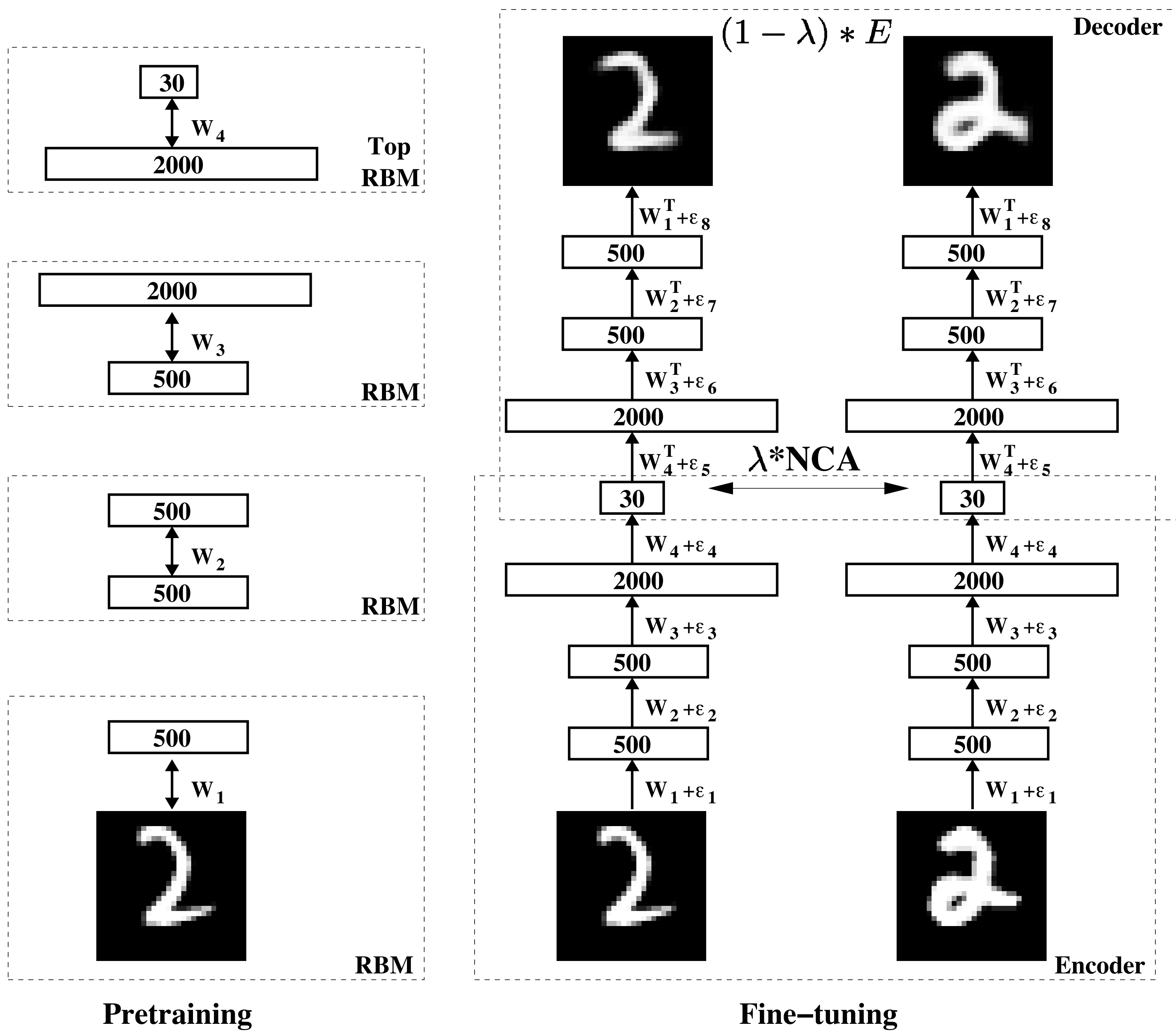
Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure

(Salakhutdinov and Hinton, AISTATS 2007)

- Peut-on adapter la représentation à autre chose qu’un classifieur réseau de neurones?
- Dans ce papier, on l’adapte à un classifieur des K plus proches voisins

Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure

(Salakhutdinov and Hinton, AISTATS 2007)



Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure

(Salakhutdinov and Hinton, AISTATS 2007)

- NCA (Neighbourhood Component Analysis)

probabilité que 'a' choisisse 'b' comme voisin

$$p_{ab} = \frac{\exp(-d_{ab})}{\sum_{z \neq a} \exp(-d_{az})}, \qquad p_{aa} = 0$$

Représentations réseau de neurones

$$d_{ab} = \| f(\mathbf{x}^a | W) - f(\mathbf{x}^b | W) \|^2$$

probabilité que ce voisin soit de la même classe

$$p(c^a = k) = \sum_{b: c^b = k} p_{ab}$$

Objectif: maximiser cette probabilité

$$O_{NCA} = \sum_{a=1}^{N} \sum_{b: c^a = c^b} p_{ab}$$

Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure

(Salakhutdinov and Hinton, AISTATS 2007)

- NCA: la fonction de coût est

$$-\sum_{a=1}^{N} \sum_{b:c^a=c^b} \frac{\exp(-d_{ab})}{\sum_{z \neq a} \exp(-d_{az})}$$

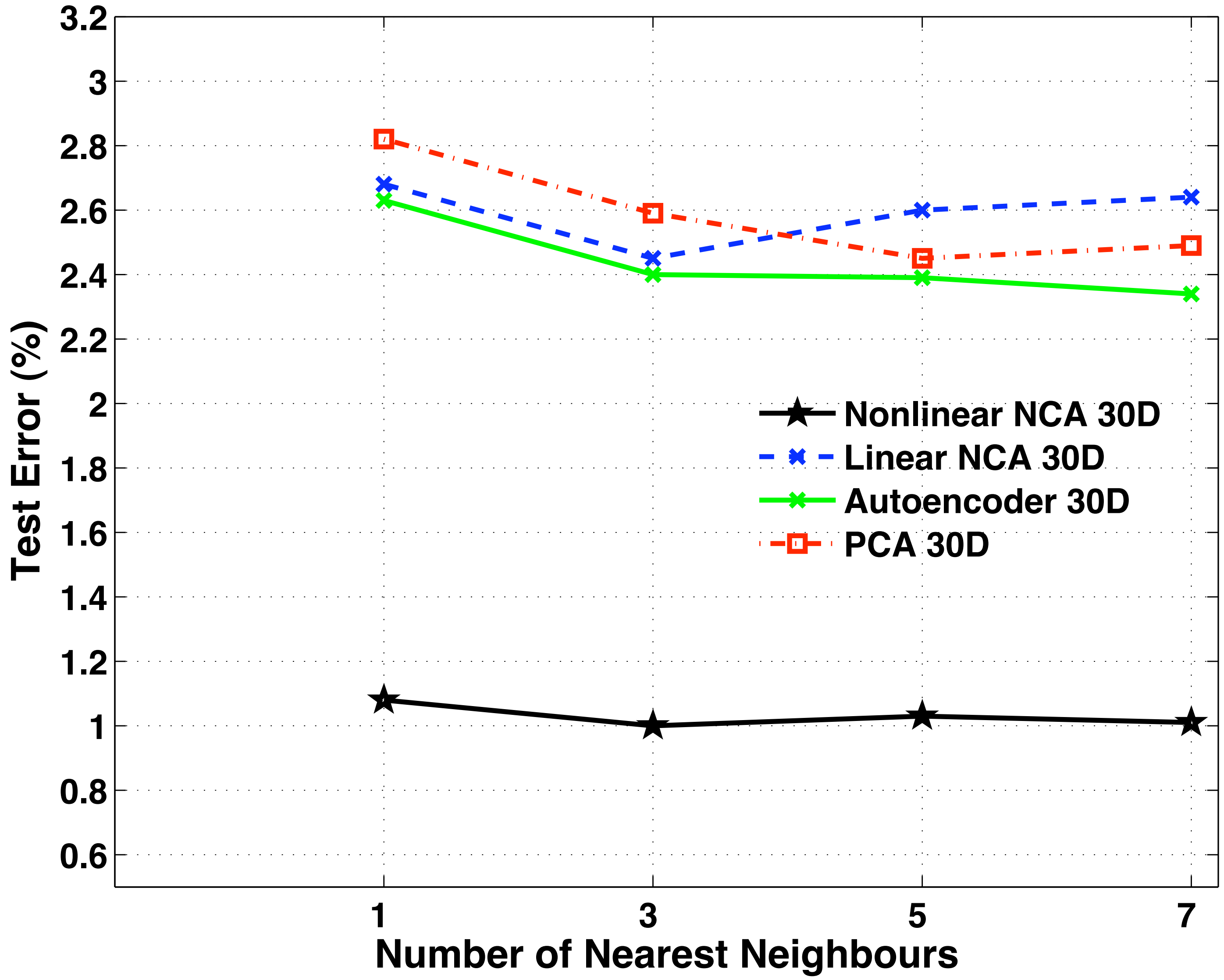
où $d_{ab} = \| f(\mathbf{x}^a|W) - f(\mathbf{x}^b|W) \|^2$

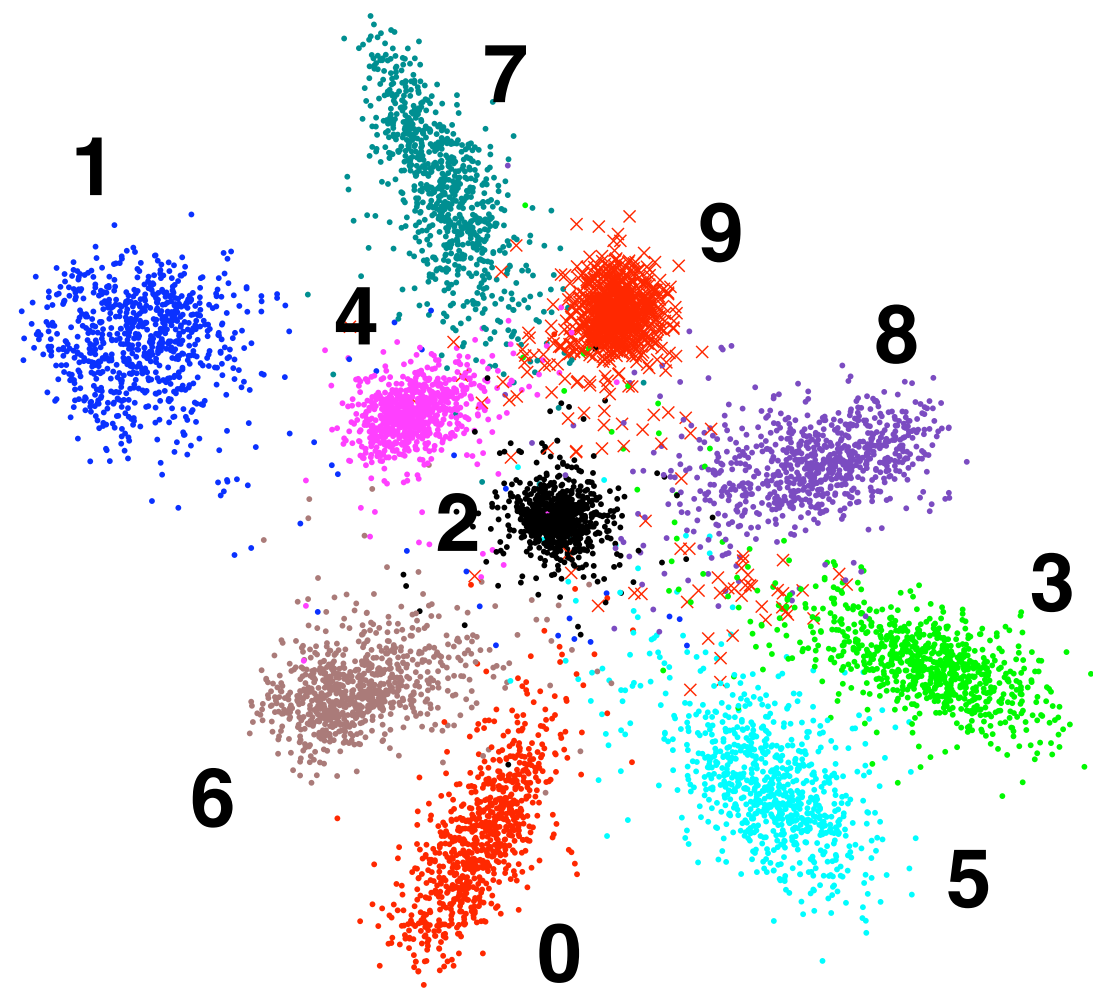
- On optimise les paramètres de $f(\cdot|W)$ par descente de gradient

Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure

(Salakhutdinov and Hinton, AISTATS 2007)

Résultats: classification (MNIST, $\lambda = 1$)

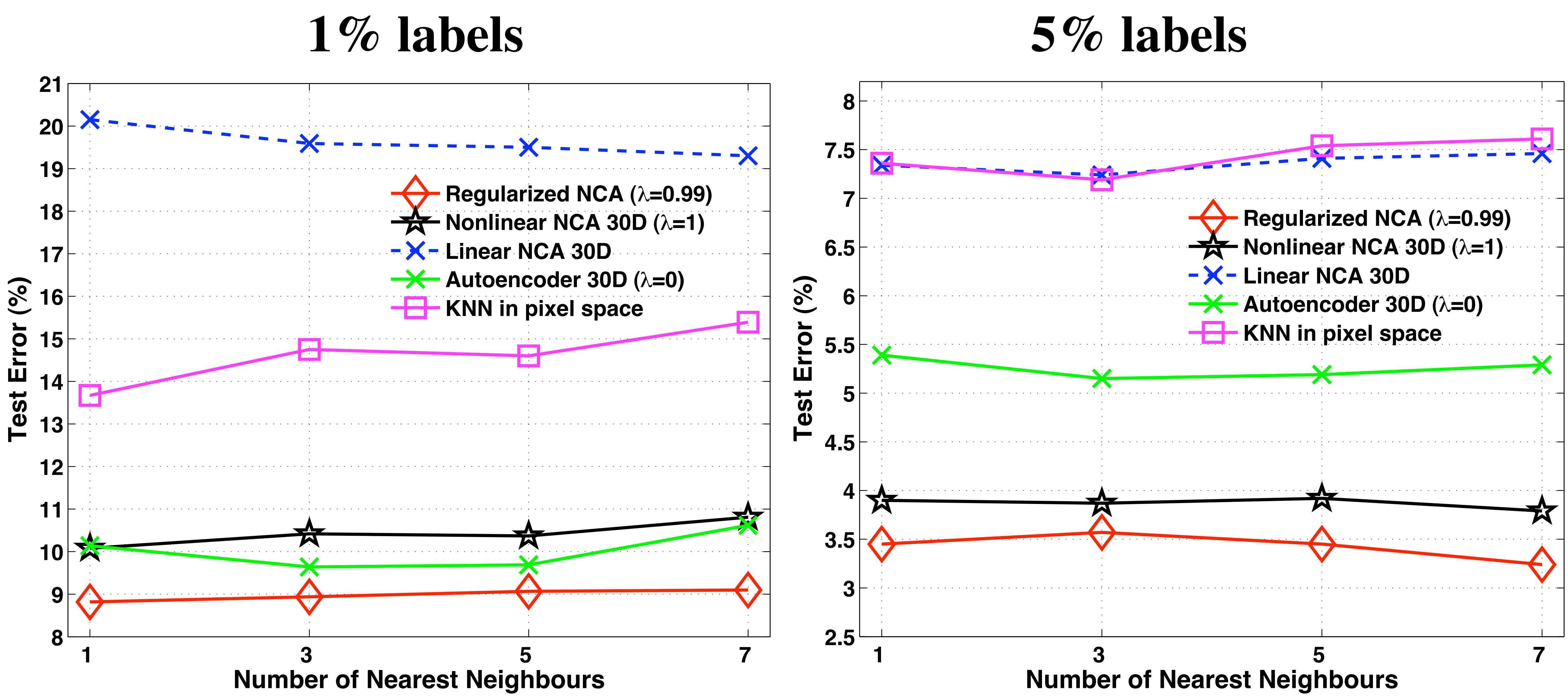




Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure

(Salakhutdinov and Hinton, AISTATS 2007)

Résultats: classification (MNIST, λ varie)



Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure

(Salakhutdinov and Hinton, AISTATS 2007)

Résultats: séparer l'information de classe

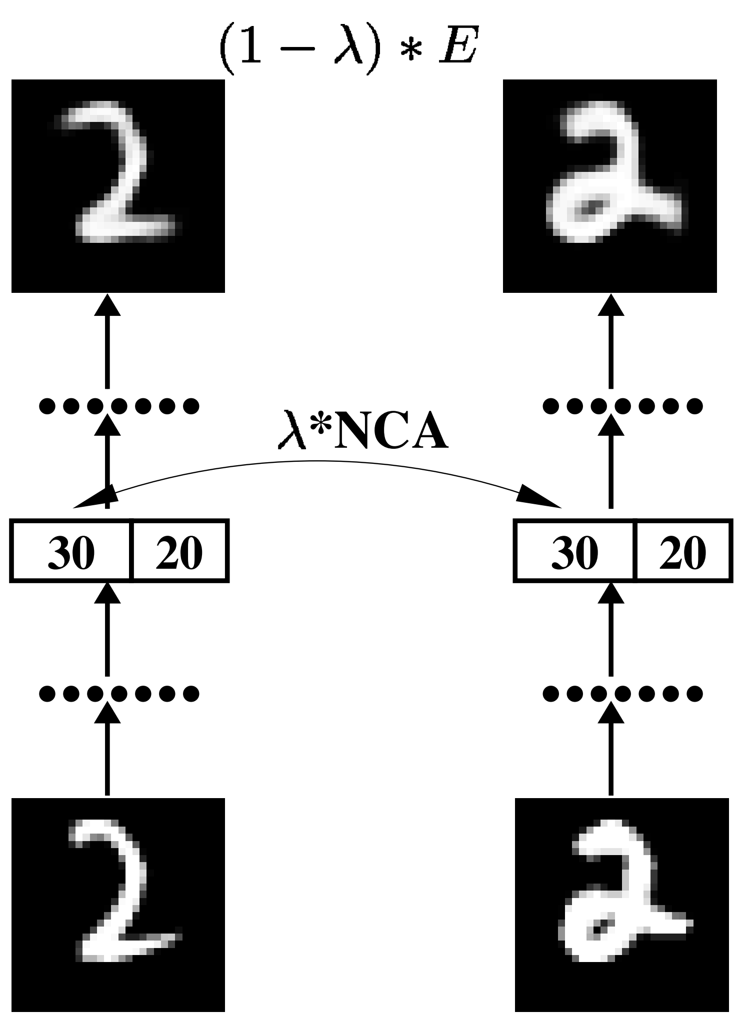


Figure 6: Left panel: The NCA objective function is only applied to the first 30 code units, but all 50 units are used for image reconstruction. Right panel: The top row shows the reconstructed images as we vary the activation of code unit 25 from 1 to -23 with a stepsize of 4. The bottom row shows the reconstructed images as we vary code unit 42 from 1 to -23.

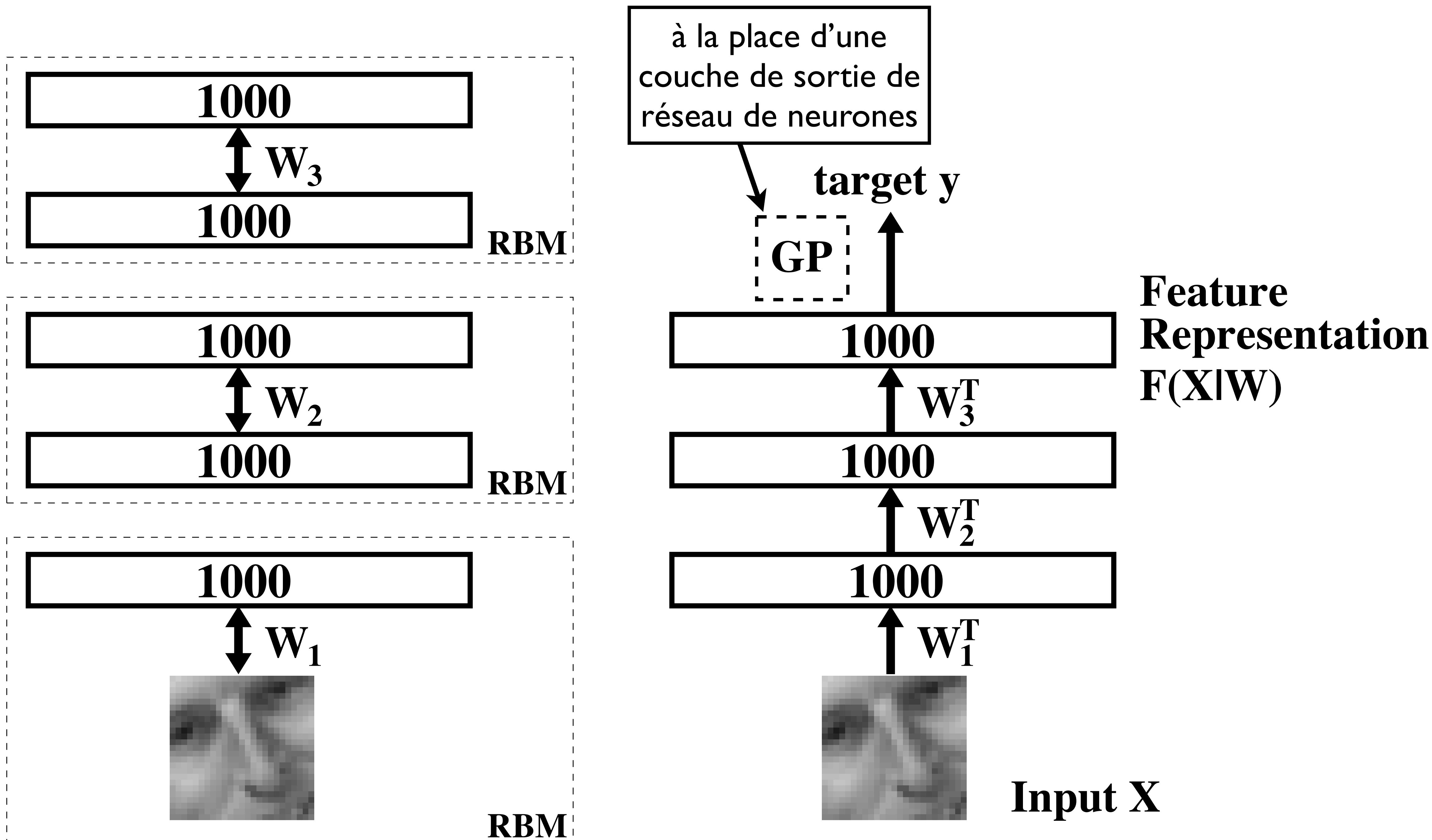
Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes

(Salakhutdinov and Hinton, NIPS 2008)

- Peut-on adapter la représentation à une tâche de régression?
- Dans ce papier, on l'adapte à un régresseur par processus Gaussien (GP)

Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes

(Salakhutdinov and Hinton, NIPS 2008)



Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes

(Salakhutdinov and Hinton, NIPS 2008)

- (Très court) rappel des GPs

GP $p(\mathbf{f}|\mathbf{X}_l) = \mathcal{N}(\mathbf{f}|0, K)$ **où** $\mathbf{f} = [f(x_1), ..., f(x_n)]^T$

matrice covariance $K_{ij} = \alpha \exp\left(-\frac{1}{2\beta}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\right)$

vraisemblance à maximiser

$$L = \log p(\mathbf{y}|\mathbf{X}_l) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|K + \sigma^2 I| - \frac{1}{2}\mathbf{y}^T(K + \sigma^2 I)^{-1}\mathbf{y}$$

Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes

(Salakhutdinov and Hinton, NIPS 2008)

- Apprendre la matrice de covariance

$$K_{ij} = \alpha \exp\big(- \frac{1}{2\beta} ||F(\mathbf{x}_i|W) - F(\mathbf{x}_j|W)||^2 \big)$$

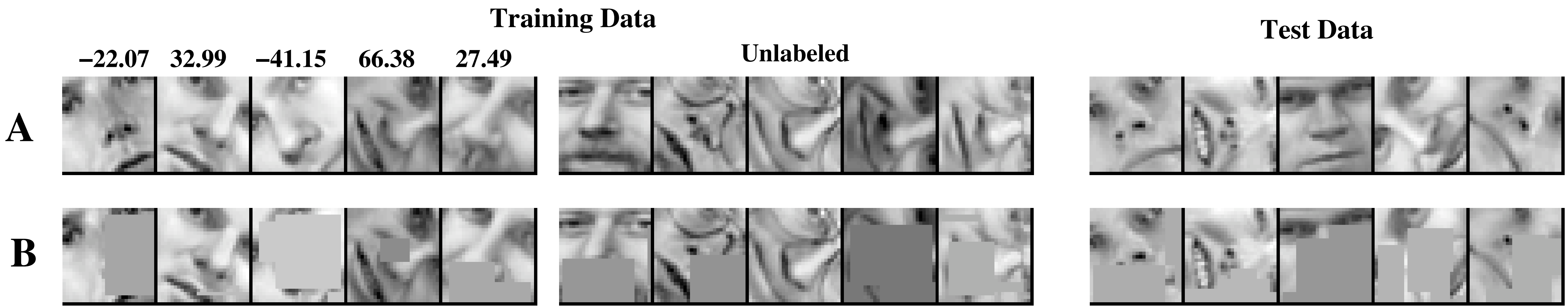
- Procède par descente de gradient

$$\frac{\partial L}{\partial K_y} = \frac{1}{2}\big(K_y^{-1}\mathbf{y}\mathbf{y}^T K_y^{-1} - K_y^{-1}\big) \qquad \frac{\partial L}{W} = \frac{\partial L}{\partial K_y}\frac{\partial K_y}{\partial F(\mathbf{x}|W)}\frac{\partial F(\mathbf{x}|W)}{\partial W}$$

Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes

(Salakhutdinov and Hinton, NIPS 2008)

Résultats: régression

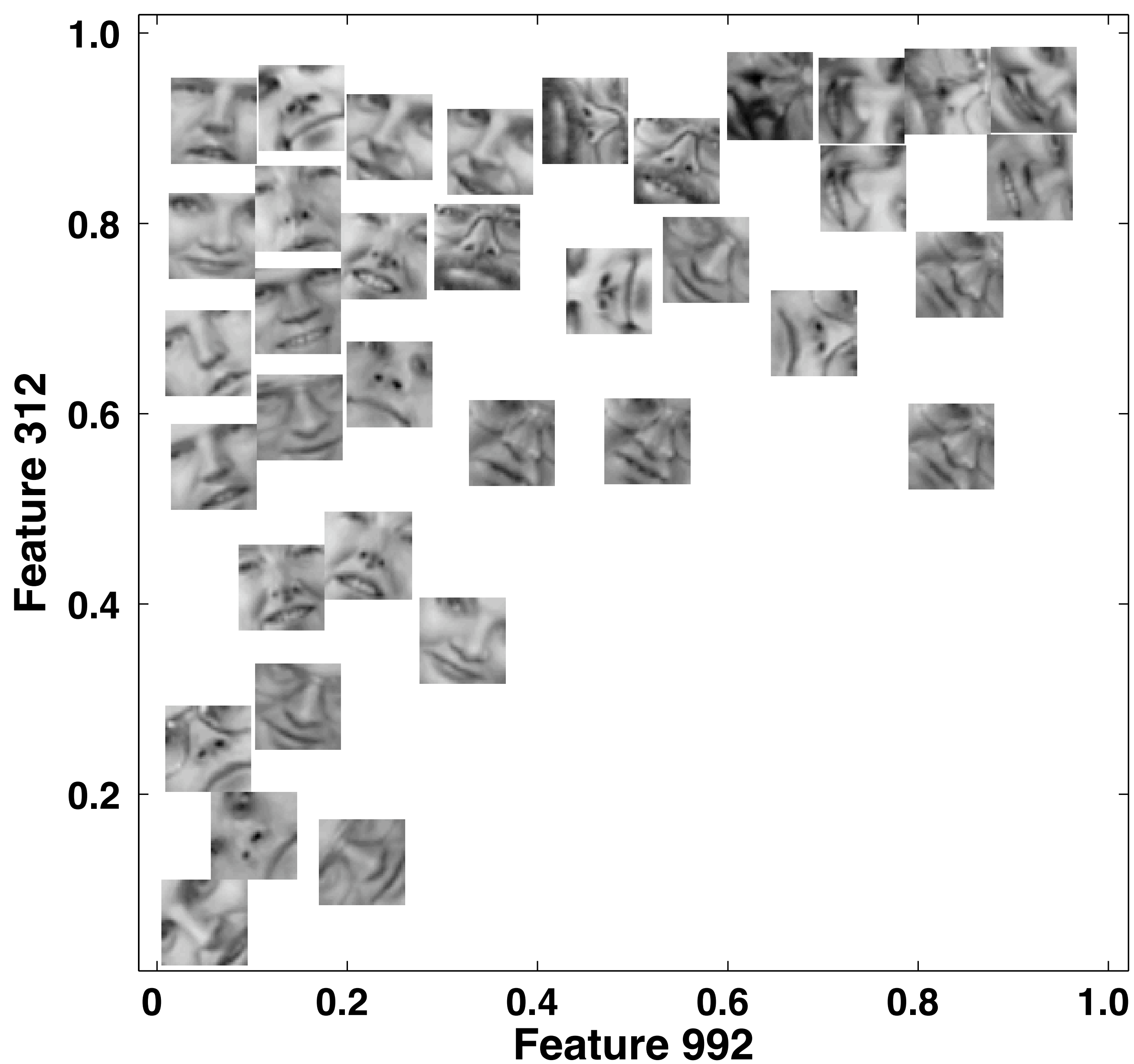


	Training labels	**GPstandard**		**GP-DBNgreedy**		**GP-DBNfine**		**GPpca**	
		Sph.	ARD	Sph.	ARD	Sph.	ARD	Sph.	ARD
A	100	22.24	28.57	17.94	18.37	15.28	**15.01**	18.13 (10)	16.47 (10)
	500	17.25	18.16	12.71	8.96	7.25	**6.84**	14.75 (20)	10.53 (80)
	1000	16.33	16.36	11.22	8.77	6.42	**6.31**	14.86 (20)	10.00 (160)
B	100	26.94	28.32	23.15	19.42	19.75	**18.59**	25.91 (10)	19.27 (20)
	500	20.20	21.06	15.16	11.01	10.56	**10.12**	17.67 (10)	14.11 (20)
	1000	19.20	17.98	14.15	10.43	**9.13**	9.23	16.26 (10)	11.55 (80)

Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes

(Salakhutdinov and Hinton, NIPS 2008)

Résultats: visualisation



Y a pas juste les réseaux de neurones dans la vie...

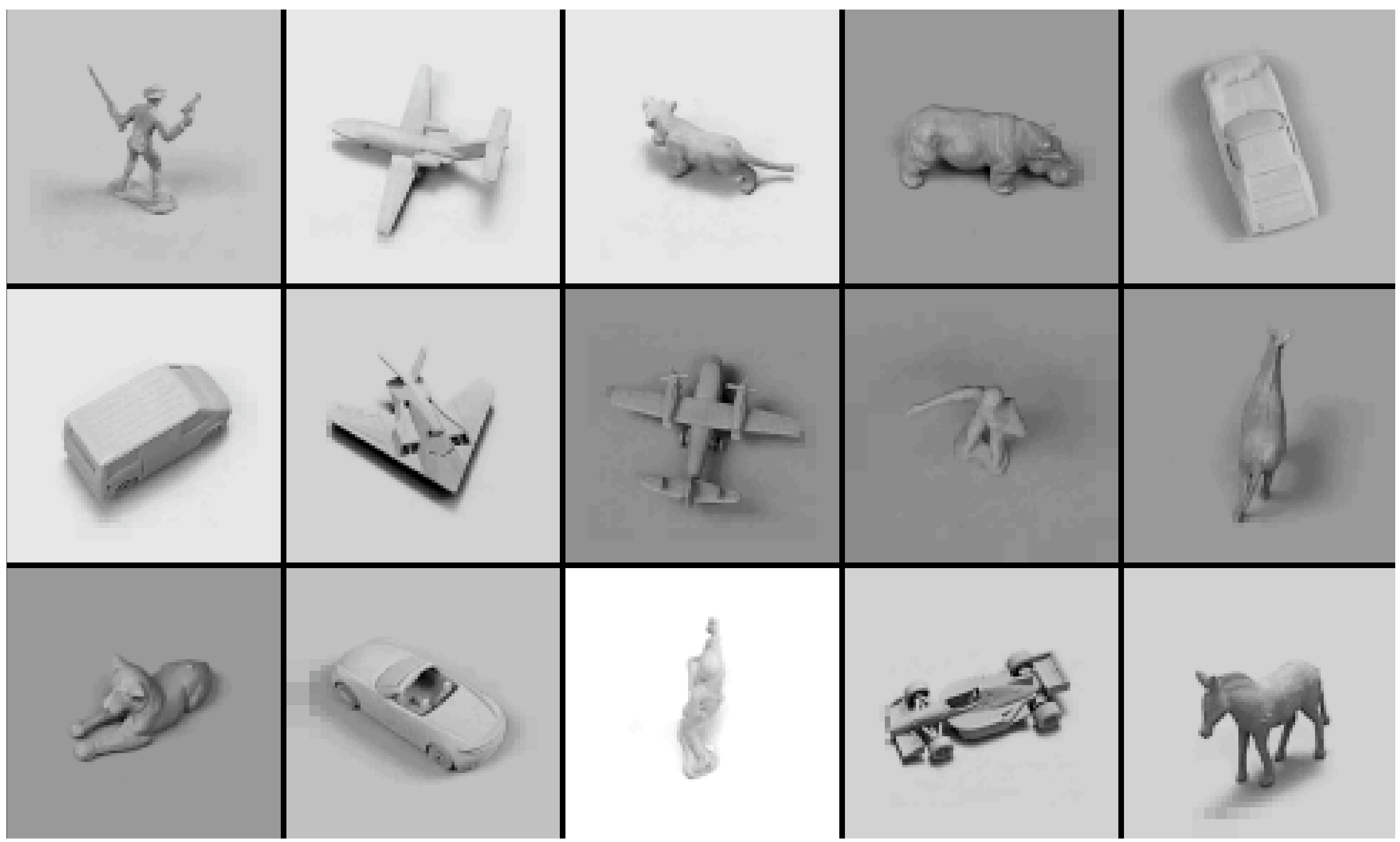
- Adaptation à KNN et GP particulièrement utile pour des petits jeux de données (avec beaucoup de données non-étiquetées)
- Y a-t-il d'autres algorithmes pour lesquels un critère d'entraînement pour réseau profond pourrat être dérivé?

! Piste de recherche !

Implicit Mixtures of Restricted Boltzmann Machines

(Nair and Hinton, NIPS 2009)

- Classification d'objets 3D

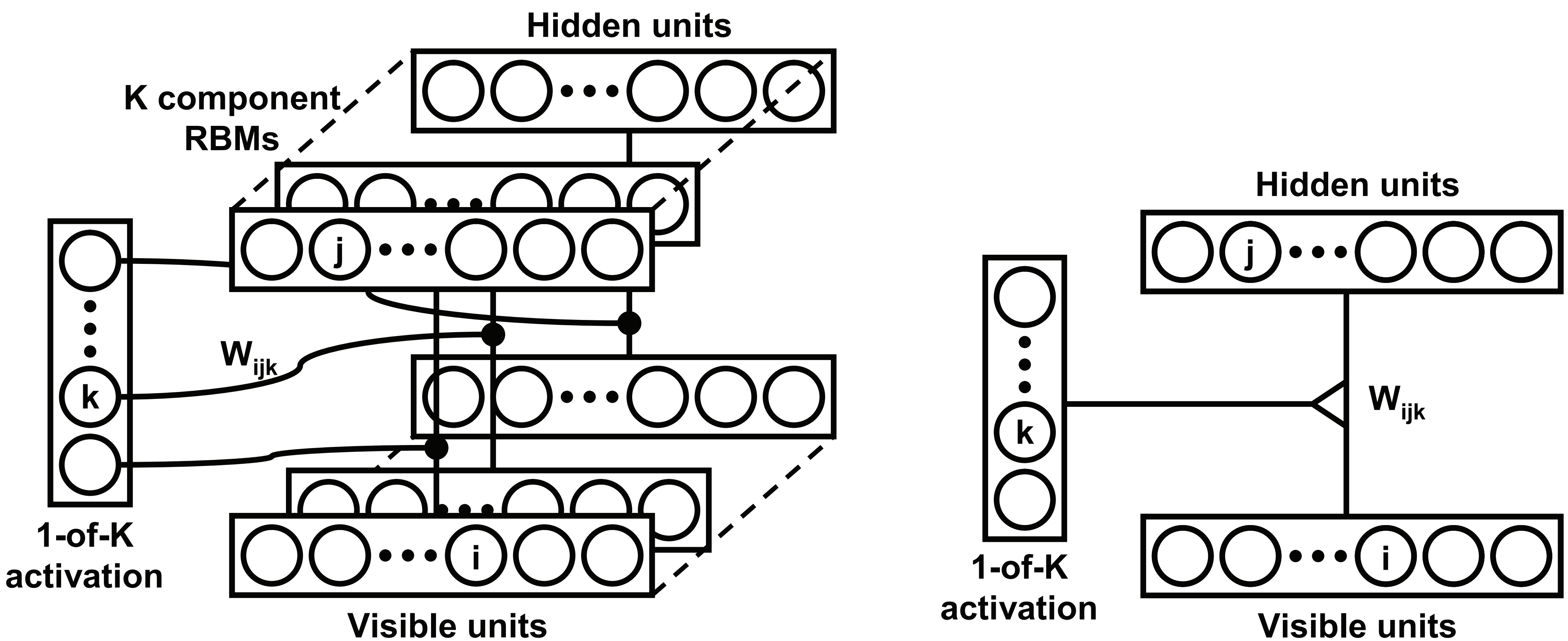


- On a besoin d'un module RBM plus puissant

Implicit Mixtures of Restricted Boltzmann Machines

(Nair and Hinton, NIPS 2009)

- Modèle RBM avec interaction d’ordre 3



$$E(\mathbf{v},\mathbf{h},\mathbf{z}) = -\sum_{i,j,k} W^{I}_{ijk} v_i h_j z_k$$

Implicit Mixtures of Restricted Boltzmann Machines

(Nair and Hinton, NIPS 2009)

- Correspond à un mélange de RBMs, mais dont les probabilités des composantes sont “implicites

distribution jointe

$$P(\mathbf{v},\mathbf{h},\mathbf{z}) = \frac{\exp(-E(\mathbf{v},\mathbf{h},\mathbf{z}))}{Z_I}$$

constante de normalization

$$Z_I = \sum_{\mathbf{u},\mathbf{g},\mathbf{y}} \exp(-E(\mathbf{u},\mathbf{g},\mathbf{y}))$$

distribution des entrées

sous forme de mixture de RBMs

$$P(\mathbf{v}) = \sum_{\mathbf{h},\mathbf{z}} P(\mathbf{v},\mathbf{h},\mathbf{z}) = \sum_{k=1}^{K} \sum_{\mathbf{h}} P(\mathbf{v},\mathbf{h}|z_k=1)P(z_k=1)$$

Implicit Mixtures of Restricted Boltzmann Machines

(Nair and Hinton, NIPS 2009)

- Possible d'échantillonner $\mathbf{z}$ étant donné $\mathbf{v}$

$$P(z_k = 1|\mathbf{v}) = \frac{\exp(-F(\mathbf{v}, z_k = 1))}{\sum_l \exp(-F(\mathbf{v}, z_l = 1))}$$

$$F(\mathbf{v}, z_k = 1) = -\sum_j \log(1 + \exp(\sum_i W^I_{ijk} v_i))$$

- Étant donné $\mathbf{z}$ et $\mathbf{v}$, on peut échantillonner $\mathbf{h}$

$$P(h_j = 1|\mathbf{v}, z_k = 1) = \frac{1}{1 + \exp(-\sum_i W^I_{ijk} v_i)}$$

Implicit Mixtures of Restricted Boltzmann Machines

(Nair and Hinton, NIPS 2009)

Contrastive divergence learning: Below is a summary of the steps in the CD learning for the implicit mixture model.

