Soutenance

Stage de fin d'étude

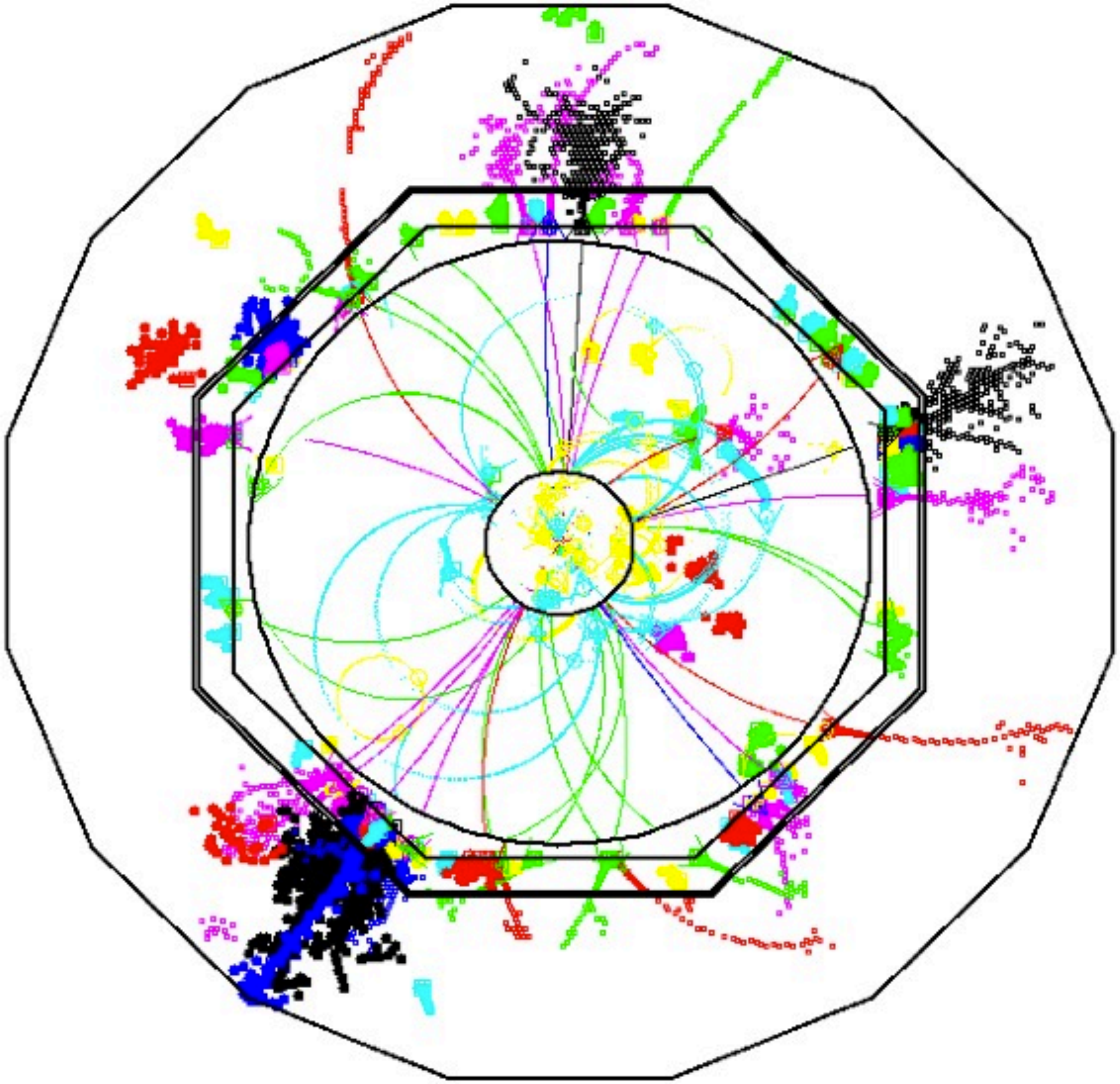
Deep Learning for Imaging Calorimeters

Superviseur : Dr. Balázs KÉGL
Février-Juillet 2013

Franck DUBARD
ING 3 - SCIA 2013

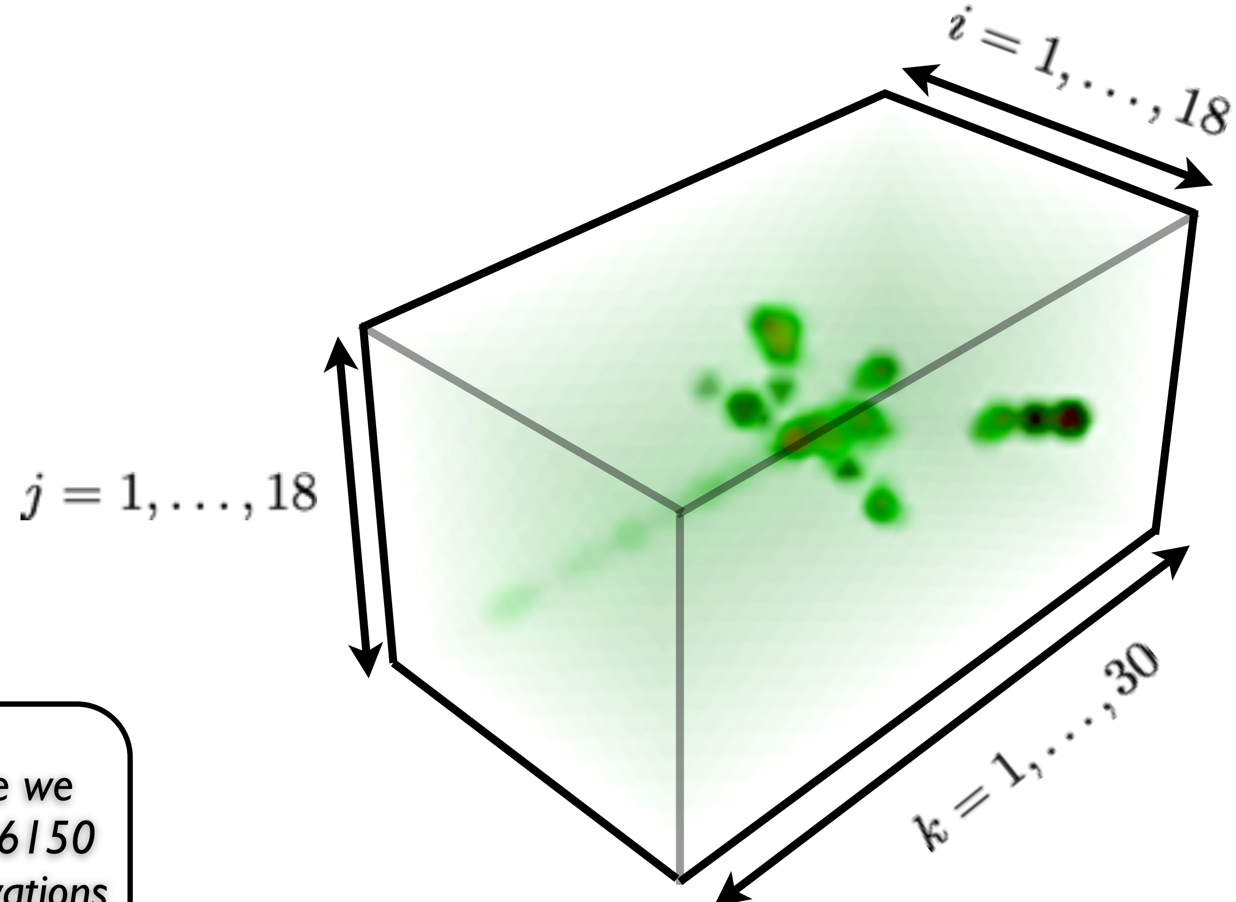
Outline

- Motivation
- Data description
- Internship project
  - Overview
  - Classification : AdaBoost
  - Pre-processing
    - Normalisation : histogram equalisation
    - Binarisation : automatic thresholding
  - Feature-extracting methods
    - Manual : Benchmark
    - Automatic : Deep Learning
  - Results
- Perspectives



