UTLC
Unsupervised Transfer Learning Challenge

Grégoire Mesnil[1,2], Yann Dauphin[1], Xavier Glorot[1],
Salah Rifai[1], Yoshua Bengio[1] *et al.*

[1] LISA, Université de Montréal, Canada
[2] LITIS, Université de Rouen, France

July 2nd 2011

Plan

UTL Challenge
Presentation

Dates :

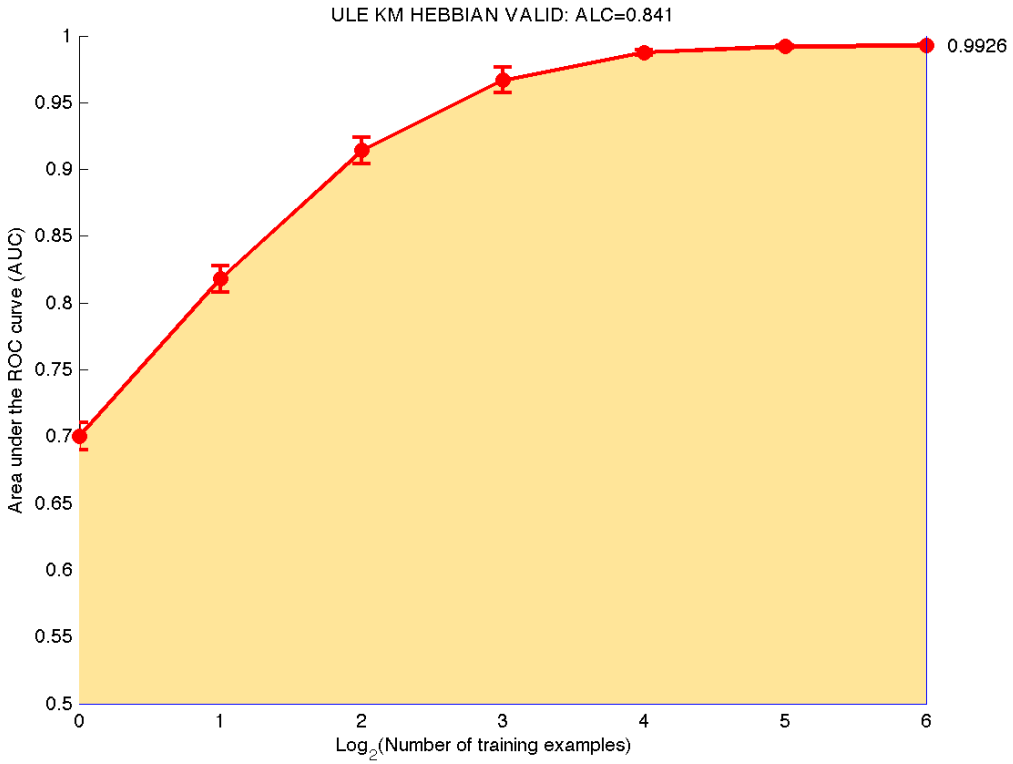
Phase 1 : Unsupervised Learning ; start : january 3, end : march 4.
Phase 2 : Transfer Learning ; start : march 4, end : april 15.

Five different Data sets :

data set		# samples	dimension	sparsity
AVICENNA	Arabic Manuscripts	150205	120	0 %
HARRY	Human actions	69652	5000	98 %
RITA	CIFAR-10	111808	7200	1 %
SYLVESTER	Ecology	572820	100	0 %
TERRY	NLP	217034	47236	99 %

UTL Challenge
Evaluation

ALC : Area under Learning Curve



1 to 64 samples per class

UTL Challenge
Performance

How to evaluate the performance of one model without any label or prior knowledge on the training set ?

UTL Challenge
Performance

How to evaluate the performance of one model without any label or prior knowledge on the training set ?

- proxy : ALC Valid versus Test (Phase 1)

UTL Challenge
Performance

How to evaluate the performance of one model without any label or prior knowledge on the training set ?

- proxy : ALC Valid versus Test (Phase 1)
- valid ALC returned by the competition servers (Phase 1 & 2)

UTL Challenge
Performance

How to evaluate the performance of one model without any label or prior knowledge on the training set ?

- proxy : ALC Valid versus Test (Phase 1)
- valid ALC returned by the competition servers (Phase 1 & 2)
- ALC with the given labels (Phase 2)

UTL Challenge
Performance

How to evaluate the performance of one model without any label or prior knowledge on the training set ?

- proxy : ALC Valid versus Test (Phase 1)
- **valid ALC returned by the competition servers (Phase 1 & 2)**
- ALC with the given labels (Phase 2)

UTL Challenge
Performance

How to evaluate the performance of one model without any label or prior knowledge on the training set ?

- proxy : ALC Valid versus Test (Phase 1)
- **valid ALC returned by the competition servers (Phase 1 & 2)**
- ALC with the given labels (Phase 2)

From phase 1 to phase 2, we over-explored the hyperparameters of the next models to grab the 1st place.