1. For a training vector $\mathbf{v}_+$, pick a component RBM by sampling the responsibilities $P(z_k = 1|\mathbf{v}_+)$. Let l be the index of the selected RBM.
2. Sample $\mathbf{h}_+ \sim P_l(\mathbf{h}|\mathbf{v}_+)$.
3. Compute the outer product $\mathbf{D}_l^+ = \mathbf{v}_+ \mathbf{h}_+^T$.
4. Sample $\mathbf{v}_- \sim P_l(\mathbf{v}|\mathbf{h}_+)$.
5. Pick a component RBM by sampling the responsibilities $P(z_k = 1|\mathbf{v}_-)$. Let m be the index of the selected RBM.
6. Sample $\mathbf{h}_- \sim P_m(\mathbf{h}|\mathbf{v}_-)$.
7. Compute the outer product $\mathbf{D}_m^- = \mathbf{v}_- \mathbf{h}_-^T$.

stats. positives pour la l^{e} RBM

stats. négative pour la m^{e} RBM

Truc pratique

$$P(z_k = 1|\mathbf{v}) = \frac{\exp(-F(\mathbf{v}, z_k = 1)/T)}{\sum_l \exp(-F(\mathbf{v}, z_l = 1)/T)}$$

Implicit Mixtures of Restricted Boltzmann Machines

(Nair and Hinton, NIPS 2009)

Résultats: apprentissage non-supervisé

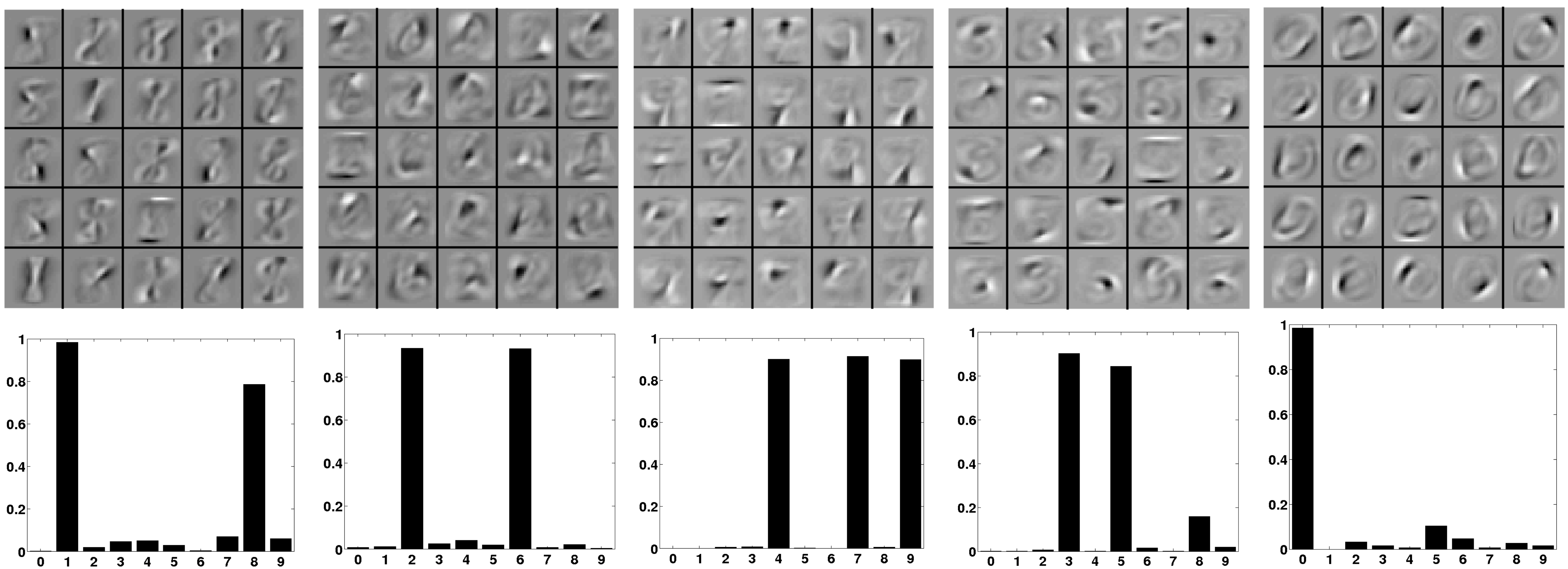


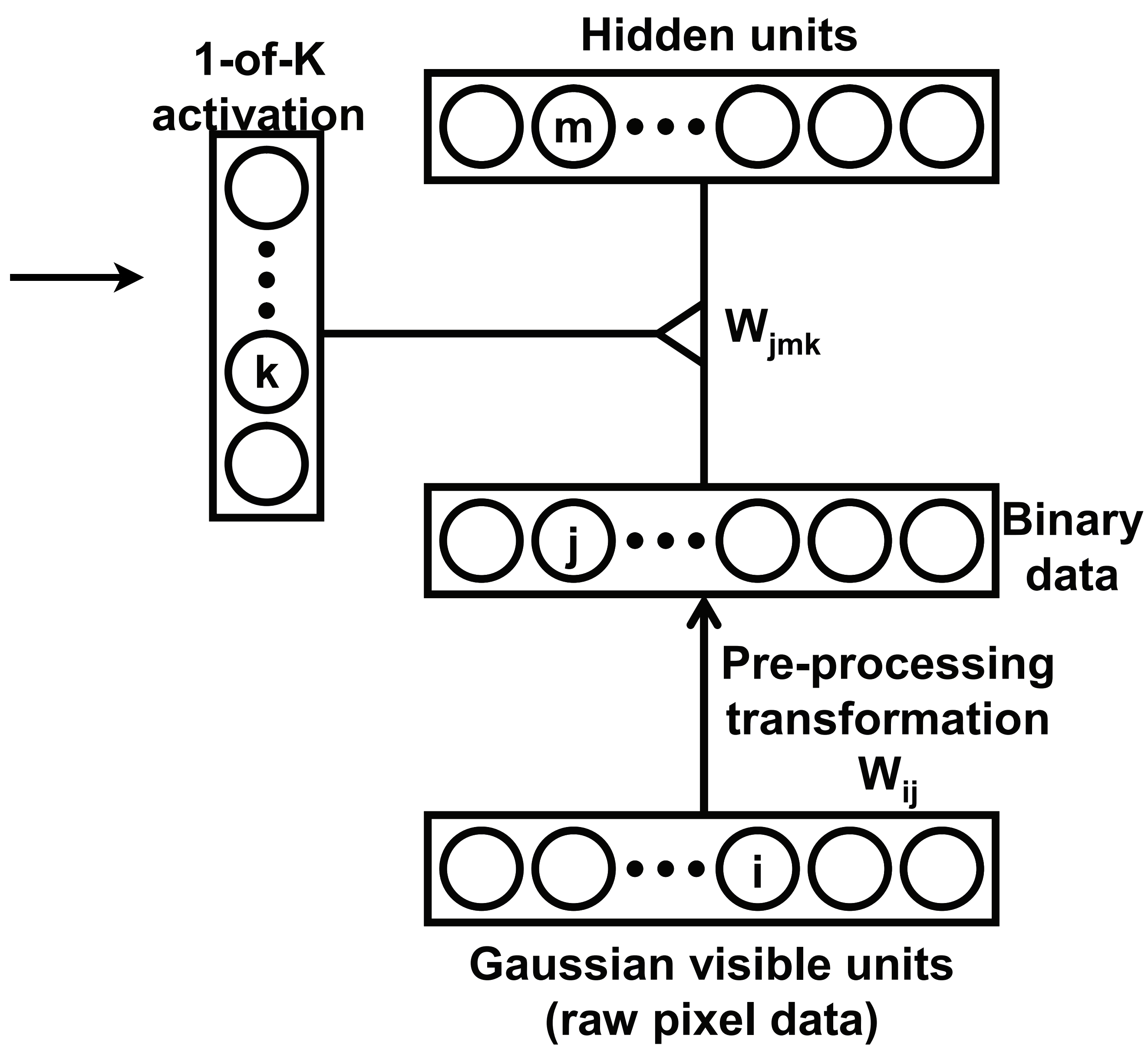
Figure 2: Features of the mixture model with five component RBMs trained on all ten classes of MNIST images.

3D Object Recognition with Deep Belief Nets

(Nair and Hinton, NIPS 2009)

- Application à la reconnaissance d'objets 3D

Étiquette (label) de l'entrée



3D Object Recognition with Deep Belief Nets

(Nair and Hinton, NIPS 2009)

Résultats: classification (NORB)

	Model	RBM with label unit	Third-order RBM
1 couche cachée →	Shallow	22.8%	20.8%
2 couches cachées →	Deep	11.9%	7.6%

Learning algorithm	RBM with label unit	Third-order RBM
CD	11.9%	7.6%
Hybrid	10.4%	6.5%

utilise $\log p(\mathbf{v}|\mathbf{l}) + \lambda \log p(\mathbf{l}|\mathbf{v})$ comme critère pour la couche du haut

3D Object Recognition with Deep Belief Nets

(Nair and Hinton, NIPS 2009)

Résultats: classification (NORB)

Top-level model (hyrbid learning only)	**Unlabeled jitter for pre-training lower layer?**	**Unlabeled jitter at the top-level?**	**Error**
RBM with label unit	No	No	10.4%
	Yes	No	9.0%
Third-order model	No	No	6.5%
	Yes	No	5.3%
	Yes	Yes	5.2%

3D Object Recognition with Deep Belief Nets

(Nair and Hinton, NIPS 2009)

Résultats: classification (NORB)

Initialization of top-level parameters	Use jittered images as labeled?	Error
Random	No	13.4%
Random	Yes	7.1%
Model with 5.2% error from table 3	Yes	5.0%

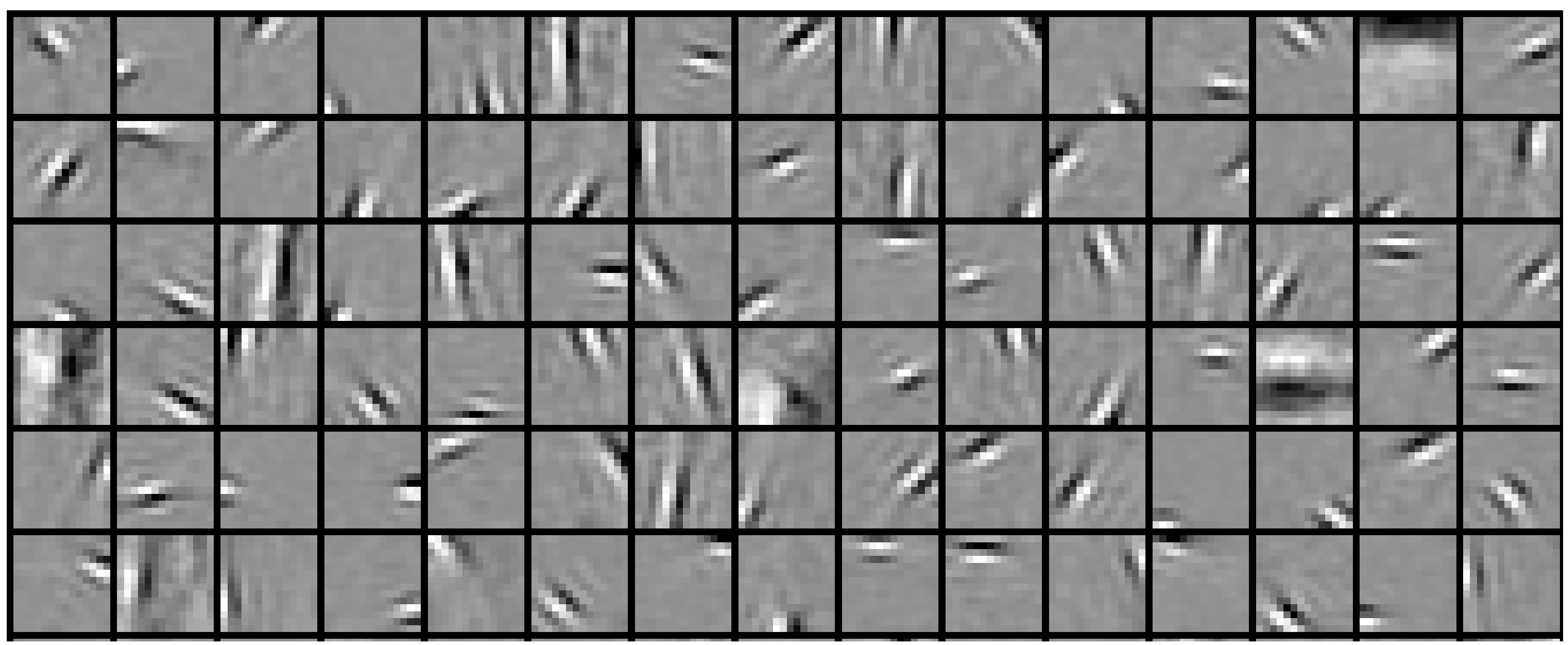
raffinement de la deuxième couche →

Réseau convolution: 6.0%

Sparse deep belief net model for visual area V2

(Lee, Ekanadham and Ng, NIPS 2009)

- "Sparsité" et la connexion avec la neuroscience



- Comment obtenir de tels filtres!

Sparse deep belief net model for visual area V2

(Lee, Ekanadham and Ng, NIPS 2009)

- Réponse: avec de la "sparsité"

$$\text{minimize}_{\{w_{ij},c_i,b_j\}} \left(\begin{array}{l} -\sum_{l=1}^{m} \log \sum_{\mathbf{h}} P(\mathbf{v}^{(l)}, \mathbf{h}^{(l)}) \\ +\lambda \sum_{j=1}^{n} \left| p - \frac{1}{m} \sum_{l=1}^{m} \mathbb{E}[h_j^{(l)} | \mathbf{v}^{(l)}] \right|^2 \end{array} \right)$$

Algorithm 1 Sparse RBM learning algorithm

1. Update the parameters using contrastive divergence learning rule. More specifically,

$$\begin{aligned} w_{ij} &:= w_{ij} + \alpha(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{recon}}) \\ c_i &:= c_i + \alpha(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{recon}}) \\ b_j &:= b_j + \alpha(\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{recon}}), \end{aligned}$$

where α is a learning rate, and $\langle \cdot \rangle_{\text{recon}}$ is an expectation over the reconstruction data, estimated using one iteration of Gibbs sampling (as in Equations 2,3).

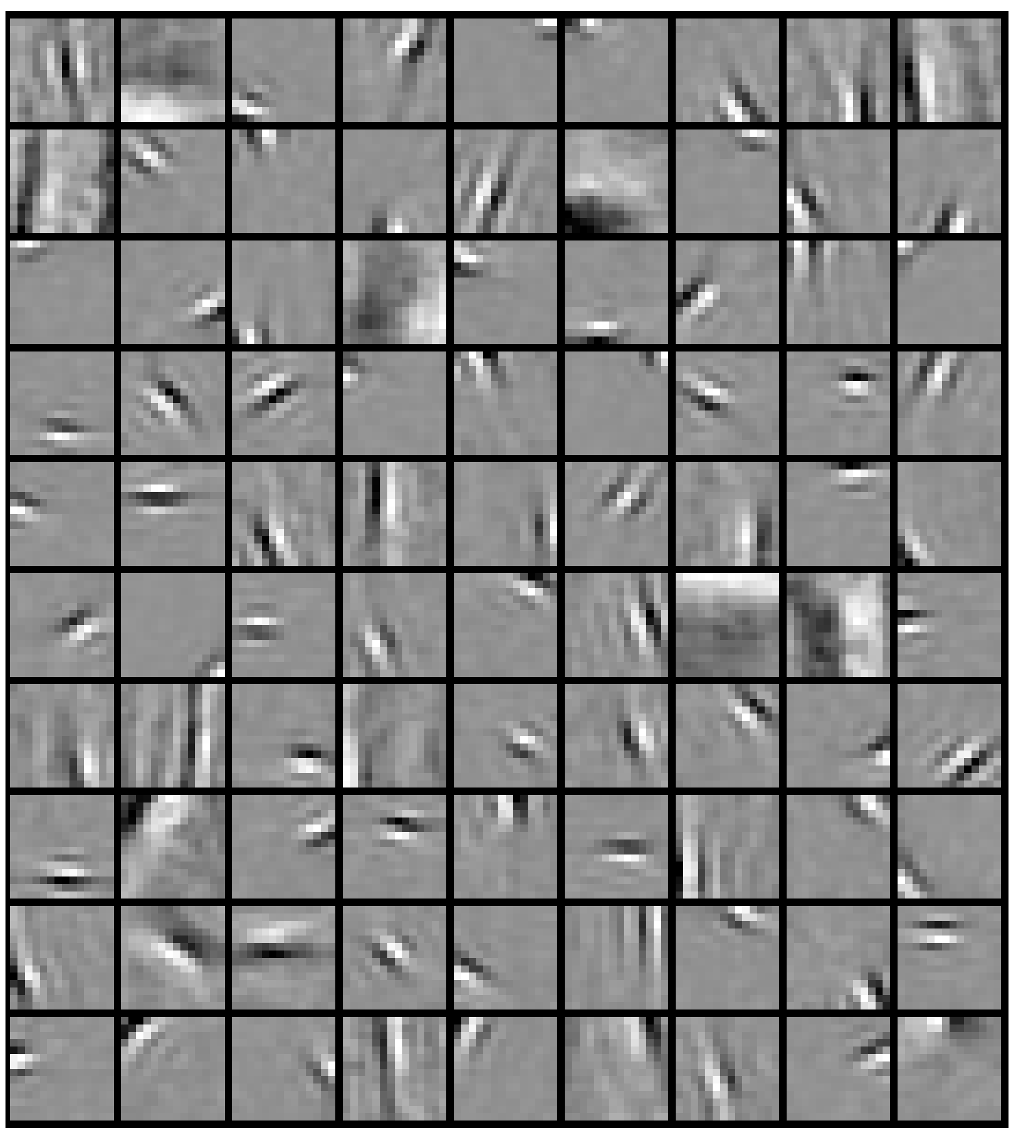
2. Update the parameters using the gradient of the regularization term.
3. Repeat Steps 1 and 2 until convergence.

Sparse deep belief net model for visual area V2

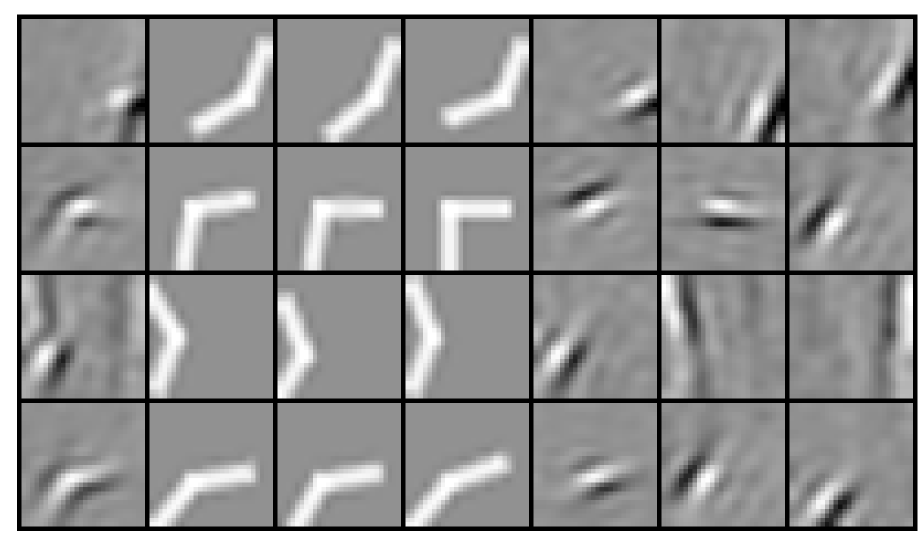
(Lee, Ekanadham and Ng, NIPS 2009)

Résultats: filtres (images naturelles)

Première couche (V1)



Deuxième couche (V2)

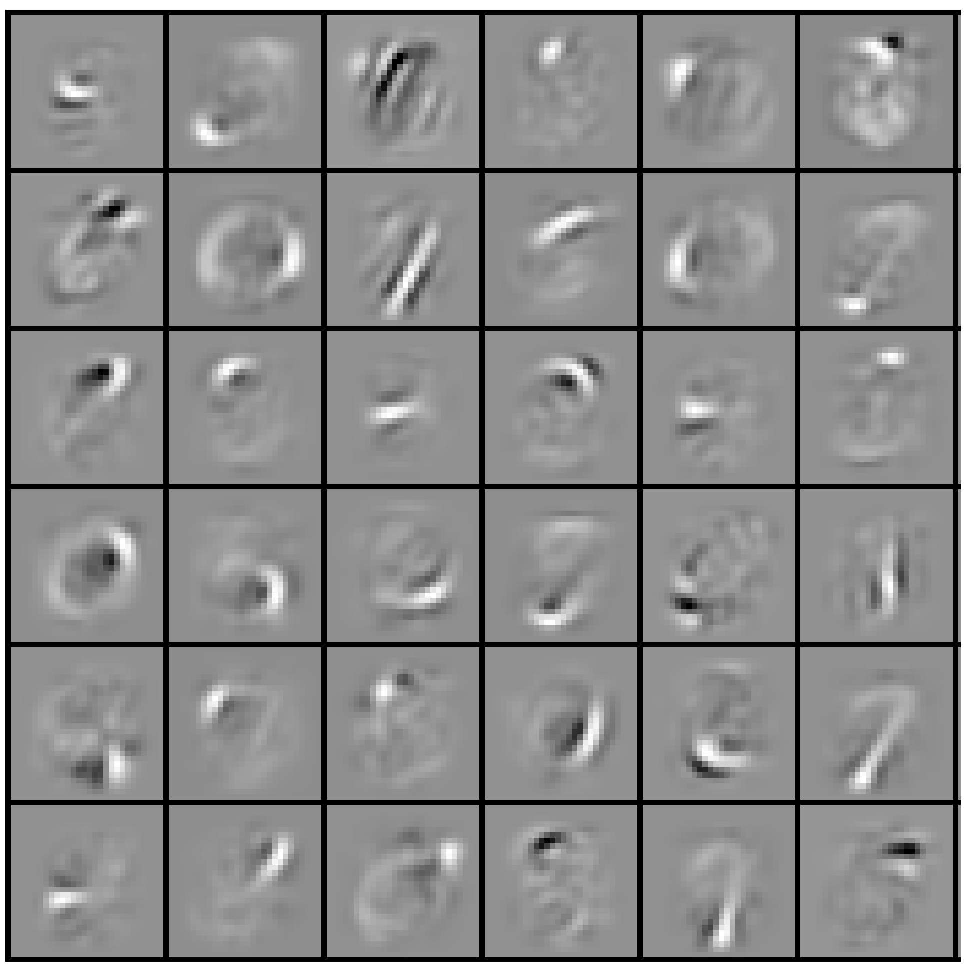


Sparse deep belief net model for visual area V2

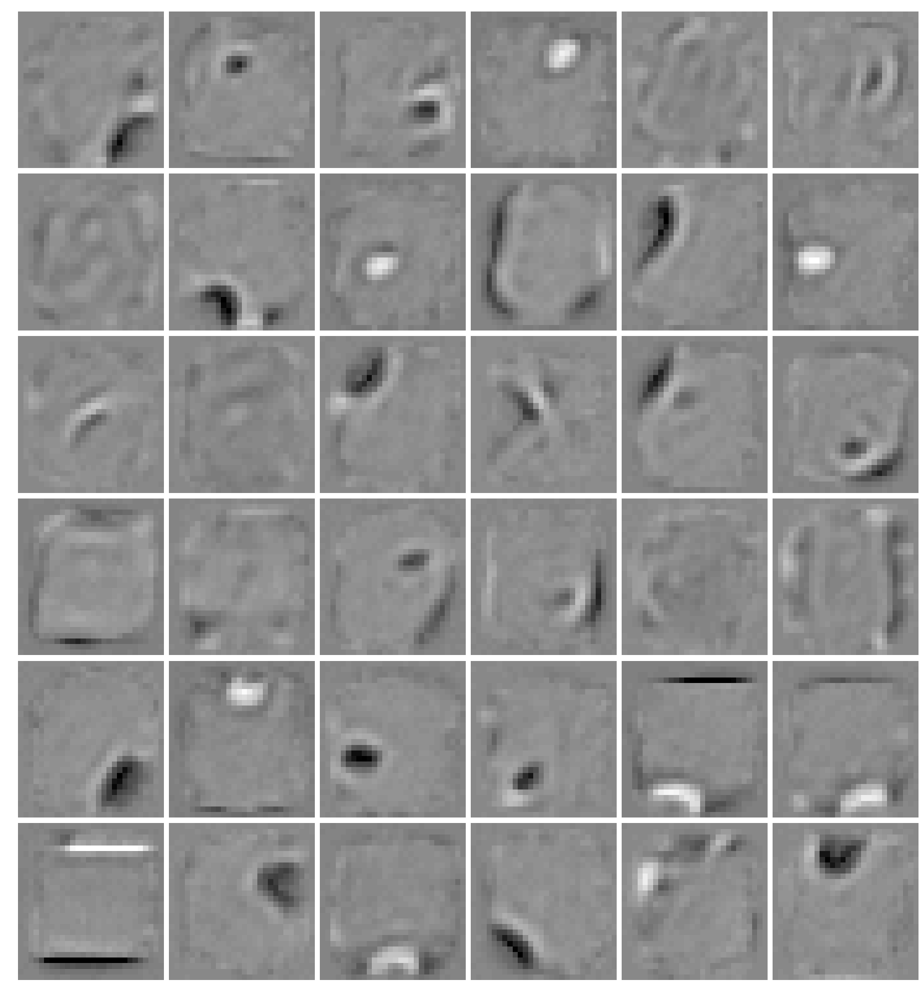
(Lee, Ekanadham and Ng, NIPS 2009)

Résultats: filtres (MNIST)

Avec sparsité

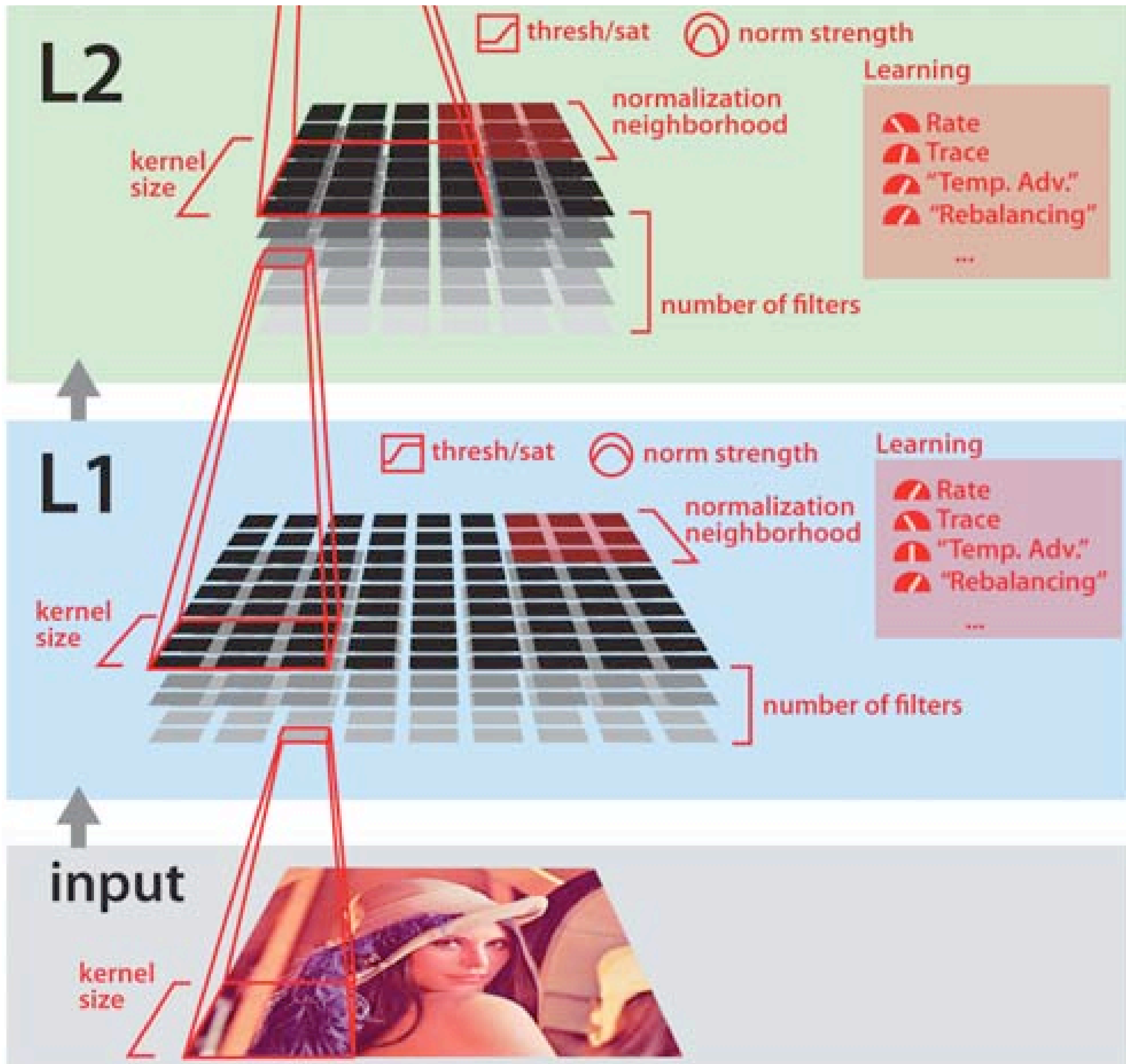


Sans sparsité



Réseaux à convolution

Rappel (en images)



Rappel (en équations)

$$\mathbf{N}^{\ell-1} \xrightarrow{\text{Filter}} \mathbf{F}^{\ell} \xrightarrow{\text{Activate}} \mathbf{A}^{\ell} \xrightarrow{\text{Pool}} \mathbf{P}^{\ell} \xrightarrow{\text{Normalize}} \mathbf{N}^{\ell}$$

$F_i^\ell = N^{\ell-1} \otimes \Phi_i^\ell \quad \longleftarrow$ convolution (linéaire)

$\mathbf{A}^\ell = \mathbf{Activate}(\mathbf{F}^\ell) \quad \longleftarrow$ transformation non-linéaire (tanh, abs, etc.)

$\mathbf{P}^\ell = \mathbf{Pool}(\mathbf{A}^\ell) \quad \longleftarrow$ "max pooling", "average pooling"

$\mathbf{N}^\ell = \mathbf{Normalize}(\mathbf{P}^\ell) \quad \longleftarrow$ "subtractive normalization" "divisive normalization"

Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition

(Kavukcuoglu, Ranzato and LeCun, techreport 2008)

Sparse Coding

$$\mathcal{L}(Y,Z;B) = \frac{1}{2}||Y - BZ||_2^2 + \lambda||Z||_1$$

Predictive Sparse Decomposition

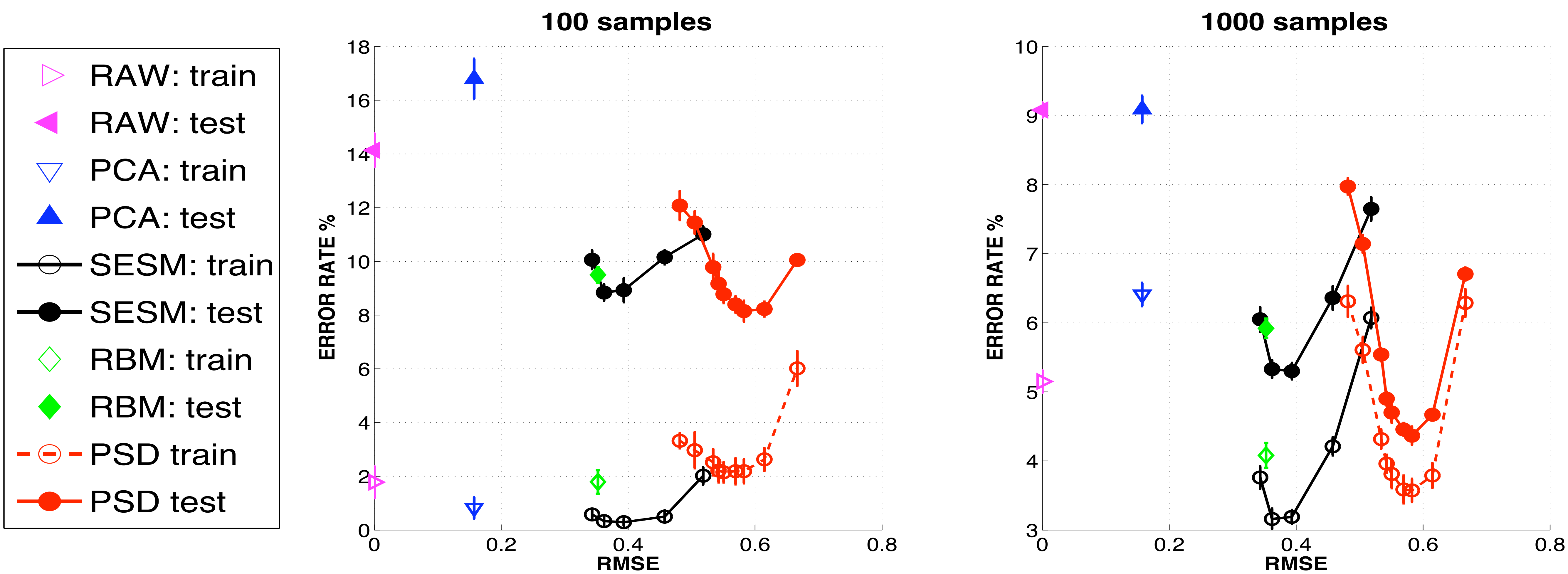
$$\mathcal{L}(Y,Z;B,P_f) = \|Y - BZ\|_2^2 + \lambda\|Z\|_1 + \alpha\|Z - F(Y;P_f)\|_2^2$$

$$F(Y;G,W,D) = G\tanh(WY + D)$$

encoder

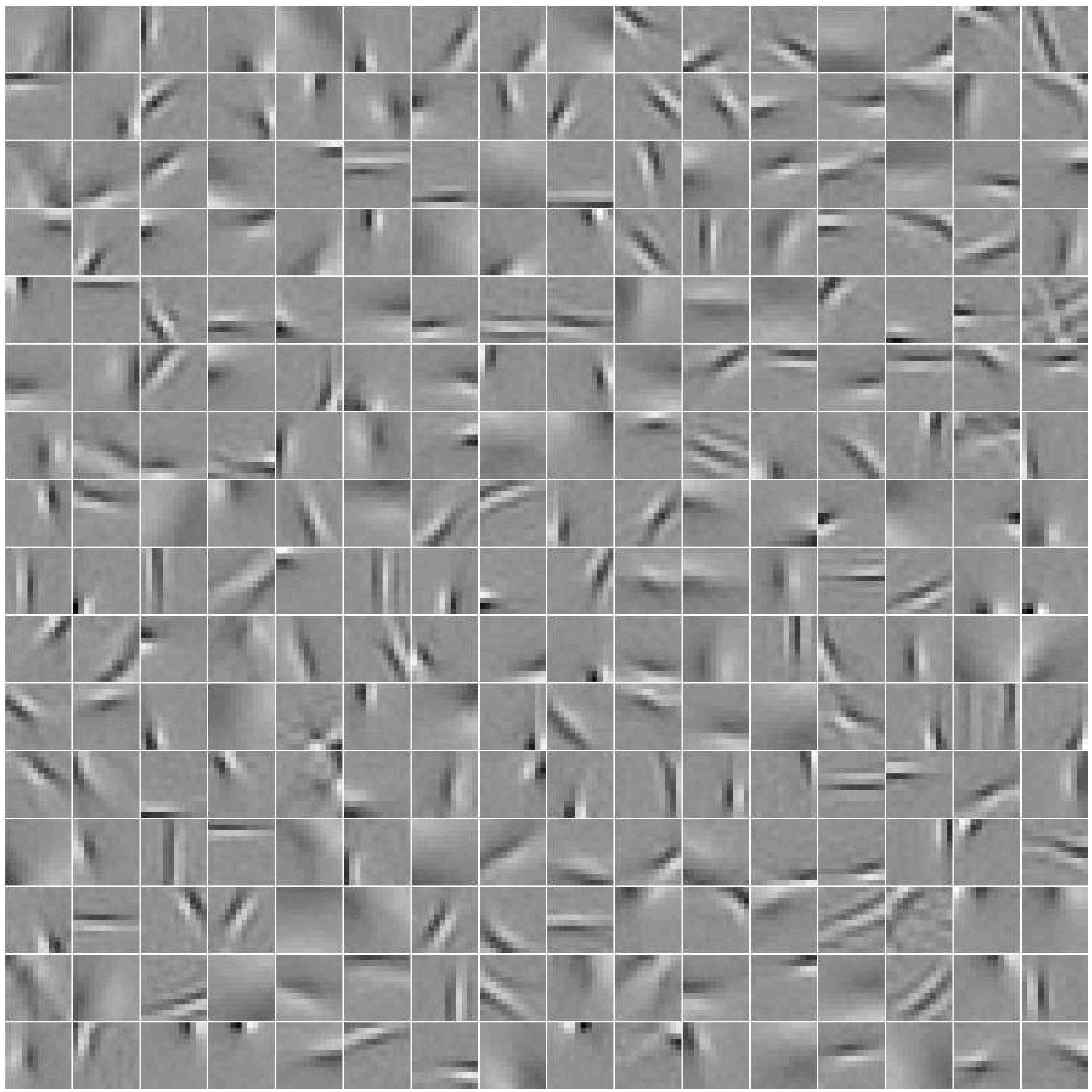
Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition

(Kavukcuoglu, Ranzato and LeCun, techreport 2008)



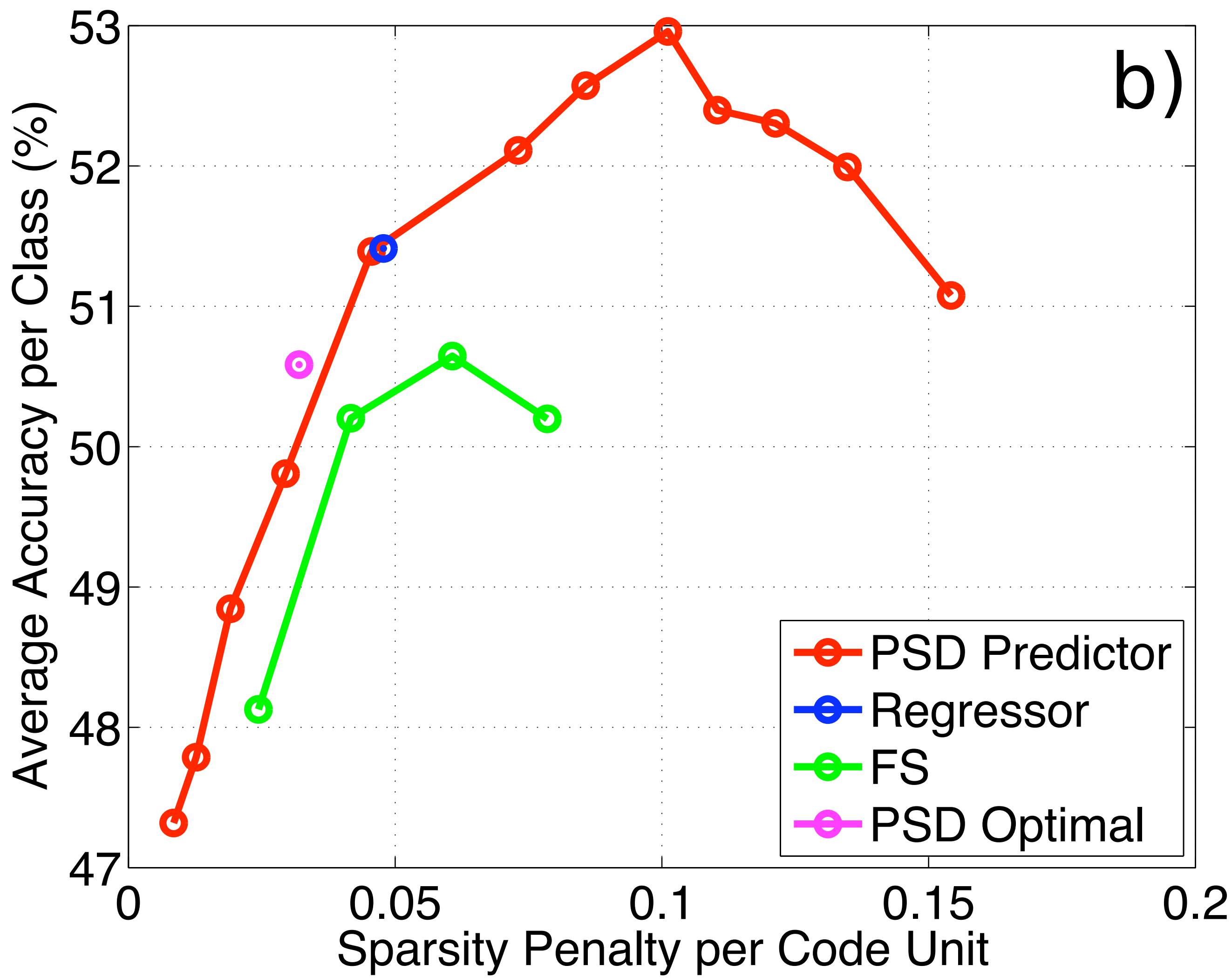
Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition

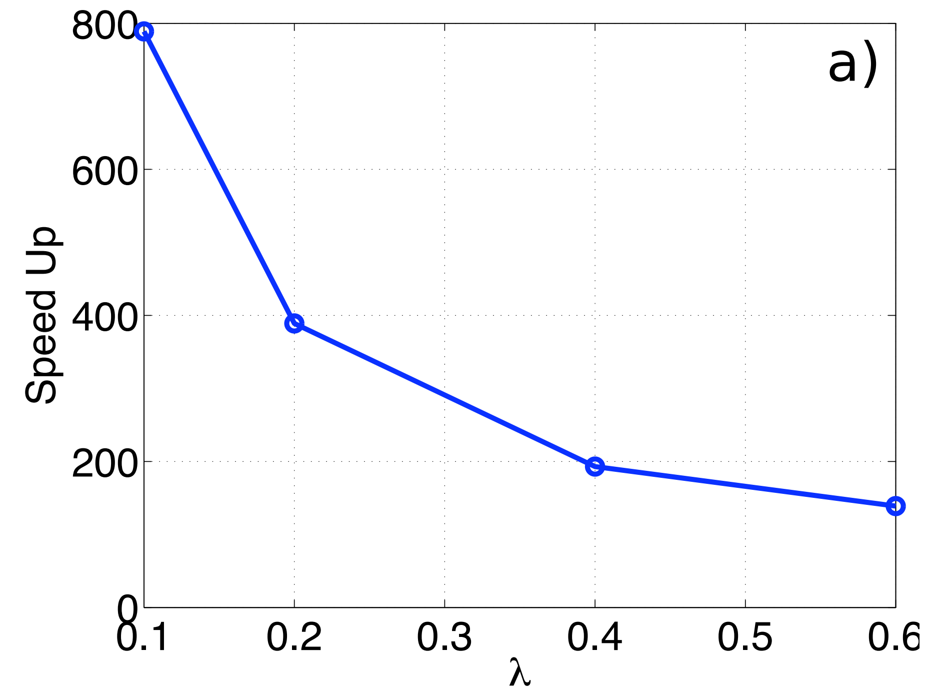
(Kavukcuoglu, Ranzato and LeCun, techreport 2008)

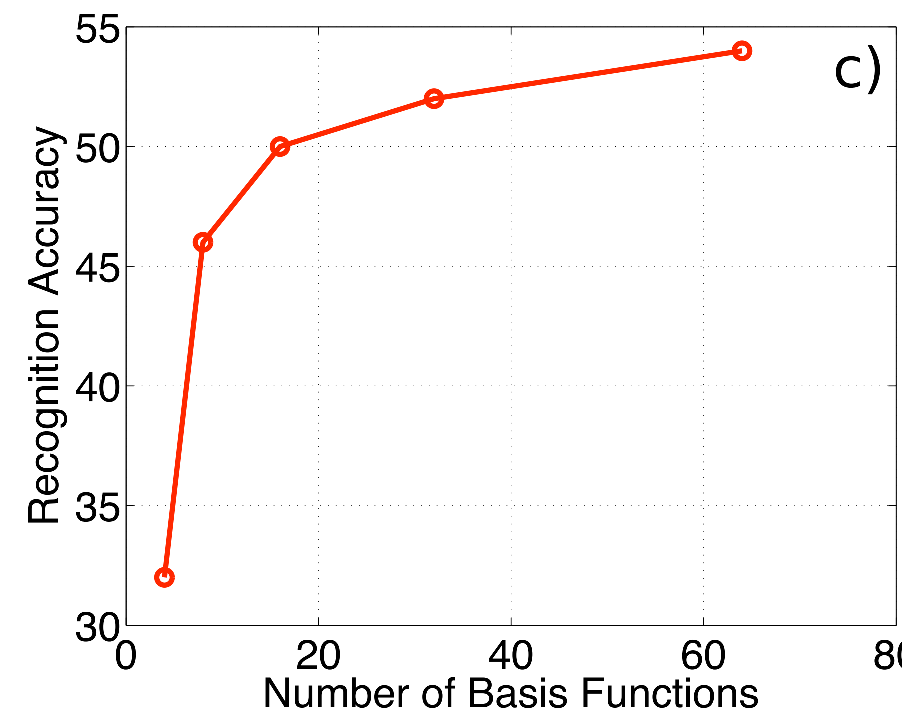


Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition

(Kavukcuoglu, Ranzato and LeCun, techreport 2008)



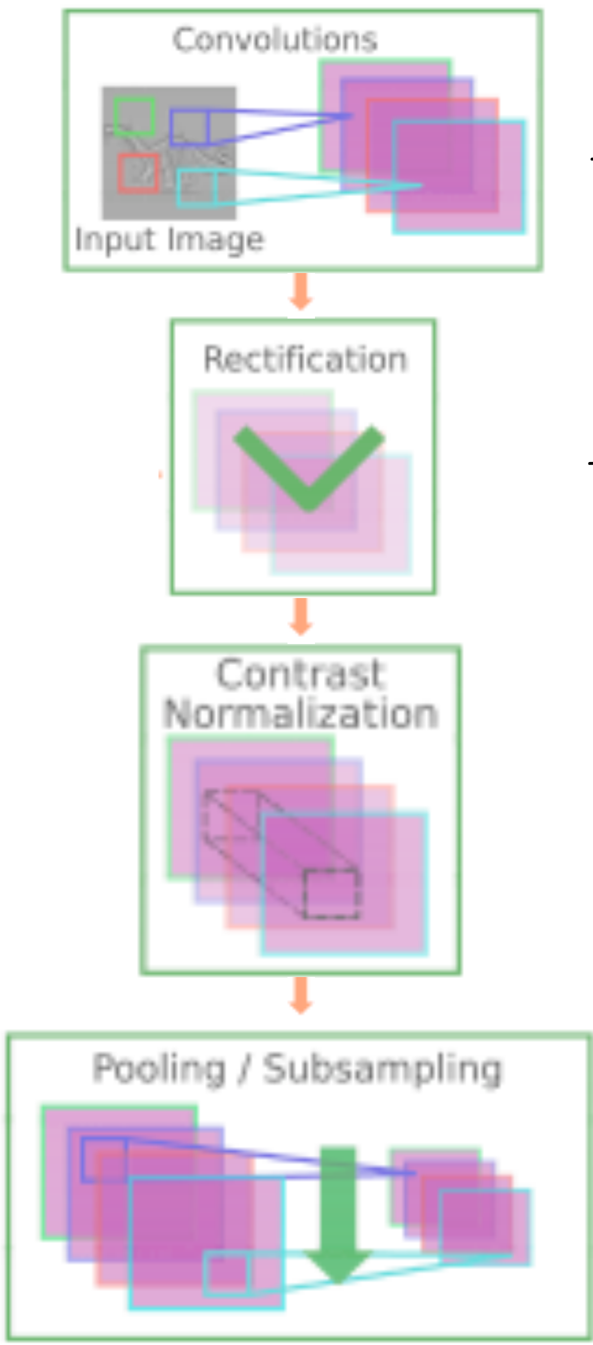




What is the Best Multi-Stage Architecture for Object Recognition?