Motivation

Data description

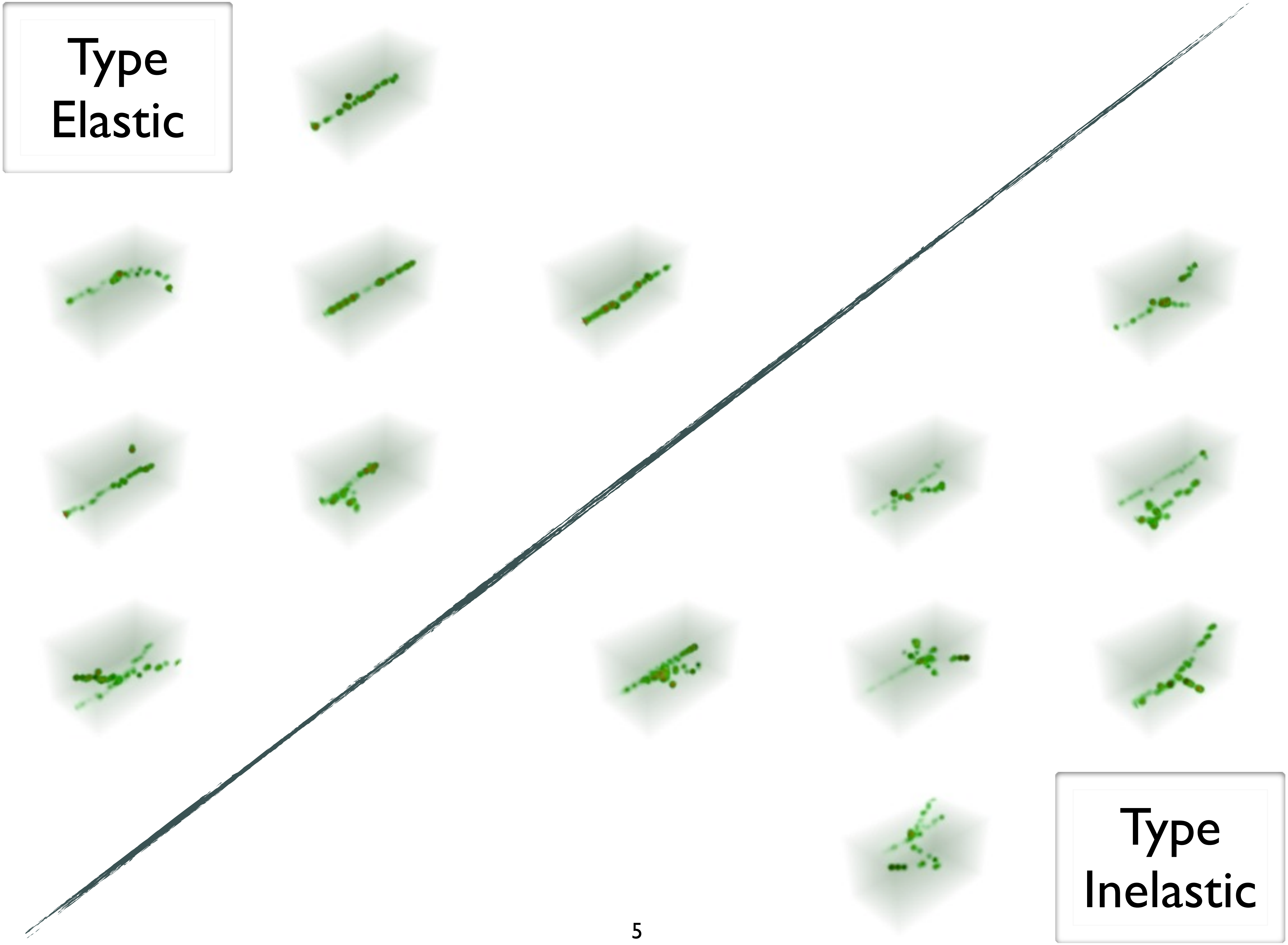


Here we have 6150 observations

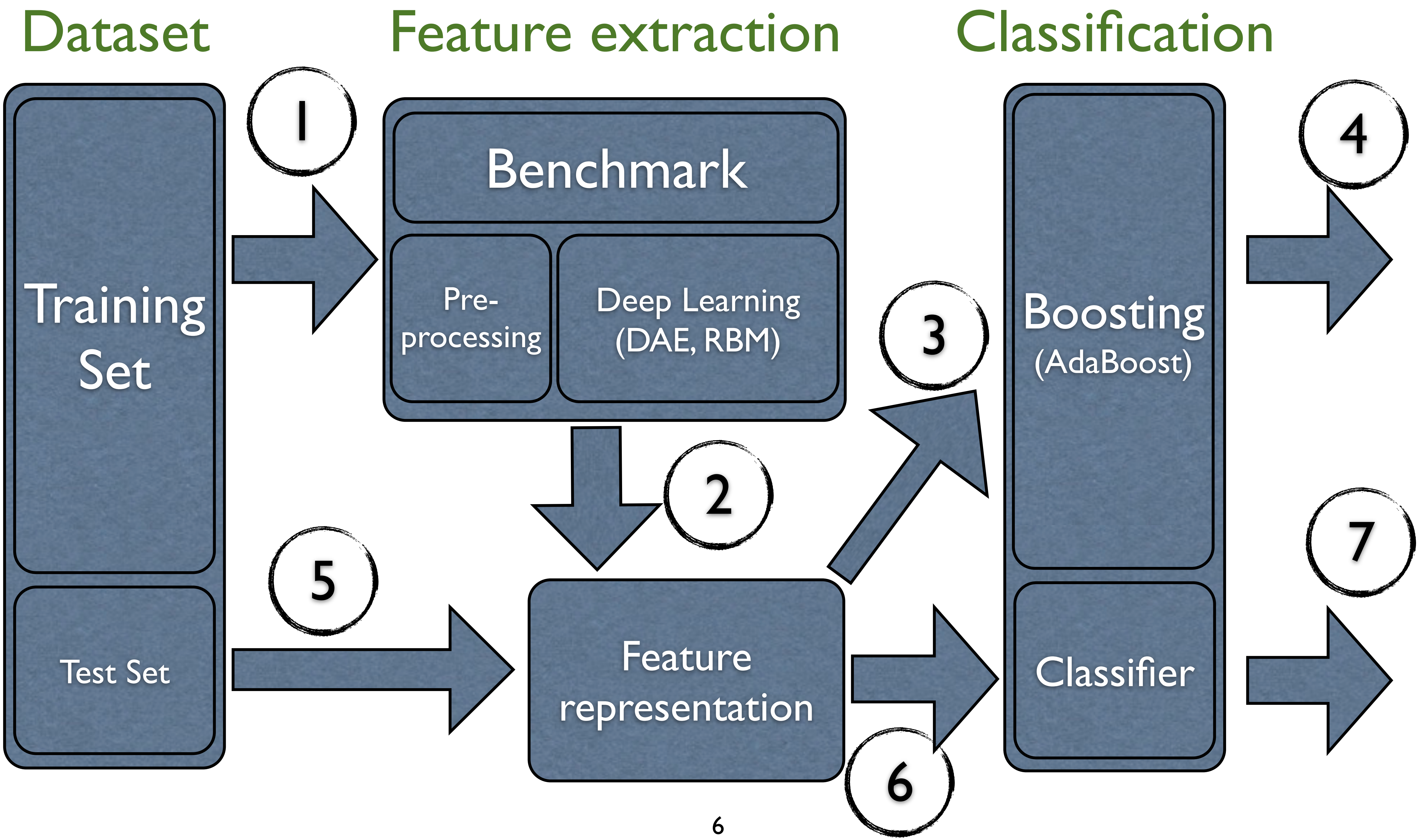
$$\text{Dataset} = \bigcup_{i=1}^{N} (x_i, y_i) \text{ where } (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{X} = \{\forall f,\ f : [\![1;18]\!]^2 \times [\![1;30]\!] \to \mathbb{R}^+\}$$

