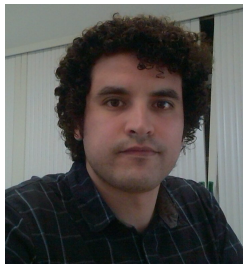
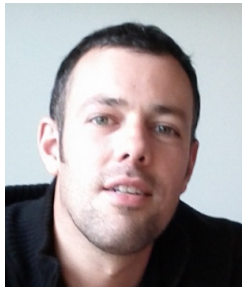
# Deep multi-task learning with evolving weights

## Normastic

Soufiane Belharbi

Romain Hérault



Clément Chatelain

Sébastien Adam

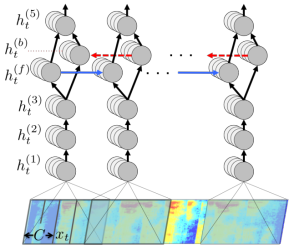
soufiane.belharbi@insa-rouen.fr

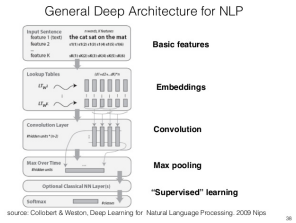
LITIS lab., Apprentissage team - INSA de Rouen, France

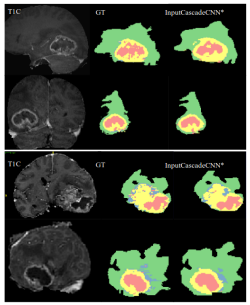

28 June, 2016

# Deep learning Today

## Deep learning state of the art

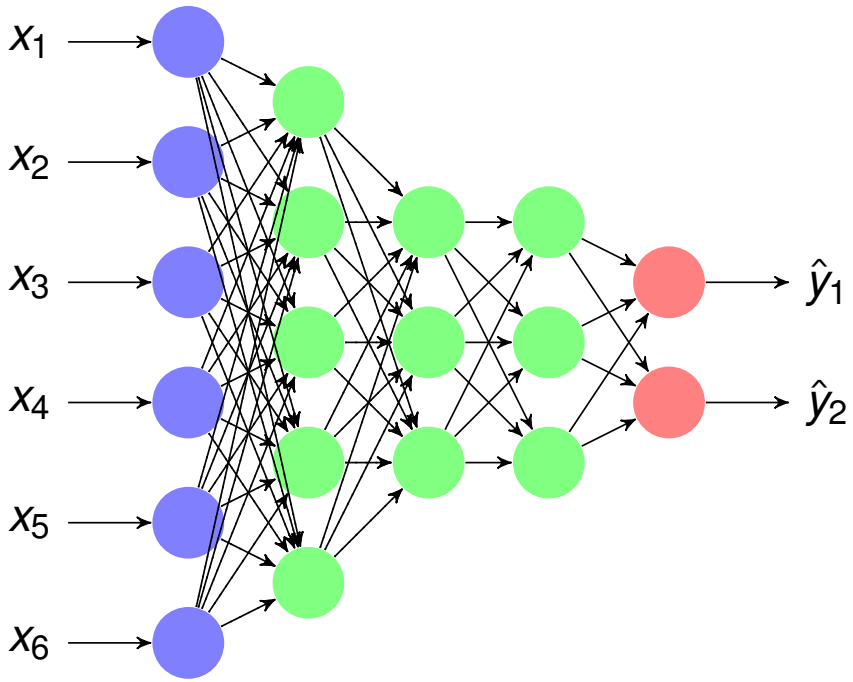

## What is new today?

- Large data
- Calculation power (GPUS, clouds)