Deep Architecture

Stack different blocks

We used this template :

1. **Pre-processing :** PCA w/wo whitening, Contrast Normalization, Uniformization
2. **Feature Extraction :** Rectifiers, DAE, CAE, μ-ss-RBM
3. **Post-processing :** Transductive PCA

Deep Architecture

Stack different blocks

We used this template :

1. **Pre-processing :** PCA w/wo whitening, Contrast Normalization, Uniformization
2. **Feature Extraction :** Rectifiers, DAE, CAE, μ-ss-RBM
3. **Post-processing :** Transductive PCA

Deep Architecture

Stack different blocks

We used this template :

1. **Pre-processing :** PCA w/wo whitening, Contrast Normalization, Uniformization
2. **Feature Extraction :** Rectifiers, DAE, CAE, μ-ss-RBM
3. **Post-processing :** Transductive PCA

Deep Architecture

Stack different blocks

We used this template :

1. **Pre-processing :** PCA w/wo whitening, Contrast Normalization, Uniformization
2. **Feature Extraction :** Rectifiers, DAE, CAE, μ-ss-RBM
3. **Post-processing :** Transductive PCA

Plan

Preprocessing

Given a training set $\mathcal{D} = \{x^{(j)}\}_{j=1\ldots n}$ where $x^{(j)} \in \mathbb{R}^d$:

- **Uniformization** (t-IDF)
  Rank all the $x_i^{(j)}$ and map them to $[0, 1]$
- **Contrast Normalization**
  For each $x^{(j)}$, compute its mean $\mu^{(j)} = \sum_{i=1}^{d} x_i^{(j)}$ and its deviation $\sigma^{(j)}$. $x^{(j)} \leftarrow (x^{(j)} - \mu^{(j)})/\sigma^{(j)}$
- **Principal Component Analysis**
  with/without whitening
  i.e divide by the squared root eigen value or not.

Preprocessing

Given a training set $\mathcal{D} = \{x^{(j)}\}_{j=1\ldots n}$ where $x^{(j)} \in \mathbb{R}^d$:

- **Uniformization** (t-IDF)
  Rank all the $x_i^{(j)}$ and map them to $[0, 1]$
- **Contrast Normalization**
  For each $x^{(j)}$, compute its mean $\mu^{(j)} = \sum_{i=1}^{d} x_i^{(j)}$ and its deviation $\sigma^{(j)}$. $x^{(j)} \leftarrow (x^{(j)} - \mu^{(j)})/\sigma^{(j)}$
- **Principal Component Analysis**
  with/without whitening
  i.e divide by the squared root eigen value or not.

Preprocessing

Given a training set $\mathcal{D} = \{x^{(j)}\}_{j=1\ldots n}$ where $x^{(j)} \in \mathbb{R}^d$:

- **Uniformization** (t-IDF)
  Rank all the $x_i^{(j)}$ and map them to $[0, 1]$
- **Contrast Normalization**
  For each $x^{(j)}$, compute its mean $\mu^{(j)} = \sum_{i=1}^{d} x_i^{(j)}$ and its deviation $\sigma^{(j)}$. $x^{(j)} \leftarrow (x^{(j)} - \mu^{(j)})/\sigma^{(j)}$
- **Principal Component Analysis**
  with/without whitening
  i.e divide by the squared root eigen value or not.

Preprocessing

Given a training set $\mathcal{D} = \{x^{(j)}\}_{j=1\ldots n}$ where $x^{(j)} \in \mathbb{R}^d$:

- **Uniformization** (t-IDF)
  Rank all the $x_i^{(j)}$ and map them to $[0, 1]$
- **Contrast Normalization**
  For each $x^{(j)}$, compute its mean $\mu^{(j)} = \sum_{i=1}^{d} x_i^{(j)}$ and its deviation $\sigma^{(j)}$. $x^{(j)} \leftarrow (x^{(j)} - \mu^{(j)})/\sigma^{(j)}$
- **Principal Component Analysis**
  with/without whitening
  i.e divide by the squared root eigen value or not.

Plan

Feature Extraction

μ-ss-RBM

μ-**Spike & Slab Restricted Boltzmann Machine** modelizes the interaction between three random vectors :

1. visible vector v representing the observed data
2. binary "**spike**" variables h
3. real-valued "**slab**" variables s

Feature Extraction

μ-ss-RBM

μ-**Spike & Slab Restricted Boltzmann Machine** modelizes the interaction between three random vectors :

1. visible vector v representing the observed data
2. binary "**spike**" variables h
3. real-valued "**slab**" variables s

It is defined by the energy function :

$$E(v,s,h) = -\sum_{i=1}^{N} v^T W_i s_i h_i + \frac{1}{2} v^T \left(\Lambda + \sum_{i=1}^{N} \Phi_i h_i \right) v$$
$$+ \sum_{i=1}^{N} \frac{1}{2} s_i^T \alpha_i s_i - \sum_{i=1}^{N} \mu_i^T \alpha_i s_i h_i - \sum_{i=1}^{N} b_i h_i + \sum_{i=1}^{N} \mu_i^T \alpha_i \mu_i h_i,$$

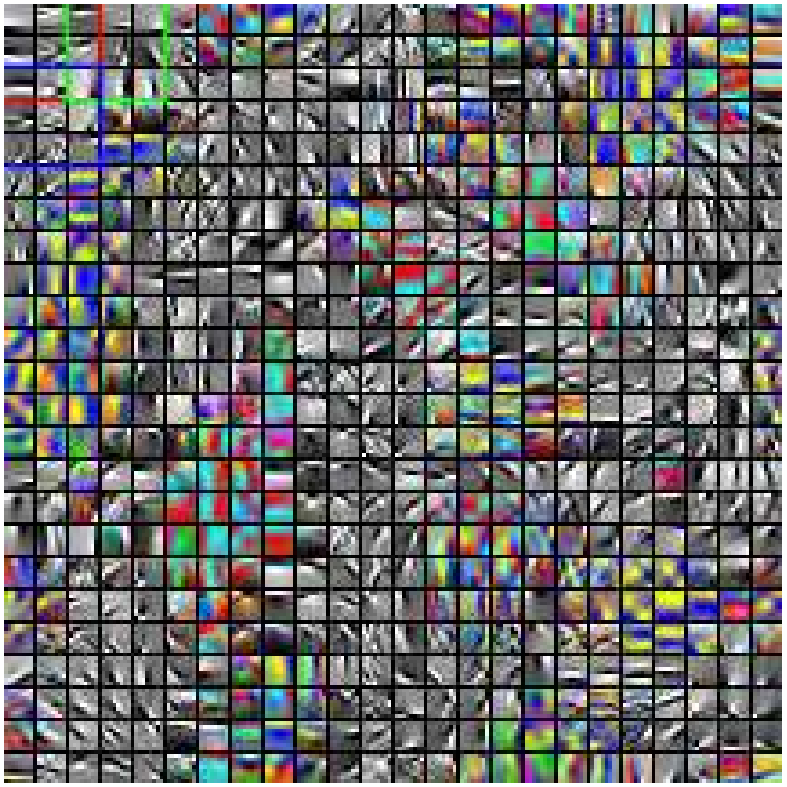
In training, we use **Persistent Contrastive Divergence** with a **Gibbs Sampling** procedure.