$$\mathcal{Y} = \{\text{Elastic, Inelastic}\}$$


Type
Elastic
Type
Inelastic

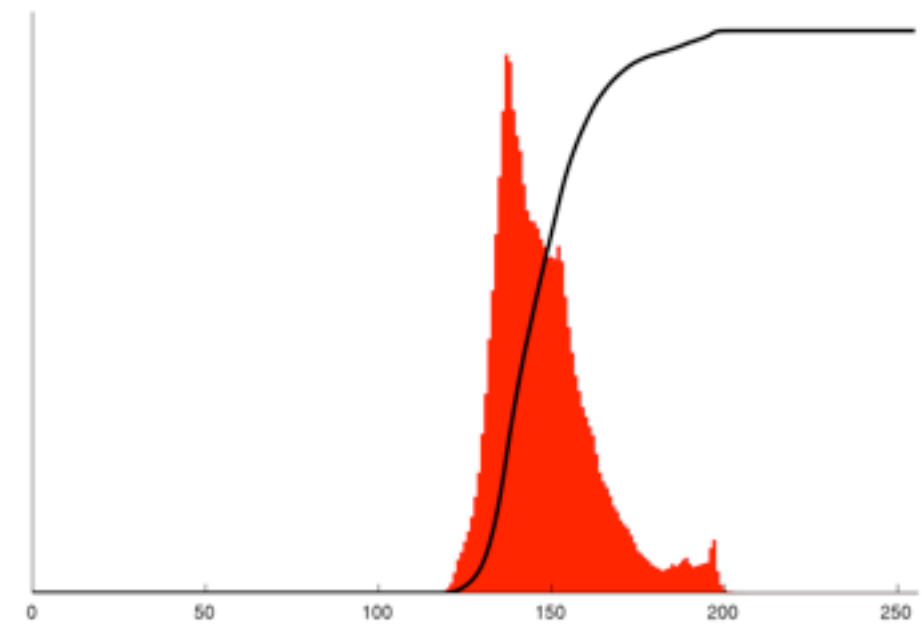
Project overview

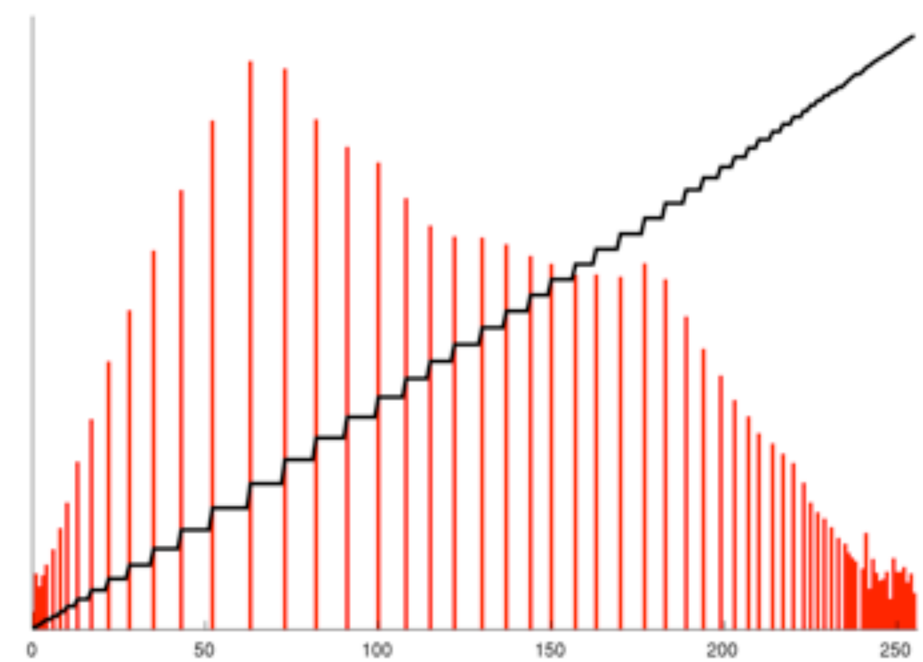


Classification

- MultiBoost package
  - Fast & efficient C++ implementation of algorithms based on the boosting paradigm
- AdaBoost
  - The classifiers it uses can be weak, but as long as their performance is slightly better than random, they will improve the final model
  - It generates and calls an new weak classifier in each series of rounds, where at each call, a distribution of weights is updated that indicates the importance of examples in the dataset for classification
    - At each round, the weights of incorrectly classified examples are increased, and the weights of the correctly classified examples are decreased