(Jarrett, Kavukcuoglu, Ranzato and LeCun, ICCV 2009)

- Variations dans les détails de l'architecture



$$F_{CSG} \qquad y_j = g_j \tanh\Big(\sum_i k_{ij} \otimes x_i\Big)$$

$$R_{abs} \qquad y_{ijk} = |x_{ijk}|$$

$$N \begin{cases} v_{ijk} = x_{ijk} - \sum_{ipq} w_{pq} \cdot x_{i,j+p,k+q} \\ y_{ijk} = v_{ijk} / \max(c, \sigma_{jk}) \\ \sigma_{jk} = \big(\sum_{ipq} w_{pq} \cdot v^2_{i,j+p,k+q}\big)^{1/2} \end{cases}$$

$$P_A \qquad y_{ijk} = \sum_{ipq} w_{pq} \cdot x_{i,j+p,k+q}$$

ou

$$P_M \qquad y_{ijk} = \max_{ipq} x_{i,j+p,k+q}$$

What is the Best Multi-Stage Architecture for Object Recognition?

(Jarrett, Kavukcuoglu, Ranzato and LeCun, ICCV 2009)

- Variations dans l'apprentissage

R	filtres aléatoires
R^+	filtres aléatoires, puis raffinement supervisé
U	filtres appris (entraînement non-supervisé)
U^+	filtres appris, puis raffinement supervisé

What is the Best Multi-Stage Architecture for Object Recognition?

(Jarrett, Kavukcuoglu, Ranzato and LeCun, ICCV 2009)

Single Stage System: $[64.\mathrm{F}^{9\times9}_{\mathrm{CSG}} - \mathrm{R/N/P}^{5\times5}]$ - log_reg					
	$\mathrm{R_{abs} - N - P_A}$	$\mathrm{R_{abs} - P_A}$	$\mathrm{N - P_M}$	$\mathrm{N - P_A}$	$\mathrm{P_A}$
U^+	54.2%	50.0%	44.3%	18.5%	14.5%
R^+	54.8%	47.0%	38.0%	16.3%	14.3%
U	52.2%	43.3%(±1.6)	44.0%	17.2%	13.4%
R	53.3%	31.7%	32.1%	15.3%	12.1%(±2.2)
G	52.3%				
Two Stage System: $[64.\mathrm{F}^{9\times9}_{\mathrm{CSG}} - \mathrm{R/N/P}^{5\times5}] - [256.\mathrm{F}^{9\times9}_{\mathrm{CSG}} - \mathrm{R/N/P}^{4\times4}]$ - log_reg					
	$\mathrm{R_{abs} - N - P_A}$	$\mathrm{R_{abs} - P_A}$	$\mathrm{N - P_M}$	$\mathrm{N - P_A}$	$\mathrm{P_A}$
$\mathrm{U}^+\mathrm{U}^+$	65.5%	60.5%	61.0%	34.0%	32.0%
$\mathrm{R}^+\mathrm{R}^+$	64.7%	59.5%	60.0%	31.0%	29.7%
UU	63.7%	46.7%	56.0%	23.1%	9.1%
RR	62.9%	33.7%(±1.5)	37.6%(±1.9)	19.6%	8.8%
GT	55.8%				
Single Stage: $[64.\mathrm{F}^{9\times9}_{\mathrm{CSG}} - \mathrm{R_{abs}/N/P}^{5\times5}_{\mathrm{A}}]$ - PMK-SVM					
U	64.0%				
Two Stages: $[64.\mathrm{F}^{9\times9}_{\mathrm{CSG}} - \mathrm{R_{abs}/N/P}^{5\times5}_{\mathrm{A}}] - [256.\mathrm{F}^{9\times9}_{\mathrm{CSG}} - \mathrm{R_{abs}/N}]$ - PMK-SVM					
UU	52.8%				

What is the Best Multi-Stage Architecture for Object Recognition?

(Jarrett, Kavukcuoglu, Ranzato and LeCun, ICCV 2009)

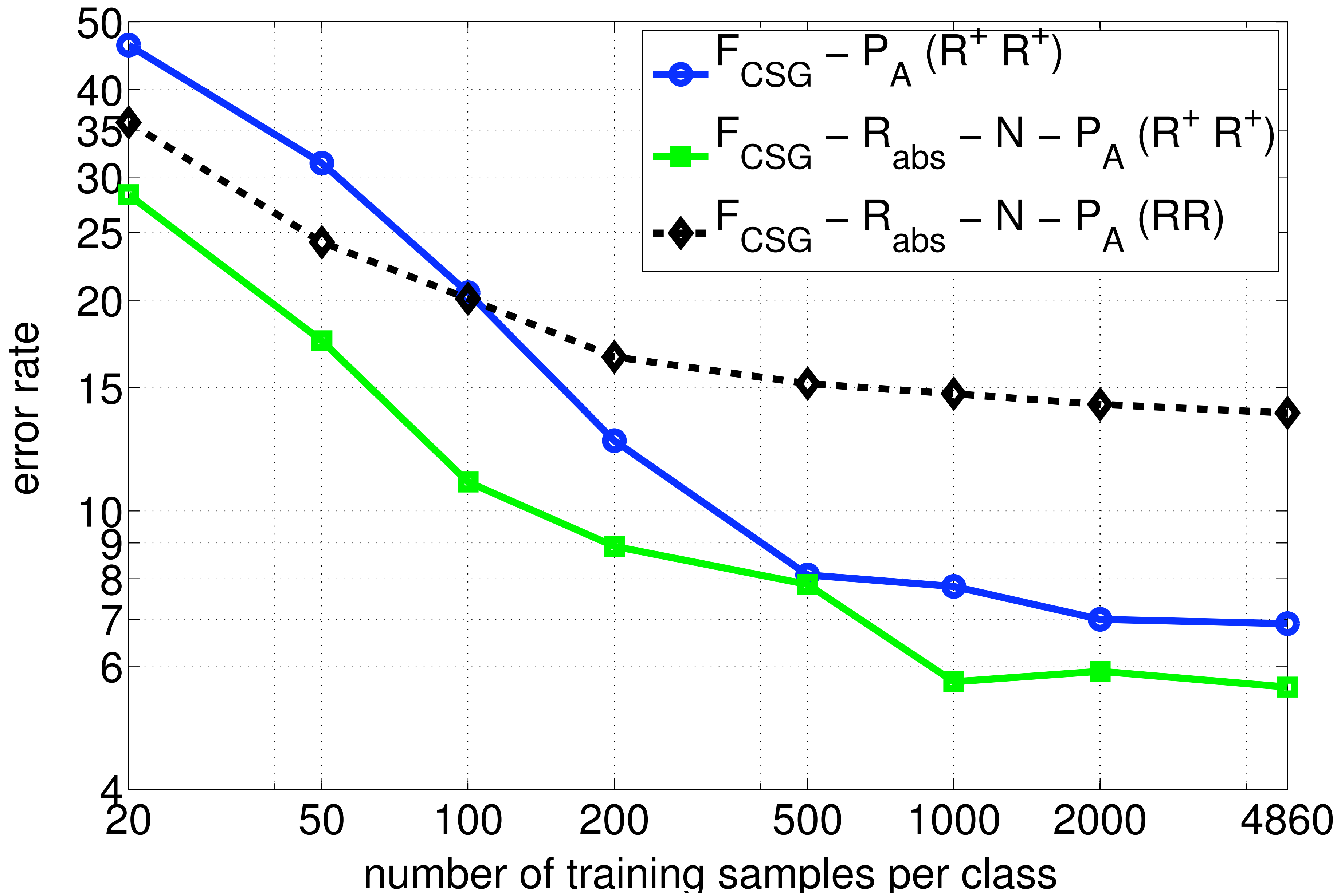
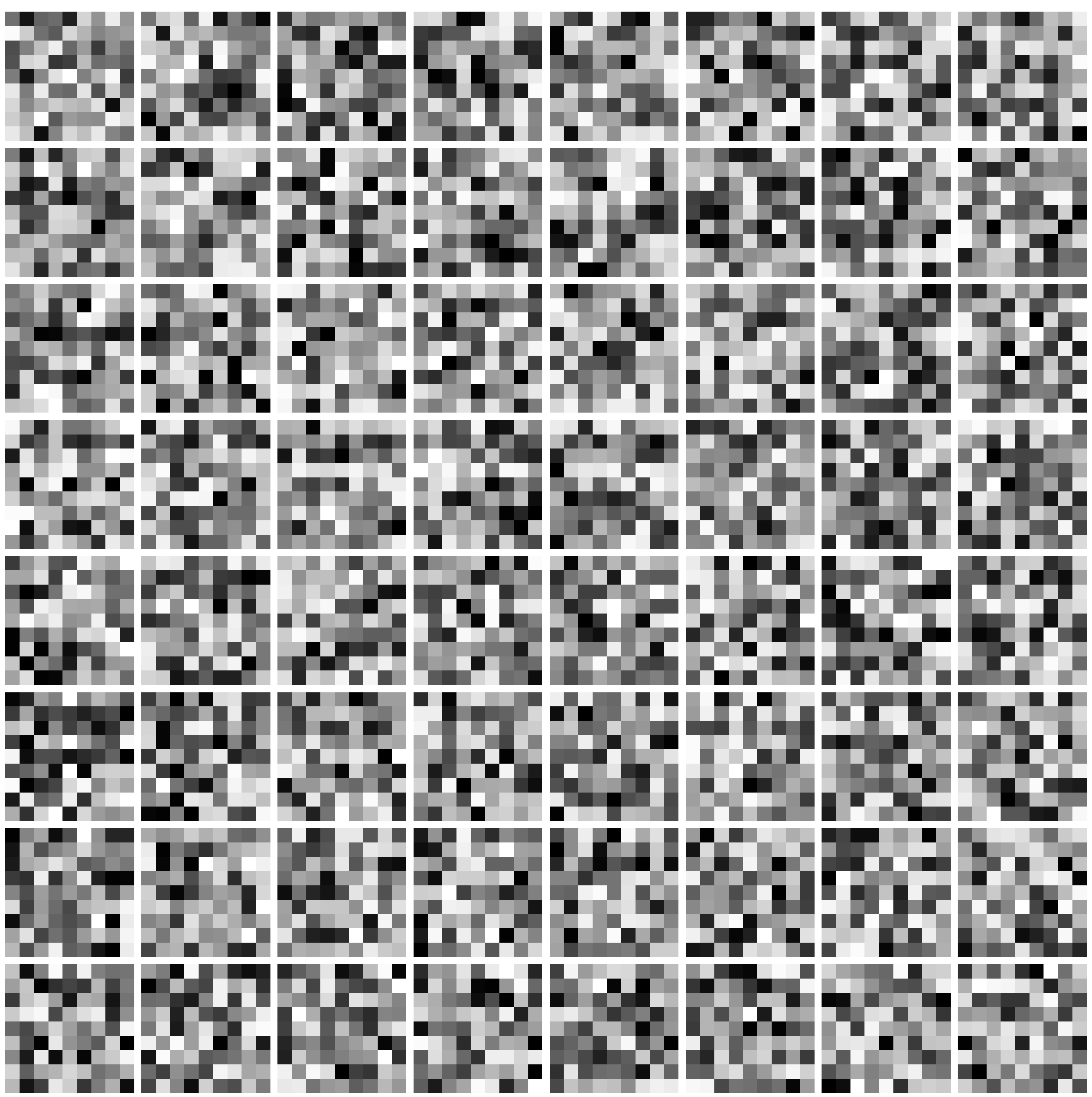


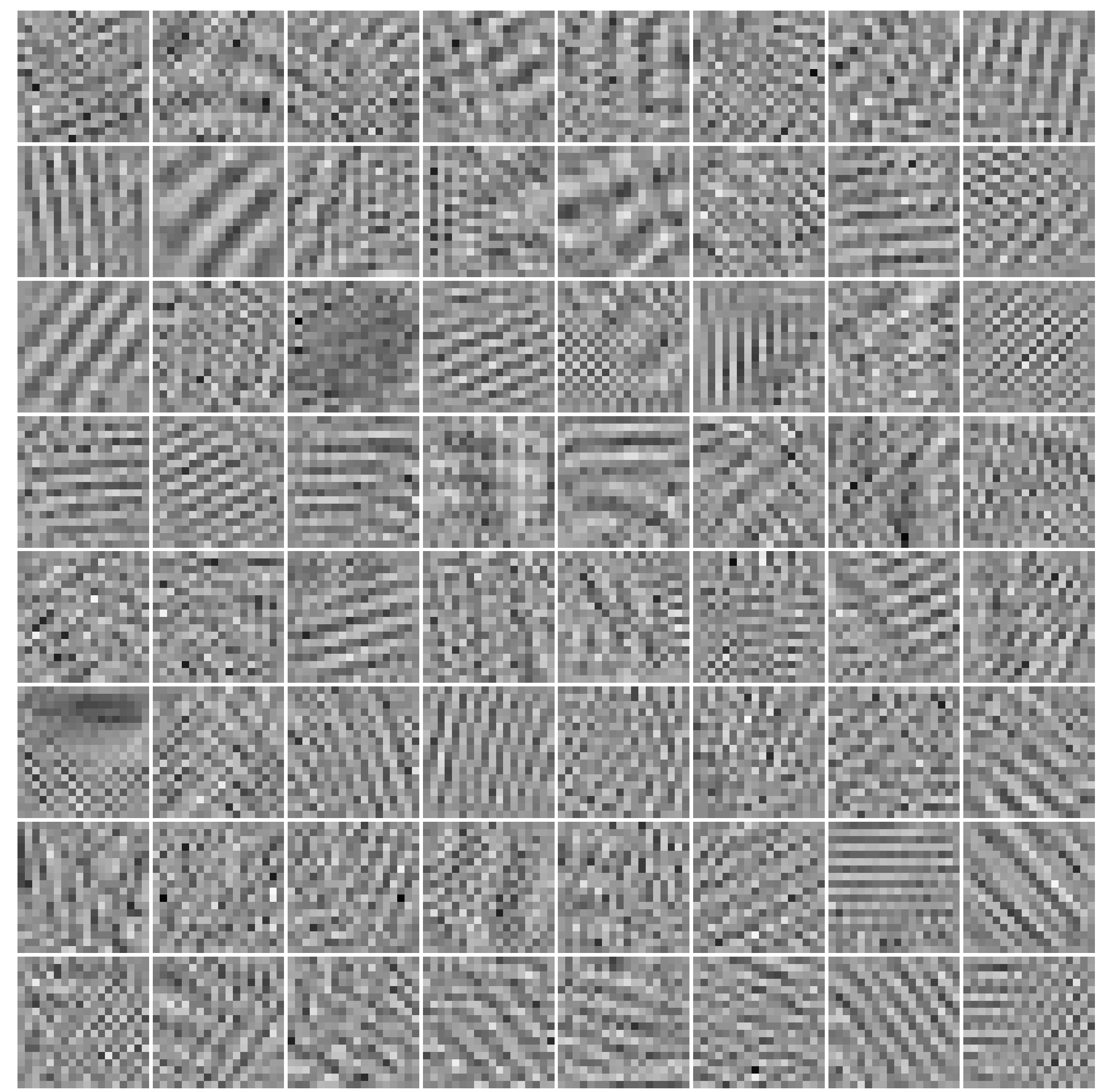
Figure 3. Test Error rate vs. number of training samples per class on NORB Dataset.

What is the Best Multi-Stage Architecture for Object Recognition?

(Jarrett, Kavukcuoglu, Ranzato and LeCun, ICCV 2009)

Filtres aléatoires

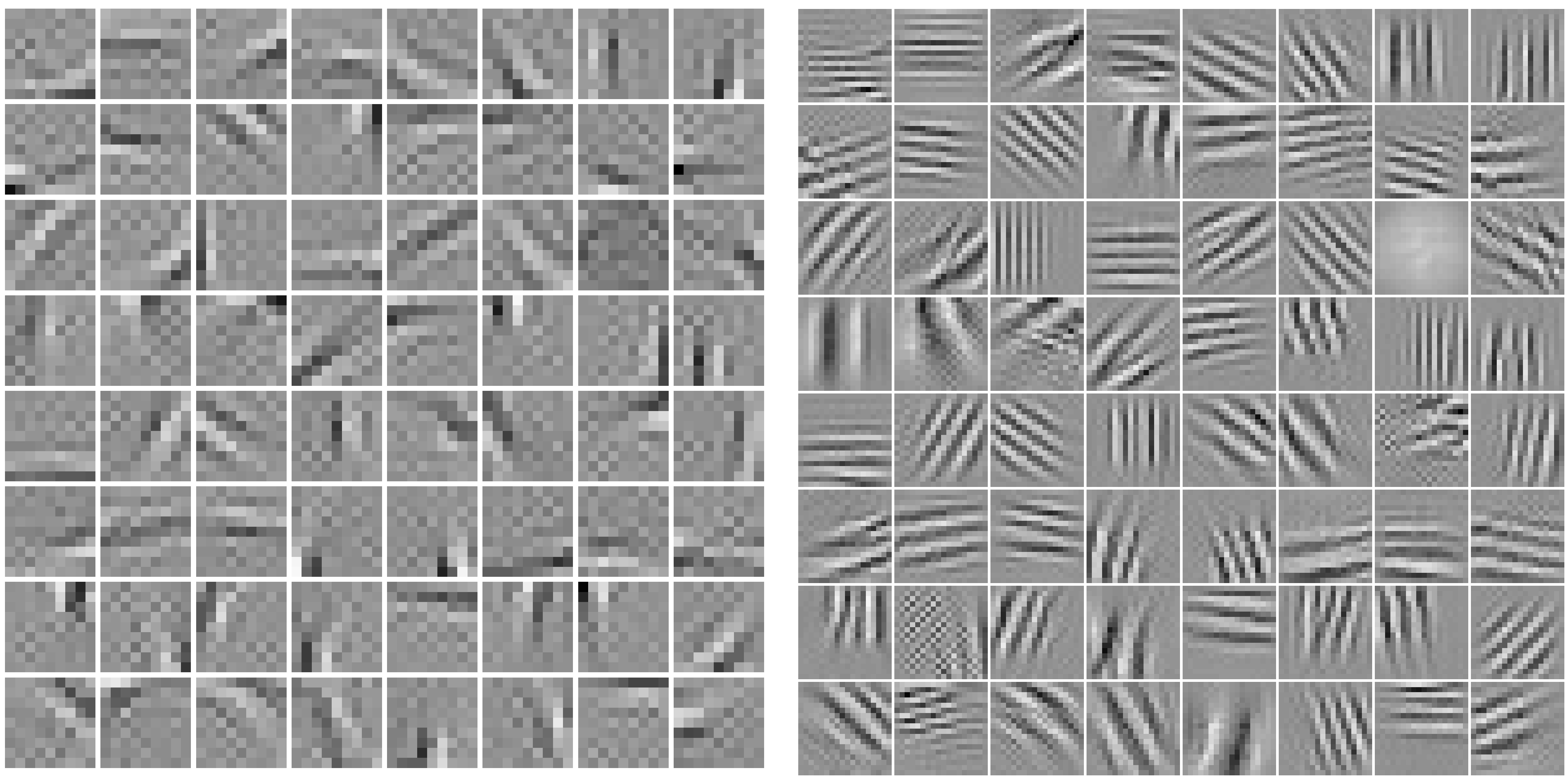




What is the Best Multi-Stage Architecture for Object Recognition?

(Jarrett, Kavukcuoglu, Ranzato and LeCun, ICCV 2009)

Filtres appris



A High-Throughput Screening Approach to Discovering Good Forms of Biologically-Inspired Visual Representation

(Pinto, Doukhan, DiCarlo and Cox, PLoS 2009)

- Une autre approche non-convolutionnelle d'apprendre les filtres

maximisation sur un voisinage

$$winner = \arg\max_i (F_i^\ell)$$

$$\Phi_{winner}^{\ell}{}' = (1 - \lambda^\ell) \cdot \Phi_{winner}^{\ell} + \lambda^\ell \cdot patch$$

$$\Phi_{winner}^{\ell}{}'' = \frac{\Phi_{winner}^{\ell}{}' - \langle \Phi_{winner}^{\ell}{}' \rangle}{\left\| \Phi_{winner}^{\ell}{}' - \langle \Phi_{winner}^{\ell}{}' \rangle \right\|_2}$$

- Une sorte de "Online K-means"

A High-Throughput Screening Approach to Discovering Good Forms of Biologically-Inspired Visual Representation

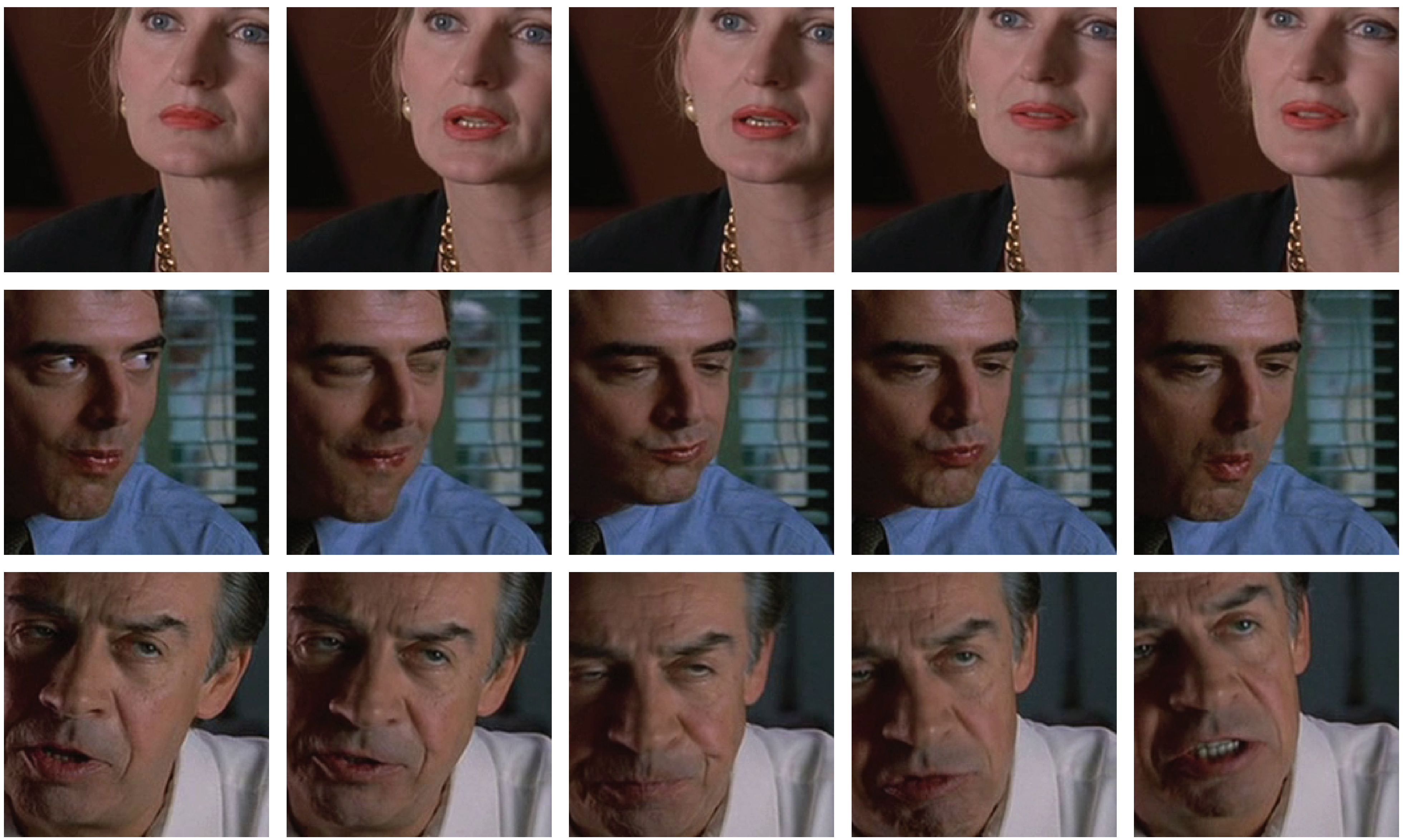
(Pinto, Doukhan, DiCarlo and Cox, PLoS 2009)

- Plusieurs options:

- *Learning rate* parameter $\lambda^{\ell} \in \{10^{-4},\ 10^{-3},\ 10^{-2}\}$
- *Patch Normalization*: normalize *patch* to unit-length, or do not normalize (2 choices)
- *Competition Neighborhood Size* $\in \{1, 3, 5, 7, 9\}$
- *Competition Neighborhood Stride* $\in \{1, 3, 5, 7, 9\}$
- *"Rebalancing"*: if the relative winning ratio [2] of a given filter Φ_i^{ℓ} is less than {1%, 10% or 50%} (3 choices), its weights are reinitialized to the values of the most-winning filter plus a random jitter. This prevents filters from never winning.
- *"Temporal Advantage"* (or *"trace"*, see also [18, 4, 19, 20] for variants): the output score of the last-winning filter is multiplied by {1, 2 or 4} (3 choices) prior to determining which filter "wins." A value of 1 is the equivalent of no advantage; a value of 2 doubles the effective output of the filter for the purposes of competition, biasing it to win again.

A High-Throughput Screening Approach to Discovering Good Forms of Biologically-Inspired Visual Representation
(Pinto, Doukhan, DiCarlo and Cox, PLoS 2009)

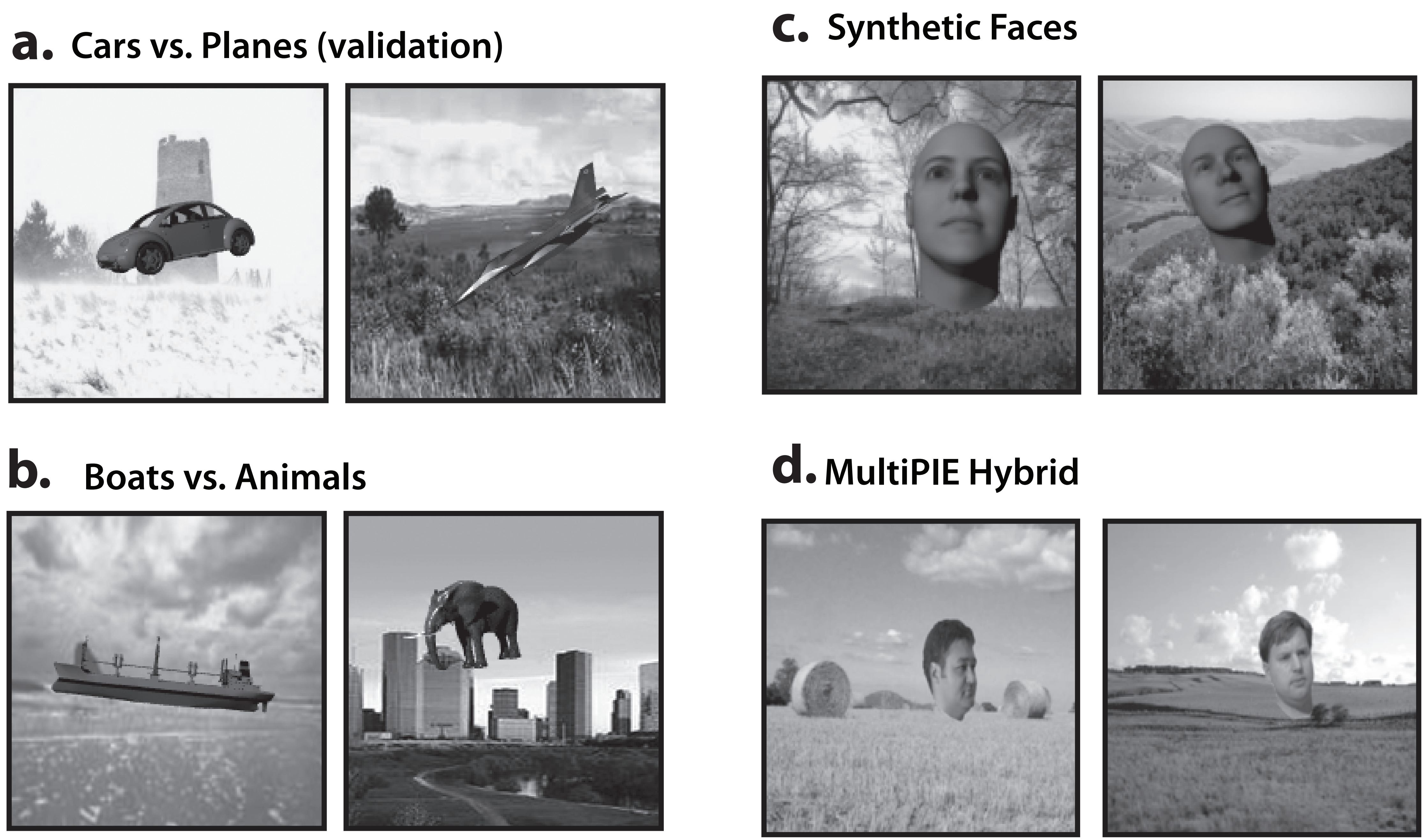
- Jeux de données (entraînement)



A High-Throughput Screening Approach to Discovering Good Forms of Biologically-Inspired Visual Representation

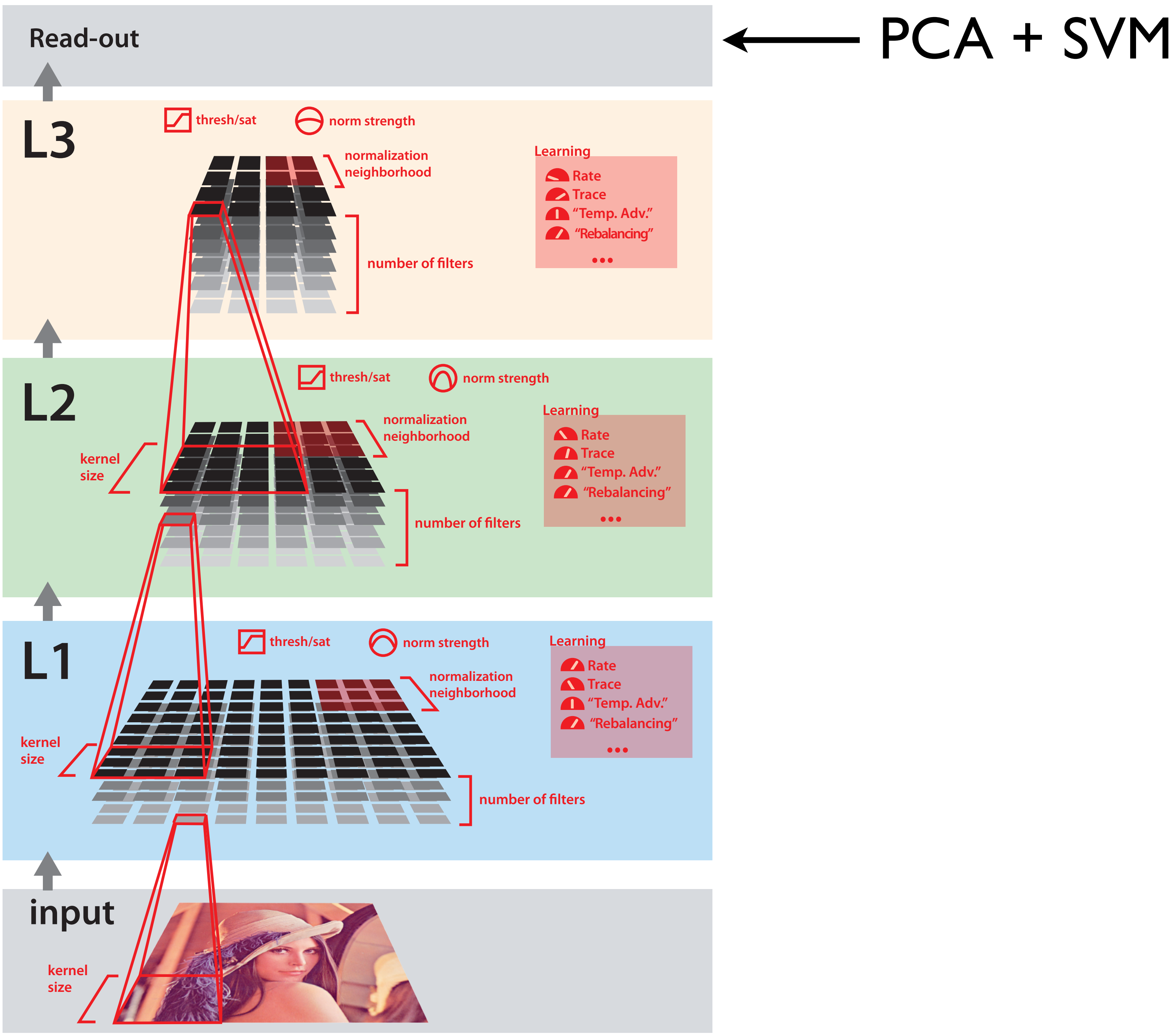
(Pinto, Doukhan, DiCarlo and Cox, PLoS 2009)

- Jeux de données (validation et test)



A High-Throughput Screening Approach to Discovering Good Forms of Biologically-Inspired Visual Representation

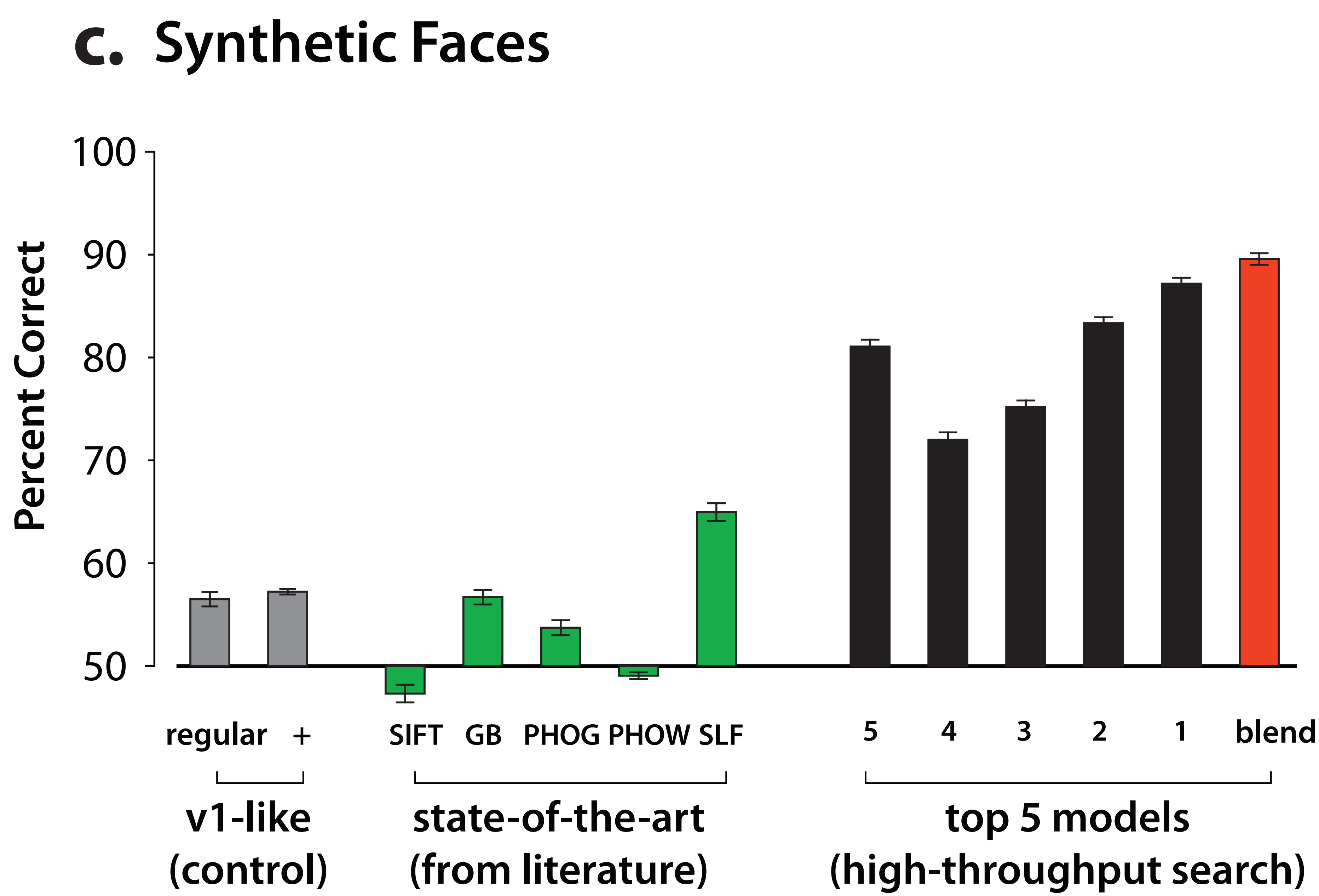
(Pinto, Doukhan, DiCarlo and Cox, PLoS 2009)



A High-Throughput Screening Approach to Discovering Good Forms of Biologically-Inspired Visual Representation

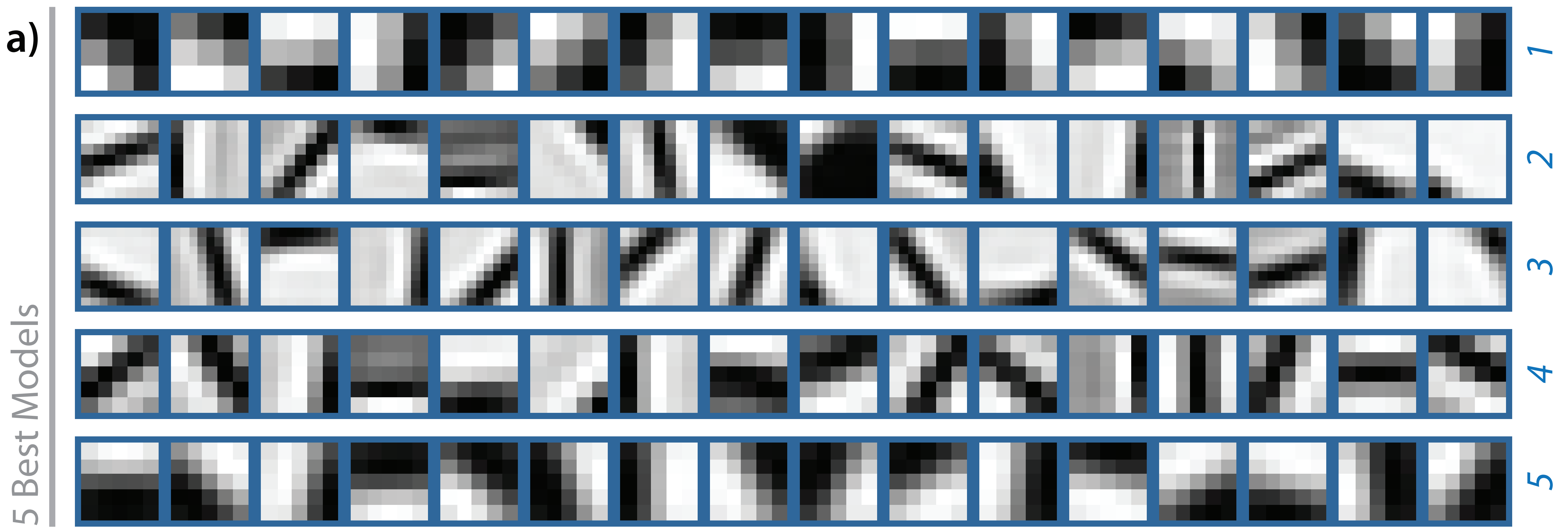
(Pinto, Doukhan, DiCarlo and Cox, PLoS 2009)

- Résultats



A High-Throughput Screening Approach to Discovering Good Forms of Biologically-Inspired Visual Representation
(Pinto, Doukhan, DiCarlo and Cox, PLoS 2009)

Filtres appris



Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)