Feature Extraction

μ-ss-RBM

more details in A.Courville, J.Bergstra and Y.Bengio, *Unsupervised Models of Images by Spike-and-Slab RBMs*, **ICML 2011**.

Pools of filters learned on CIFAR-10



Feature Extraction

Denoising Autoencoders

A **Denoising Autoencoder** is an autoencoder trained to **denoise** artificially corrupted training samples.

Corruption e.g $\tilde{x} = x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$

Encoder : $h(\tilde{x}) = s(W\tilde{x} + b)$ where s is the sigmoid function.

Decoder : $r(\tilde{x}) = W^T h(\tilde{x}) + b^{'}$ (tied weights).

Feature Extraction

Denoising Autoencoders

A **Denoising Autoencoder** is an autoencoder trained to **denoise** artificially corrupted training samples.

Corruption e.g $\tilde{x} = x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$

Encoder : $h(\tilde{x}) = s(W\tilde{x} + b)$ where s is the sigmoid function.

Decoder : $r(\tilde{x}) = W^T h(\tilde{x}) + b^{'}$ (tied weights).

Different loss functions to be minimized using **stochastic gradient descent** :

- $\|r(\tilde{x}) - x\|_2^2$ (linear reconstruction and MSE)
- $\|s(r(\tilde{x})) - x\|_2^2$ (non-linear reconstruction)
- $-\sum_i x_i \log r(\tilde{x}_i) - (1 - x_i) \log(1 - r(\tilde{x}_i))$ (cross-entropy)

Feature Extraction

Contractive Autoencoders

A **Contractive Autoencoder** encourages an **invariance** of the representation by penalizing the *sensitivity* of its encoder to the training inputs characterized with :

$$\|J_f(x)\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2$$

Feature Extraction

Contractive Autoencoders

A **Contractive Autoencoder** encourages an **invariance** of the representation by penalizing the *sensitivity* of its encoder to the training inputs characterized with :

$$\|J_f(x)\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2$$

To avoid useless constant representations, this term is counterbalanced by a **reconstruction error** and use **tied weights** (decoder and encoder share the same weights) :

$$\|s(r(x)) - x\|_2^2 + \lambda \|J_f(x)\|_F^2$$

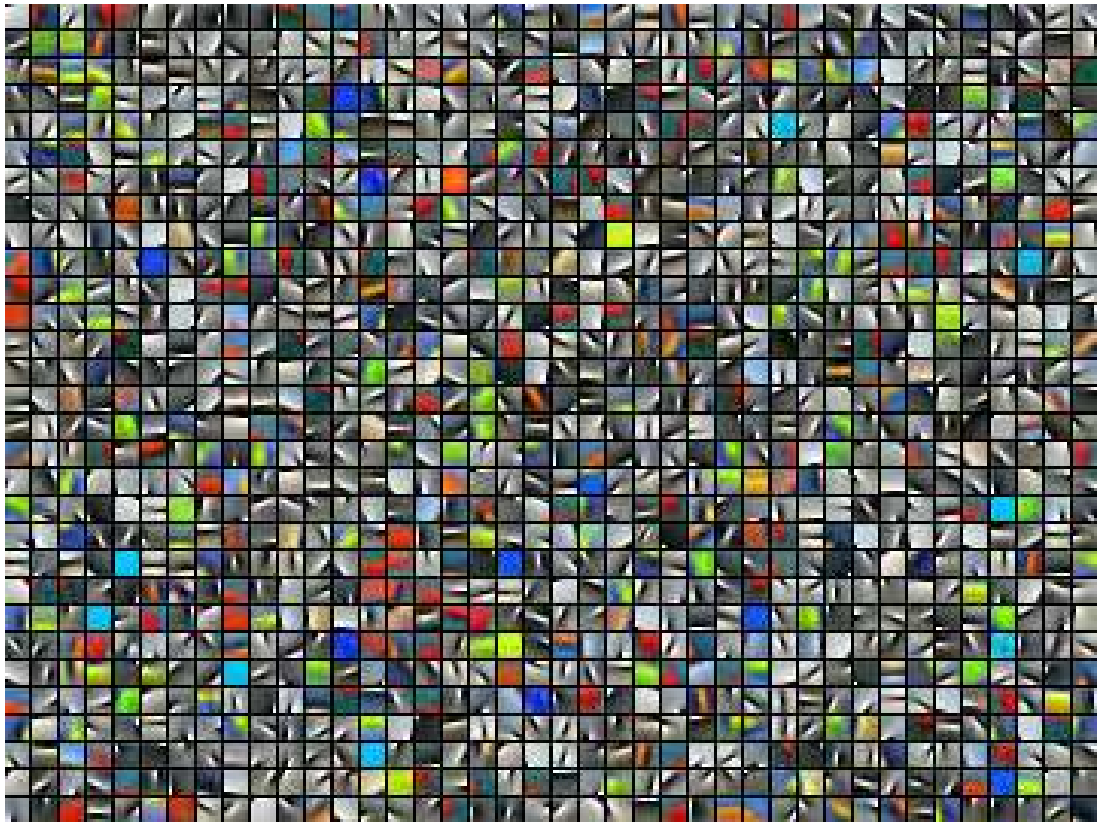
where λ controls the tradeoff between both penalities.