    - In that way, the new classifier focuses on ”difficult” examples (which are hard to classify)

Pre-processing (1|2)
Input normalisation using Histogram equalisation





Pre-processing (2|2) Binarising with automatic thresholding by entropy maximisation

Maximizing

$$E(S) = \sum_{i=0}^{S} \left(\frac{h(i)}{N_0^S} \cdot \; Log\left(\frac{h(i)}{N_0^S}\right)\right) - \sum_{i=S+1}^{255} \left(\frac{h(i)}{N_1^S} \cdot \; Log\left(\frac{h(i)}{N_1^S}\right)\right)$$

where h is the histogram of the data

$$\text{and} \begin{cases} N_0^S & \text{is the number of pixel which values are less than } S \\ N_1^S & \text{is the number of pixel which values are higher than } S+1 \end{cases}$$

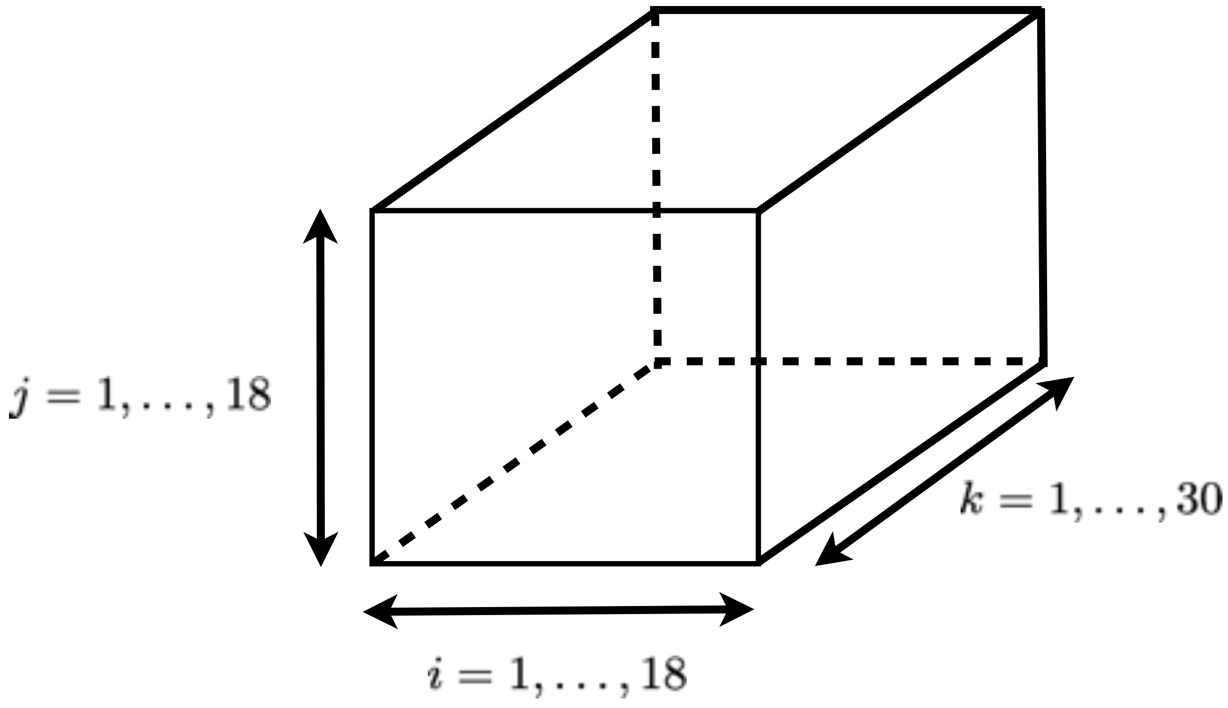
Feature extracting methods (1|2) Benchmark

