type: SIFT
Keypoint detector: SIFT
Resize dimension: 300
Distance metric: euclidean
```

params['max_size'] denotes the width to use to resize the images in the feature extraction step. The height is calculated in proportion to preserve the aspect ratio.

Feature extraction with Bag of Words

The first thing we need to do is to generate a single descriptor for every image in the training and validation sets. As we explained in earlier sessions, we will use the Bag of Words aggregation technique, which consists in training a codebook of visual words and building an image descriptor encoding the number of times each word in the codebook appears in the image.

Step 1: Local feature extraction with the training set

To build the codebook we only use the images in the training set, thus we need to compute local descriptors for all training images and stack them together in a single numpy array. The function *stack_features* does exactly this procedure.

Classification: Instant Recognition with Caffe

In this example we'll classify an image with the bundled CaffeNet model (which is based on the network architecture of Krizhevsky et al. for ImageNet).

We'll compare CPU and GPU modes and then dig into the model to inspect features and the output.

1. Setup

- First, set up Python, `numpy`, and `matplotlib`.

In [1]:
```
# set up Python environment: numpy for numerical routines, and matplotlib for plotting
import numpy as np
import matplotlib.pyplot as plt
# display plots in this notebook
%matplotlib inline

# set display defaults
plt.rcParams['figure.figsize'] = (10, 10)        # large images
plt.rcParams['image.interpolation'] = 'nearest'  # don't interpolate: show square pixels
plt.rcParams['image.cmap'] = 'gray'  # use grayscale output rather than a (potentially misleading) color h
eatmap
```

- Load `caffe`.

In [2]:
```
# The caffe module needs to be on the Python path;
#  we'll add it here explicitly.
import sys
caffe_root = '../'  # this file should be run from {caffe_root}/examples (otherwise change this line)
sys.path.insert(0, caffe_root + 'python')

import caffe
# If you get "No module named _caffe", either you have not built pycaffe or you have the wrong path.
```

- If needed, download the reference model ("CaffeNet", a variant of AlexNet).

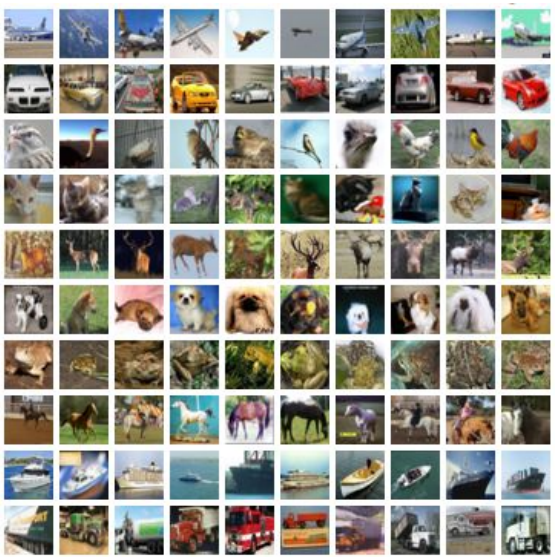
In [3]:
```
import os
if os.path.isfile(caffe_root + 'models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel'):
    print 'CaffeNet found.'
else:
    print 'Downloading pre-trained CaffeNet model...'
    !../scripts/download_model_binary.py ../models/bvlc_reference_caffenet
```

Development

At the end of the project, publish your results on a GitHub page.

Datasets

MNIST



CIFAR-10

Terrassa Buildings 900

Tasks

Block	Title	Goal
1	Architecture	Experiment with the layers and their parameters.
2	Training	Data augmentation, Batch size, Overfitting & Regularization
3	Visualization	Visualization of filters, Visualization over images
4	Transfer learning	Fine-tuning, domain adaptation
5	Free choice	

Task 1: Architecture

- Build your own network to solve a classification task.
  - Choose which layers to use and how to sort them.
  - Study the memory requirements and computational load for each type of layer.
- Recommended:
  - Experiment with small-ish architectures
  - Avoid using many conv layers when training on CPU
  - Start with MNIST

Task 2: Training

- Study the impact in performance of:
  - Data augmentation.
  - Sizes of the training batches.
  - Batch normalization
- Draw your training and validation curves.
- Overfitting
  - Force an overfitting problem.
  - Investigate if regularization (eg. drop out) reduces/solves it.

Task 3: Visualization

- Visualize filter responses.
  - From your own network.
  - From a pre-trained network.
- t-SNE
- Off-the-shelf AlexNet:
  - Visualize local classification over a small set of images.

Task 4: Transfer learning

- Train a network over CIFAR-10 and fine-tune over Terrassa Buildings 900.
- Off-the-shelf convnet:
  - Freeze weight in all layers but the last one, and replace it with a softmax to solve Terrassa Buildings 900.

Task 5: Open project

Bring your ideas to the project sessions for discussion.
Some ideas:

- Generative Adversarial Networks with MNIST.
- Neural Style, Deep Dream.

Missing GitHub users

dlcv05 - Michele de Compri

dlcv02 - Àlex Nowak

- Add your username to the spreadsheet!
- Repo admins: add all your teammates.

Access to the server

Check Atenea for the instructions.

Installation Guidelines

- Project page

Server Access & Setup

The instructions for the access to the server will be provided to students by e-mail.

Once you are logged in the server, you will need to setup your working environment. The simplest way will be to use a virtual environment to install your dependencies. As an example, if you are working with keras:

```
dlcv@imatge-dlcv:~$ virtualenv keras-env
dlcv@imatge-dlcv:~$ source keras-env/bin/activate
(keras-env) dlcv@imatge-dlcv:~$ pip install theano
(keras-env) dlcv@imatge-dlcv:~$ pip install keras
```

Installation in personal laptops

Here is a list of resources to install the project dependencies in your laptops:

- Docker for Deep Learning. Contains most deep learning libraries. It works for Linux, MacOSX and Windows. However, GPU support is only available for the first two.
- If you are using Windows 10 & Keras & want to have GPU support, here is a detailed installation guide.
- TensorFlow, CUDA, OpenCV Installation Guide by Teaching Assistant Andrea Ferri. Please use the issues section in his repository for any questions you may have about it.

Project sessions

- Part I: Open issues of general interest.
- Part II: Individual discussions with the teams.

Oral session

- 12 minutes presentation + 5 minutes questions
- Each student must present one block. Instructors will decide which one each.

Registration in Piazza

- Register to Piazza you have not done it yet.
- Check that you can access today's test.