Feature Extraction

Contractive Autoencoders

more details in S.Rifai, P.Vincent, X.Muller, X.Glorot and Y.Bengio *Contractive Auto-Encoders : Explicit Invariance During Feature Extraction*, **ICML 2011**.

Random selection of 4000 filters learned on CIFAR-10



Feature Extraction

Rectifiers

Rectifiers use the activation function $\max(0, Wx + b)$ and therefore create sparse representation with **true** zeros. Those are used to be trained as Denoising Autoencoders.

more details in X.Glorot, A.Bordes and Y.Bengio, *Domain Adaptation for Large-Scale Sentiment Classification : A Deep Learning Approach*, **ICML 2011**.

Feature Extraction

Rectifiers

Rectifiers use the activation function $\max(0, Wx + b)$ and therefore create sparse representation with **true** zeros. Those are used to be trained as Denoising Autoencoders.

more details in X.Glorot, A.Bordes and Y.Bengio, *Domain Adaptation for Large-Scale Sentiment Classification : A Deep Learning Approach*, **ICML 2011**.

For huge sparse distributions, e.g :

- input dimension is $50,000$
- embedding dimension is $1,000$
- $\Longrightarrow$ **decoding** requires $50,000,000$ operations. **Expensive**...

Feature Extraction

Reconstruction Sampling

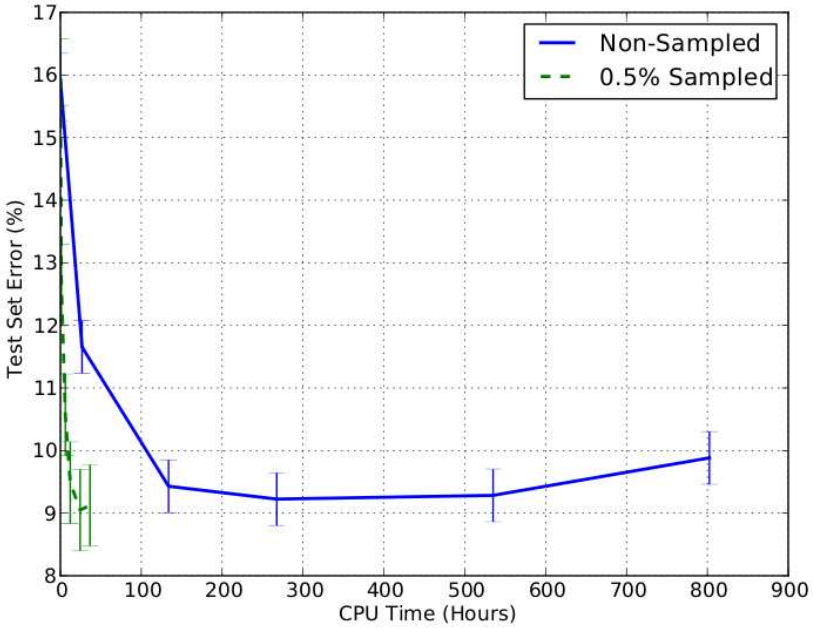
Reconstruction sampling : reconstruct all the non-zeros elements and a small random subset of the zeros elements and **speed-up** training.

Feature Extraction

Reconstruction Sampling

Reconstruction sampling : reconstruct all the non-zeros elements and a small random subset of the zeros elements and **speed-up** training.

more details in Y.Dauphin, X.Glorot and Y.Bengio, *Large-Scale Learning of Embeddings with Reconstruction Sampling*, **ICML 2011**.



Plan

Postprocessing
Transductive PCA

The feature extraction is performed on the training set while a **Transductive PCA** is a PCA trained not on the training set but on the valid (or test) set.

- Trained on the **representation** learned by the feature extraction process.
- Only retains dominant variations on the test or validation test.
- Validation of the **number of components** on the valid set (assume there is the same number of classes in the test and valid set).

Postprocessing
Transductive PCA

The feature extraction is performed on the training set while a **Transductive PCA** is a PCA trained not on the training set but on the valid (or test) set.

- Trained on the **representation** learned by the feature extraction process.
- Only retains dominant variations on the test or validation test.
- Validation of the **number of components** on the valid set (assume there is the same number of classes in the test and valid set).

Postprocessing
Transductive PCA

The feature extraction is performed on the training set while a **Transductive PCA** is a PCA trained not on the training set but on the valid (or test) set.

- Trained on the **representation** learned by the feature extraction process.
- Only retains dominant variations on the test or validation test.
- Validation of the **number of components** on the valid set (assume there is the same number of classes in the test and valid set).

Postprocessing
Transductive PCA

The feature extraction is performed on the training set while a **Transductive PCA** is a PCA trained not on the training set but on the valid (or test) set.

- Trained on the **representation** learned by the feature extraction process.
- Only retains dominant variations on the test or validation test.
- Validation of the **number of components** on the valid set (assume there is the same number of classes in the test and valid set).

Computation

How much time ?

From preprocessing to postprocessing, the time spent for training is at most 12 hours for every model...

Computation

How much time ?

From preprocessing to postprocessing, the time spent for training is at most 12 hours for every model...

Once you have found the good hyperparameters ! And there is a lot.

Computation

How much time ?

From preprocessing to postprocessing, the time spent for training is at most 12 hours for every model...
Once you have found the good hyperparameters ! And there is a lot.