- Approche d'apprentissage non-supervisé **convolutionnel**

$$\mathcal{L}(S) = \frac{1}{P}\sum_{i=1}^{P} ||X^i - F_{dec}(F_{enc}(X^i))||_2$$

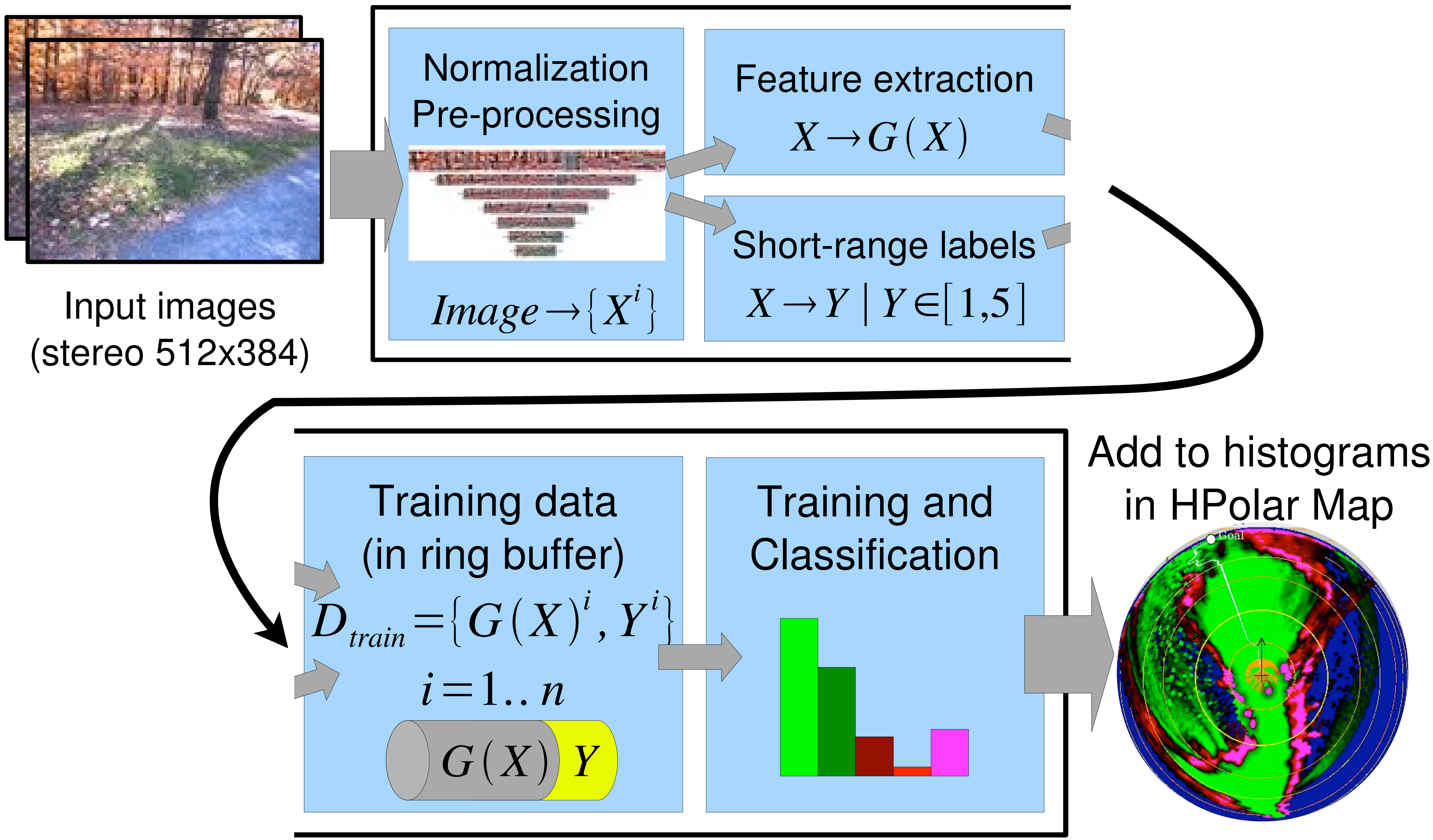
$$z_j = \tanh(c_j(\sum_i x_i * f_{ij}) + b_j)$$

$$z_i = \max_{i \in N_i}(x)$$

- Entraînement de plusieurs couchent à l'aide de la procédure de préentraînement

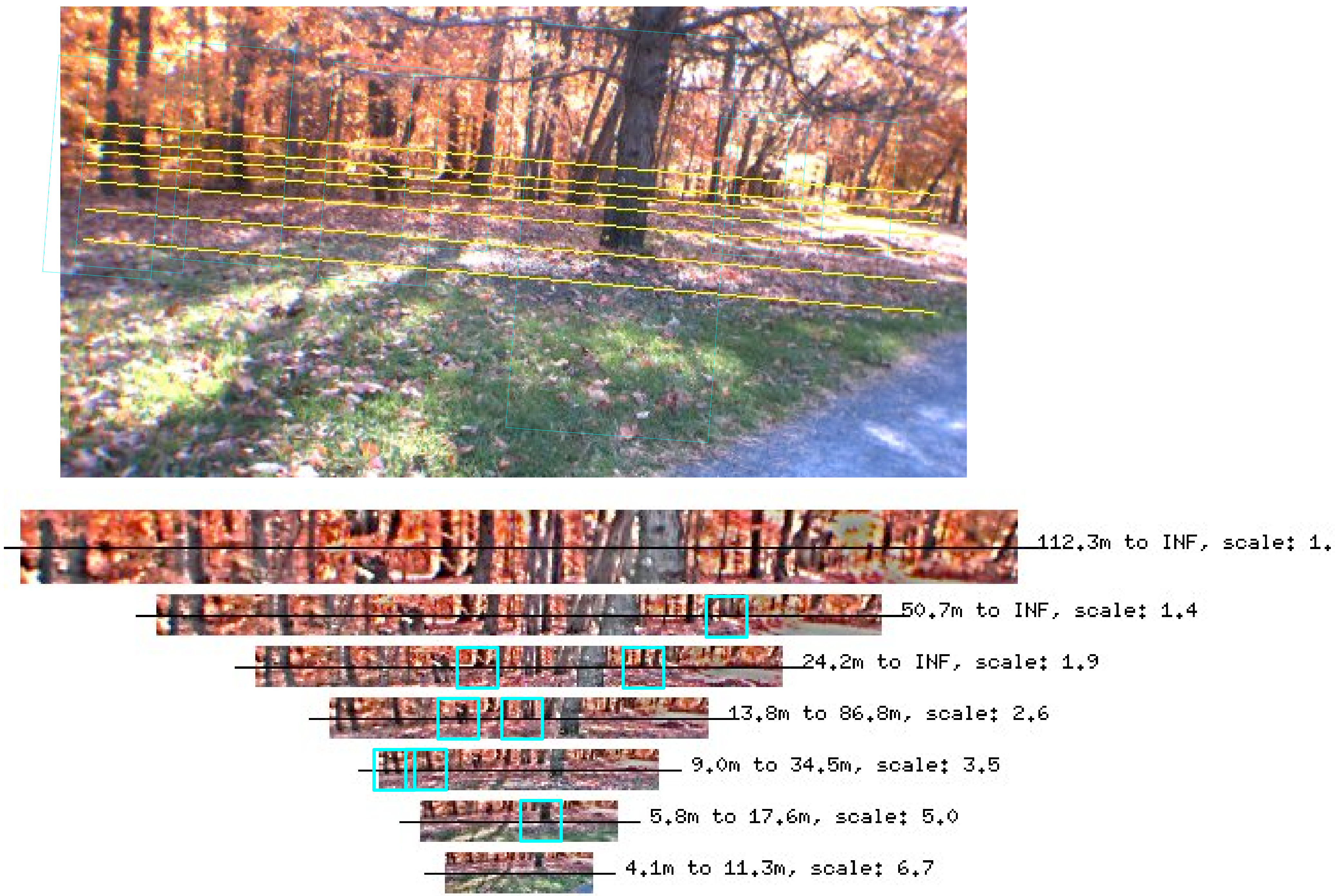
Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)



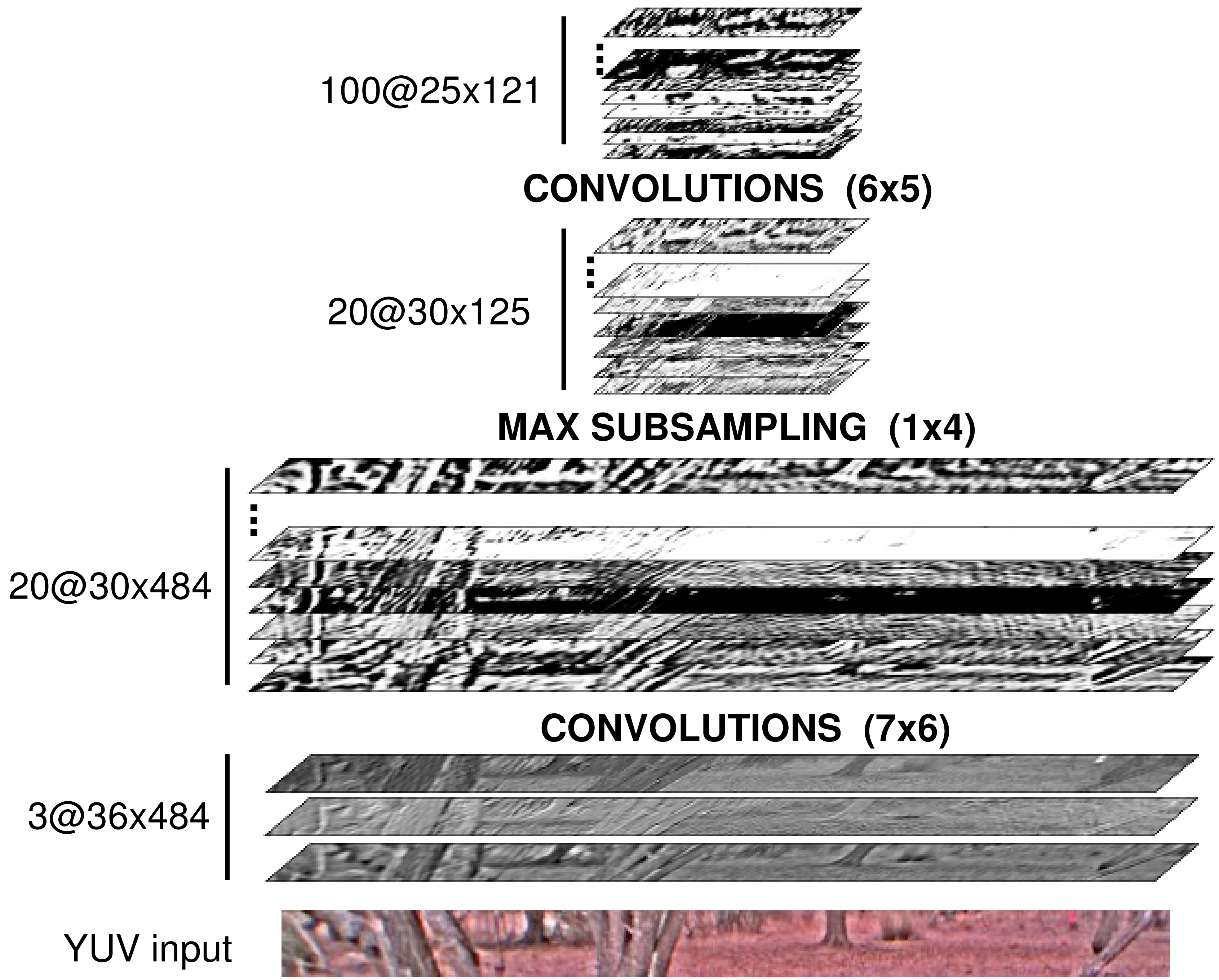
Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)



Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)



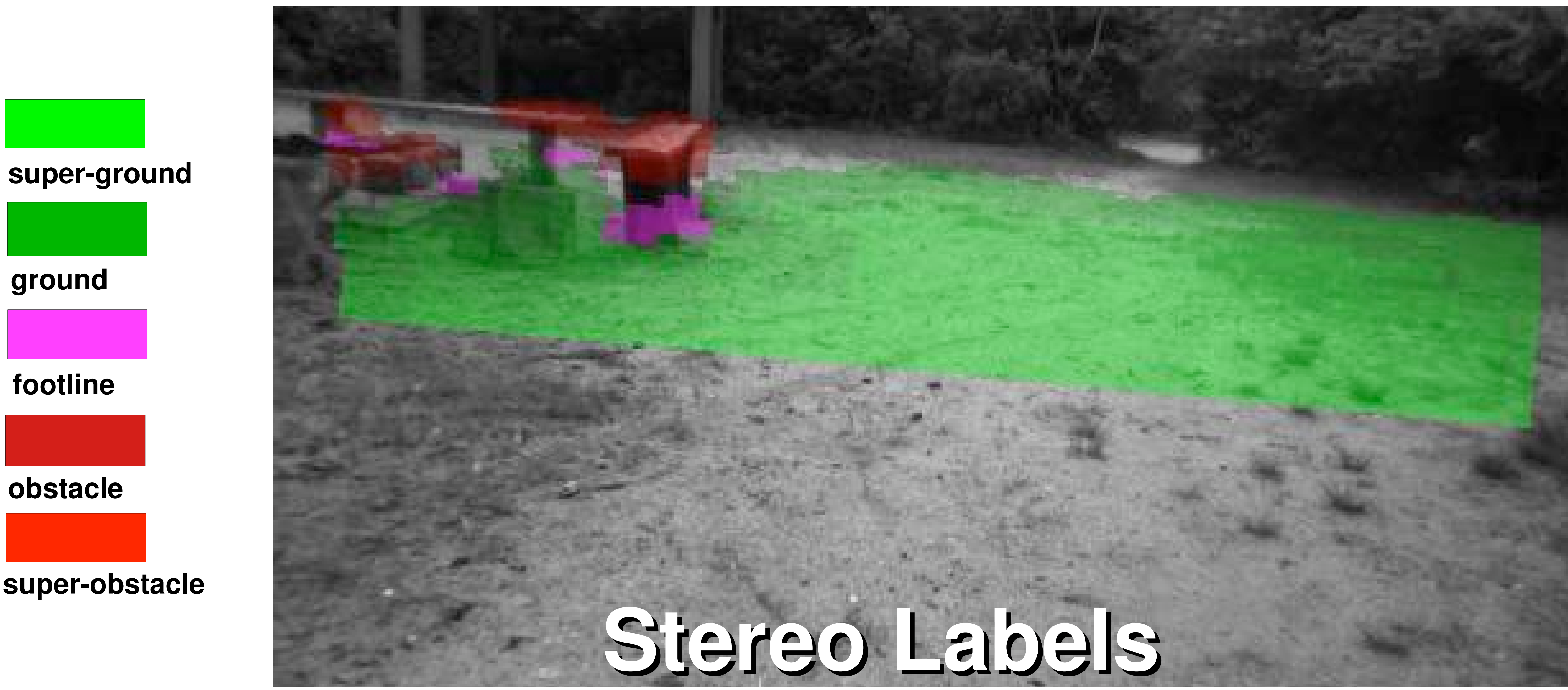
Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)

Super ground | Ground | Footline | Obstacle | Super obstacle

Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)



Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)

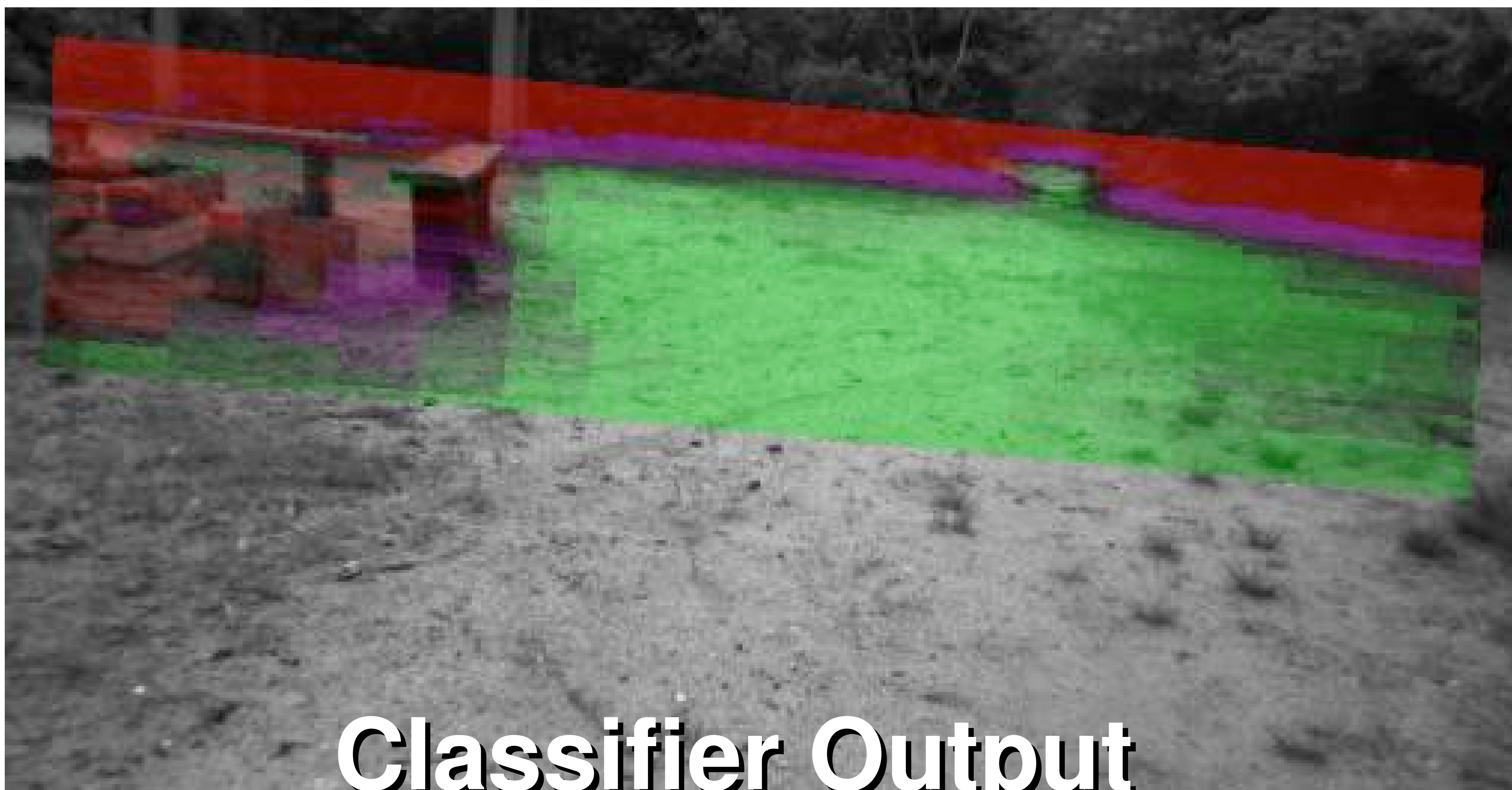
super-ground

ground

footline

obstacle

super-obstacle



Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)

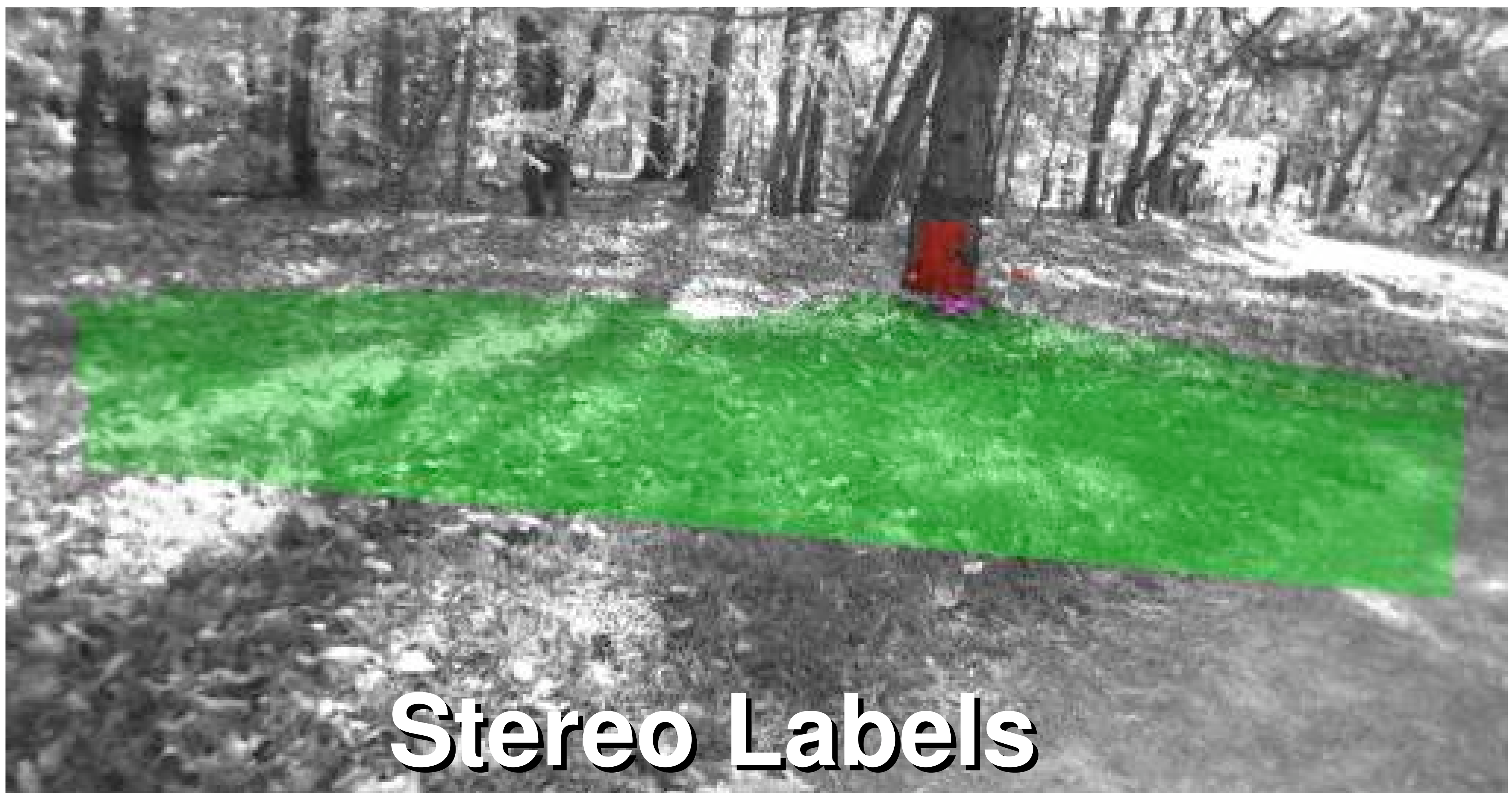
super-ground

ground

footline

obstacle

super-obstacle



Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)

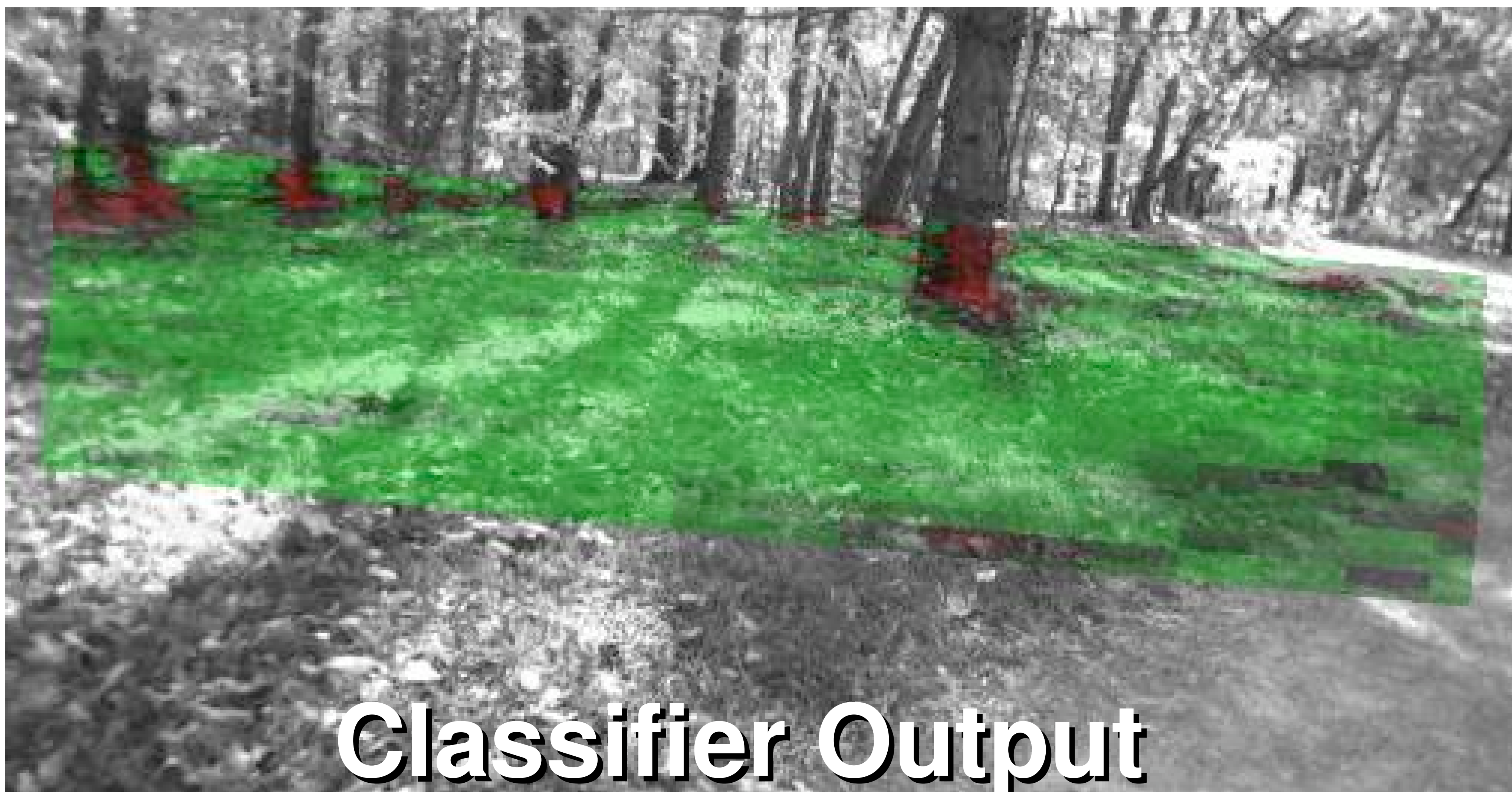
super-ground

ground

footline

obstacle

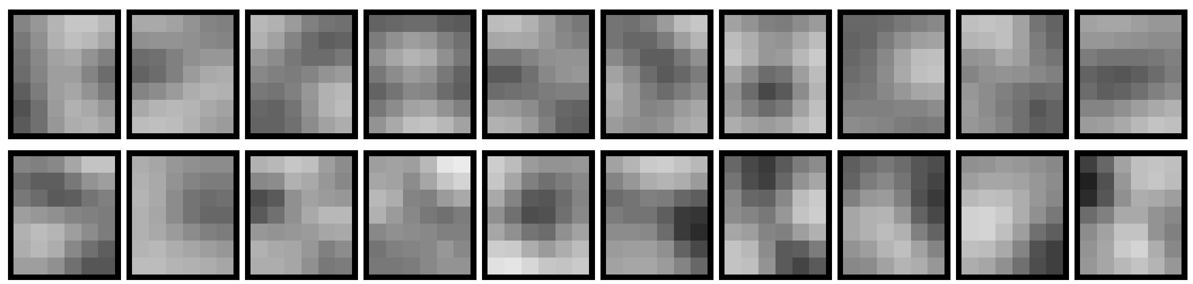
super-obstacle

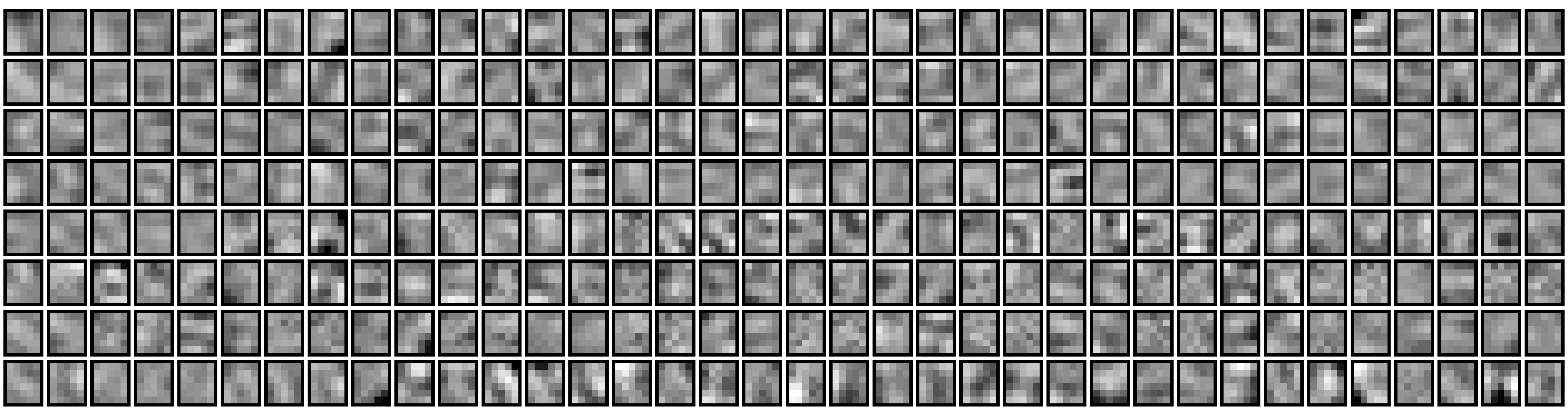


Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)

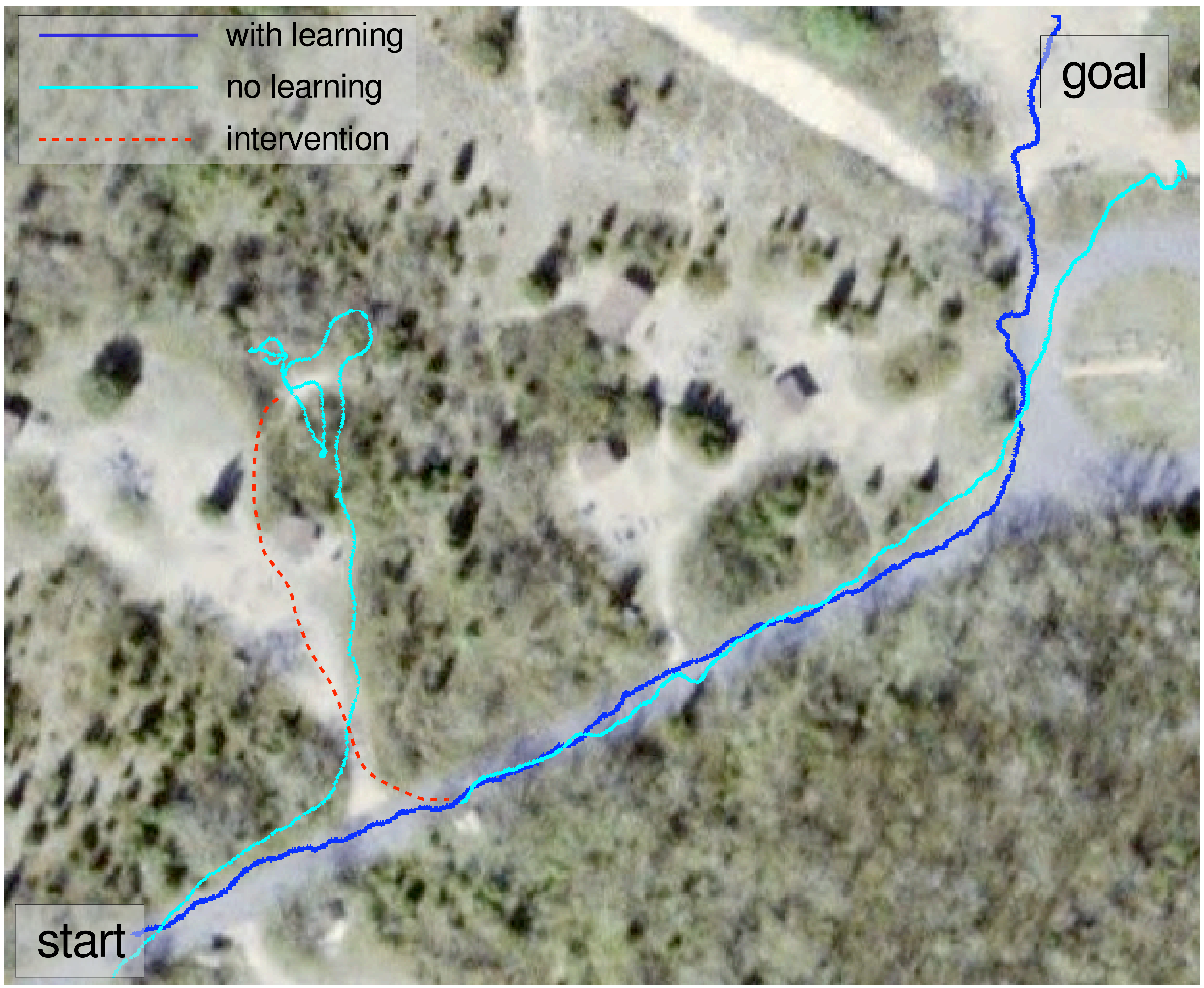
Filtres appris





Deep Belief Net Learning in a Long-Range Vision System for Autonomous Off-Road Driving

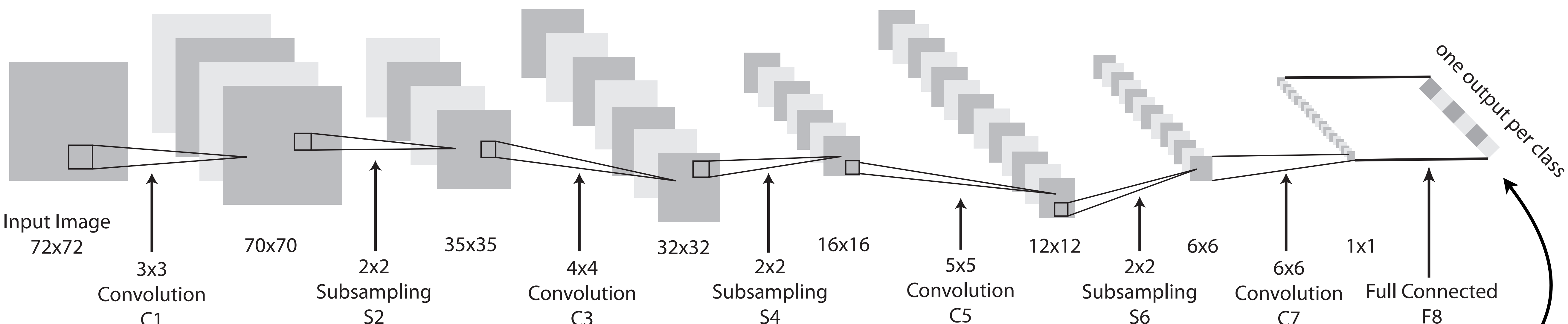
(Hadsell, Erkan, Sermanet, Scoffier, Muller and LeCun, IROS 2008)



Deep Learning from Temporal Coherence in Video

(Mobahi, Collobert and Weston, ICML 2009)

- Application du critère "semi-supervised embedding" à un réseau à convolution



$$L_{coh}(\boldsymbol{\theta}, \boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{cases} ||\boldsymbol{z}^l_{\boldsymbol{\theta}}(\boldsymbol{x}_1) - \boldsymbol{z}^l_{\boldsymbol{\theta}}(\boldsymbol{x}_2)||_1, & \text{if } \boldsymbol{x}_1,\ \boldsymbol{x}_2 \text{ consecutive} \\ \max(0,\ \delta - ||\boldsymbol{z}^l_{\boldsymbol{\theta}}(\boldsymbol{x}_1) - \boldsymbol{z}^l_{\boldsymbol{\theta}}(\boldsymbol{x}_2)||_1), & \text{otherwise} \end{cases}$$

juste avant la "softmax"

Deep Learning from Temporal Coherence in Video

(Mobahi, Collobert and Weston, ICML 2009)

- Les paires similaires sont extraites de séquences

Algorithm 1 Stochastic Gradient with Video Coherence.

Input: Labeled data $(\boldsymbol{x}_n, y_n)$, $n = 1, ...N$, unlabeled video data $\boldsymbol{x}_n$, $n = N+1, ...N+U$
repeat
  Pick a random *labeled* example $(\boldsymbol{x}_n, y_n)$
  Make a gradient step to decrease $L(\boldsymbol{\theta}, \boldsymbol{x}_n, y_n)$
  Pick a random pair of consecutive images $\boldsymbol{x}_m, \boldsymbol{x}_n$ in the video
  Make a gradient step to decrease $L_{coh}(\boldsymbol{\theta}, \boldsymbol{x}_m, \boldsymbol{x}_n)$
  Pick a random pair of images $\boldsymbol{x}_m, \boldsymbol{x}_n$ in the video
  Make a gradient step to decrease $L_{coh}(\boldsymbol{\theta}, \boldsymbol{x}_m, \boldsymbol{x}_n)$
until Stopping criterion is met

Deep Learning from Temporal Coherence in Video

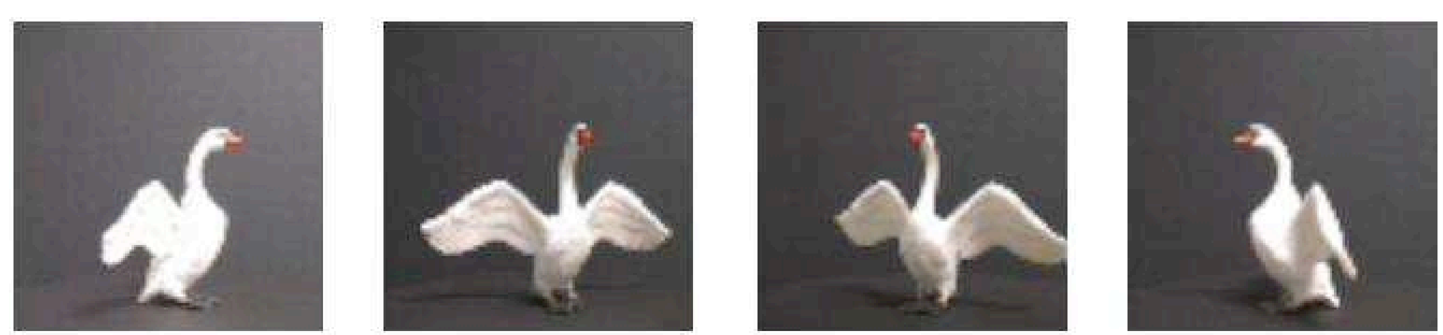
(Mobahi, Collobert and Weston, ICML 2009)

- Jeux de données

COIL100

COIL100-Like

Animal Set



Deep Learning from Temporal Coherence in Video

(Mobahi, Collobert and Weston, ICML 2009)

Method	**30 objects**	**100 objects**
Nearest Neighbor	81.8	70.1
SVM	84.9	74.6
SpinGlass MRF	82.79	69.41
Eigen Spline	84.6	77.0
VTU	89.9	79.1
Standard CNN	84.88	71.49
*video*CNN V:COIL100	-	92.25
*video*CNN V:COIL"70"	95.03	-
*video*CNN V:COIL-Like	-	79.77
*video*CNN V:Animal	-	78.67

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

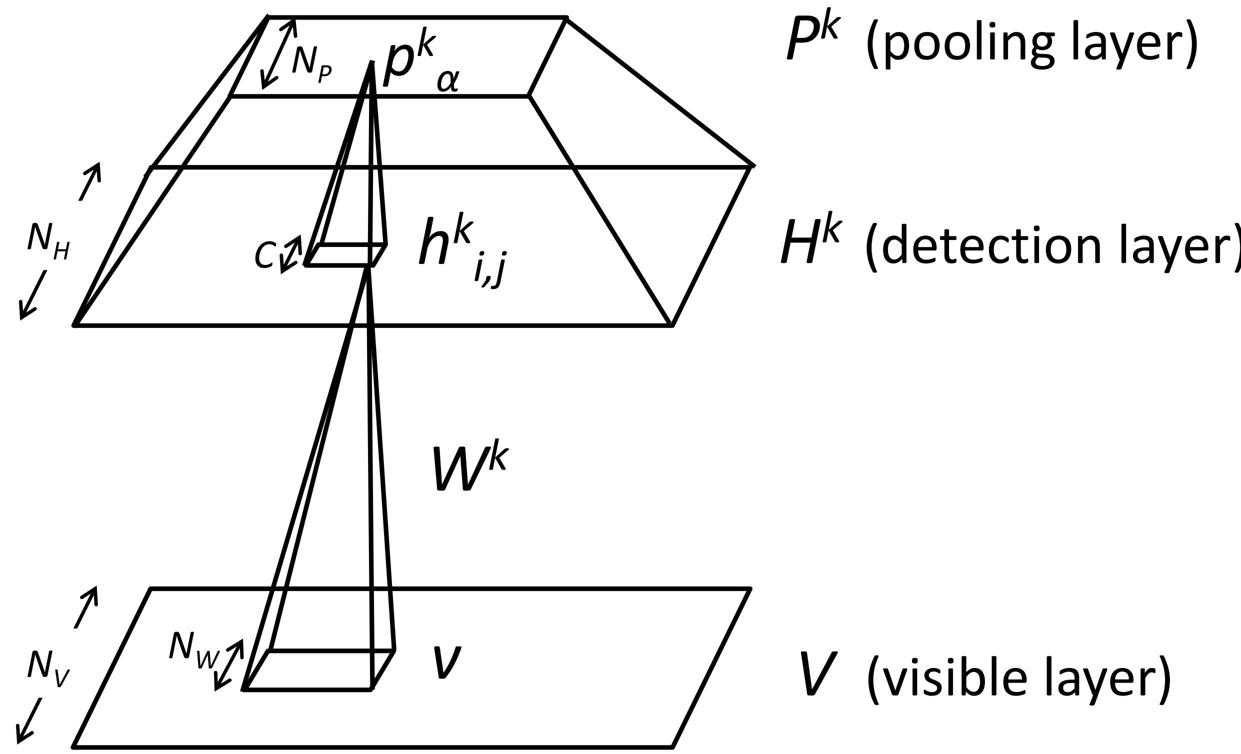
(Lee, Grosse, Ranganath and Ng, ICML 2009)

$$P(v_{ij}=1|\mathbf{h}) = \sigma((\sum_k W^k * h^k)_{ij} + c)$$

$$P(h^k_{i,j}=1|\mathbf{v}) = \frac{\exp(I(h^k_{i,j}))}{1+\sum_{(i',j')\in B_\alpha}\exp(I(h^k_{i',j'}))}$$

$$P(p^k_\alpha=0|\mathbf{v}) = \frac{1}{1+\sum_{(i',j')\in B_\alpha}\exp(I(h^k_{i',j'}))}$$

$$I(h^k_{ij}) \triangleq b_k + (\tilde{W}^k * v)_{ij}$$



$$E(\mathbf{v},\mathbf{h}) = -\sum_{k=1}^{K} h^k \bullet (\tilde{W}^k * v) - \sum_{k=1}^{K} b_k \sum_{i,j} h^k_{i,j} - c\sum_{i,j} v_{ij}$$

$$\text{subj. to} \qquad \sum_{(i,j)\in B_\alpha} h^k_{i,j} \le 1, \quad \forall k, \alpha$$