$\Rightarrow$ optimization

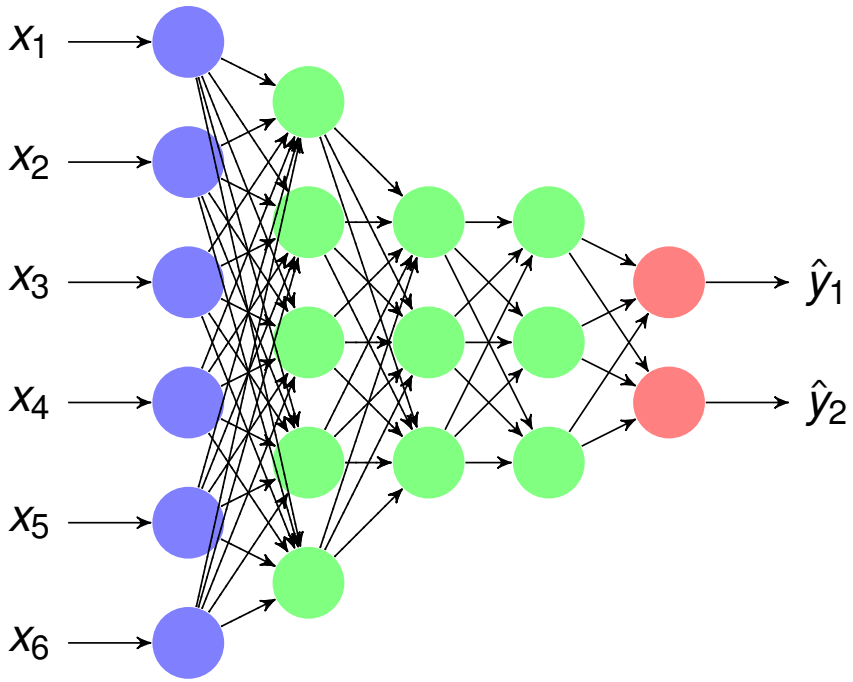
- Dropout
- Momentum, AdaDelta, AdaGrad, RMSProp, Adam, Adamax
- Maxout, Local response normalization, local contrast normalization, batch normalization
- RELU
- CNN, RBM, RNN

# Deep neural networks (DNN)




- Feed-forward neural network
- Back-propagation error
- Training **deep** neural networks is **difficult**
  $\Rightarrow$ Vanishing gradient
  $\Rightarrow$ Pre-training technique [Y.Bengio et al. 06, G.E.Hinton et al. 06]
  $\Rightarrow$ More parameters $\Rightarrow$ Need more data
  $\Rightarrow$ Use unlabeled data

# Deep neural networks (DNN)




- Feed-forward neural network
- Back-propagation error
- Training **deep** neural networks is **difficult**
  $\Rightarrow$ Vanishing gradient
  $\Rightarrow$ Pre-training technique [Y.Bengio et al. 06, G.E.Hinton et al. 06]
  $\Rightarrow$ More parameters $\Rightarrow$ Need more data
  $\Rightarrow$ Use unlabeled data

# Semi-supervised learning

General case:

$$Data = \{ \underbrace{labeled\ data\ (\mathbf{x}, \mathbf{y})}_{\text{expensive (money, time), few}}, \underbrace{unlabeled\ data\ (\mathbf{x}, --)}_{\text{cheap, abundant}} \}$$

E.g:

- Collect images from the internet
- Medical images

$\Rightarrow$ semi-supervised learning:

Exploit unlabeled data to improve the **generalization**

# Semi-supervised learning

General case:

$$Data = \{ \underbrace{\textit{labeled data } (\mathbf{x}, \mathbf{y})}_{\text{expensive (money, time), few}} , \underbrace{\textit{unlabeled data } (\mathbf{x}, --)}_{\text{cheap, abundant}} \}$$

E.g:

- Collect images from the internet
- Medical images

$\Rightarrow$ semi-supervised learning:

Exploit unlabeled data to improve the **generalization**

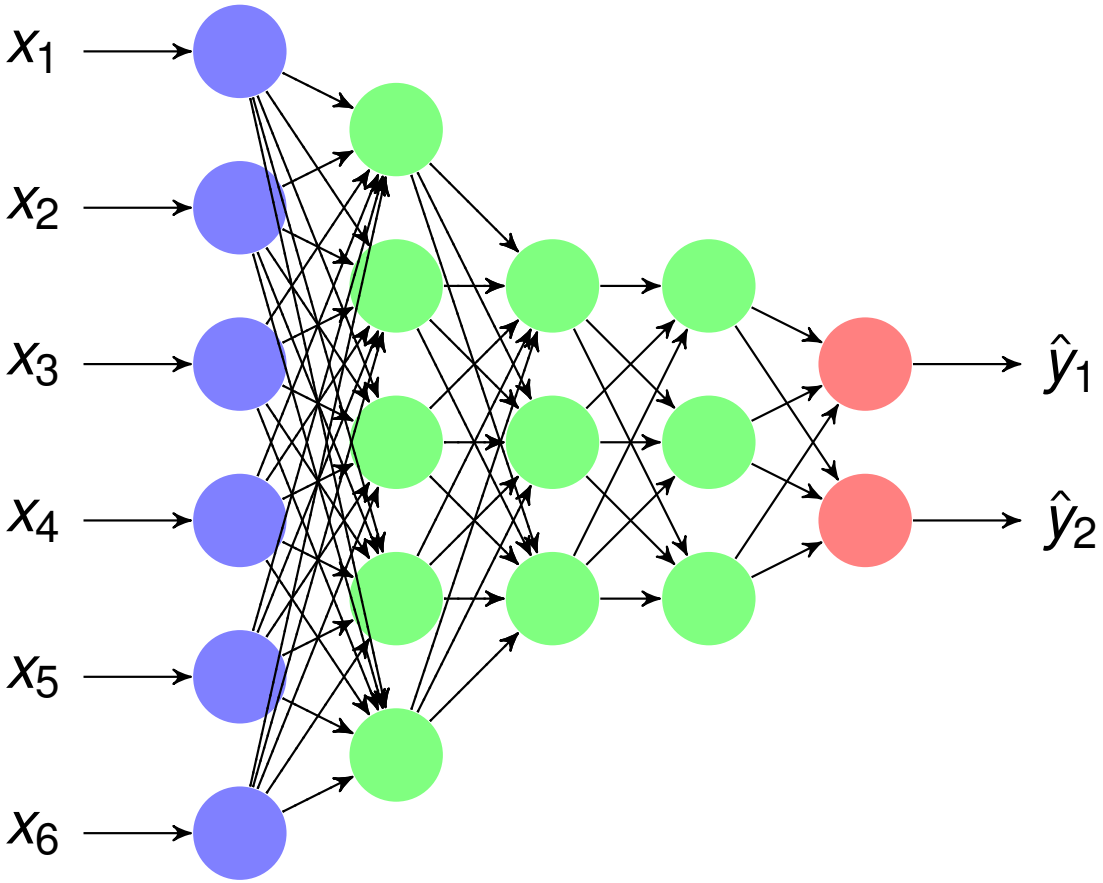
# Pre-training and semi-supervised learning

The pre-training technique can exploit the unlabeled data

A **sequential** transfer learning performed in 2 steps:

1. **Unsupervised task** (**x** labeled and unlabeled data)
2. **Supervised task** ( (**x**, **y**) labeled data)