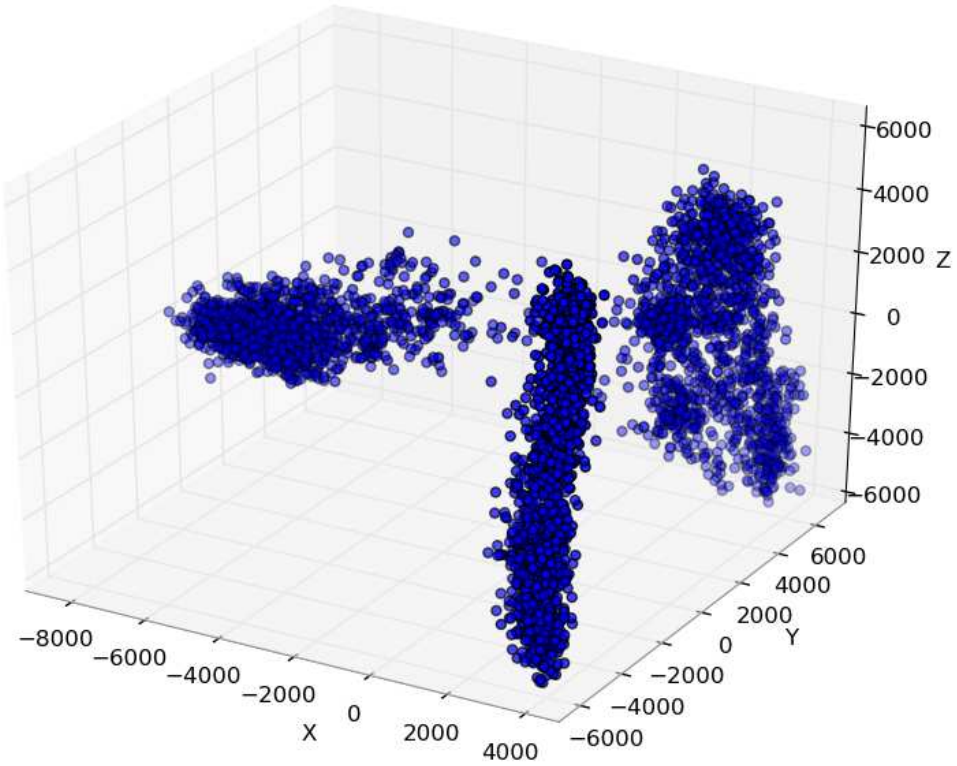
Software : Theano (Python Library)
Hardware : GPU (Geforce GTX 580)