- Used as a reference baseline to which we will compare new results
- Using insight from physicists : what features discriminate the classes

$$Y_{\bullet,\bullet,k} = \sum_{i,j} X_{i,j,k}$$

$$\Delta Y_{\bullet,\bullet,k} = X_{\bullet,\bullet,k+1} - X_{\bullet,\bullet,k}$$

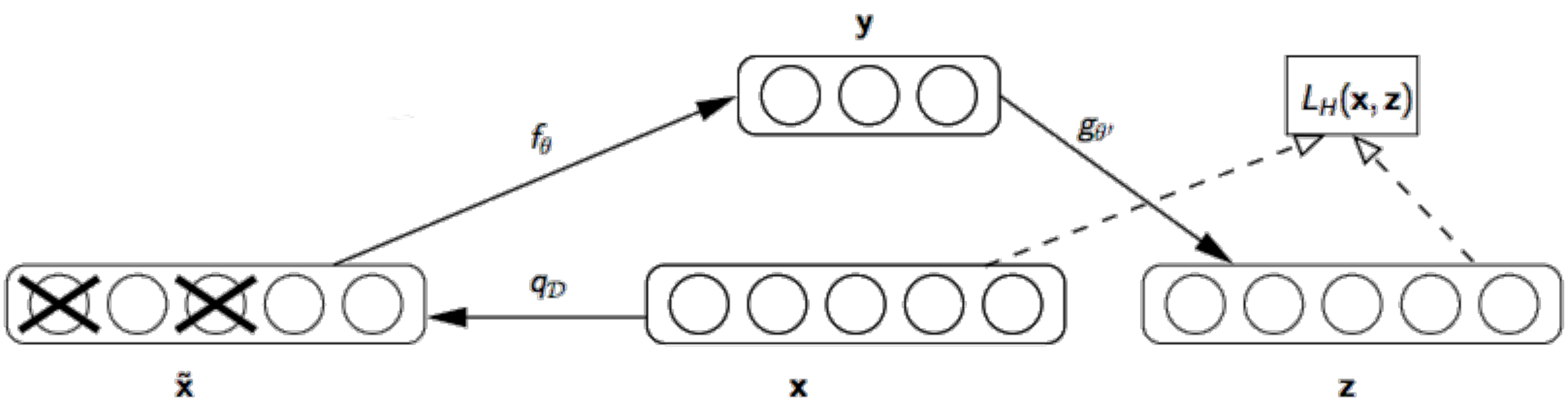
$$Y_{i,j,\bullet} = \sum_{k} X_{i,j,k}$$



Feature extracting methods (2|2) Deep Learning (1|3)

- New paradigm of Machine Learning
  - Recent rebirth in 06's
  - Strong impact in the field of Computer Vision or Natural Language Processing
- Main interest
  - Add an unsupervised pre-learning step before the supervised classification that automatically extracts relevant features from the training dataset
- List of algorithms commonly used
  - Auto-Associators|Auto-Encoders and their variants
  - Restricted Boltzmann Machines
  - Sparse Coding
  - Convolutionnal Networks

Feature extracting methods (2|2)
Deep Learning (2|3)
Denoising Auto-Encoder (1|2)



- Clean input $\mathbf{x} \in [0,1]^d$ is partially destroyed, yielding corrupted input: $\tilde{\mathbf{x}} \sim q_\mathcal{D}(\tilde{\mathbf{x}}|\mathbf{x})$.
- $\tilde{\mathbf{x}}$ is mapped to hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$.
- From $\mathbf{y}$ we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y})$.
- Train parameters to minimize the cross-entropy "reconstruction error"

ICML 2008, *Extracting and Composing Robust Features with Denoising Autoencoders*

Feature extracting methods (2|2)
Deep Learning (2|3)
Denoising Auto-Encoder (2|2)

Cross-Entropy Loss Function

$$L_H(\mathbf{x}, \mathbf{z}) = -\sum_{k=1}^{d} [\mathbf{x}_k \log \mathbf{z}_k + (1 - \mathbf{x}_k) \log(1 - \mathbf{z}_k)]$$

To be used when $\mathbf{x} \in [0, 1]^d$

Mean Square Error

$$L_H(\mathbf{x}, \mathbf{z}) = ||\mathbf{x} - \mathbf{z}||^2$$

Feature extracting methods (2|2)
Deep Learning (3|3)
Restricted Boltzmann Machine (1|5)

- Particular type of energy-based model
  - Define a probability function *P(x)* through an energy function

$$P(x) = \frac{e^{-\mathrm{Energy}(x)}}{Z}$$

$$Z = \sum_x e^{-\mathrm{Energy}(x)}, \text{ which are called } \textbf{partition function}$$

Feature extracting methods (2|2)
Deep Learning (3|3)
Restricted Boltzmann Machine (2|5)