# Layer-wise pre-training: auto-encoders

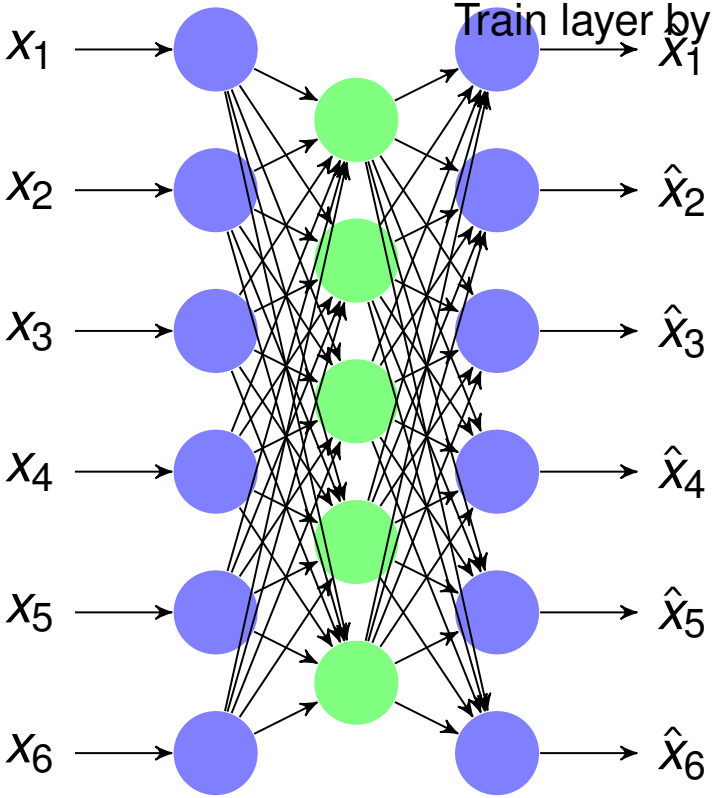



A DNN to train

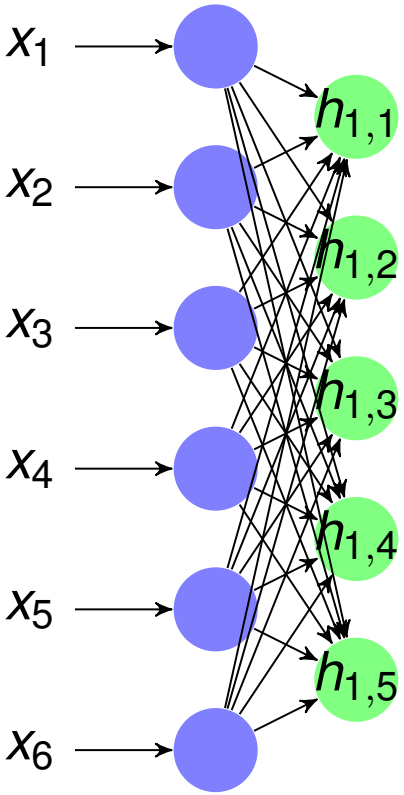
# Layer-wise pre-training: auto-encoders

1) Step 1: **Unsupervised layer-wise pre-training**

Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



# Layer-wise pre-training: auto-encoders



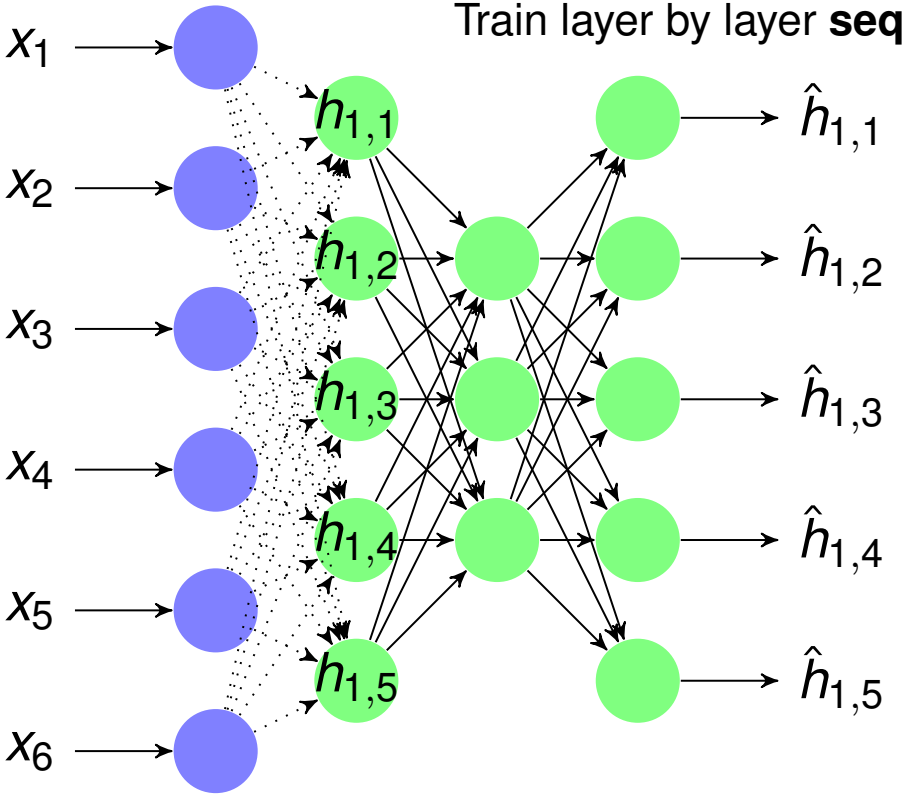

1) Step 1: **Unsupervised layer-wise pre-training**

Train layer by layer **sequentially** using **only x** (labeled or unlabeled)