theano

http ://deeplearning.net/

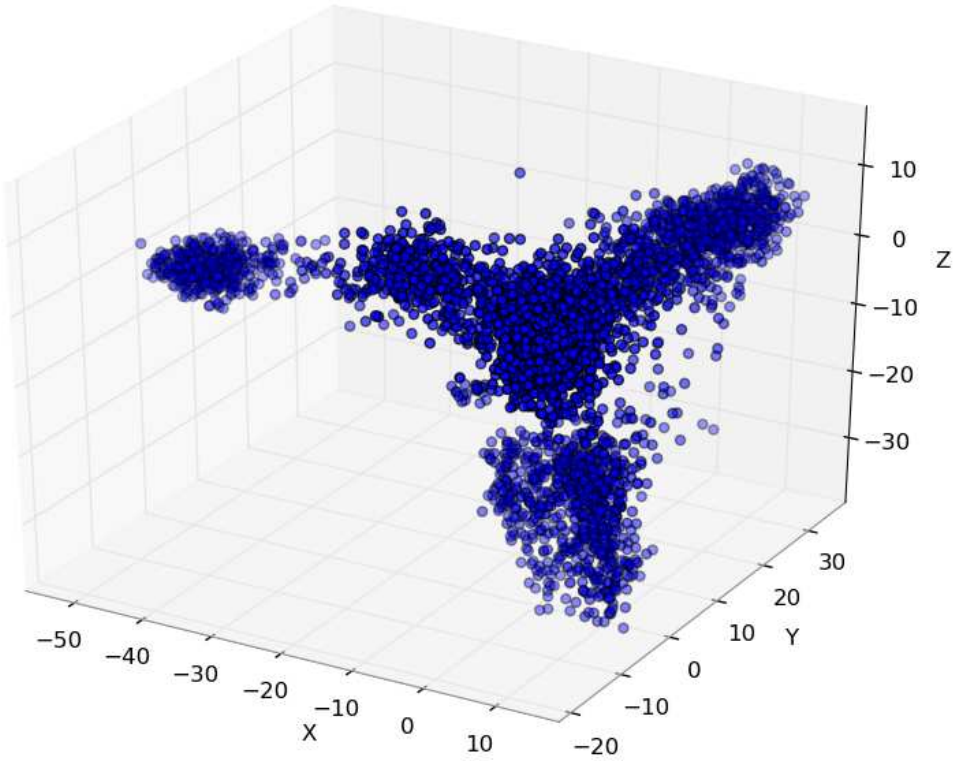
Harry
Best model

input dimension is 5,000 (98% sparse) **Human actions**



Terry
Best model

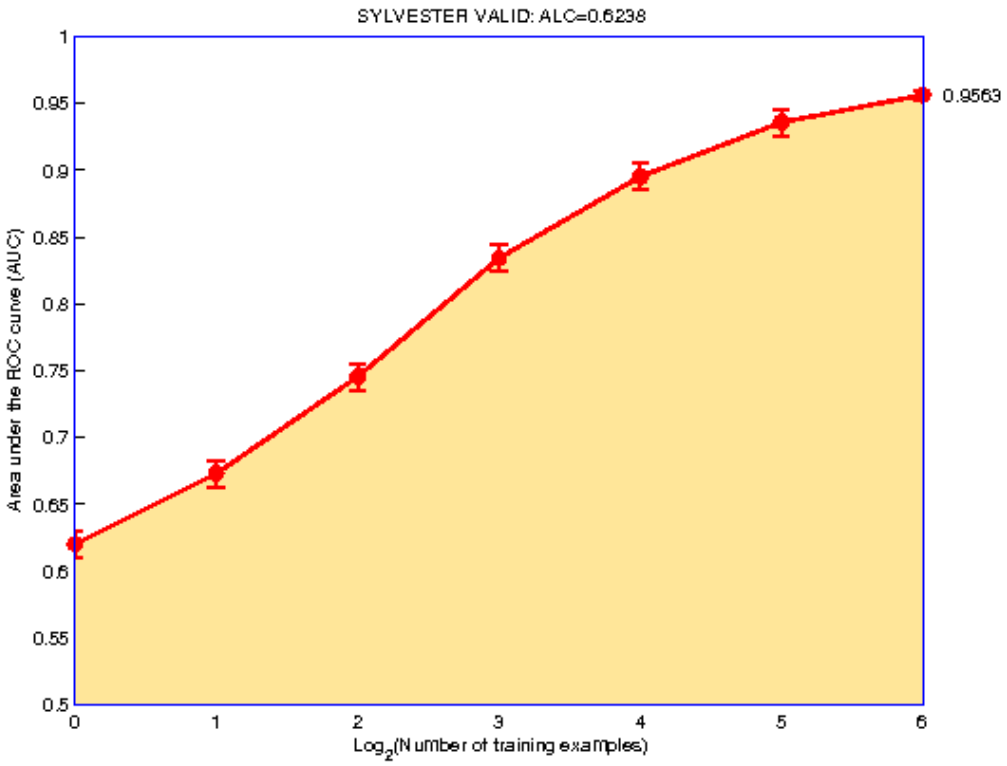
input dimension is 47,236 (99% sparse) **Natural Language Processing**