EBM with hidden variables

New Probability function

The energy function depends now on x **and** h. But we only observe x and not (x,h), so we need to replace it on the previous equation by FreeEnergy(x) which is

$$\text{FreeEnergy}(x) = -log \sum_h e^{-\text{Energy}(x,h)}$$

$$P(x) = \frac{e^{-\text{FreeEnergy}(x)}}{Z}$$

Feature extracting methods (2|2)
Deep Learning (3|3)
Restricted Boltzmann Machine (3|5)

General Boltzmann Machines are EBMs with hidden variables

Restricted Boltzmann Machines are special case of Boltzmann Machines

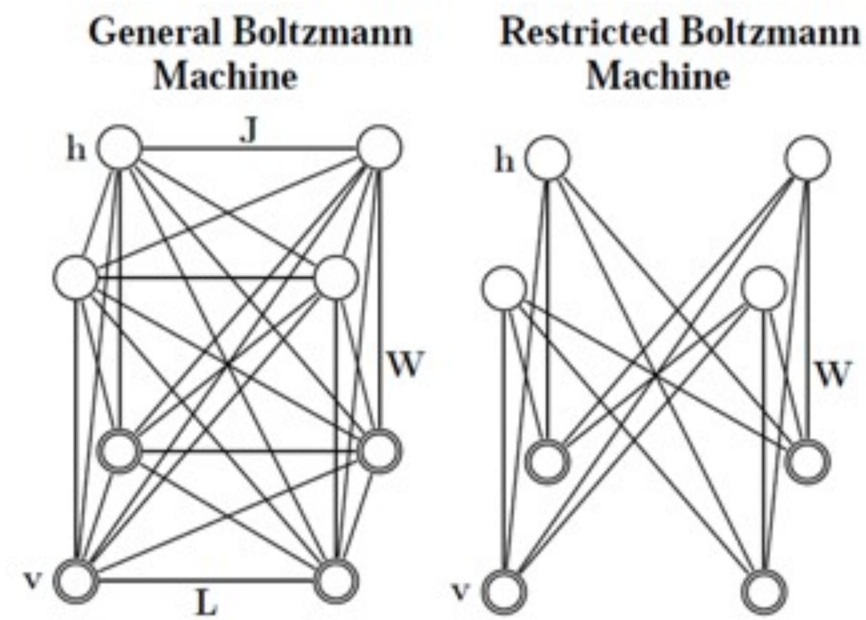


Figure 1: **Left**: A general Boltzmann machine. The top layer represents a vector of stochastic binary "hidden" features and the bottom layer represents a vector of stochastic binary "visible" variables. **Right**: A restricted Boltzmann machine with no hidden-to-hidden and no visible-to-visible connections.

$$\text{Boltzmann Machine} : \text{Energy}(x, h) = -b^T x - c^T h - h^T W x - x^T L x - h^T J h$$

$$\text{Restricted Boltzmann Machine} : \text{Energy}(x, h) = -b^T x - c^T h - h^T W x$$

Feature extracting methods (2|2) Deep Learning (3|3) Restricted Boltzmann Machine (4|5)

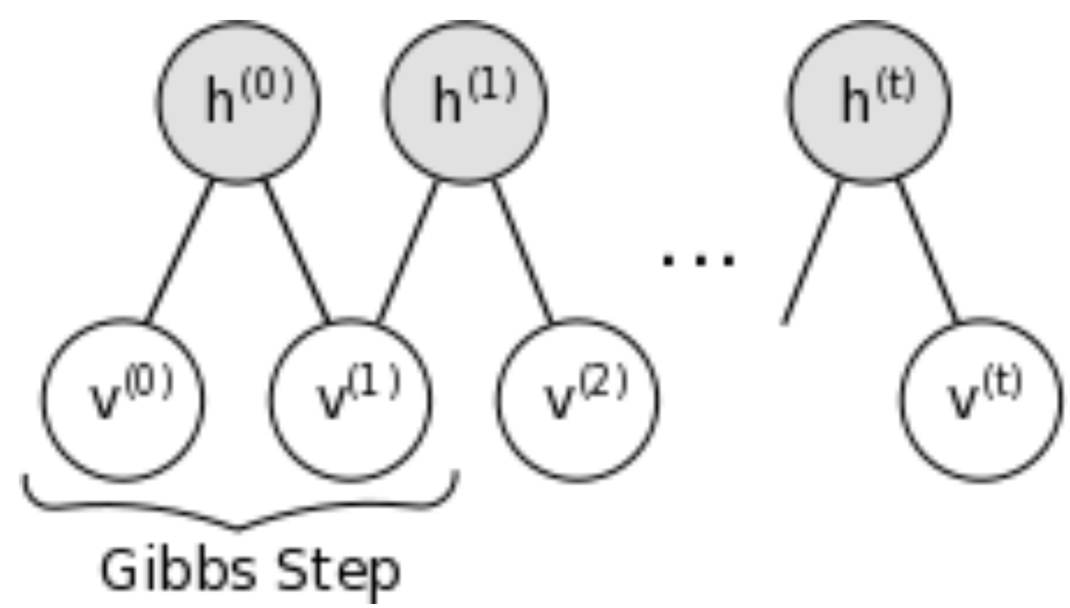
We run a Markov chain to convergence, using Gibbs sampling as the transition operator.

In RBM, visible and hidden units are independent conditionally.