# Layer-wise pre-training: auto-encoders

1) Step 1: **Unsupervised layer-wise pre-training**

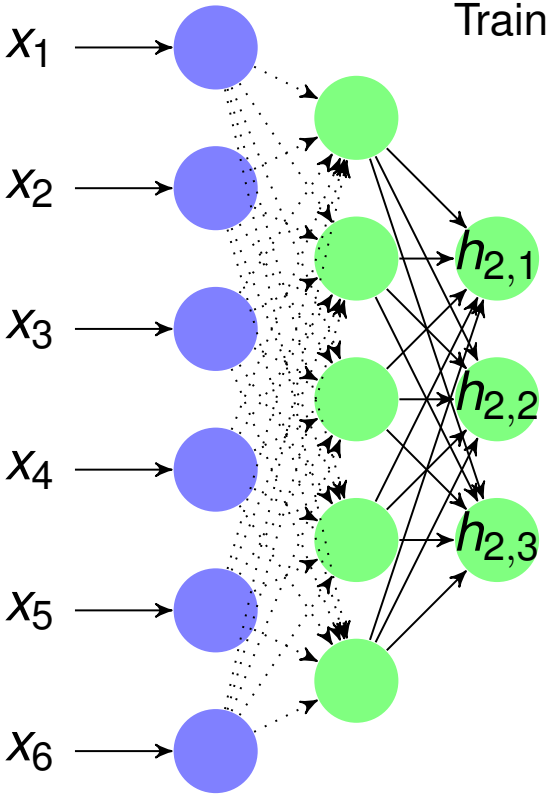
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



# Layer-wise pre-training: auto-encoders

1) Step 1: **Unsupervised layer-wise pre-training**

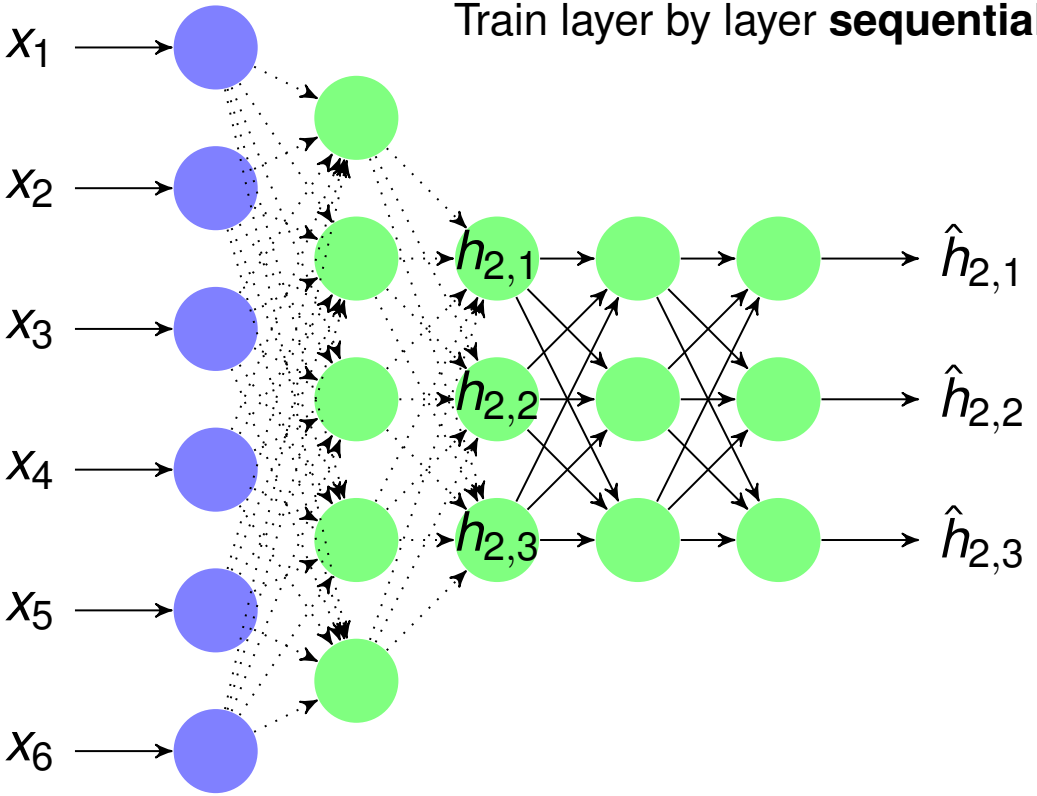
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