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

(Lee, Grosse, Ranganath and Ng, ICML 2009)

- Combiner les couches

$$\boxed{\begin{aligned} E(\mathbf{v},\mathbf{h},\mathbf{p},\mathbf{h}') &= -\sum_k v \bullet (W^k * h^k) - \sum_k b_k \sum_{ij} h^k_{ij} \\ &\quad - \sum_{k,\ell} p^k \bullet (\Gamma^{k\ell} * {h'}^{\ell}) - \sum_\ell {b'}_\ell \sum_{ij} {h'}^{\ell}_{ij} \end{aligned}}$$

$$P(h^k_{i,j} = 1|\mathbf{v},\mathbf{h}') = \frac{\exp(I(h^k_{i,j}) + I(p^k_\alpha))}{1 + \sum_{(i',j')\in B_\alpha} \exp(I(h^k_{i',j'}) + I(p^k_\alpha))}$$

$$P(p^k_\alpha = 0|\mathbf{v},\mathbf{h}') = \frac{1}{1 + \sum_{(i',j')\in B_\alpha} \exp(I(h^k_{i',j'}) + I(p^k_\alpha))}$$

vers le haut

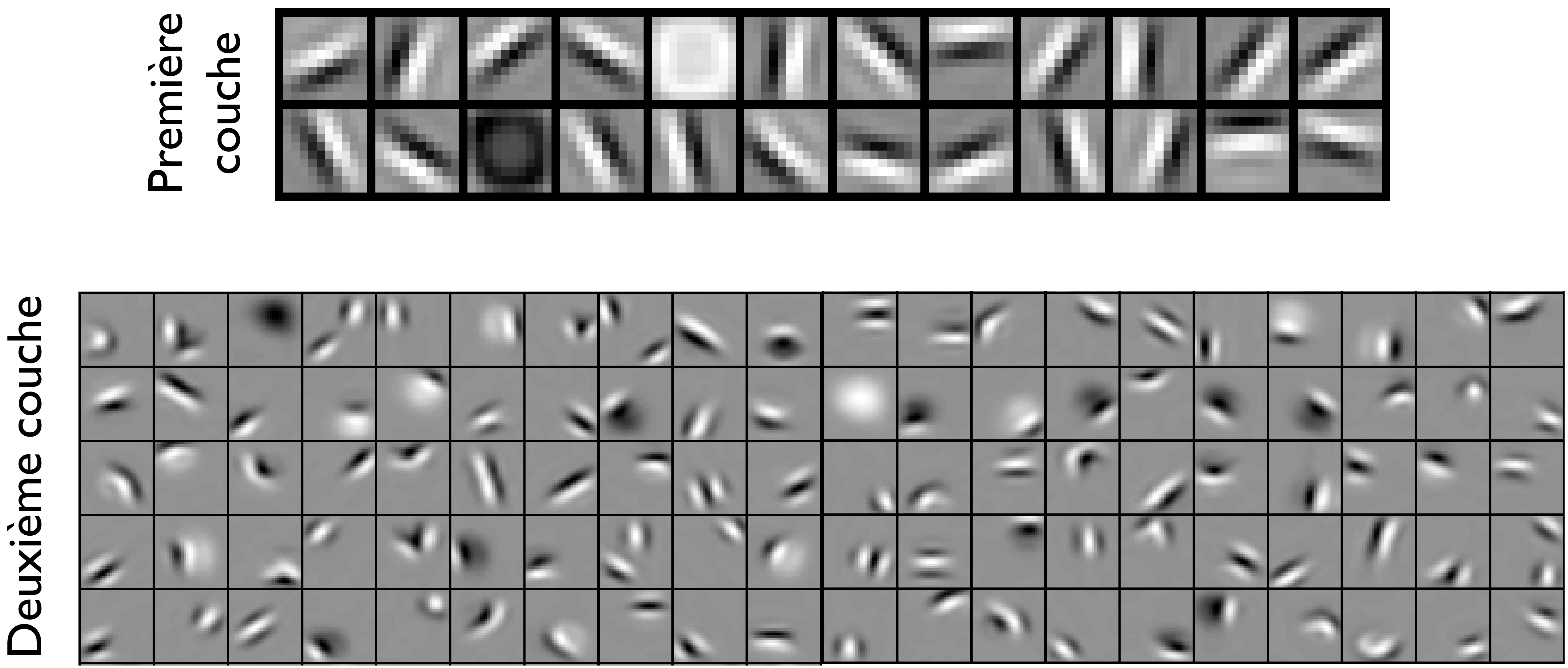
vers le bas

$$I(h^k_{ij}) \triangleq b_k + (\tilde{W}^k * v)_{ij} \qquad I(p^k_\alpha) \triangleq \sum (\Gamma^{k\ell} * {h'}^{\ell})_\alpha$$

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

(Lee, Grosse, Ranganath and Ng, ICML 2009)

Filtres appris



Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

(Lee, Grosse, Ranganath and Ng, ICML 2009)

Table 1. Classification accuracy for the Caltech-101 data

Training Size	15	30
CDBN (first layer)	53.2±1.2%	60.5±1.1%
CDBN (first+second layers)	57.7±1.5%	65.4±0.5%
Raina et al. (2007)	46.6%	-
Ranzato et al. (2007)	-	54.0%
Mutch and Lowe (2006)	51.0%	56.0%
Lazebnik et al. (2006)	54.0%	64.6%
Zhang et al. (2006)	59.0±0.56%	66.2±0.5%

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

(Lee, Grosse, Ranganath and Ng, ICML 2009)

Table 2. Test error for MNIST dataset

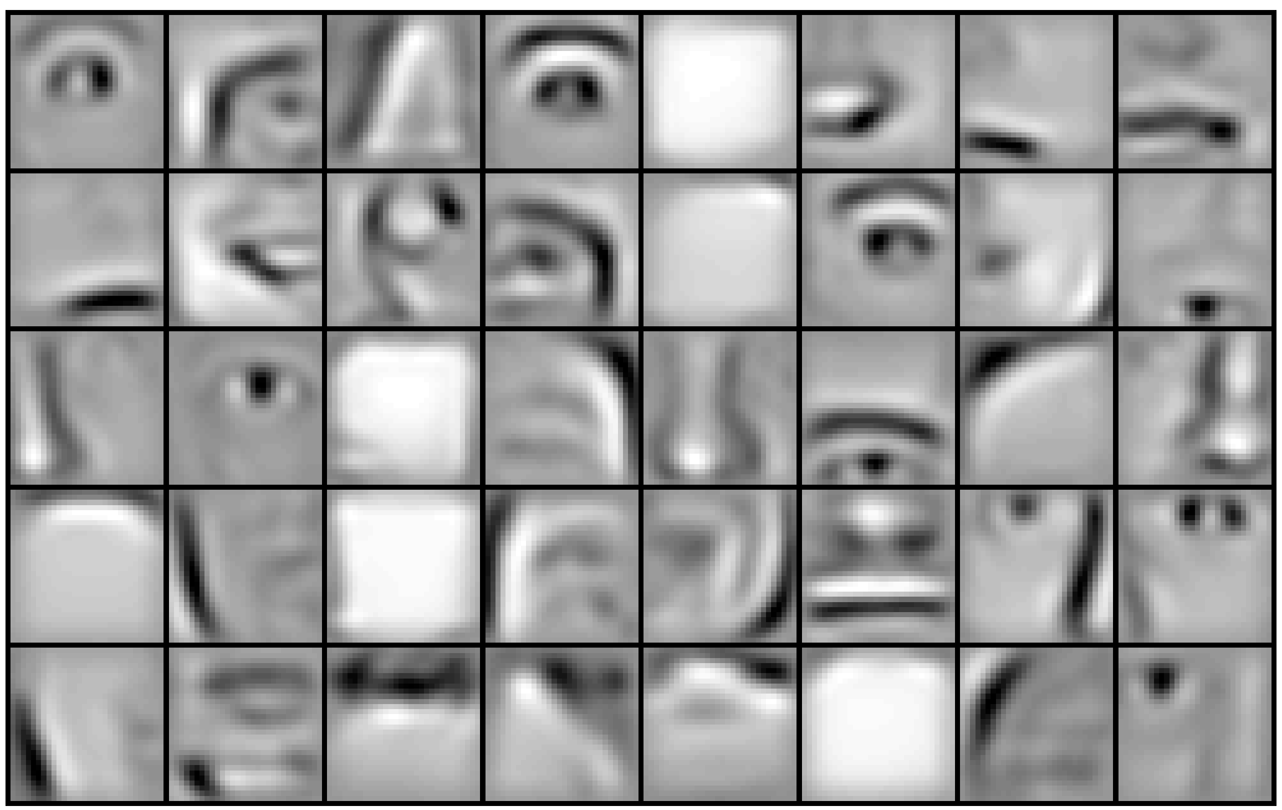
Labeled training samples	1,000	2,000
CDBN	2.62±0.12%	2.13±0.10%
Ranzato et al. (2007)	3.21%	2.53%
Hinton and Salakhutdinov (2006)	-	-
Weston et al. (2008)	2.73%	-

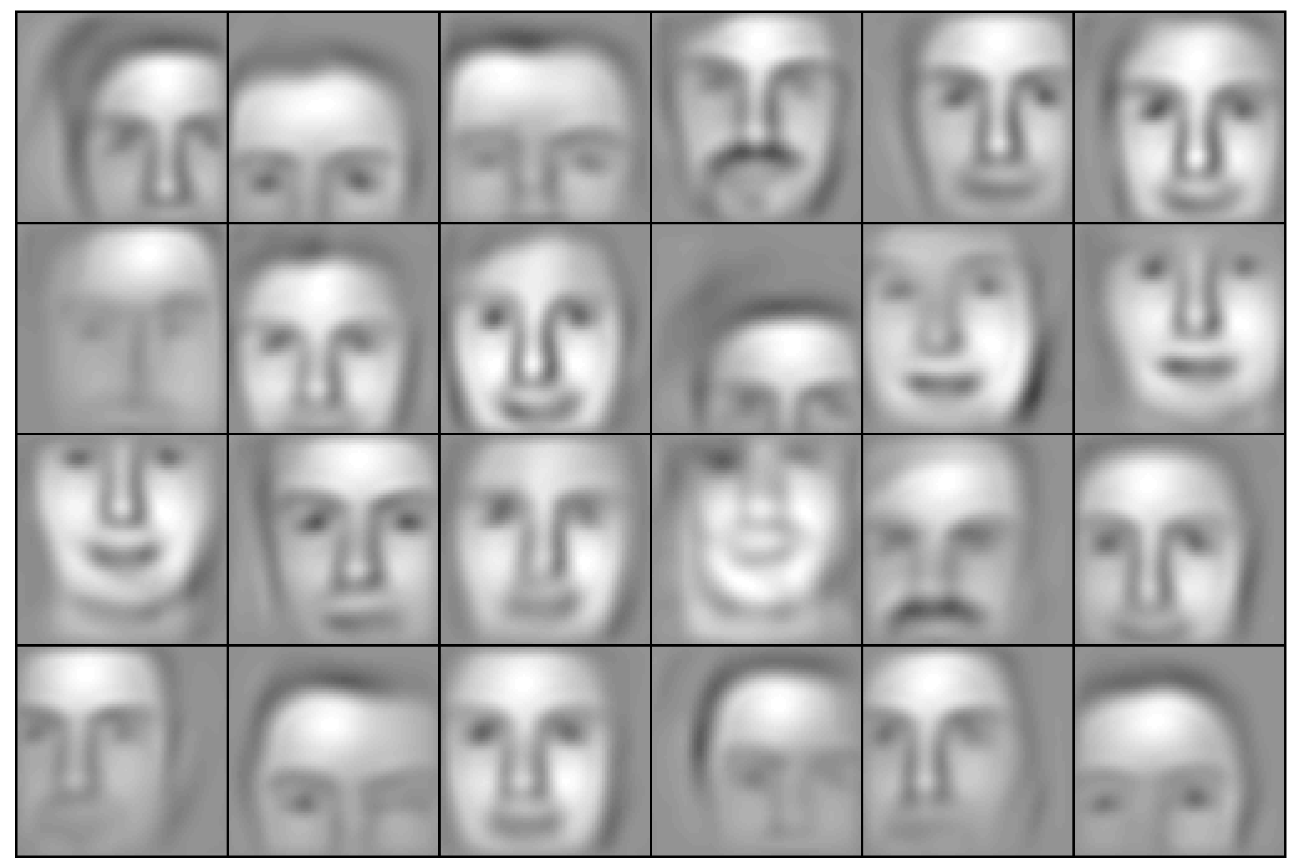
Labeled training samples	3,000	5,000	60,000
CDBN	1.91±0.09%	1.59±0.11%	0.82%
Ranzato et al. (2007)	-	1.52%	0.64%
Hinton and Salakhutdinov (2006)	-	-	1.20%
Weston et al. (2008)	1.83%	-	1.50%

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

(Lee, Grosse, Ranganath and Ng, ICML 2009)

Visages

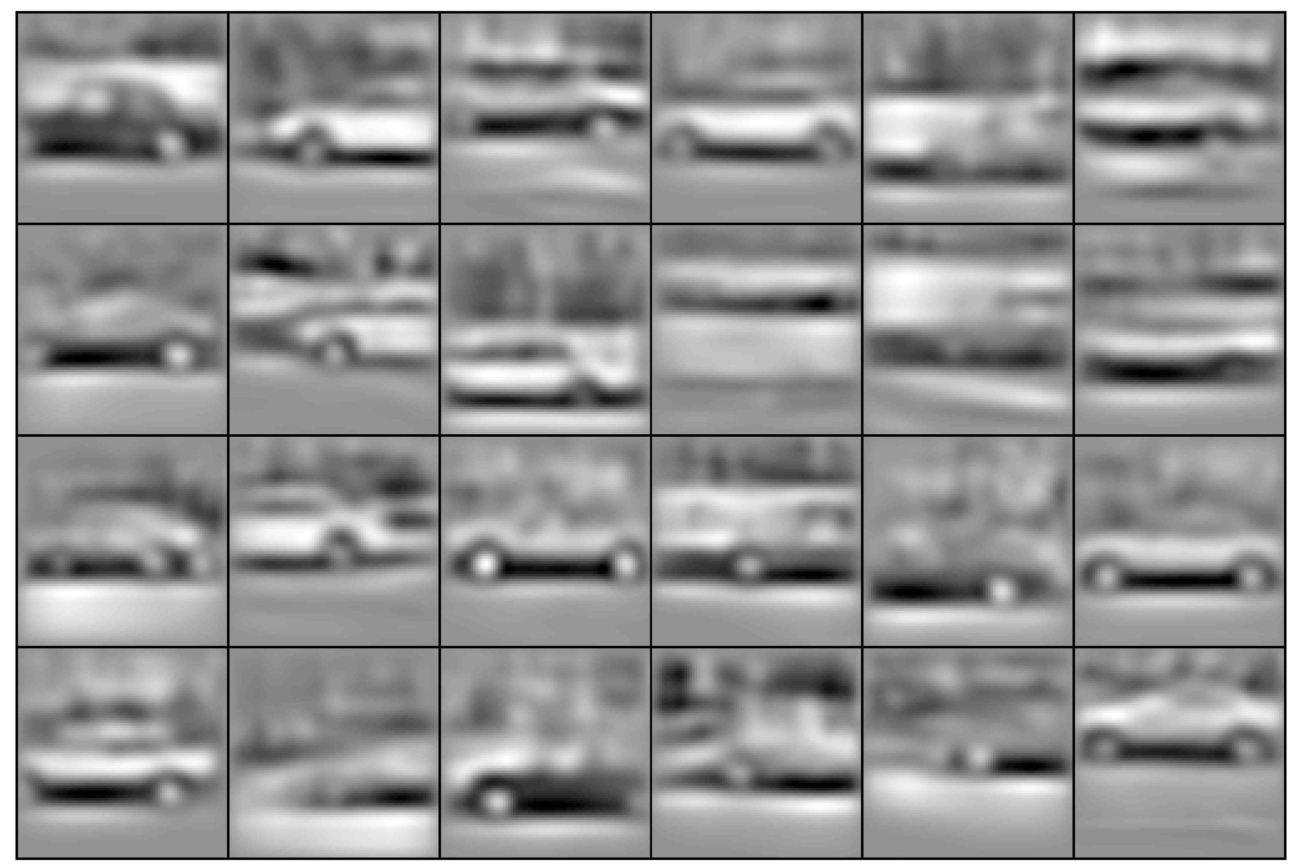




Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

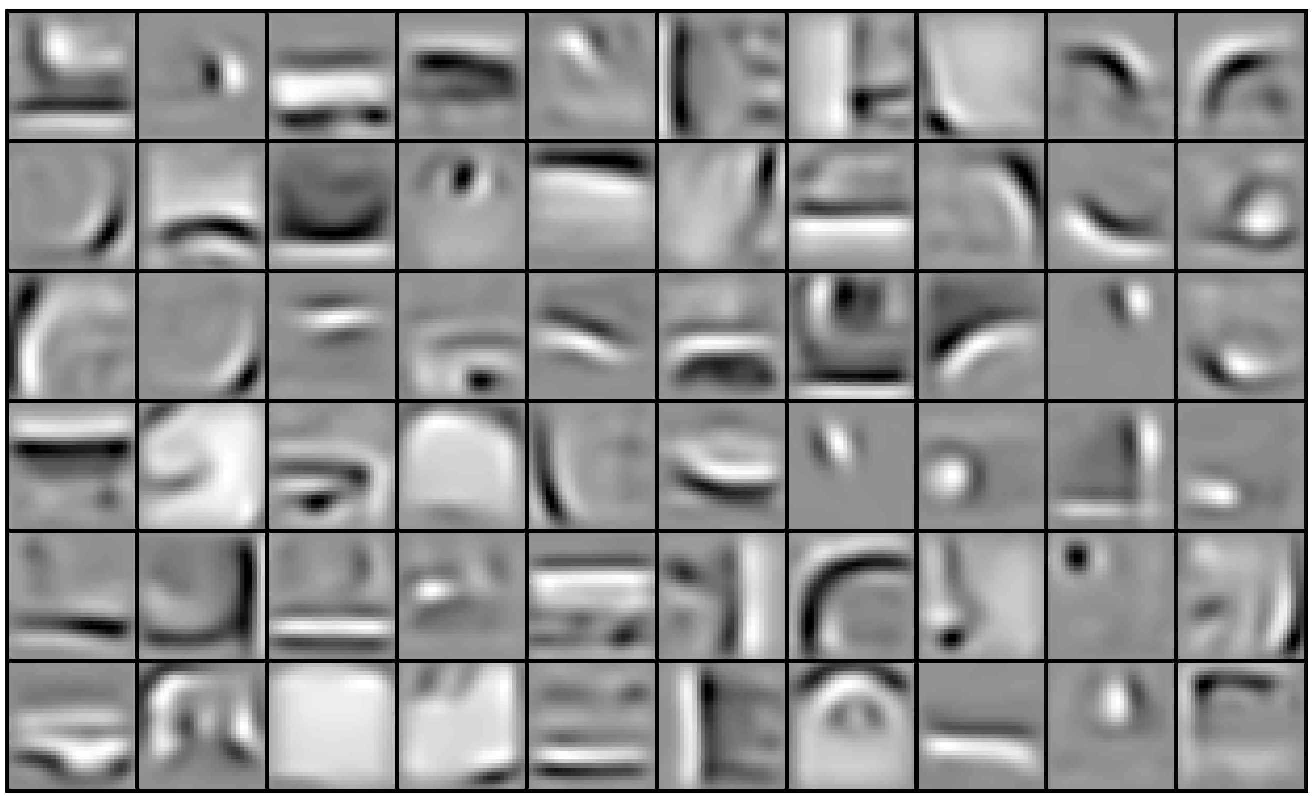
(Lee, Grosse, Ranganath and Ng, ICML 2009)

Voitures



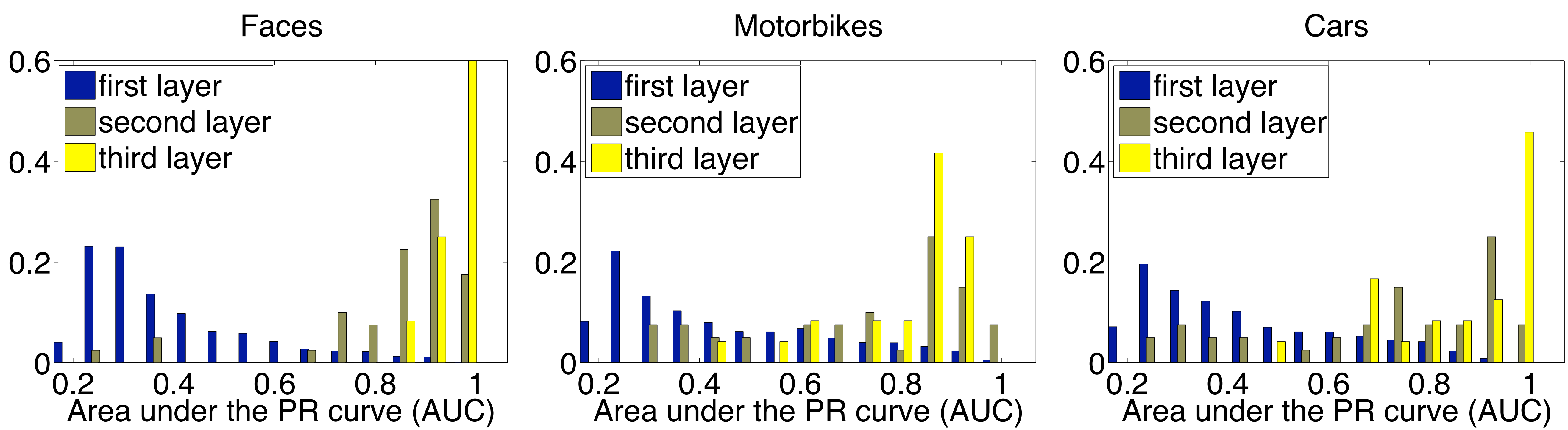
Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations
(Lee, Grosse, Ranganath and Ng, ICML 2009)

Visages, voitures, éléphants et chaises



Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

(Lee, Grosse, Ranganath and Ng, ICML 2009)



Features	Faces	Motorbikes	Cars
First layer	0.39±0.17	0.44±0.21	0.43±0.19
Second layer	0.86±0.13	0.69±0.22	0.72±0.23
Third layer	0.95±0.03	0.81±0.13	0.87±0.15

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

(Lee, Grosse, Ranganath and Ng, ICML 2009)

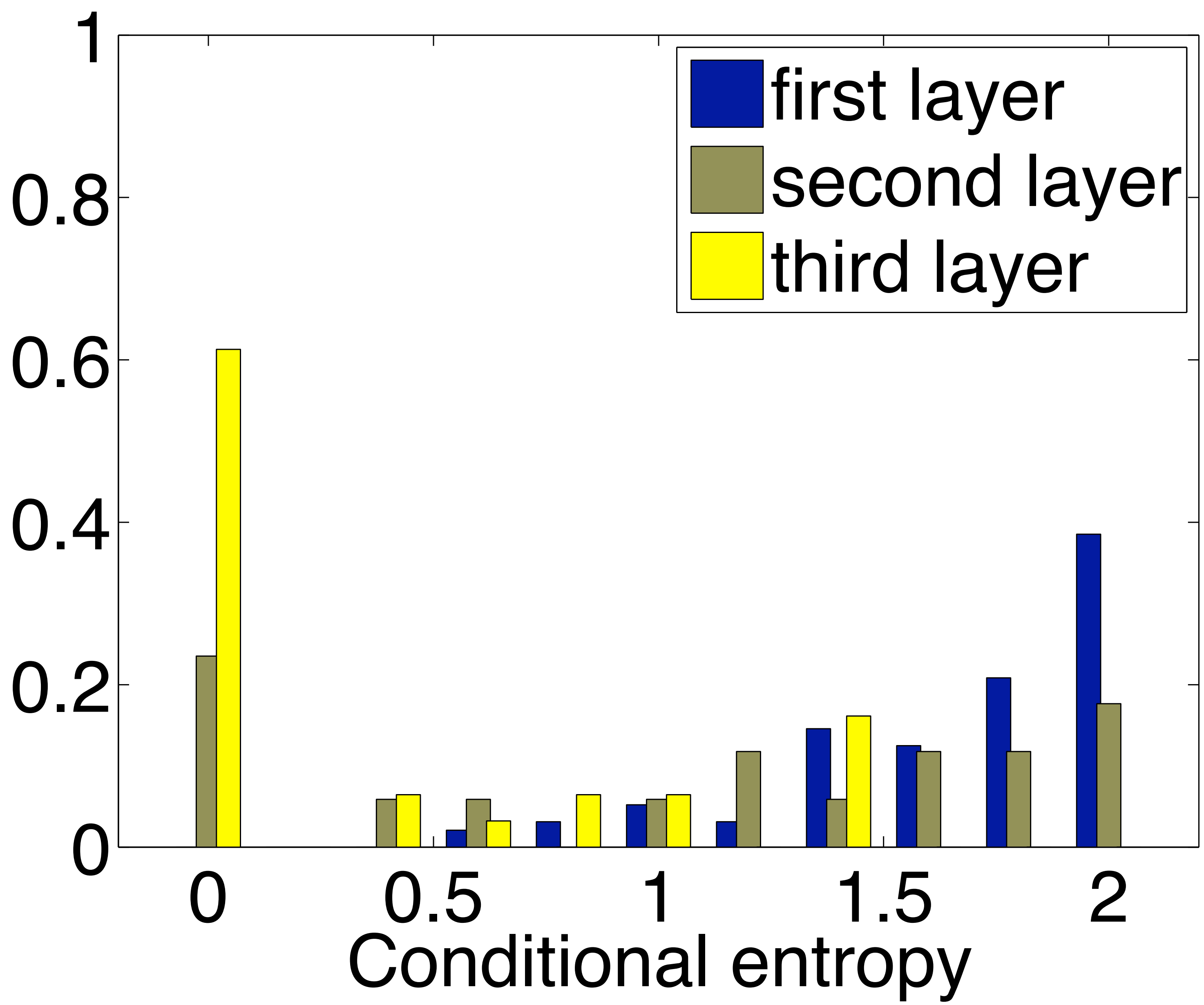
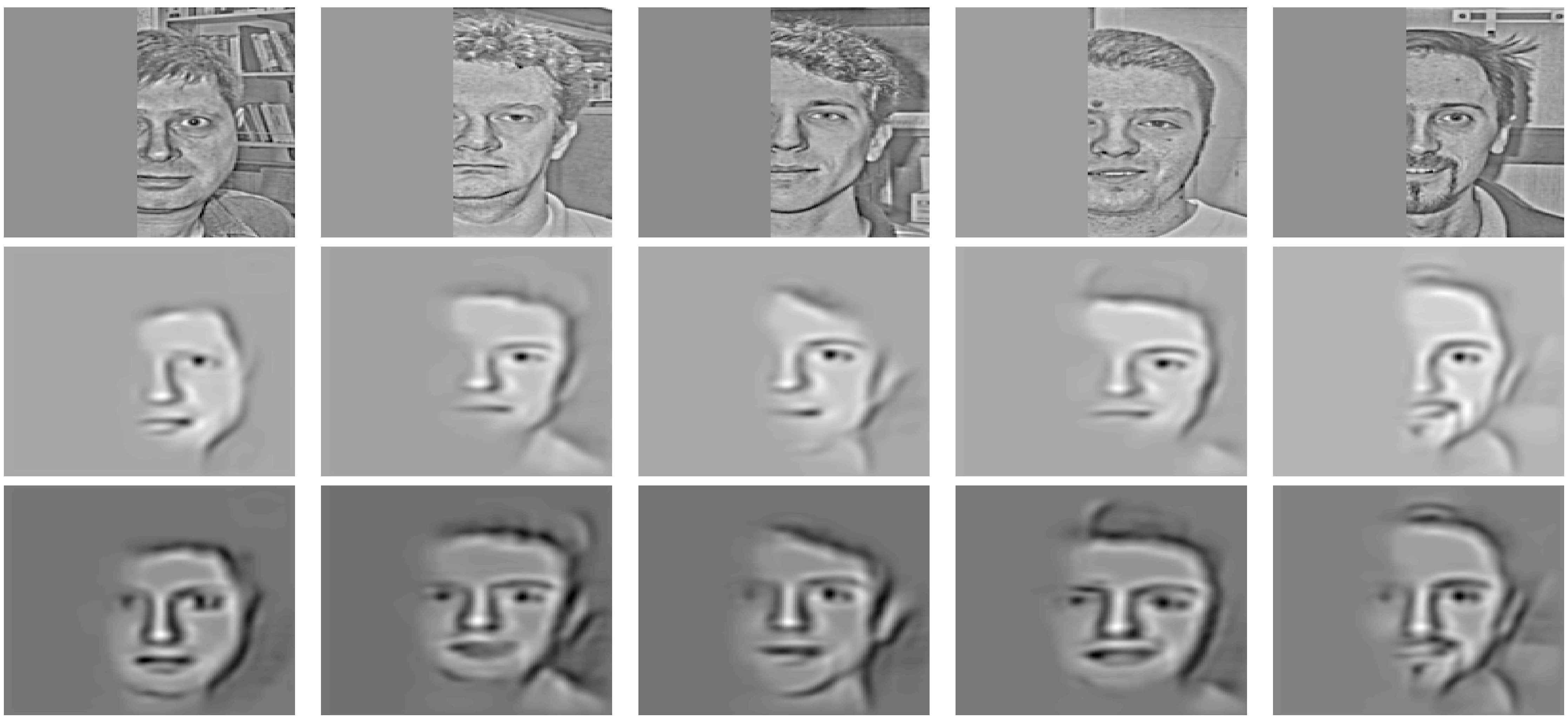


Figure 5. Histogram of conditional entropy for the representation learned from the mixture of four object classes.

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

(Lee, Grosse, Ranganath and Ng, ICML 2009)

Reconstruction de visages



Unsupervised feature learning for audio classification using convolutional deep belief networks

(Lee, Largman, Pham and Ng, NIPS 2009)

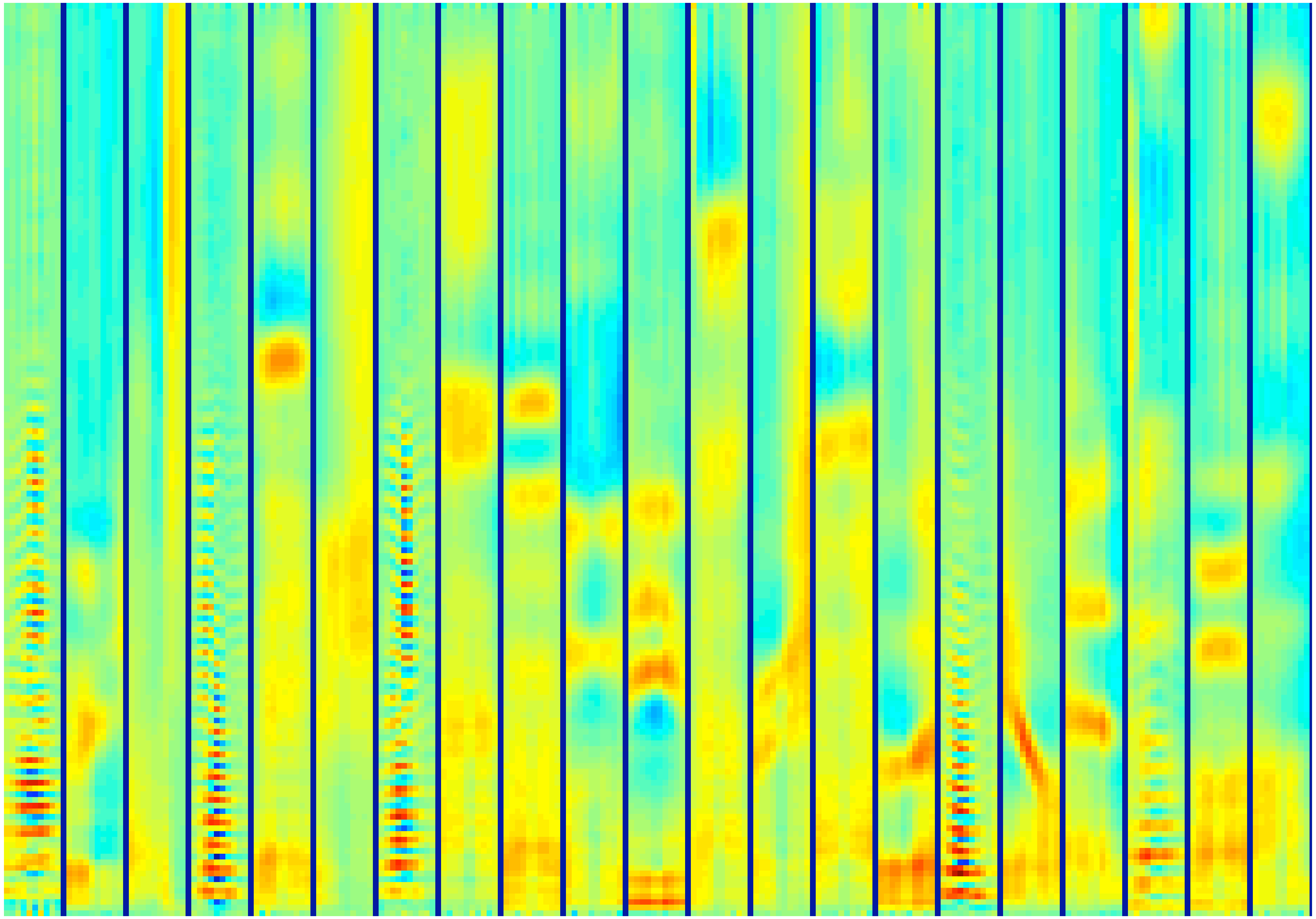
- Possible de considérer un signal sonore comme une image, via son spectrogramme
- Un filtre couvre tous les canaux de fréquences, mais seulement 6 “frames”
- Utilise la PCA pour réduire le nombre de canaux de fréquences

Unsupervised feature learning for audio classification using convolutional deep belief networks

(Lee, Largman, Pham and Ng, NIPS 2009)

Filtres appris

high freq.

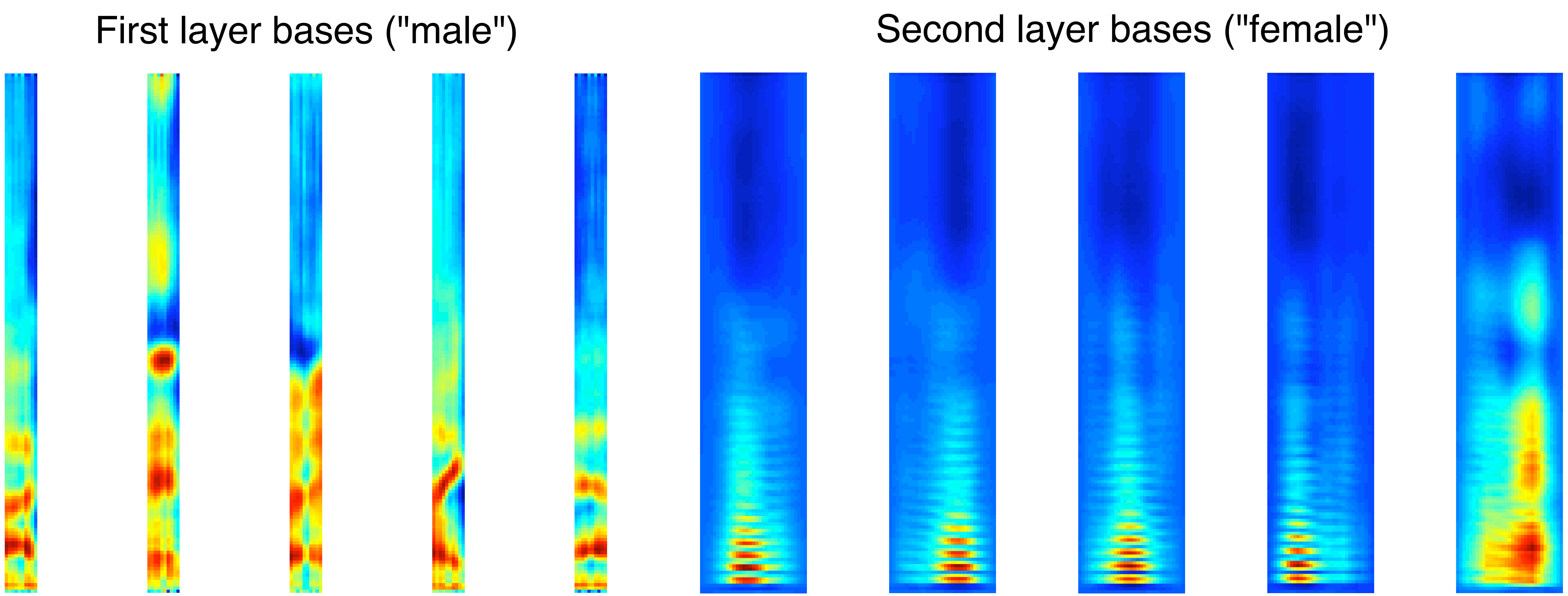


low freq.