Sylvester
Best model

input dimension is 100 (no sparsity) **Ecology**

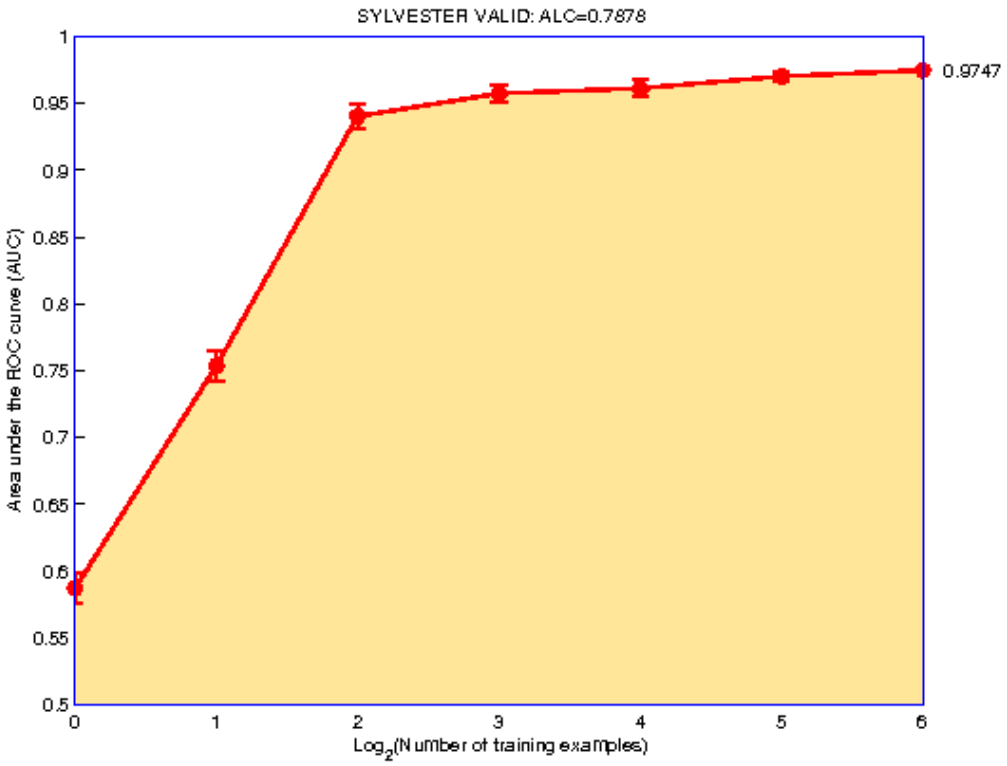
Stacking effect PCA-8



Sylvester
Best model

input dimension is 100 (no sparsity) **Ecology**

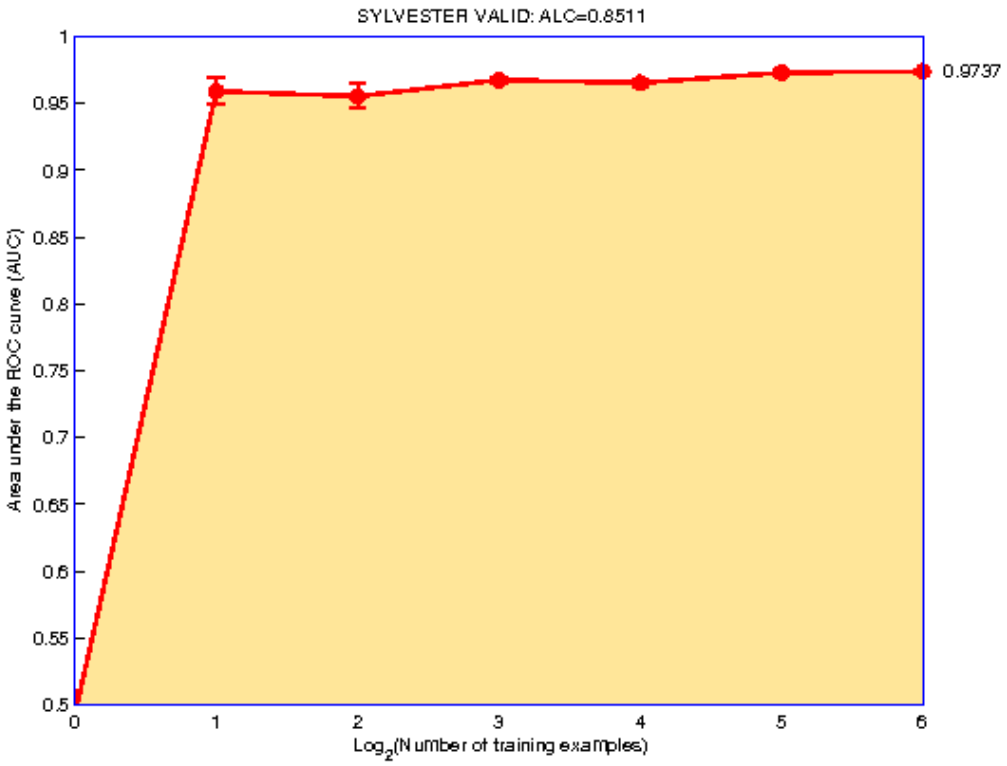
Stacking effect PCA-8 // CAE-6



Sylvester
Best model

input dimension is 100 (no sparsity) **Ecology**

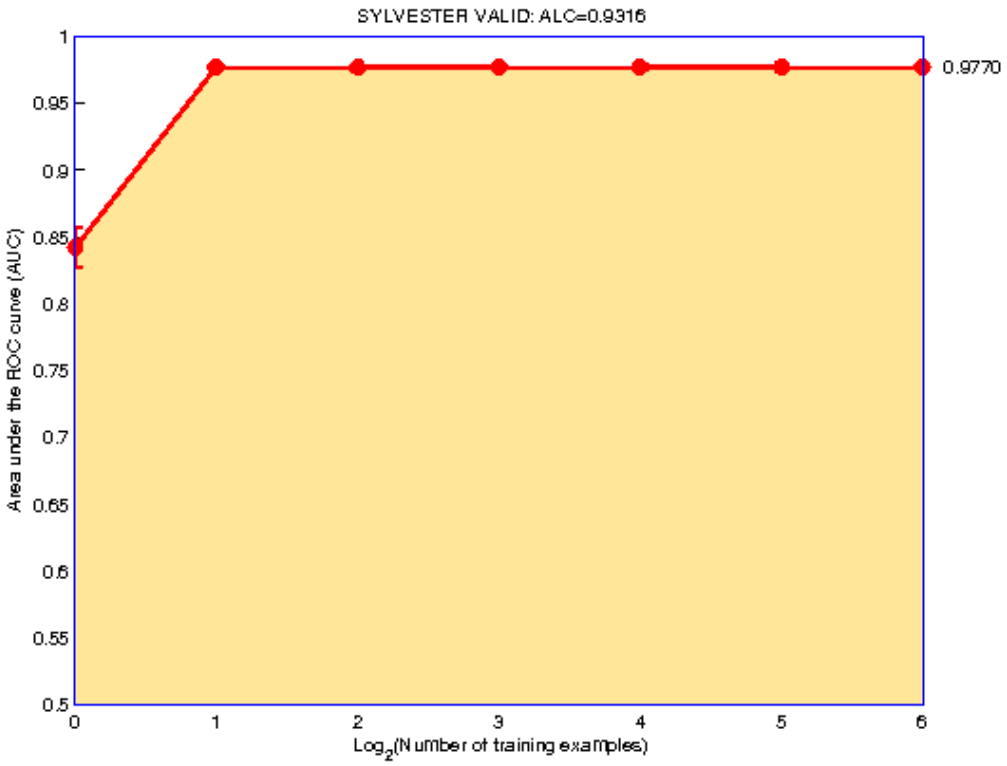
Stacking effect PCA-8 // CAE-6 // CAE-6



Sylvester
Best model

input dimension is 100 (no sparsity) **Ecology**

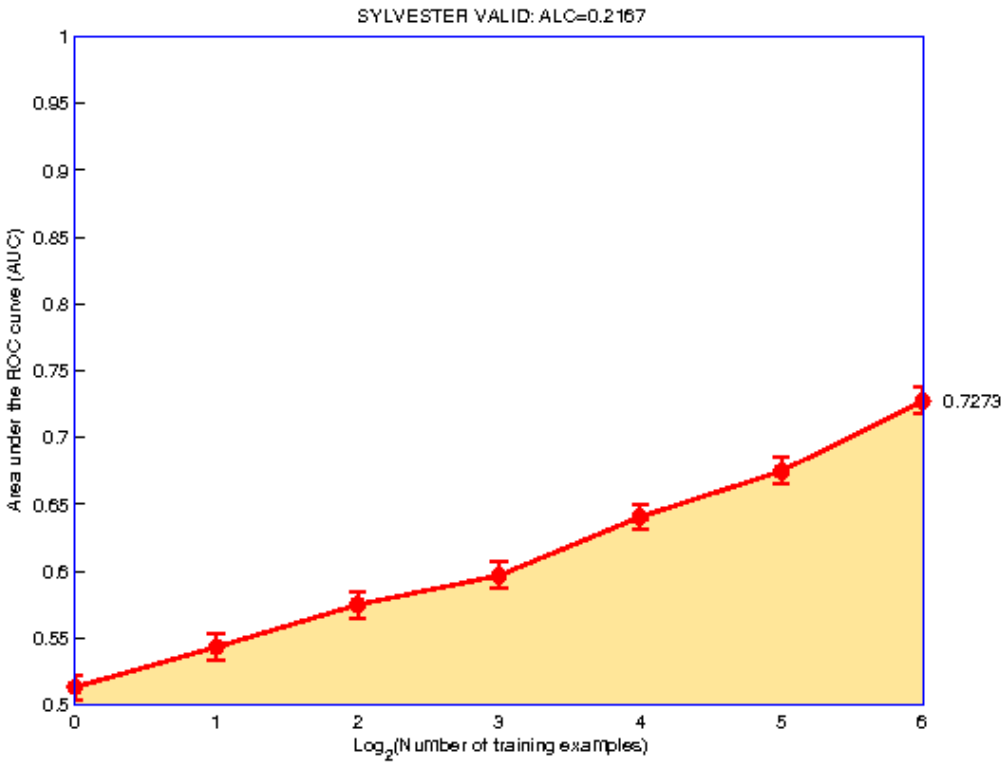
Stacking effect PCA-8 // CAE-6 // CAE-6 // PCA-1