A step in the Markov chain is thus taken as follows:

$$h^{(n+1)} \sim sigm(W^T v^{(n)} + c)$$

$$v^{(n+1)} \sim sigm(W h^{(n+1)} + b)$$



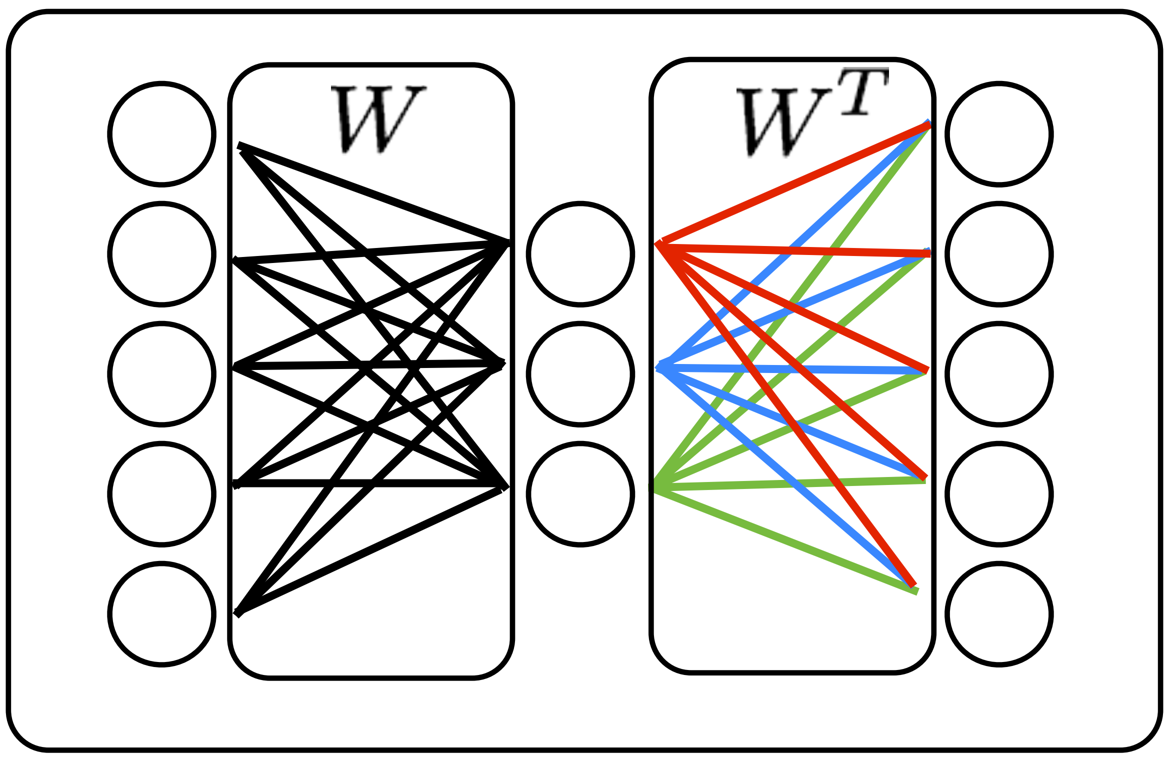
Problem : Too expensive to achieve convergence

Feature extracting methods (2|2)
Deep Learning (3|3)
Restricted Boltzmann Machine (5|5)

- Contrastive Divergence
  - Does not wait for the chain to converge
    - Samples are obtained after only k-steps of Gibbs sampling
- Persistent Contrastive Divergence
  - We do not restart the chain for each observed example
  - The state of the chain is preserved all along

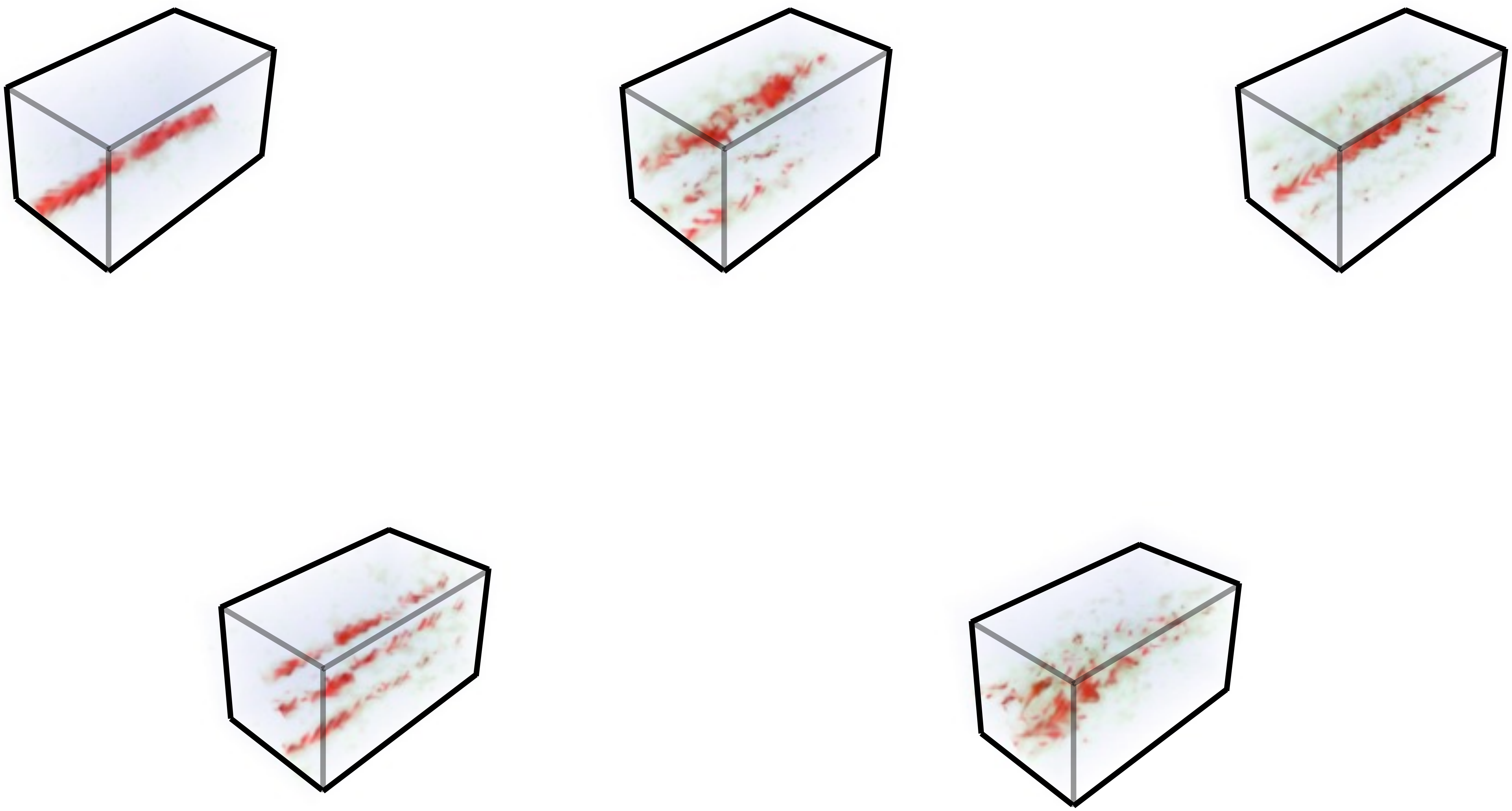
Feature extracting methods (2|2)
Results (1|2)
Examples of filters obtained (1|2)