Unsupervised feature learning for audio classification using convolutional deep belief networks

(Lee, Largman, Pham and Ng, NIPS 2009)

Filtres appris



Unsupervised feature learning for audio classification using convolutional deep belief networks

(Lee, Largman, Pham and Ng, NIPS 2009)

Table 1: Test classification accuracy for speaker identification using summary statistics

#training utterances per speaker	RAW	MFCC	CDBN L1	CDBN L2	CDBN L1+L2
1	46.7%	54.4%	**74.5%**	62.8%	72.8%
2	43.5%	69.9%	**76.7%**	66.2%	**76.7%**
3	67.9%	76.5%	91.3%	84.3%	**91.8%**
5	80.6%	82.6%	93.7%	89.6%	**93.8%**
8	90.4%	92.0%	**97.9%**	95.2%	97.0%

Table 2: Test classification accuracy for speaker identification using all frames

#training utterances per speaker	MFCC ([16]'s method)	CDBN	MFCC ([16]) + CDBN
1	40.2%	90.0%	**90.7%**
2	87.9%	97.9%	**98.7%**
3	95.9%	98.7%	**99.2%**
5	99.2%	99.2%	**99.6%**
8	99.7%	99.7%	**100.0%**

Unsupervised feature learning for audio classification using convolutional deep belief networks

(Lee, Largman, Pham and Ng, NIPS 2009)

Table 3: Test accuracy for gender classification problem

#training utterances per gender	RAW	MFCC	CDBN L1	CDBN L2	CDBN L1+L2
1	68.4%	58.5%	78.5%	**85.8%**	83.6%
2	76.7%	78.7%	86.0%	**92.5%**	92.3%
3	79.5%	84.1%	88.9%	**94.2%**	**94.2%**
5	84.4%	86.9%	93.1%	**95.8%**	95.6%
7	89.2%	89.0%	94.2%	**96.6%**	96.5%
10	91.3%	89.8%	94.7%	**96.7%**	96.6%

Table 4: Test accuracy for phone classification problem

#training utterances	RAW	MFCC	MFCC ([15]'s method)	CDBN L1	MFCC+CDBN L1 ([15])
100	36.9%	58.3%	66.6%	53.7%	**67.2%**
200	37.8%	61.5%	70.3%	56.7%	**71.0%**
500	38.7%	64.9%	74.1%	59.7%	**75.1%**
1000	39.0%	67.2%	76.3%	61.6%	**77.1%**
2000	39.2%	69.2%	78.4%	63.1%	**79.2%**
3696	39.4%	70.8%	79.6%	64.4%	**80.3%**

Unsupervised feature learning for audio classification using convolutional deep belief networks

(Lee, Largman, Pham and Ng, NIPS 2009)

Table 5: Test accuracy for 5-way music genre classification

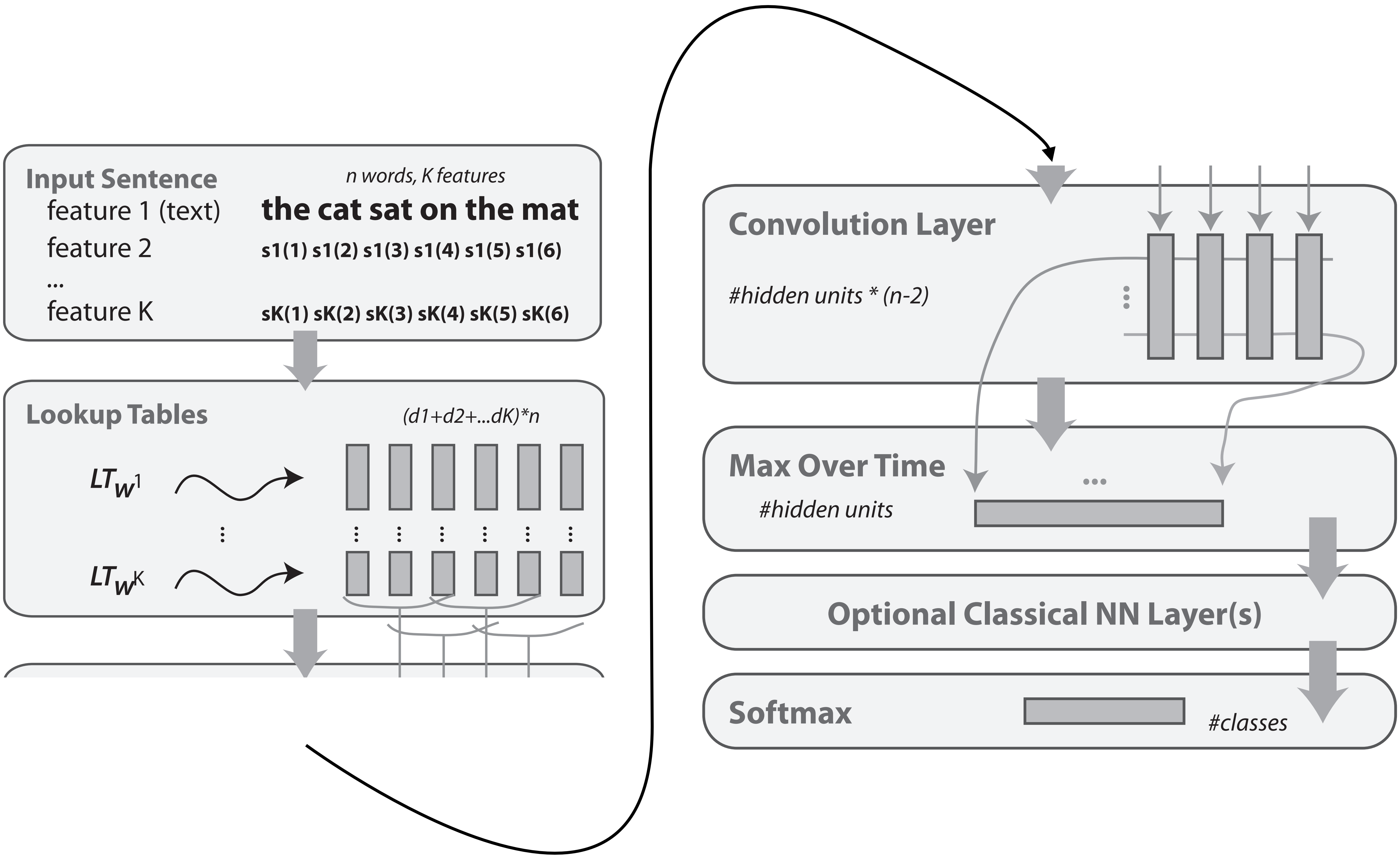
Train examples	RAW	MFCC	CDBN L1	CDBN L2	CDBN L1+L2
1	51.6%	54.0%	**66.1%**	62.5%	64.3%
2	57.0%	62.1%	**69.7%**	67.9%	69.5%
3	59.7%	65.3%	**70.0%**	66.7%	69.5%
5	65.8%	68.3%	**73.1%**	69.2%	72.7%

Table 6: Test accuracy for 4-way artist identification

Train examples	RAW	MFCC	CDBN L1	CDBN L2	CDBN L1+L2
1	56.0%	63.7%	67.6%	67.7%	**69.2%**
2	69.4%	66.1%	76.1%	74.2%	**76.3%**
3	73.9%	67.9%	78.0%	75.8%	**78.7%**
5	79.4%	71.6%	80.9%	**81.9%**	81.4%

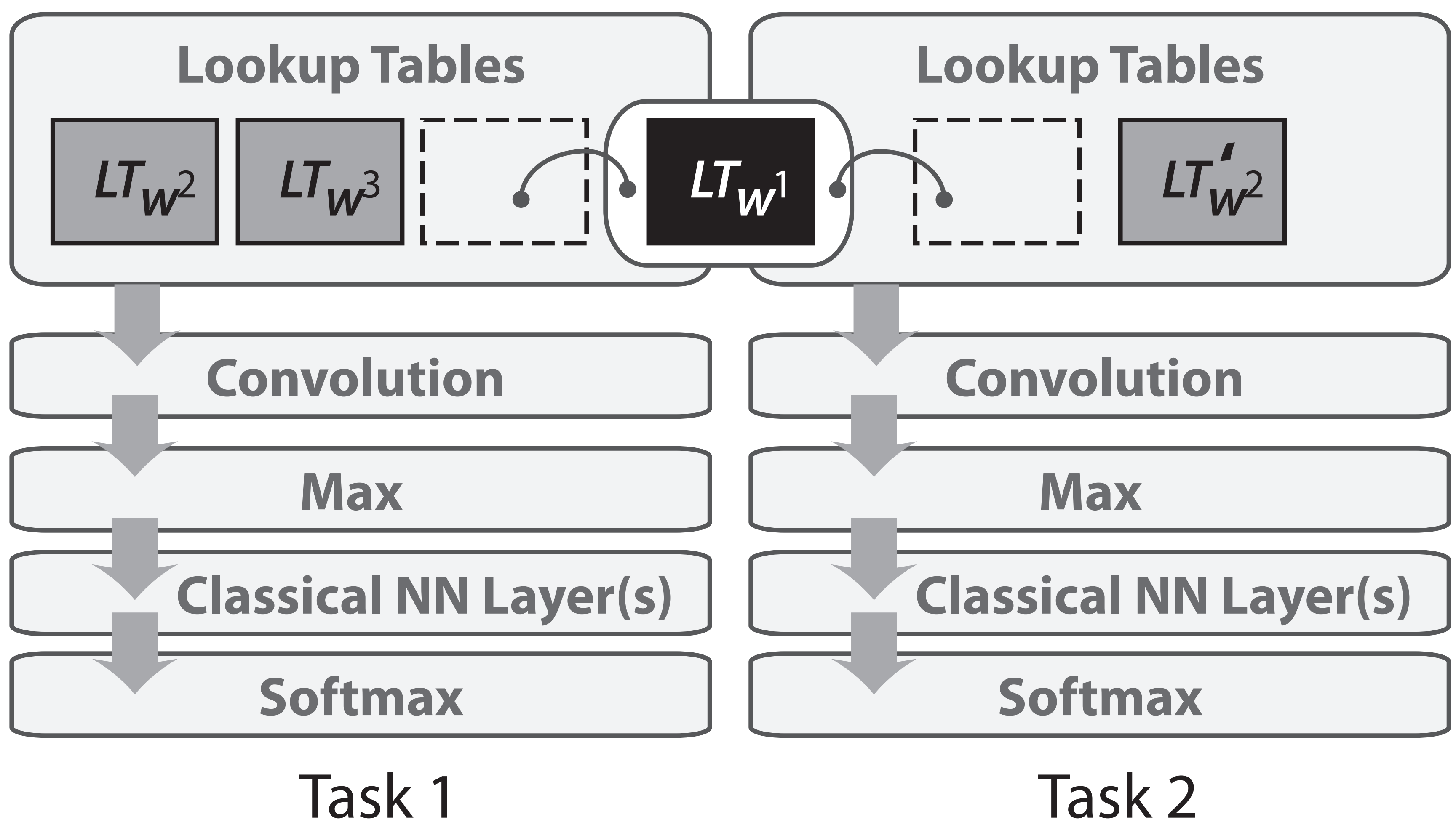
A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning

(Collobert and Weston, ICML 2008)



A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning

(Collobert and Weston, ICML 2008)



A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning

(Collobert and Weston, ICML 2008)

- Utilise une tâche de modélisation du langage comme tâche non-supervisée

$$\sum_{s \in \mathcal{S}} \sum_{w \in \mathcal{D}} \max\left(0,\, 1 - f(s) + f(s^{w})\right)$$

représentation d'une fenêtre de texte de Wikipedia

même fenêtre, mais avec le mot w au milieu

A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning

(Collobert and Weston, ICML 2008)

	wsz=15	wsz=50	wsz=100
SRL	16.54	17.33	18.40
SRL + POS	15.99	16.57	16.53
SRL + Chunking	16.42	16.39	16.48
SRL + NER	16.67	17.29	17.21
SRL + Synonyms	15.46	15.17	15.17
SRL + Language model	14.42	14.30	14.46
SRL + POS + Chunking	16.46	15.95	16.41
SRL + POS + NER	16.45	16.89	16.29
SRL + POS + Chunking + NER	16.33	16.36	16.27
SRL + POS + Chunking + NER + Synonyms	15.71	14.76	15.48
SRL + POS + Chunking + NER + Language model	14.63	14.44	14.50
État de l'art (Pradhan et al. 2004)	—	16.54	—

A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning

(Collobert and Weston, ICML 2008)

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869
SPAIN	CHRIST	PLAYSTATION	YELLOWISH	SMASHED
ITALY	GOD	DREAMCAST	GREENISH	RIPPED
RUSSIA	RESURRECTION	PSNUMBER	BROWNISH	BRUSHED
POLAND	PRAYER	SNES	BLUISH	HURLED
ENGLAND	YAHWEH	WII	CREAMY	GRABBED
DENMARK	JOSEPHUS	NES	WHITISH	TOSSED
GERMANY	MOSES	NINTENDO	BLACKISH	SQUEEZED
PORTUGAL	SIN	GAMECUBE	SILVERY	BLASTED
SWEDEN	HEAVEN	PSP	GREYISH	TANGLED
AUSTRIA	SALVATION	AMIGA	PALER	SLASHED

(montrer visualisation t-SNE)

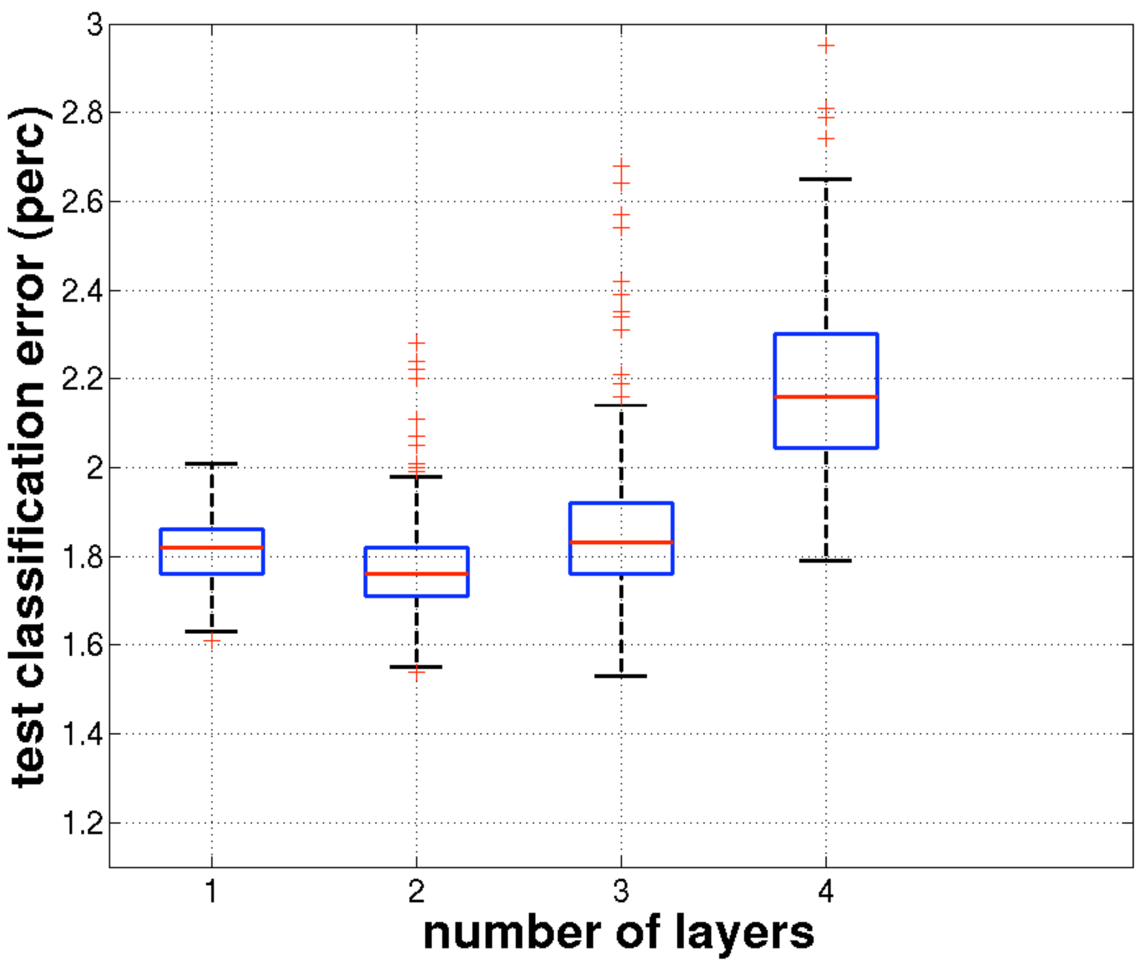
Réseaux à convolution et “deep learning”

- Y a-t-il d’autres façon d’initialiser des réseaux à convolution de façon non-supervisée?
- Y a-t-il d’autres tâches qui seraient mieux résolues par un réseau à convolution?
- Y a-t-il de meilleures architectures de réseaux à convolution?

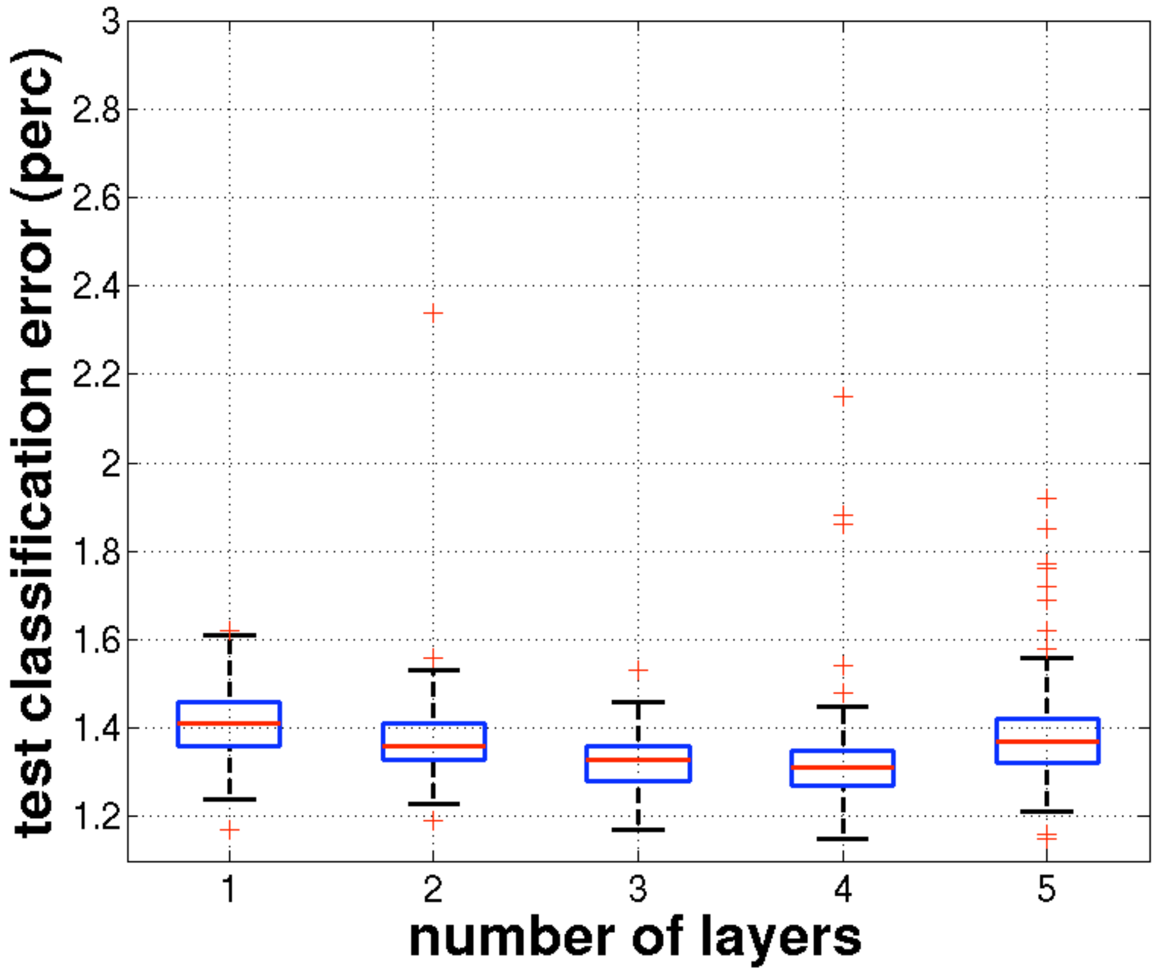
! Pistes de recherche !

Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)



A) sans pré-entraînement

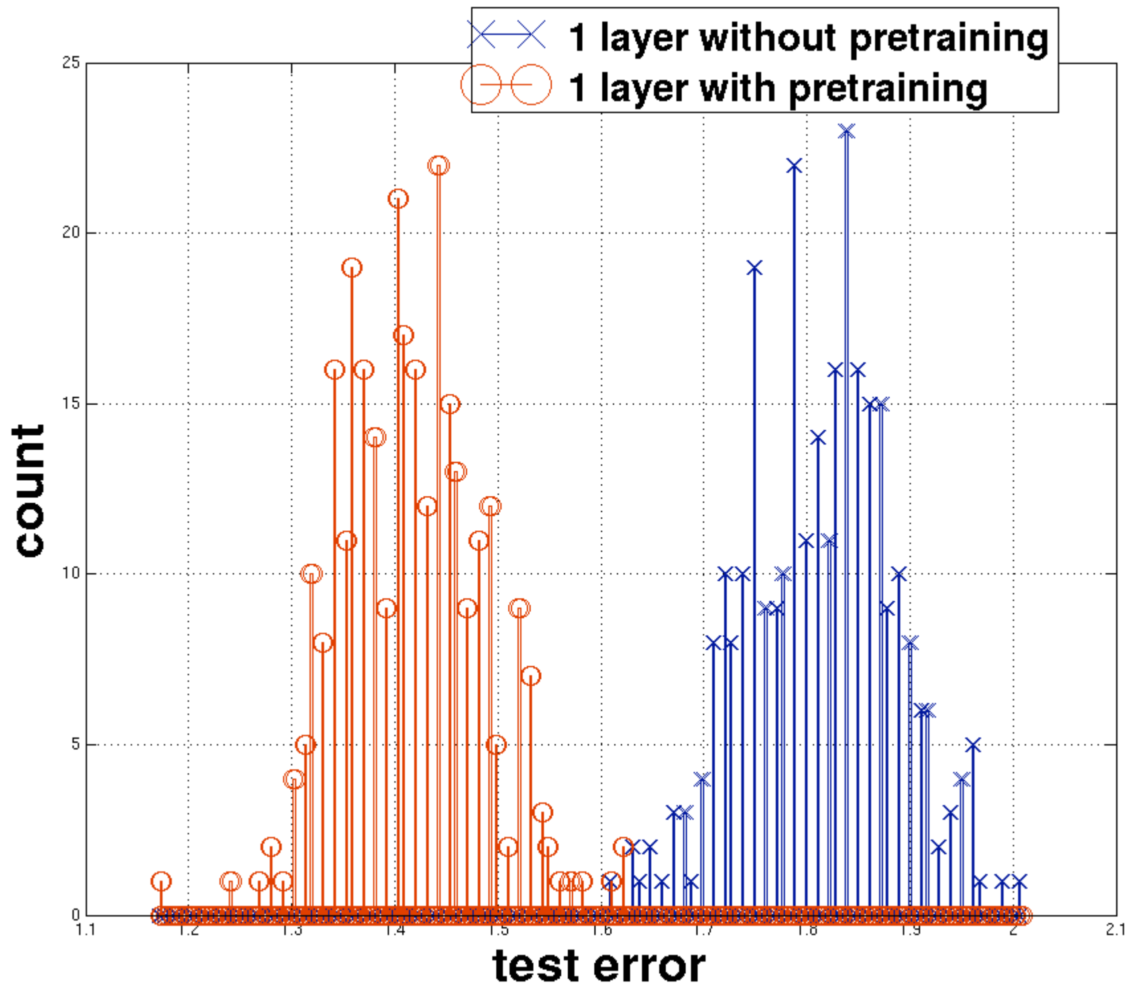


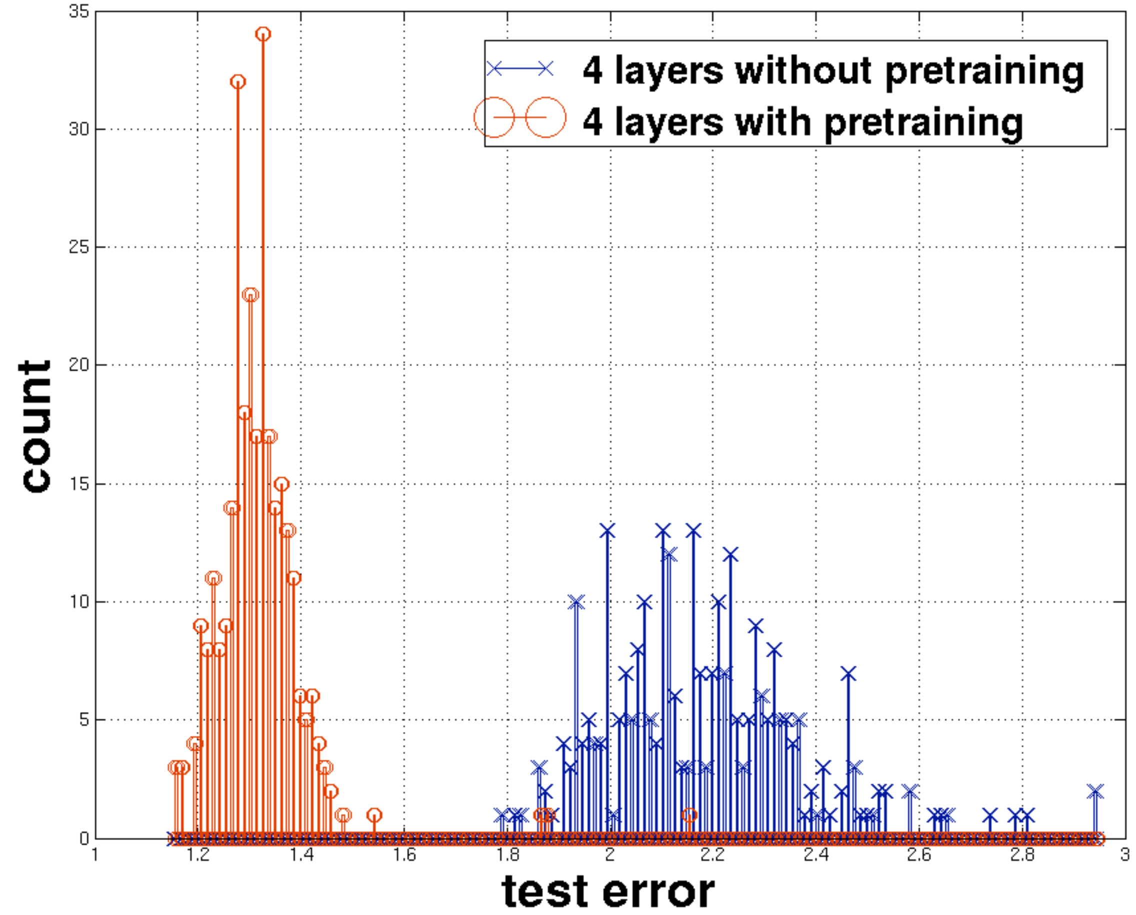
B) avec pré-entraînement

Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

Variance liée à l'initialisation aléatoire

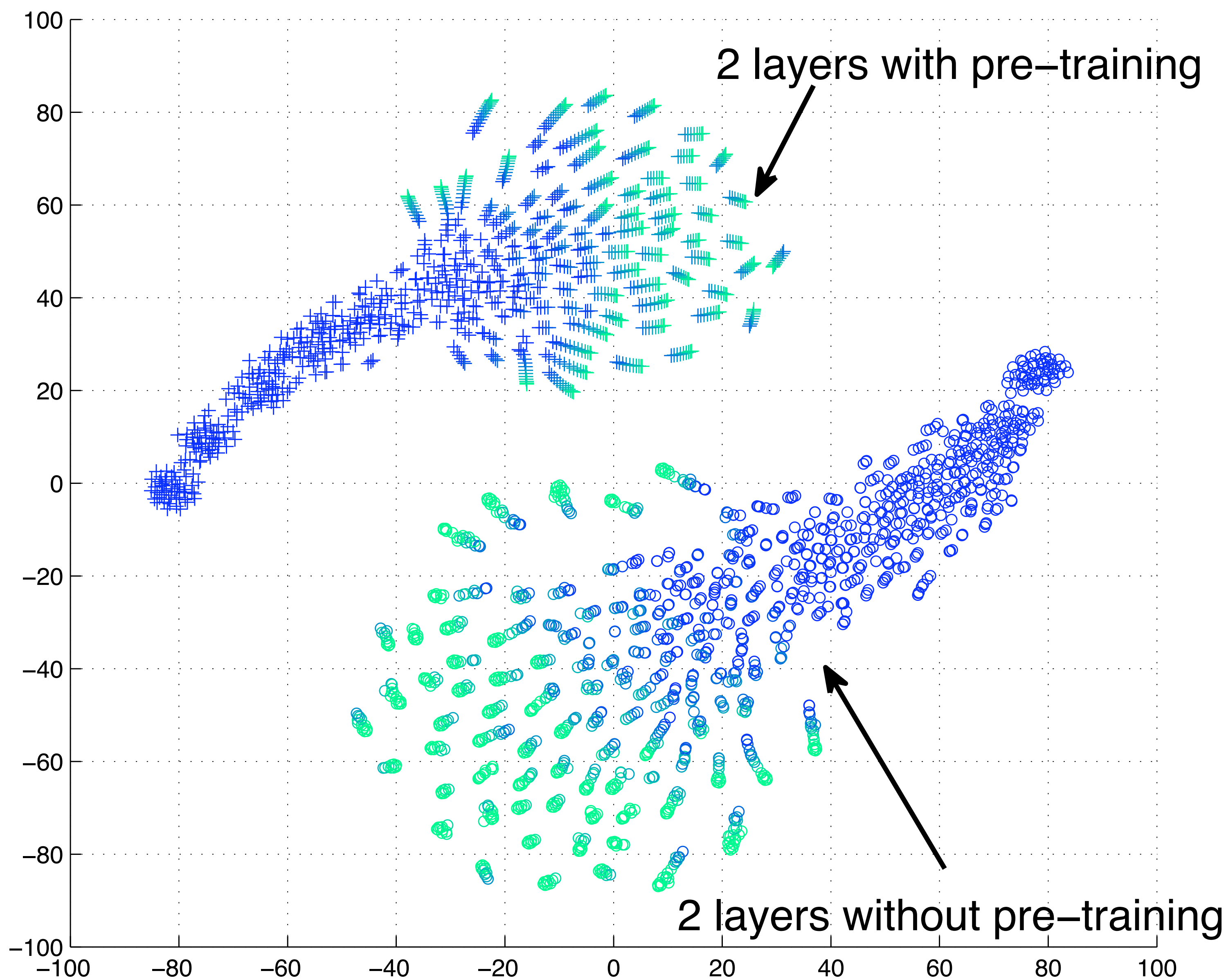




Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

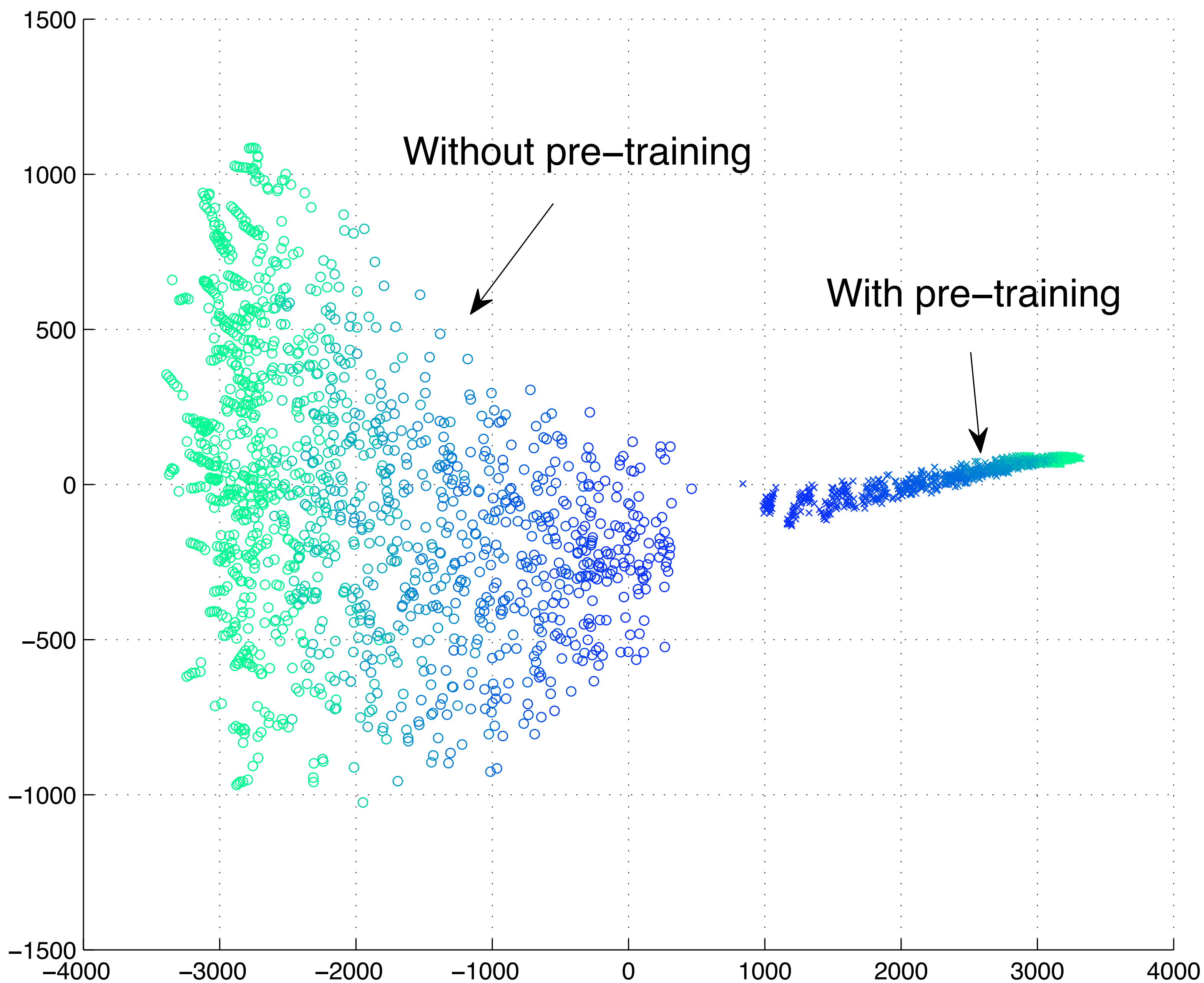
Variance liée à l'initialisation aléatoire (t-SNE)



Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

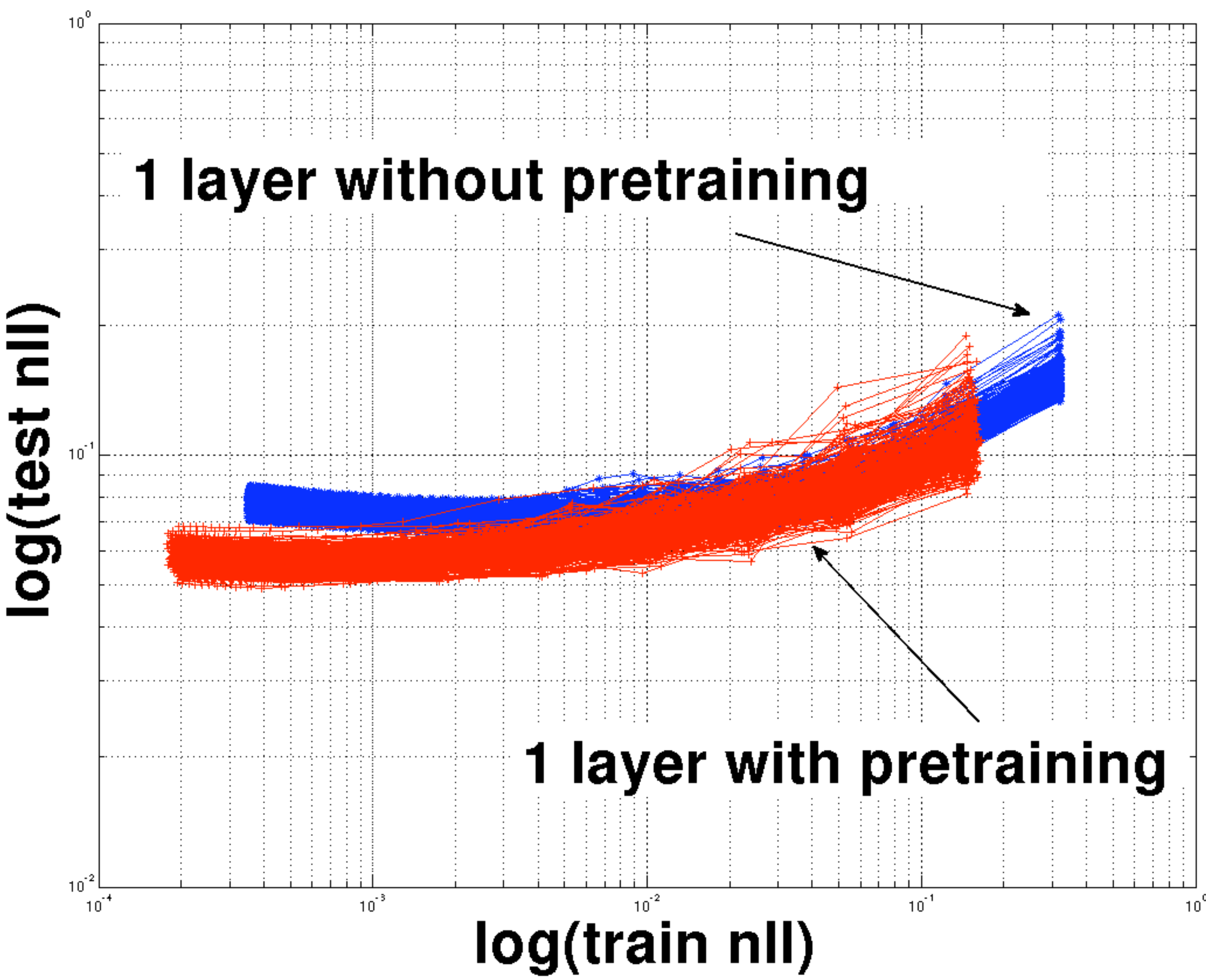
Variance liée à l'initialisation aléatoire (ISOMAP)

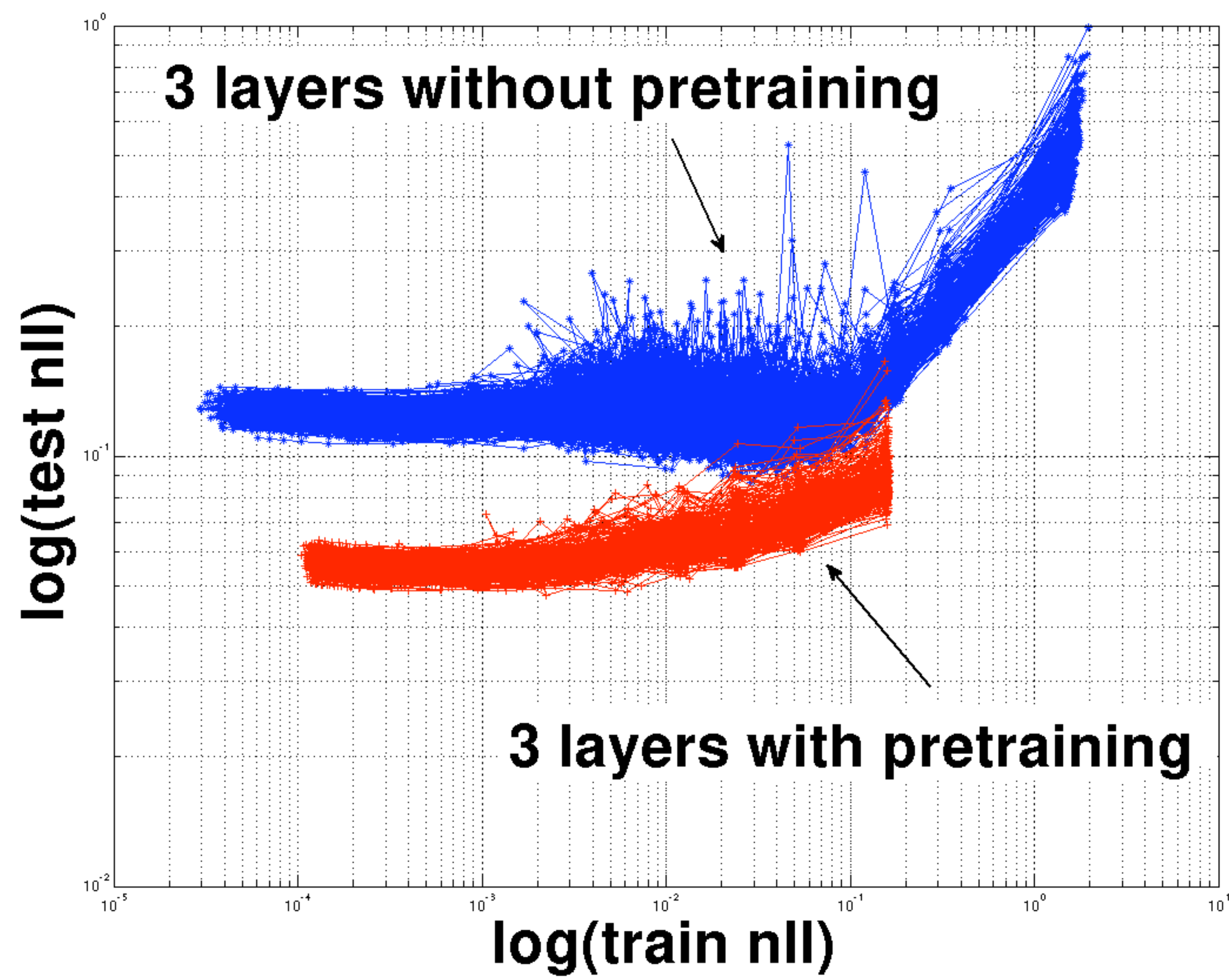


Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

Erreur d'entraînement vs erreur de test

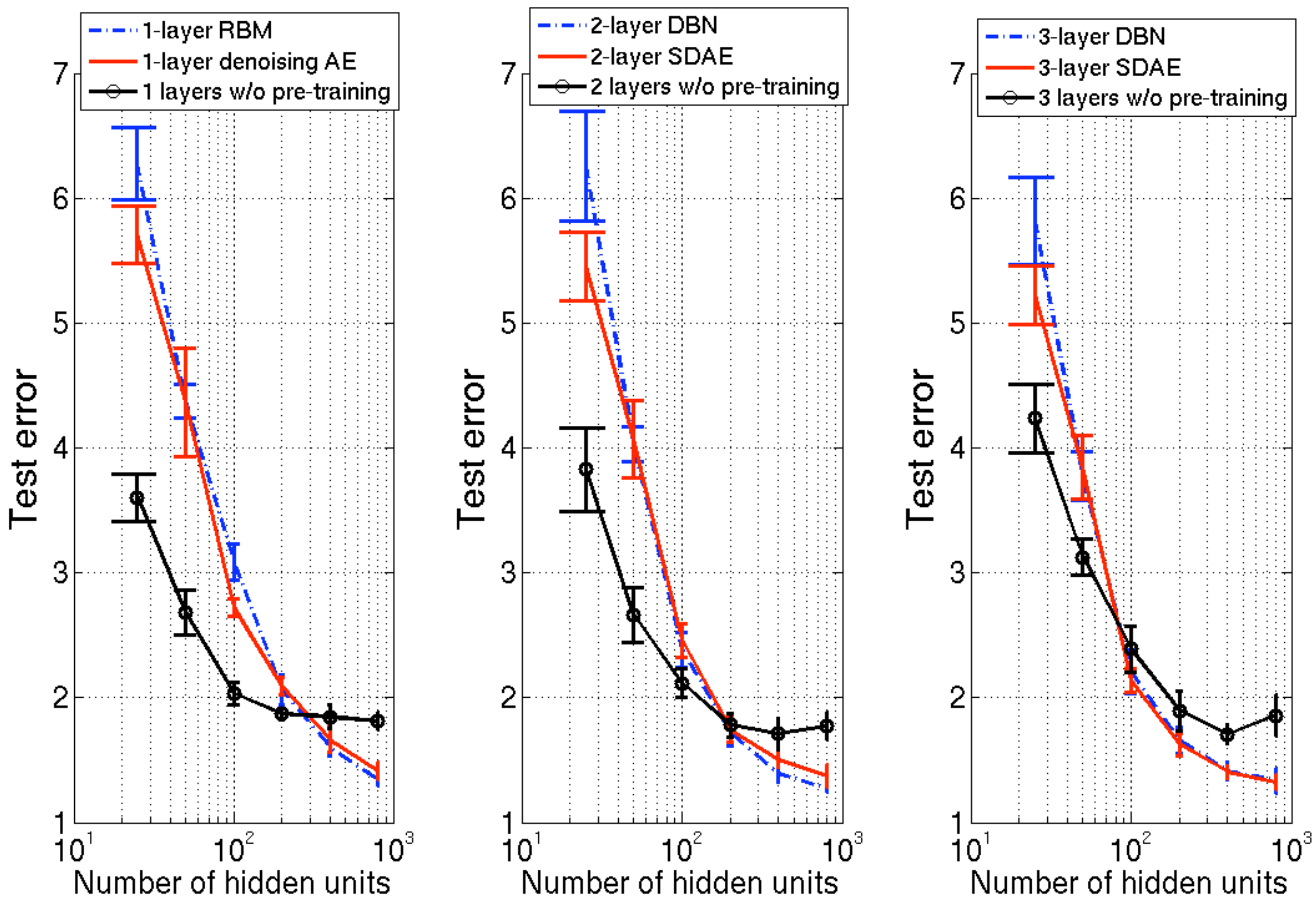




Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

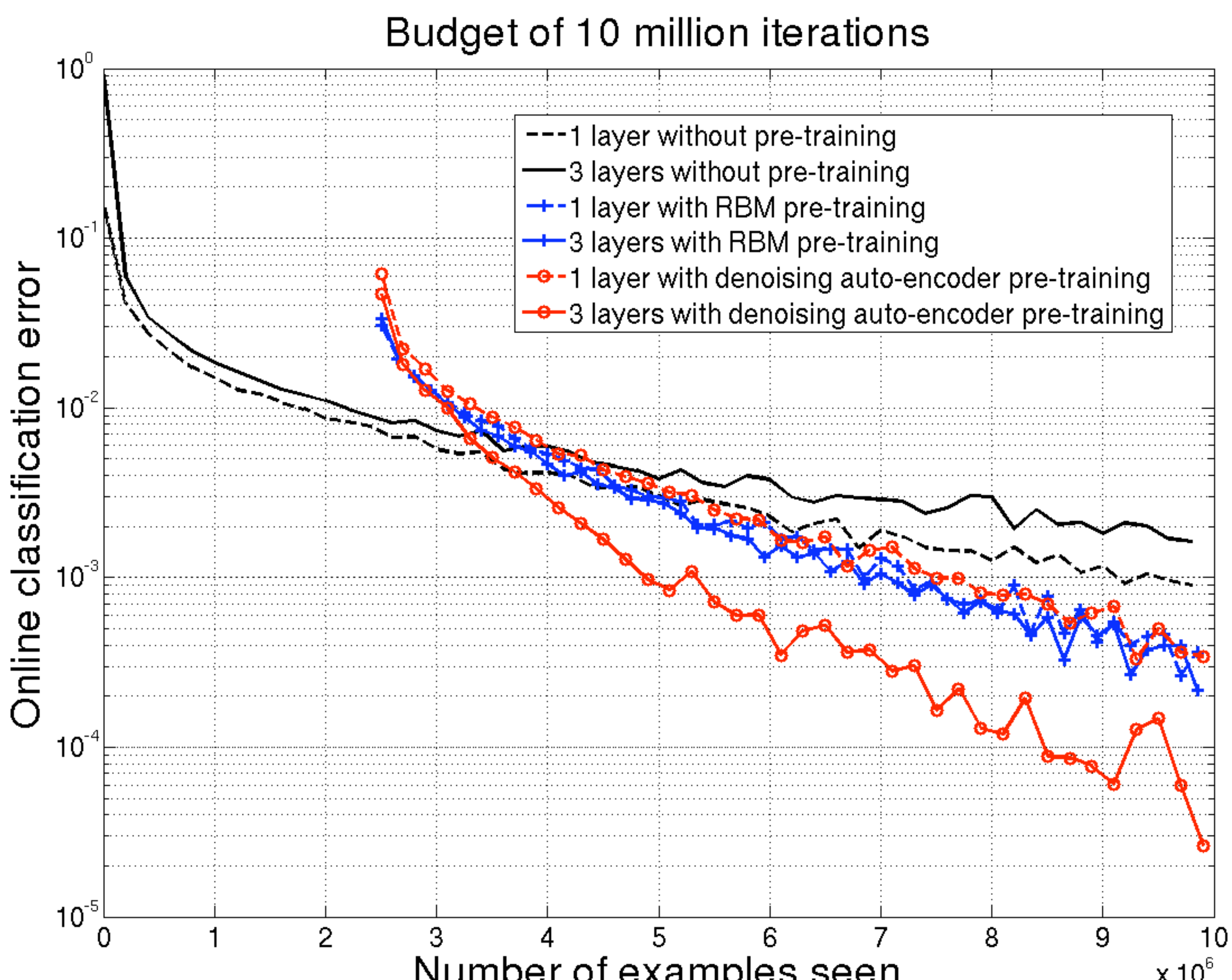
Variation du nombre de neurones par couche



Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

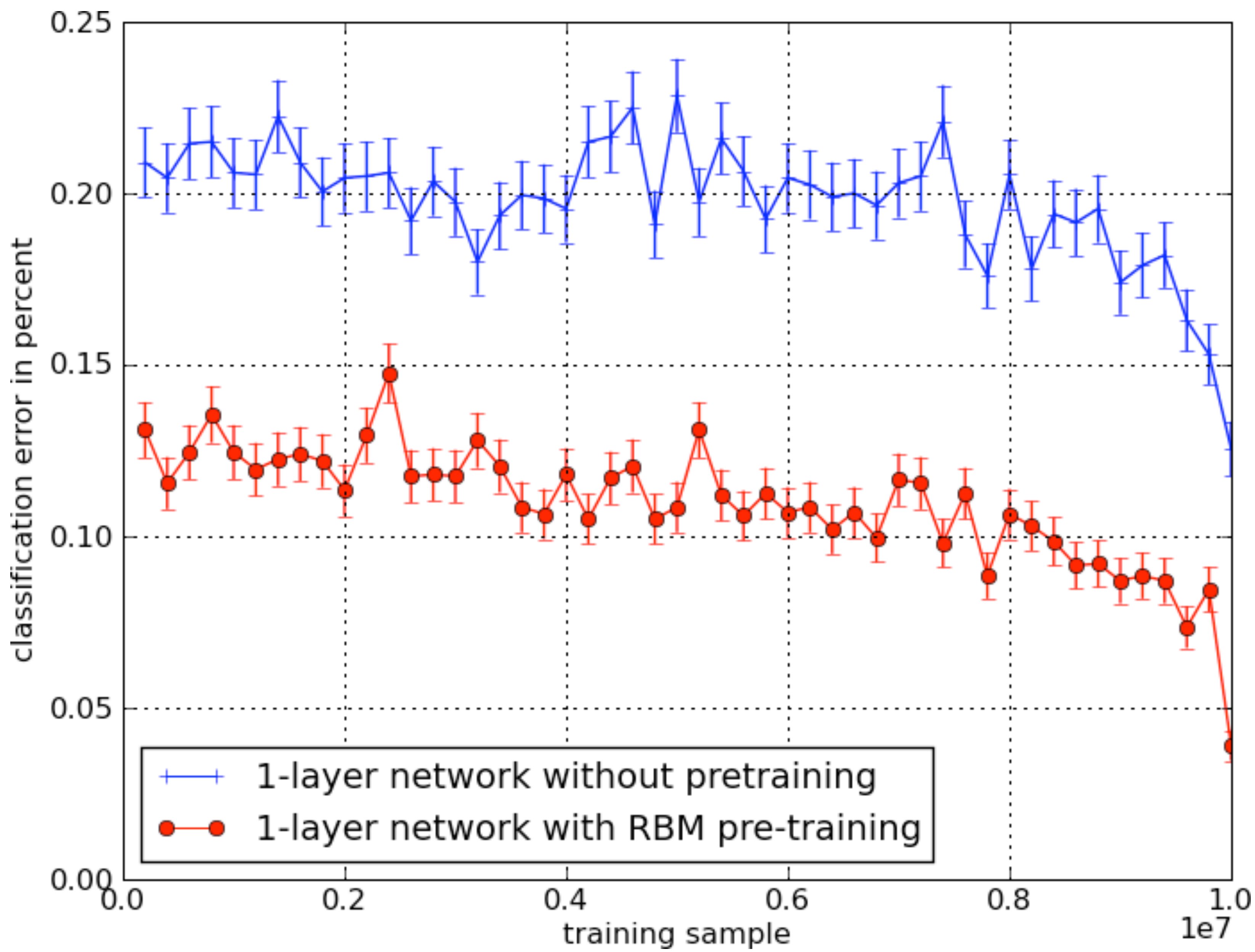
Cas spécial: infinité d'exemples



Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

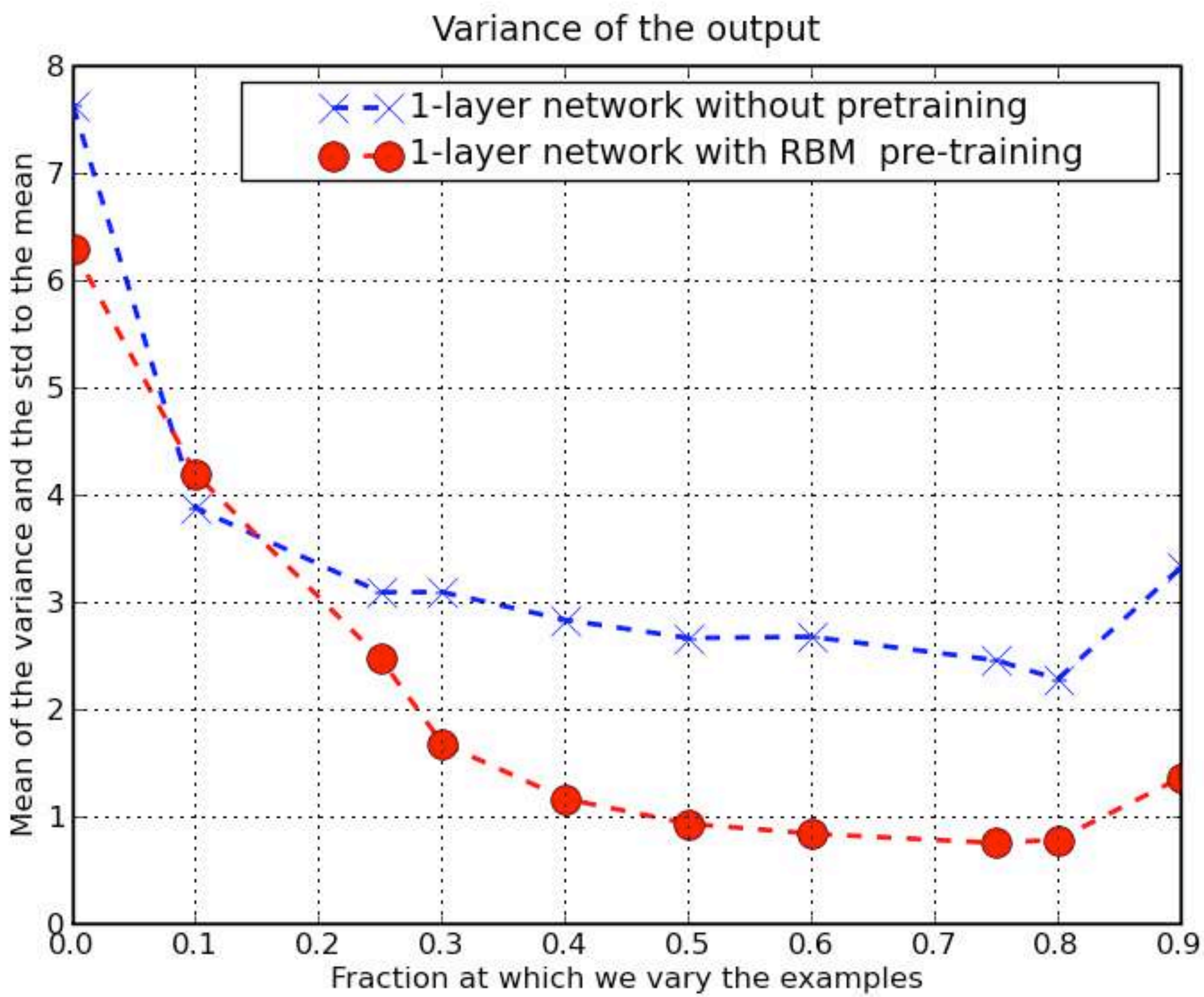
Cas spécial: infinité d'exemples



Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

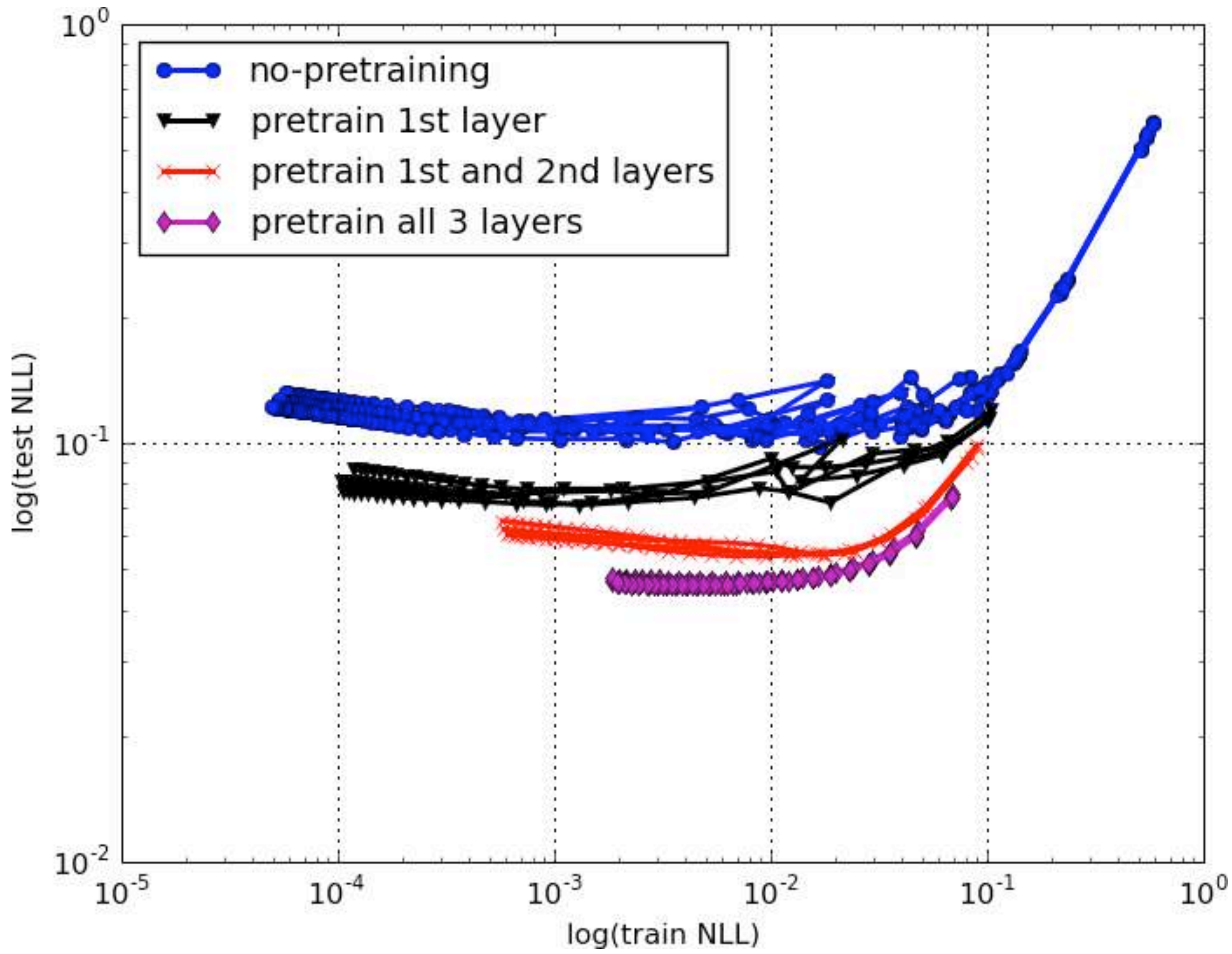
Cas spécial: infinité d'exemples



Why Does Unsupervised Pre-training Help Deep Learning?

(Erhan, Bengio, Courville, Manzagol and Vincent, JMLR 2010)

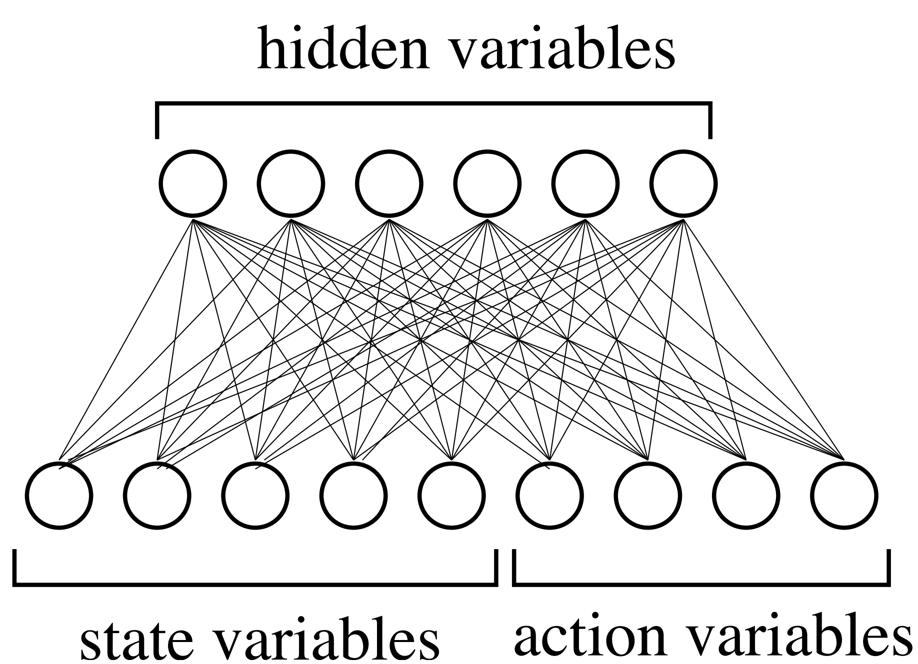
Pré-entraînement du différentes couches

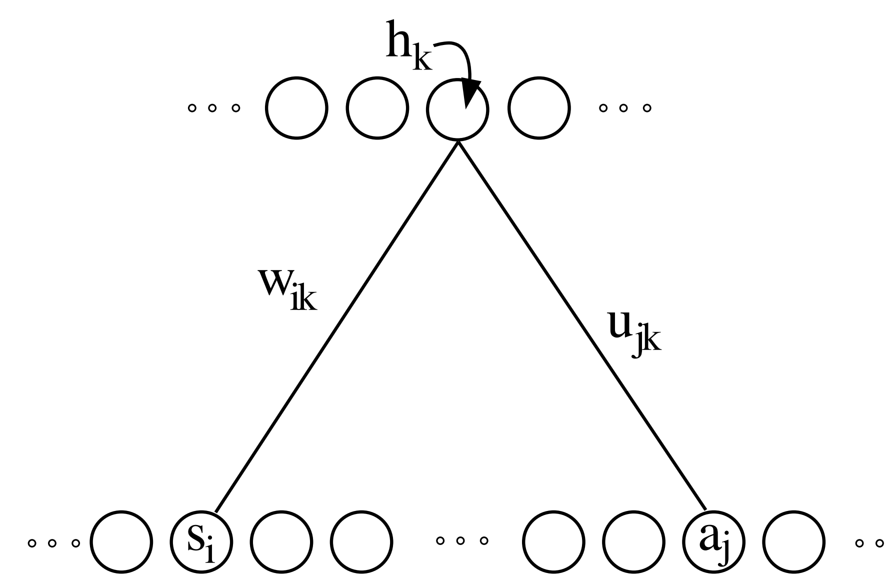


Reinforcement Learning with Factored States and Actions

(Sallans and Hinton, JMLR 2004)

- Idée: utiliser la "free energy" comme prédicteur des "Q values"





entraîné par SARSA

$$\widehat{Q}(\mathbf{s},\mathbf{a}) = -F(\mathbf{s},\mathbf{a}) = \sum_{k=1}^{K} \log(1+\exp(\mathbf{W}_{\cdot k}^{\top}\mathbf{s} + \mathbf{U}_{\cdot k}^{\top}\mathbf{a}))$$

Reinforcement Learning with Factored States and Actions

(Sallans and Hinton, JMLR 2004)

- Exploration/exploitation

$$P(\mathbf{a}|\mathbf{s}) = \frac{e^{-F(\mathbf{s},\mathbf{a})/T}}{Z} \approx \frac{e^{Q(\mathbf{s},\mathbf{a})/T}}{Z}$$

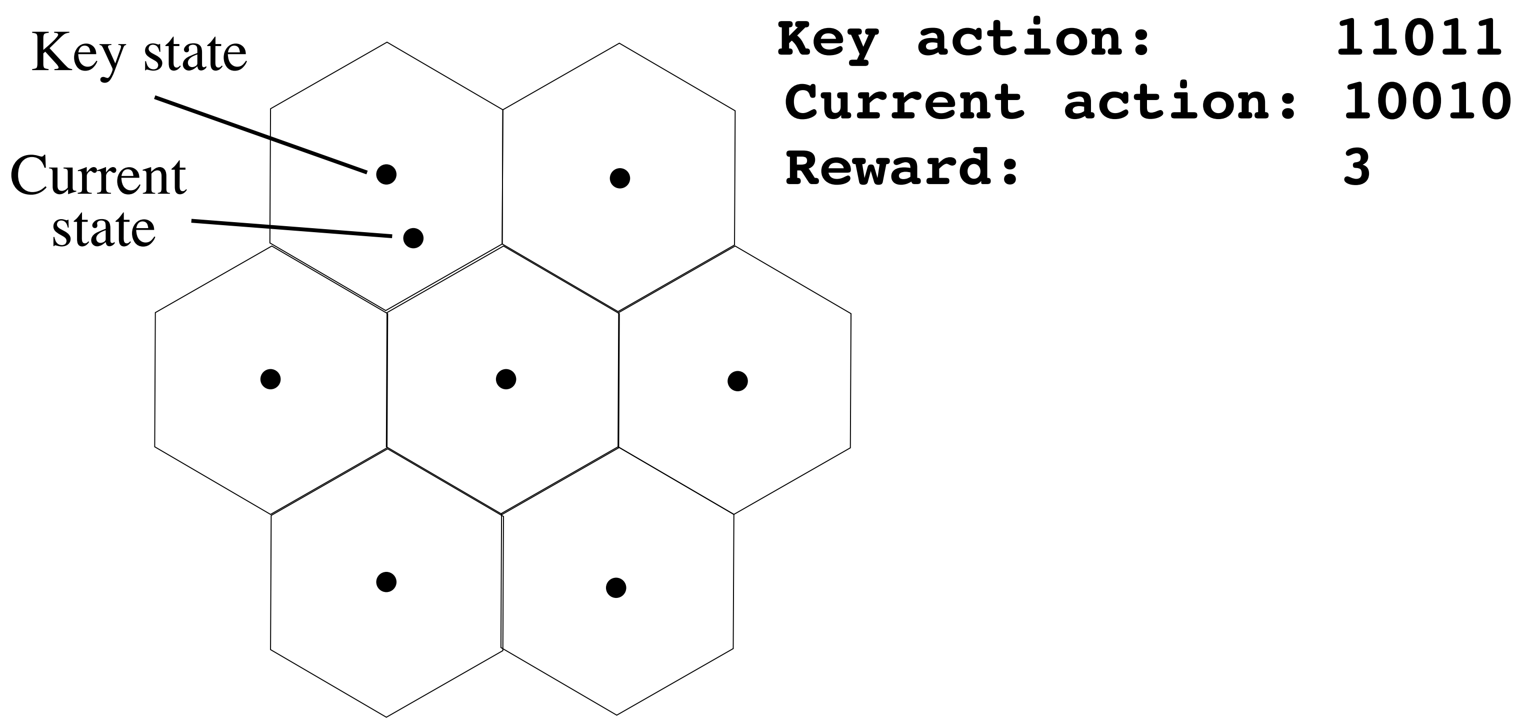
à T = 1, c'est une RBM!

- Suffit d'échantillonner de $P(\mathbf{a}|\mathbf{s})$ (Gibbs sampling)
- Si T grand, se rapproche de la maximisation

Reinforcement Learning with Factored States and Actions

(Sallans and Hinton, JMLR 2004)

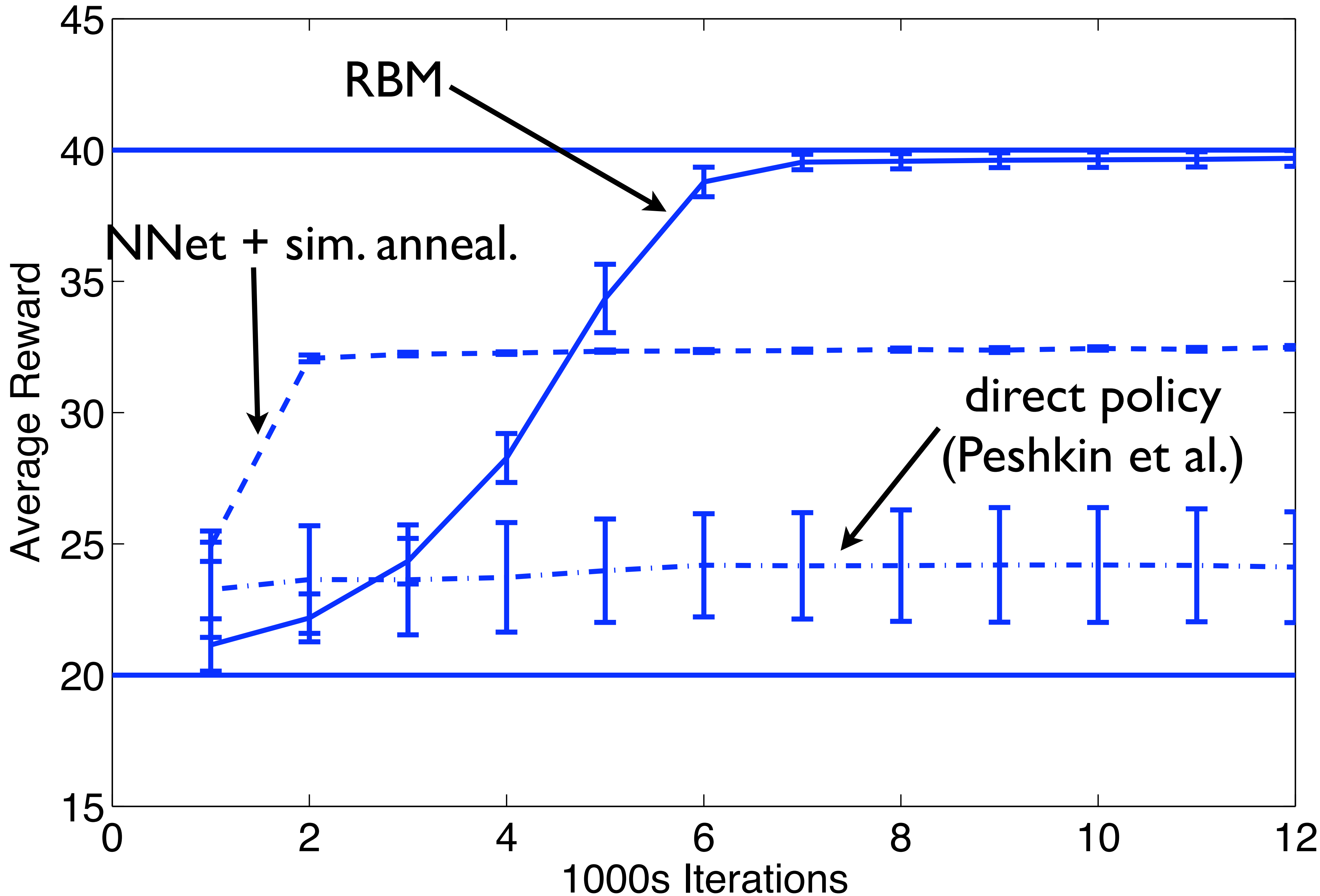
- Tâche: grand espace d'action



- État = 12 bits, action = 40 bits
- Transition d'état uniforme, renforcement immédiat

Reinforcement Learning with Factored States and Actions

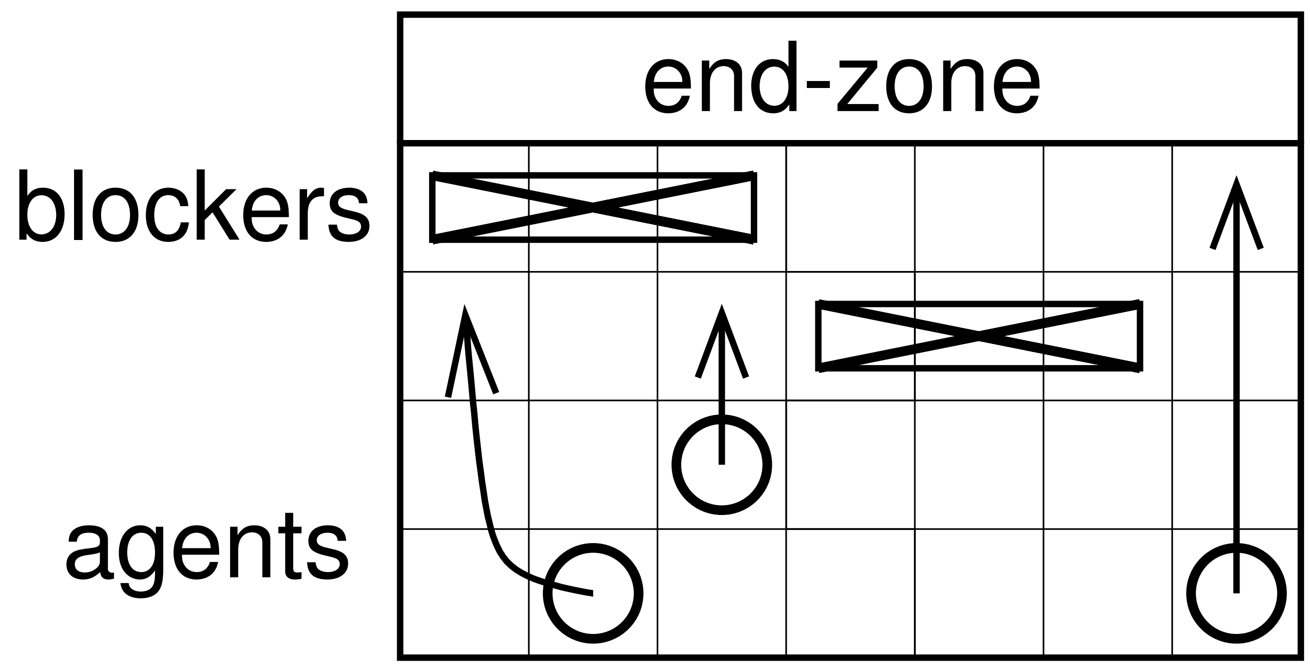
(Sallans and Hinton, JMLR 2004)



Reinforcement Learning with Factored States and Actions

(Sallans and Hinton, JMLR 2004)

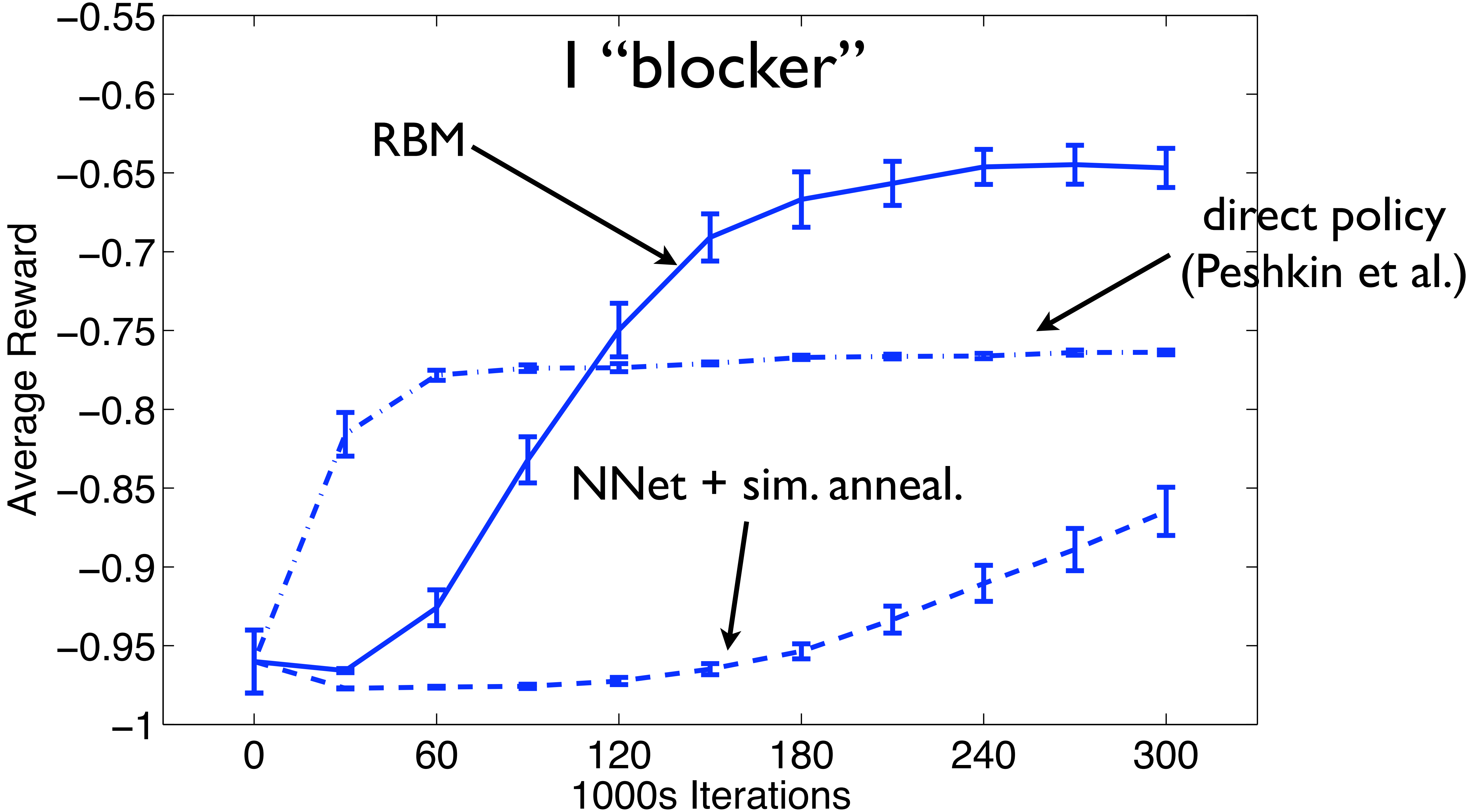
- Tâche: “blocker”



- Tâche collaborative: +1 si l’équipe gagne, -1 sinon
- Une partie dure 20/40 actions pour 1/2 “blockers”

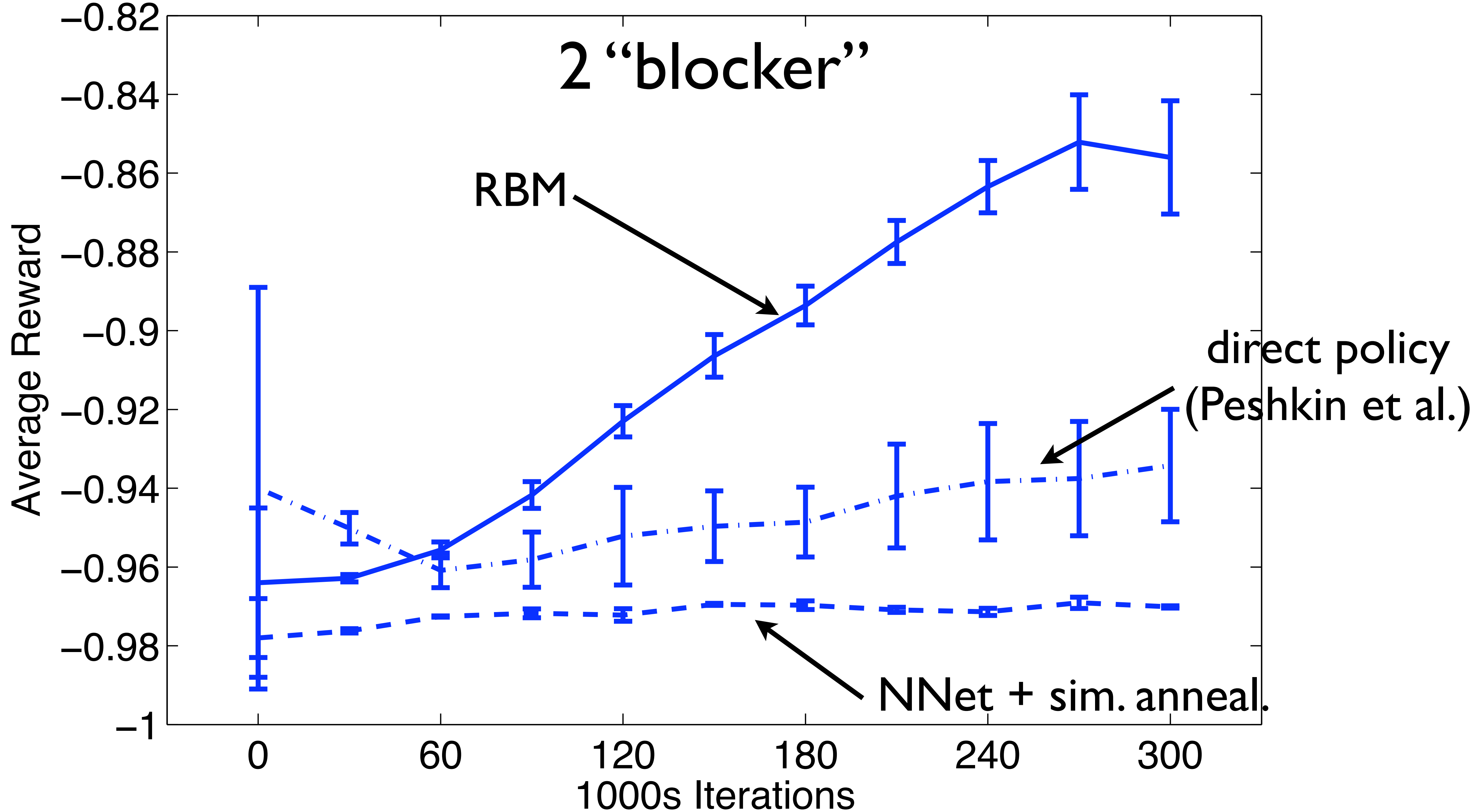
Reinforcement Learning with Factored States and Actions

(Sallans and Hinton, JMLR 2004)



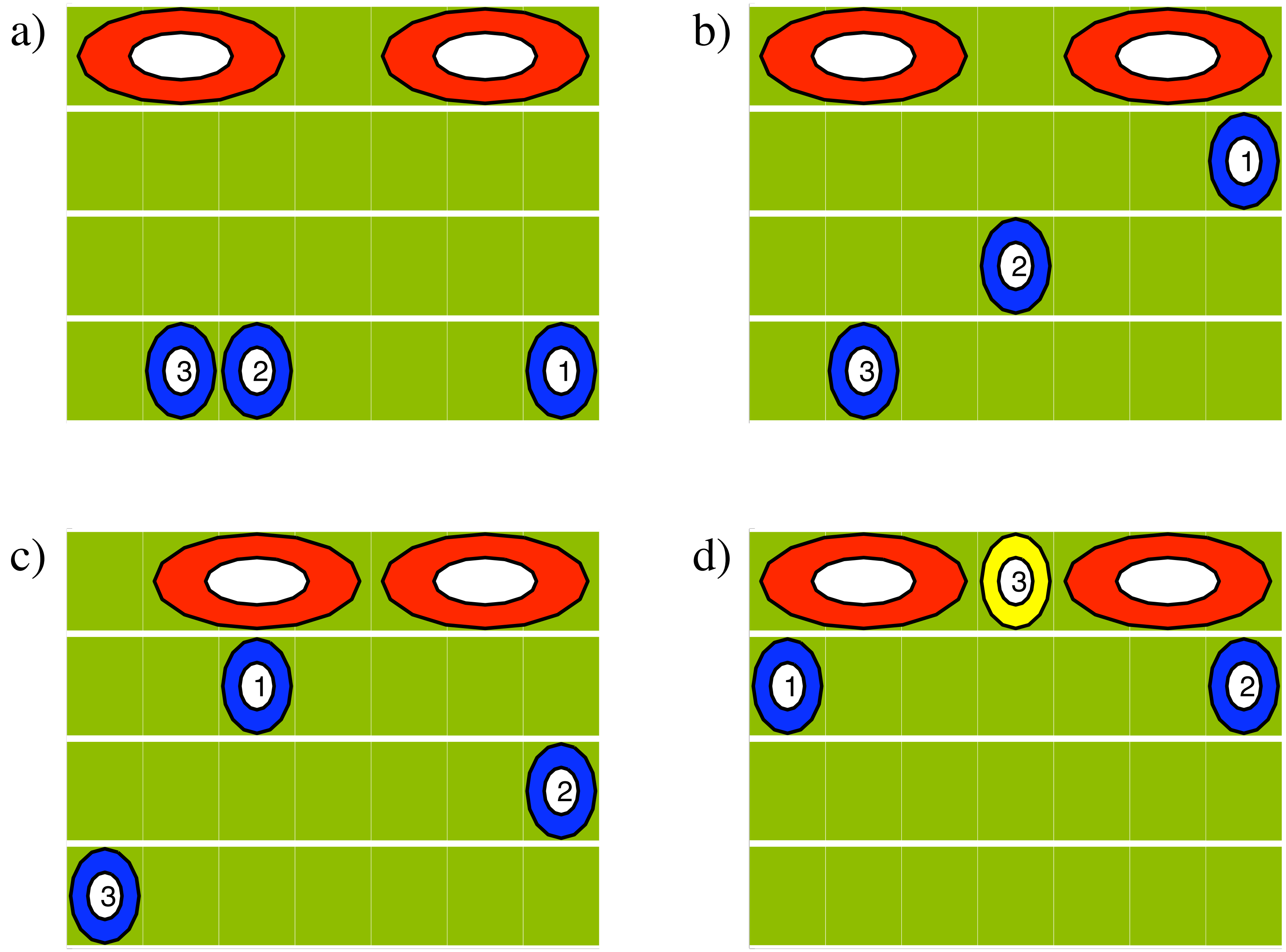
Reinforcement Learning with Factored States and Actions

(Sallans and Hinton, JMLR 2004)



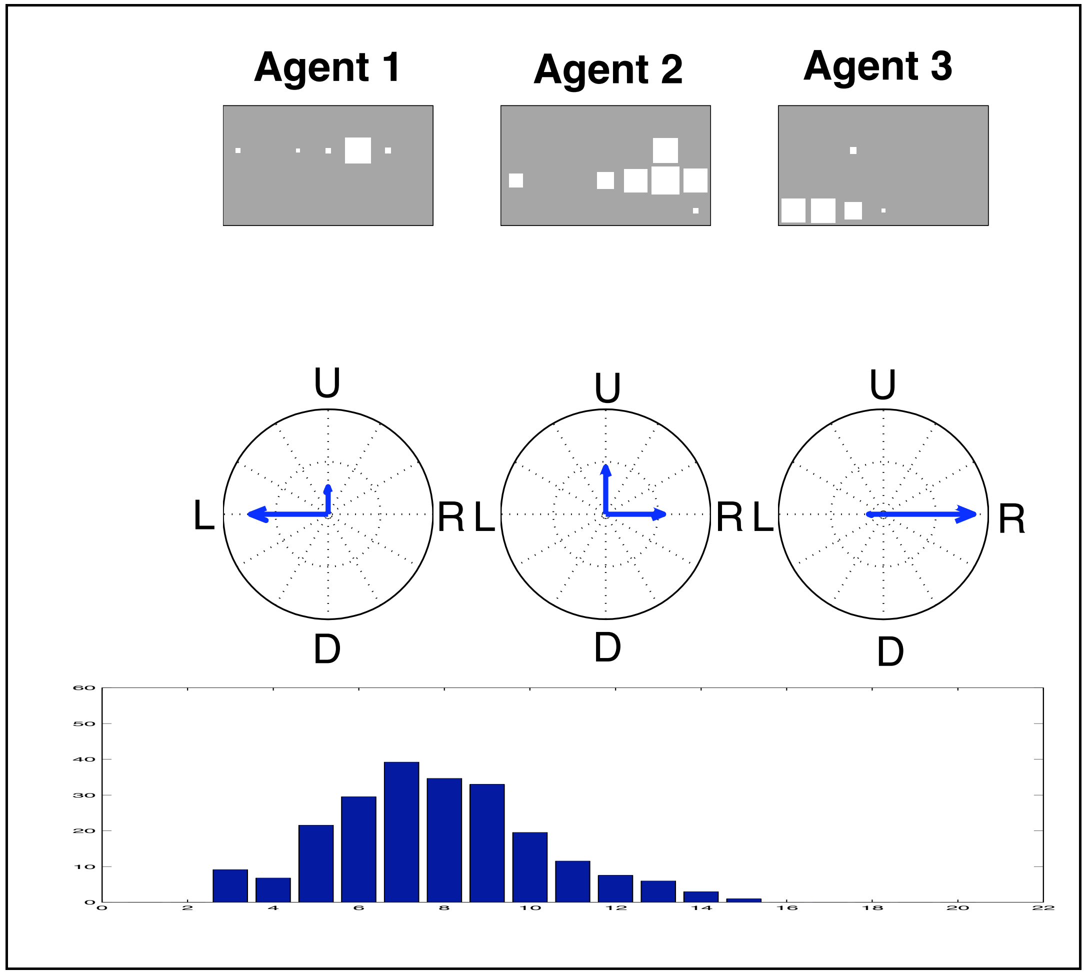
Reinforcement Learning with Factored States and Actions

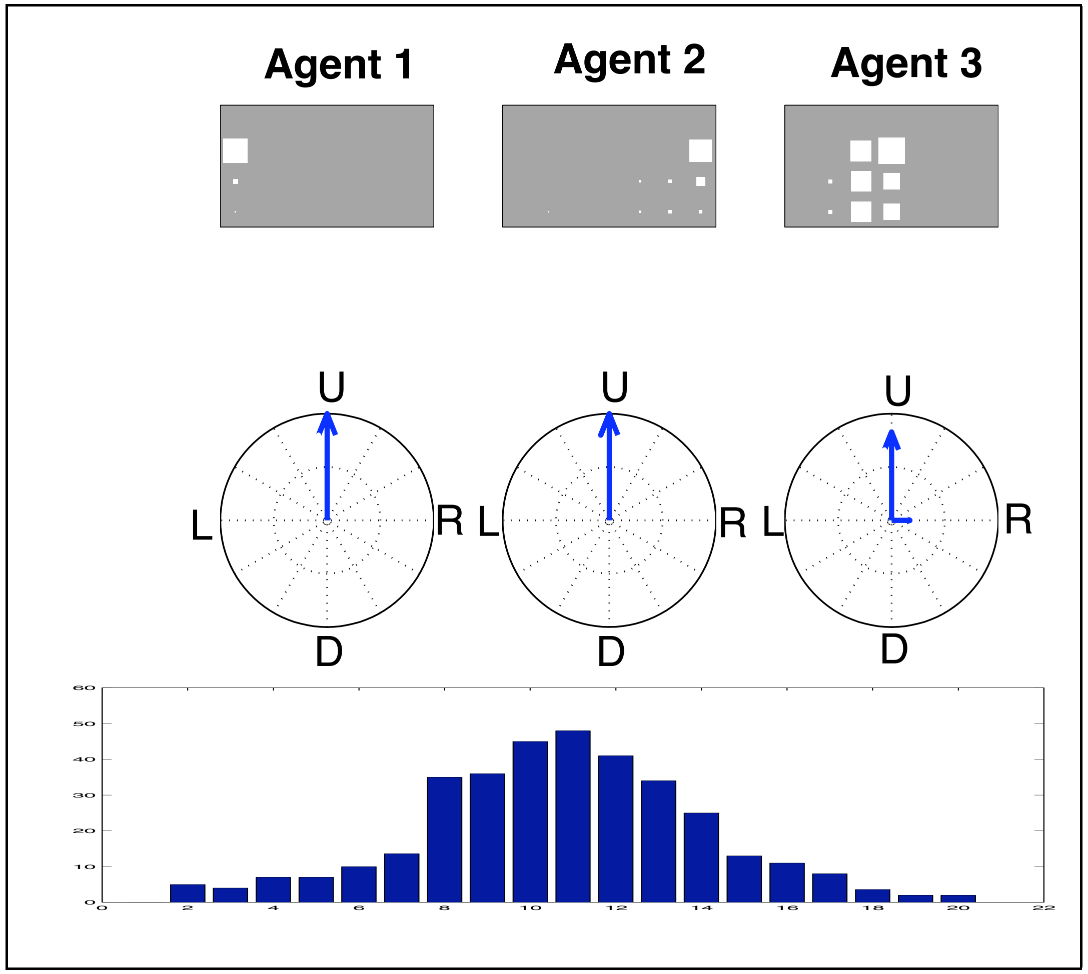
(Sallans and Hinton, JMLR 2004)



Reinforcement Learning with Factored States and Actions

(Sallans and Hinton, JMLR 2004)





Reinforcement Learning with Factored States and Actions

(Sallans and Hinton, JMLR 2004)

- Serait-il possible d'étendre cette approche à des réseaux profonds?
- La deuxième couche pourrait modéliser des "macro" actions
- Comment combiner avec l'apprentissage non-supervisé?