Sylvester
Best model

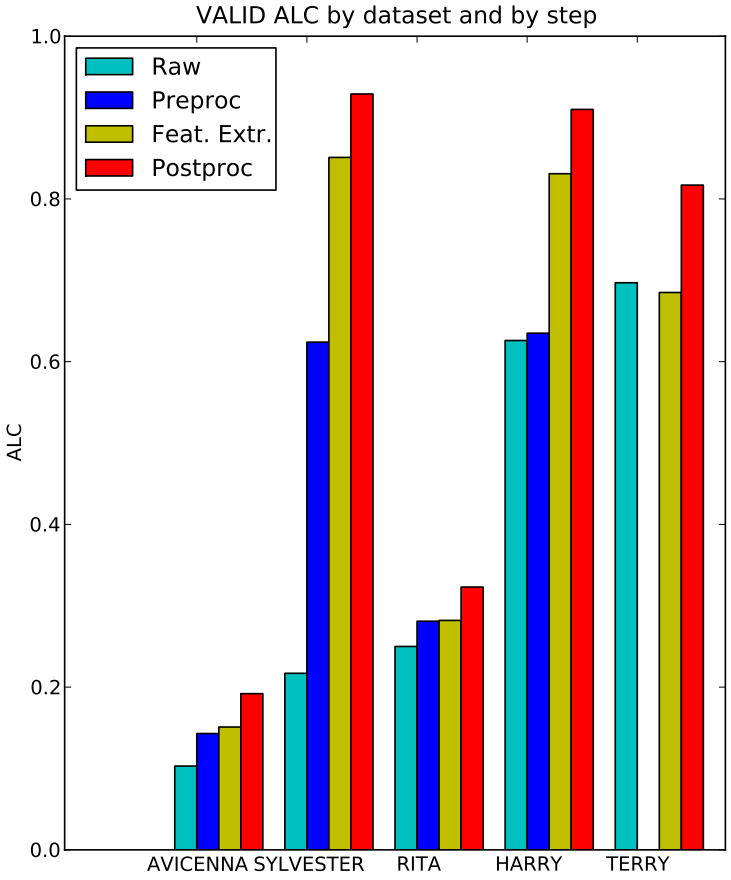
input dimension is 100 (no sparsity) **Ecology**

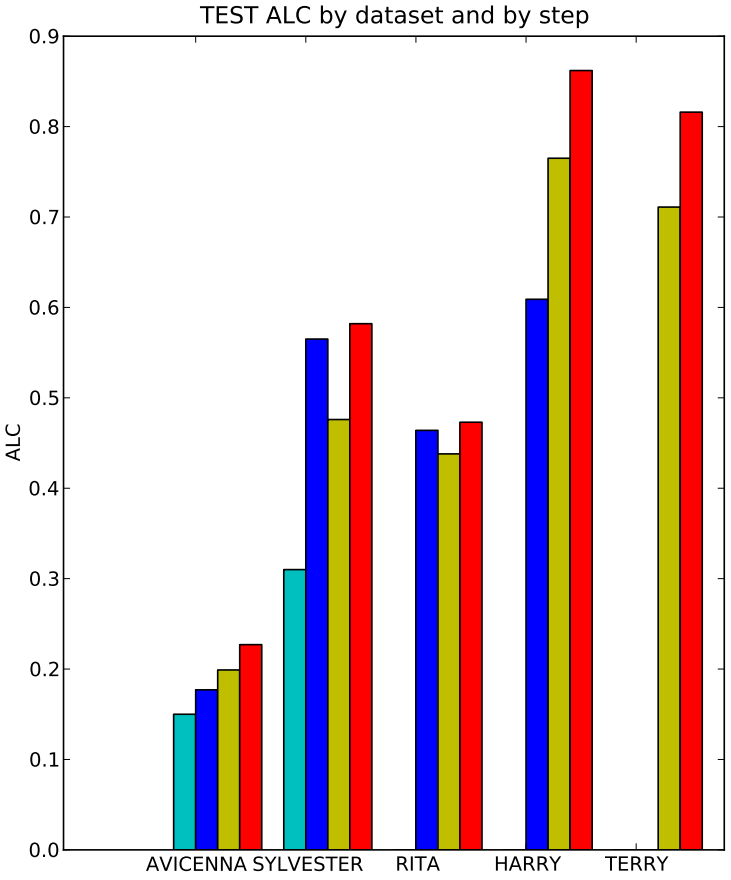
Stacking effect compared to raw data



Overall
Best models

ALC computed at each stage on the five data sets.





Summary

We proposed a successful deep approach decomposed in three steps :

1. Preprocessing
2. Feature Extraction
3. Postprocessig

We ranked 4th in the phase 1 and 1st in the phase 2.

more details in our **JMLR** paper :

G.Mesnil, Y.Dauphin, X.Glorot, Y.bengio, *et al*. **Unsupervised and Transfer Learning Challenge : a Deep Learning Approach**. (to appear)

UTLC
Unsupervised Transfer Learning Challenge

Grégoire Mesnil[1,2], Yann Dauphin[1], Xavier Glorot[1],
Salah Rifai[1], Yoshua Bengio[1] *et al.*

[1] LISA, Université de Montréal, Canada
[2] LITIS, Université de Rouen, France

Thanks for your attention. Questions ?