Where do they come from ?

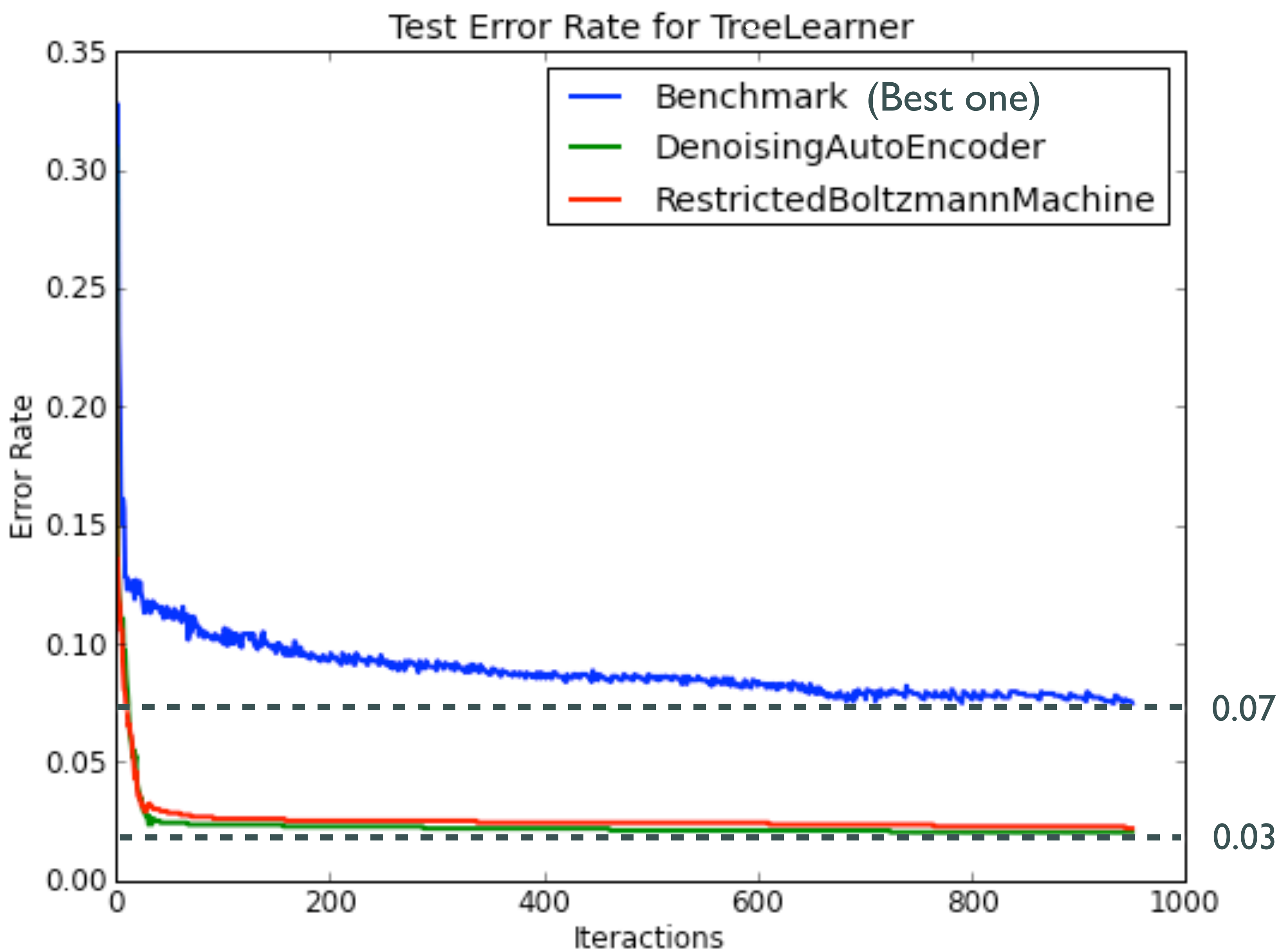


Auto-Associator

Feature extracting methods (2|2)
Results (1|2)
Examples of filters obtained (2|2)



Feature extracting methods (2|2)
Results (2|2)
Automatic VS Manually feature extracting methods



Perspectives (1|2)

- Multi-layer architectures
  - Stacked Denoising Auto-Encoders
    (Output layer N = Input layer N+1)
  - Stacked Restricted Boltzmann Machines
    (Hidden layer N = Visible layer N+1)
    - Deep Belief Networks
    - Deep Boltzmann Machines

Perspectives (2|2)
Future work

- Improving existing models or making new ones
- More classes, multi-label / multi-task
- Managing the complete detection process
- Real-time decision-making (trigger)
  - might have constraints on hardware