! Piste de recherche !

Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style

(Taylor and Hinton, ICML 2009)

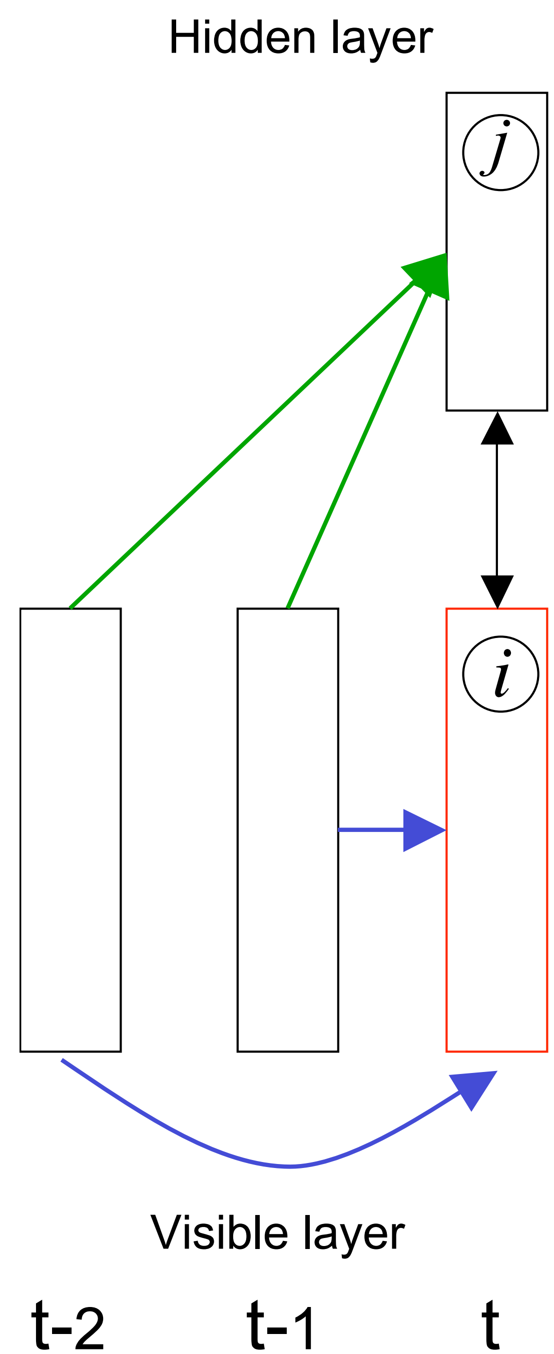
- Modélisation de séquences

$$p(\mathbf{v}_t, \mathbf{h}_t | \mathbf{v}_{<t}, \theta) = \exp\left(-E\left(\mathbf{v}_t, \mathbf{h}_t | \mathbf{v}_{<t}, \theta\right)\right)/Z$$

$$E = \sum_i \frac{(\hat{a}_{i,t} - v_{i,t})^2}{2\sigma_i^2} - \sum_j \hat{b}_{j,t} h_{j,t} - \sum_{ij} W_{ij} \frac{v_{i,t}}{\sigma_i} h_{j,t}$$

$$\hat{a}_{i,t} = a_i + \sum_k A_{ki} v_{k,<t}$$

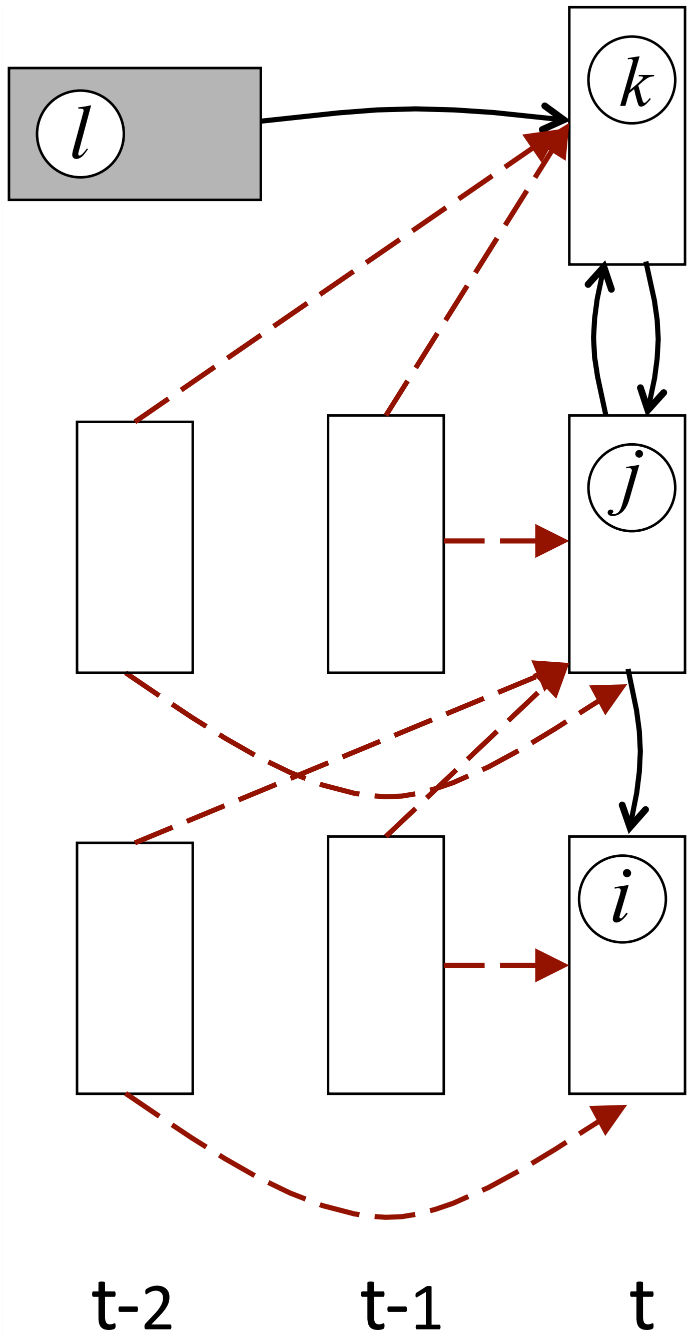
$$\hat{b}_{j,t} = b_j + \sum_k B_{kj} v_{k,<t}$$



Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style

(Taylor and Hinton, ICML 2009)

- Combinaison en plusieurs couches



Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style

(Taylor and Hinton, ICML 2009)

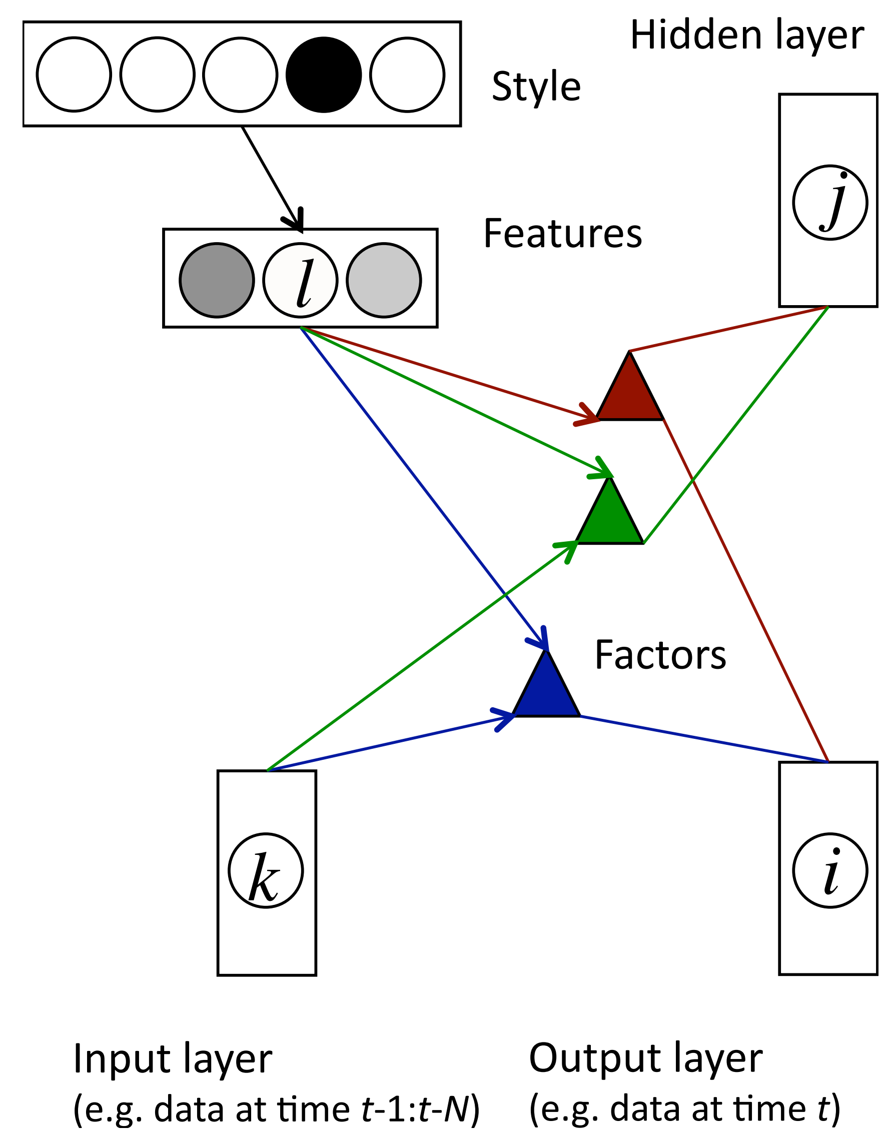
- Utilisation de connections de haut ordre

$$E\left(\mathbf{v}_t, \mathbf{h}_t | \mathbf{v}_{<t}, \mathbf{y}_t, \theta\right) = \frac{1}{2}\sum_i \left(\hat{a}_{i,t} - v_{i,t}\right)^2 - \sum_f \sum_{ijl} \boxed{W^{\mathbf{v}}_{if} W^{\mathbf{h}}_{jf} W^{\mathbf{z}}_{lf}}\, v_{i,t} h_{j,t} z_{l,t} - \sum_j \hat{b}_{j,t} h_{j,t}$$

$$\hat{a}_{i,t} = a_i + \sum_m \boxed{A^{\mathbf{v}}_{im}} \sum_k \boxed{A^{\mathbf{v}_{<t}}_{km}}\, v_{k,<t} \sum_l \boxed{A^{\mathbf{z}}_{lm}}\, z_{l,t}$$

$$\hat{b}_{j,t} = b_j + \sum_n \boxed{B^{\mathbf{h}}_{jn}} \sum_k \boxed{B^{\mathbf{v}_{<t}}_{kn}}\, v_{k,<t} \sum_l \boxed{B^{\mathbf{z}}_{ln}}\, z_{l,t}$$

= connections factorisées



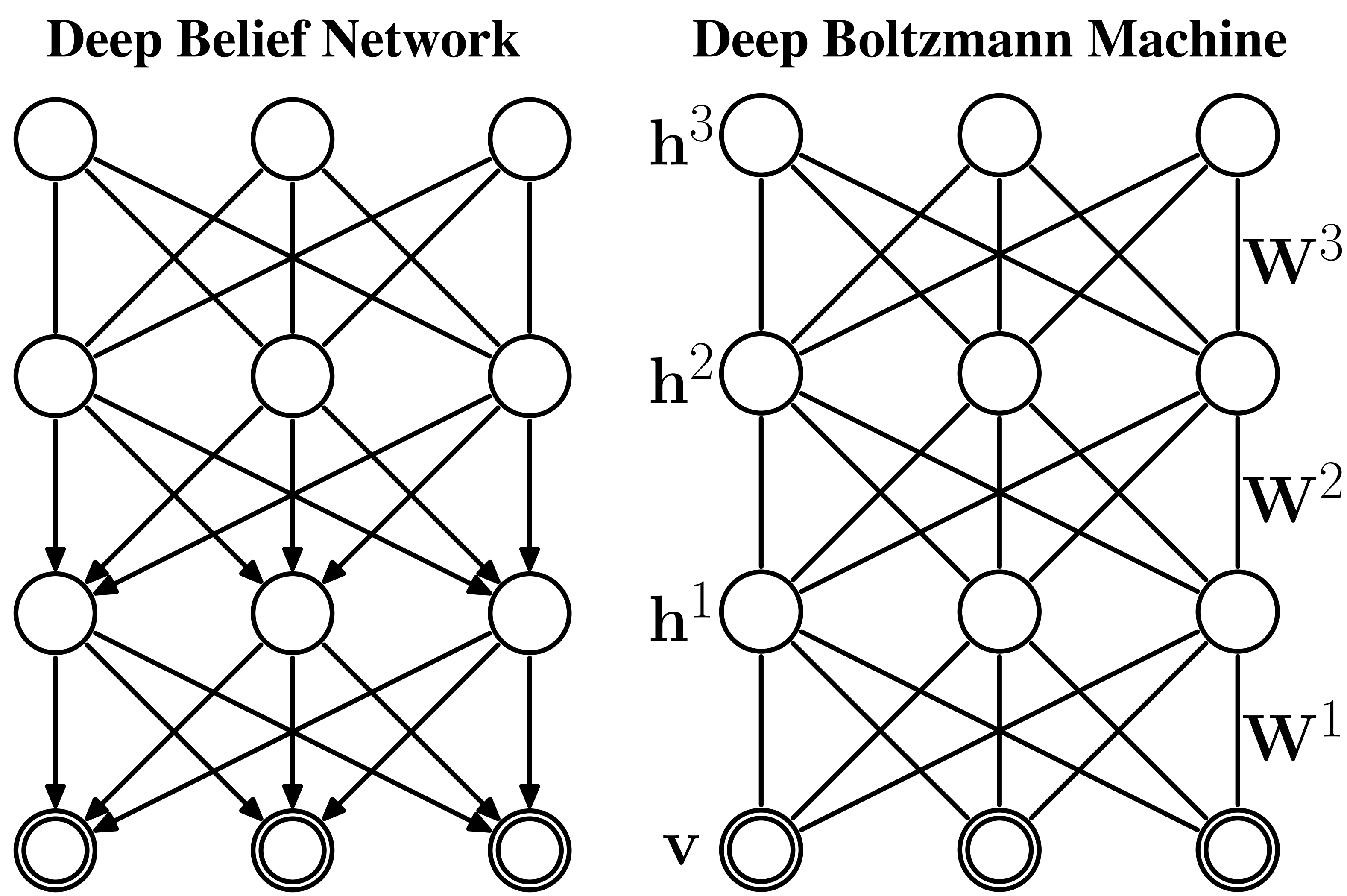
Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style

(Taylor and Hinton, ICML 2009)

(montrer démo web)

Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)



$$E(\mathbf{v},\mathbf{h};\theta) \;=\; -\mathbf{v}^\top \mathbf{W}^1\mathbf{h}^1 - \mathbf{h}^{1\top}\mathbf{W}^2\mathbf{h}^2 - \mathbf{h}^{2\top}\mathbf{W}^3\mathbf{h}^3$$

Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

$$P(\mathbf{v};\theta) = \frac{P^*(\mathbf{v};\theta)}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \underbrace{\sum_{\mathbf{h}} \exp\left(-E(\mathbf{v},\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3;\theta)\right)}_{\text{n'est plus facile à calculer}}$$

$$\frac{\partial \log P(\mathbf{v};\theta)}{\partial W^1} = \overbrace{\mathrm{E}_{P_{\mathrm{data}}}[\mathbf{v}{\mathbf{h}^1}^\top]}^{\text{n'est plus facile à calculer}} - \overbrace{\mathrm{E}_{P_{\mathrm{model}}}[\mathbf{v}{\mathbf{h}^1}^\top]}^{\text{toujours aussi difficile à calculer}}$$

Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

- Solution: approche variationnelle

$$\log P(\mathbf{v};\theta) \;\geq\; \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) \log P(\mathbf{v},\mathbf{h};\theta) + \mathcal{H}(Q)$$

$$Q^{MF}(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) = \prod_{j=1}^{F_1}\prod_{k=1}^{F_2}\prod_{m=1}^{F_3} q(h_j^1)q(h_k^2)q(h_m^3)$$

approx.
"mean field"

$$\log P(\mathbf{v};\theta) \quad \geq \quad \mathbf{v}^\top W^1 \boldsymbol{\mu}^1 + {\boldsymbol{\mu}^1}^\top W^2 \boldsymbol{\mu}^2 + {\boldsymbol{\mu}^2}^\top W^3 \boldsymbol{\mu}^2$$
$$- \log \mathcal{Z}(\theta) + \mathcal{H}(Q).$$

$$\boxed{q(h_i^l = 1) = \mu_i^l}$$

Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

- Calcul de l'approximation "mean-field"

$$
\begin{aligned}
\mu_j^1 &\leftarrow \sigma\left(\sum_{i=1}^{D} W_{ij}^1 v_i + \sum_{k=1}^{F_2} W_{jk}^2 \mu_k^2 \right), \\
\mu_k^2 &\leftarrow \sigma\left(\sum_{j=1}^{F_1} W_{jk}^2 \mu_j^1 + \sum_{m=1}^{F_3} W_{km}^3 \mu_m^3 \right) \\
\mu_m^3 &\leftarrow \sigma\left(\sum_{k=1}^{F_2} W_{km}^3 \mu_k^2 \right).
\end{aligned}
$$

Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

- Accélération de l'approximation "mean-field"

$$Q^{rec}(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) = \prod_{j=1}^{F_1}\prod_{k=1}^{F_2}\prod_{m=1}^{F_3} q^{rec}(h_j^1)q^{rec}(h_k^2)q^{rec}(h_m^3).$$

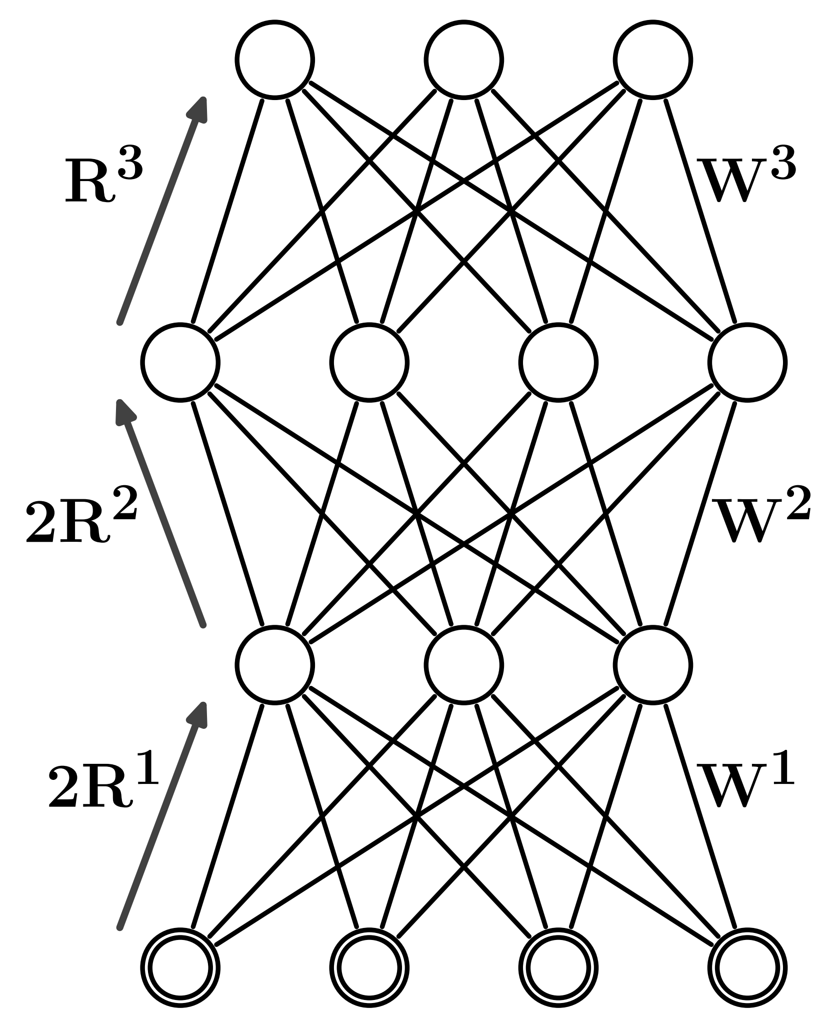
$$\nu_j^1 = \sigma\Big(\sum_{i=1}^{D} 2R_{ij}^1 v_i\Big),$$

$$\nu_k^2 = \sigma\Big(\sum_{j=1}^{F_1} 2R_{jk}^2 \nu_j^1\Big),$$

$$\nu_m^3 = \sigma\Big(\sum_{k=1}^{F_3} R_{km}^3 \nu_k^2\Big),$$

$$\boxed{q^{rec}(h_i^l = 1) = \nu_i^l}$$

Deep Boltzmann Machine



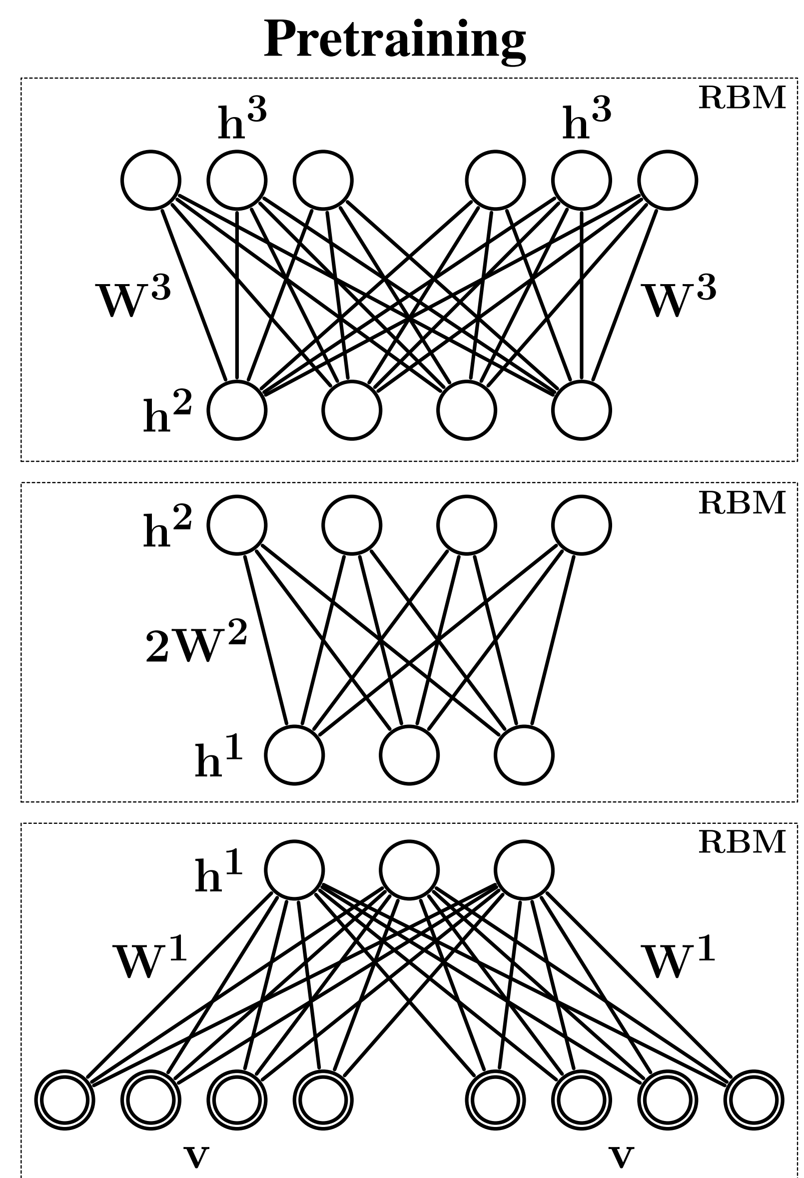
Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

- Pré-entraînement

Algorithm 1 Greedy Pretraining Algorithm for a Deep Boltzmann Machine with 3-layers.

1: Make two copies of the visible vector and tie the visible-to-hidden weights $\mathbf{W}^1$. Fit $\mathbf{W}^1$ of the 1^{st} layer RBM to data.
2: Freeze $\mathbf{W}^1$ that defines the 1^{st} layer of features, and use samples $\mathbf{h}^l$ from $P(\mathbf{h}^1|\mathbf{v}, 2\mathbf{W}^1)$ as the data for training the next layer RBM with weight vector $2\mathbf{W}^2$.
3: Freeze $\mathbf{W}^2$ that defines the 2^{nd} layer of features and use the samples $\mathbf{h}^2$ from $P(\mathbf{h}^2|\mathbf{h}^1, 2\mathbf{W}^2)$ as the data for training the 3^{rd} layer RBM with weight vector $2\mathbf{W}^3$.
4: When learning the top-level RBM, double the number of hidden units and tie the visible-to-hidden weights $\mathbf{W}^3$.
5: Use the weights $\{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3\}$ to compose a Deep Boltzmann Machine.



Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

- Entraînement du réseau de reconnaissance

// **Variational Inference**:

for each training example $\mathbf{v}_n$, $n = 1$ to N **do**

In a single deterministic bottom-up pass, use the recognition model (Eqs. 8, 9, 10) to obtain a parameter vector $\boldsymbol{\nu}$ of the approximate factorial posterior Q^{rec}.

Set $\boldsymbol{\mu} = \boldsymbol{\nu}$ and run the mean-field updates (Eqs. 4, 5, 6) for K steps to obtain the mean-field approximate posterior Q^{MF}.

Adjust the recognition parameters by taking a single gradient step in Eq. 11:

$$\theta^{rec}_{t+1} = \theta^{rec}_t + \alpha_t \frac{\partial \mathrm{KL}(Q^{MF}||Q^{rec})}{\partial \theta^{rec}}$$

Set $\boldsymbol{\mu}_n = \boldsymbol{\mu}$.

end for

Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

- Mise à jour des paramètres

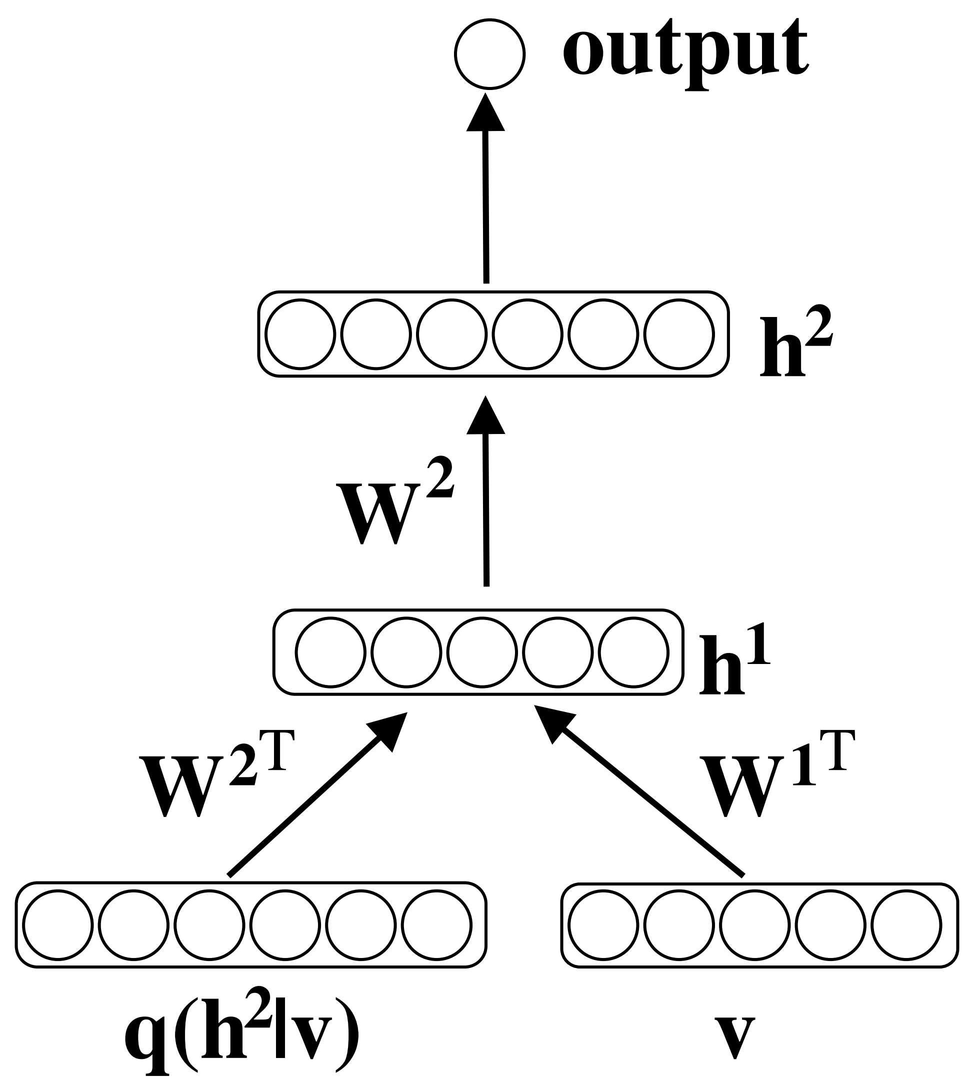
$$
\begin{aligned}
W^1_{t+1} &= W^1_t + \alpha_t \Big(\tfrac{1}{N} \textstyle\sum_{n=1}^{N} \mathbf{v}_n (\boldsymbol{\mu}^1_n)^\top - \\
&\qquad \tfrac{1}{M} \textstyle\sum_{m=1}^{M} \tilde{\mathbf{v}}_{t+1,m} (\tilde{\mathbf{h}}^1_{t+1,m})^\top \Big) \\
W^2_{t+1} &= W^2_t + \alpha_t \Big(\tfrac{1}{N} \textstyle\sum_{n=1}^{N} \boldsymbol{\mu}^1_n (\boldsymbol{\mu}^2_n)^\top - \\
&\qquad \tfrac{1}{M} \textstyle\sum_{m=1}^{M} \tilde{\mathbf{h}}^1_{t+1,m} (\tilde{\mathbf{h}}^2_{t+1,m})^\top \Big) \\
W^3_{t+1} &= W^3_t + \alpha_t \Big(\tfrac{1}{N} \textstyle\sum_{n=1}^{N} \boldsymbol{\mu}^2_n (\boldsymbol{\mu}^3_n)^\top - \\
&\qquad \tfrac{1}{M} \textstyle\sum_{m=1}^{M} \tilde{\mathbf{h}}^2_{t+1,m} (\tilde{\mathbf{h}}^3_{t+1,m})^\top \Big)
\end{aligned}
$$

- Utilise la “Persistent CD”

Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

- Classification et raffinement supervisé



Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS 2010)

Résultats: classification

Dataset	DBM inference procedures						DBN	SVM	K-NN
	MF-0	MF-1	MFRec-1	MF-5	MFRec-5	MF-Full			
MNIST	1.38%	1.15%	**1.00%**	**1.01%**	**0.96%**	**0.95%**	1.17%	1.40%	3.09%
OCR letters	**8.68%**	**8.44%**	**8.40%**	**8.50%**	**8.48%**	**8.58%**	9.68%	9.70%	18.92%
NORB	9.32%	7.96%	**7.62%**	**7.67%**	**7.46%**	**7.23%**	8.31%	11.60%	18.40%

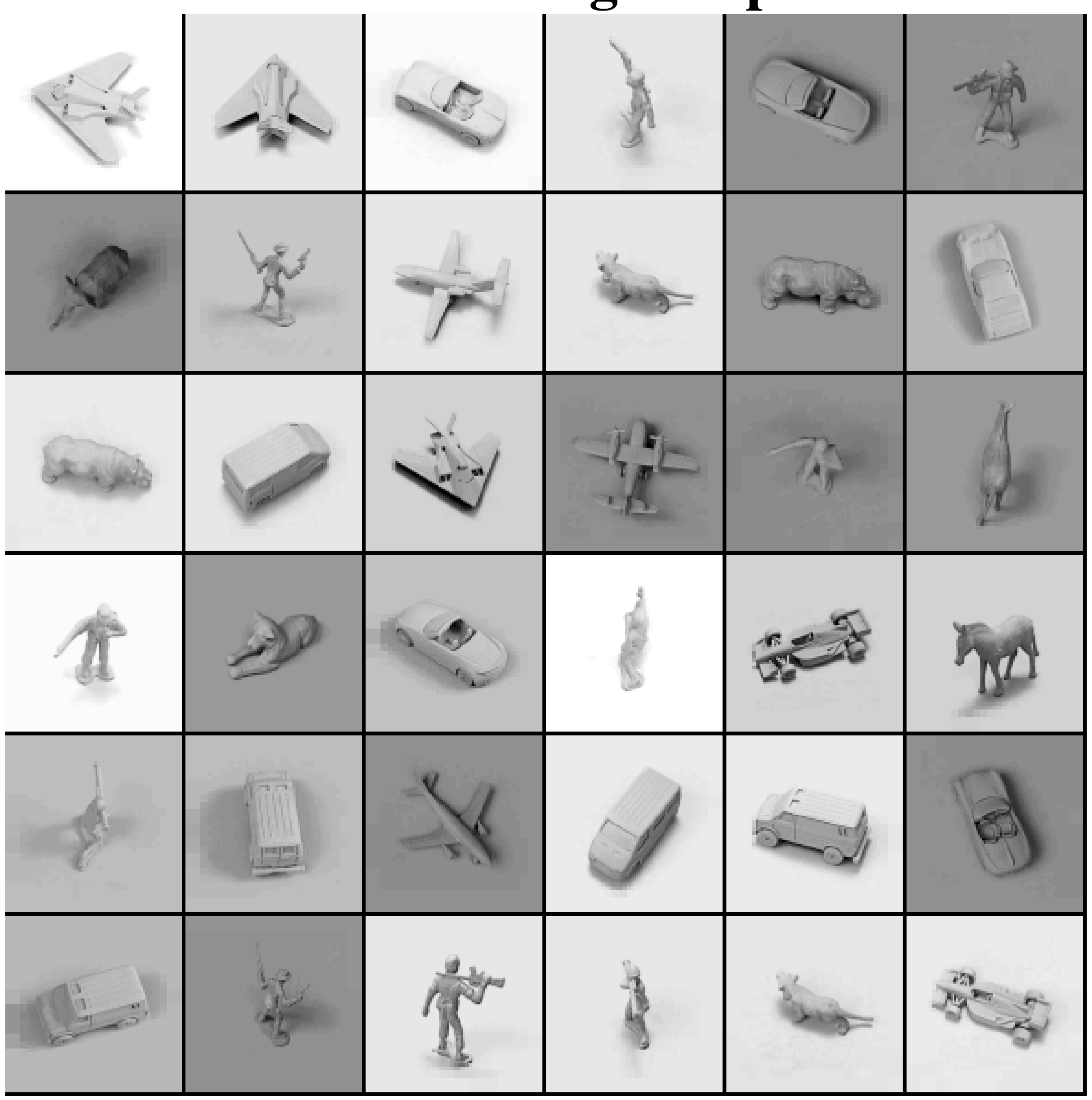
Résultats: estimation de densité

Models	Datasets		
	MNIST	OCR Letters	NORB
MF-0	-96.75	-43.40	-624.75
MF-1	-89.97	-37.21	-612.08
MFRec-1	-86.47	-35.29	-598.34
MF-5	-86.21	-34.87	-596.92
MFRec-5	-85.36	-34.73	-595.98
MF-Full	-84.97	-34.24	-593.58

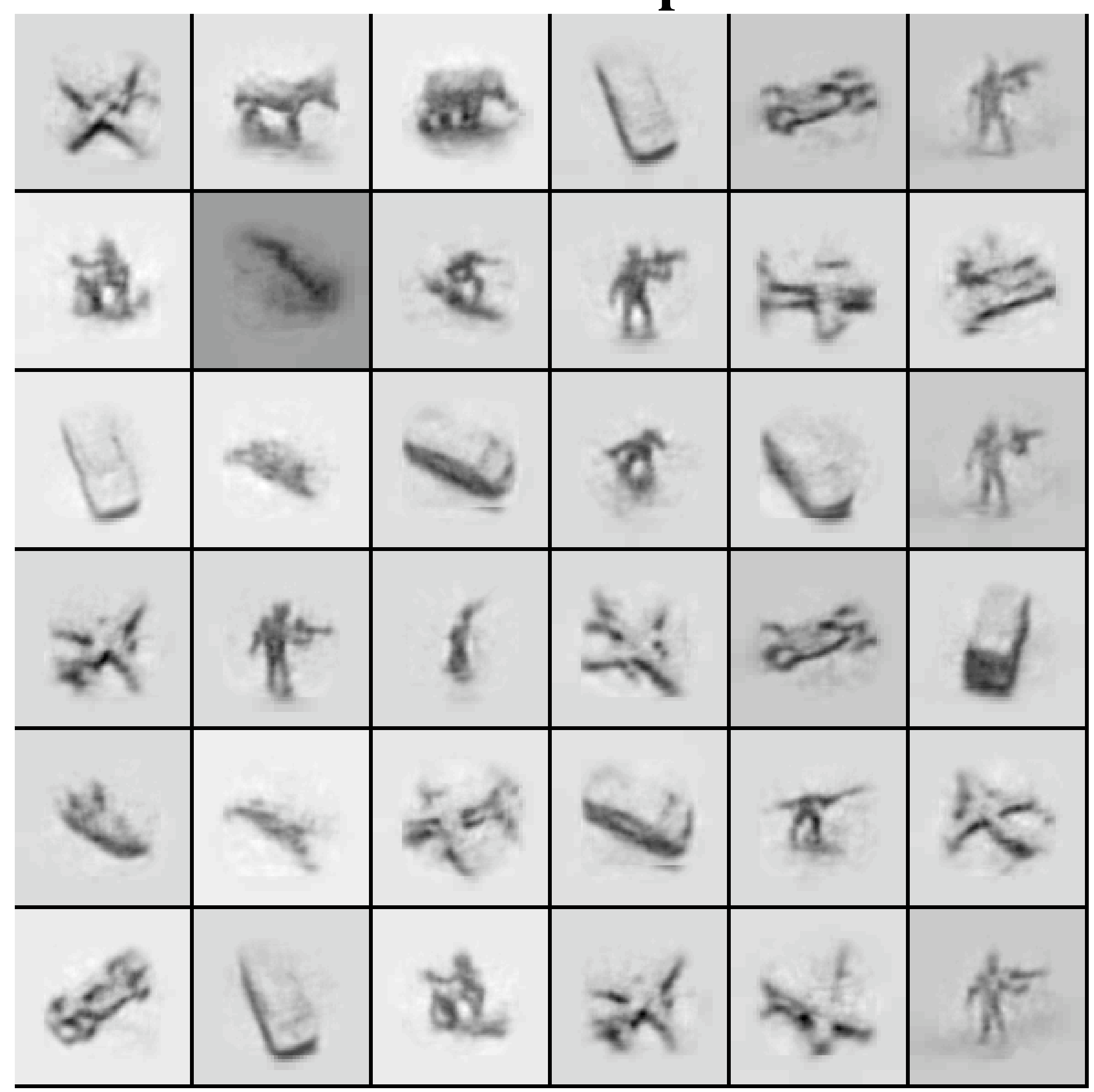
Deep Boltzmann Machines

(Salakhutdinov and Hinton, AISTATS 2009)

Training Samples



Generated Samples



Entraînement non-supervisé global

- Serait-il avantageux d'entraîner des autoencodeurs globalement de façon non-supervisée?
- Serait-il possible d'entraîner des réseaux à convolution (probabiliste ou pas) globalement de façon non-supervisée?

! Pistes de recherche !

Processus “bottom-up” et “top-down”

- Serait-il possible d’introduire des processus “bottom-up” et “top-down” dans les réseaux autoencodeurs?

! Piste de recherche !

Ressources sur le “deep learning”

- Pour en savoir plus et suivre les développement les plus récents:

http://deeplearning.net/

- Vous y trouverez:
  - ★ listes de lectures
  - ★ logiciels
  - ★ jeux de données
  - ★ tutoriels (vidéo) et démos