# Layer-wise pre-training: auto-encoders

1) Step 1: **Unsupervised layer-wise pre-training**

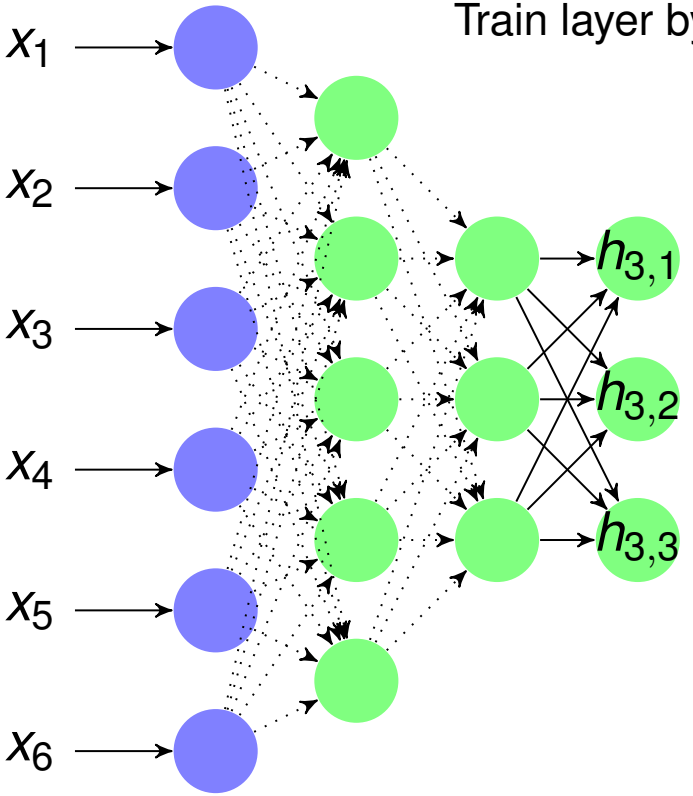
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



# Layer-wise pre-training: auto-encoders

1) Step 1: **Unsupervised layer-wise pre-training**

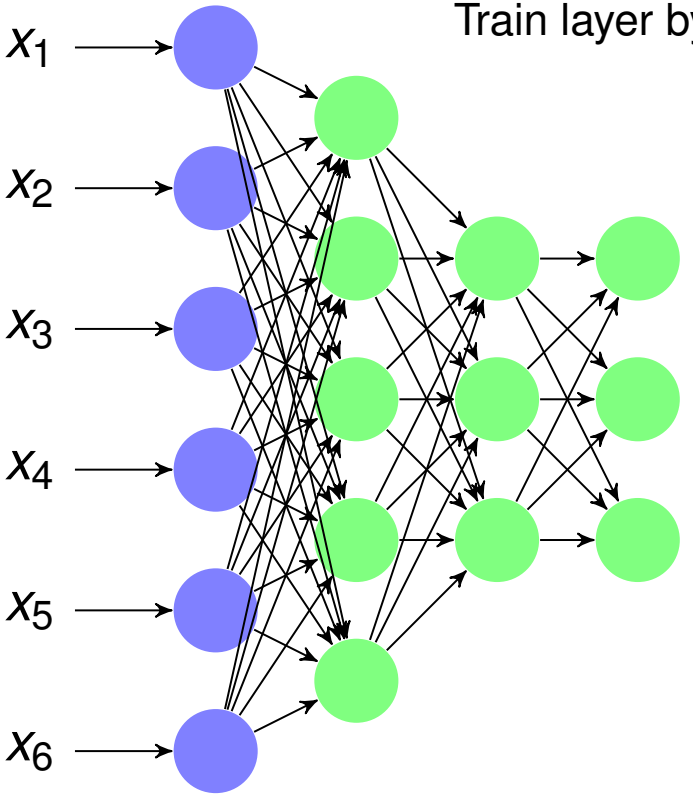
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



# Layer-wise pre-training: auto-encoders

1) Step 1: **Unsupervised layer-wise pre-training**

Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



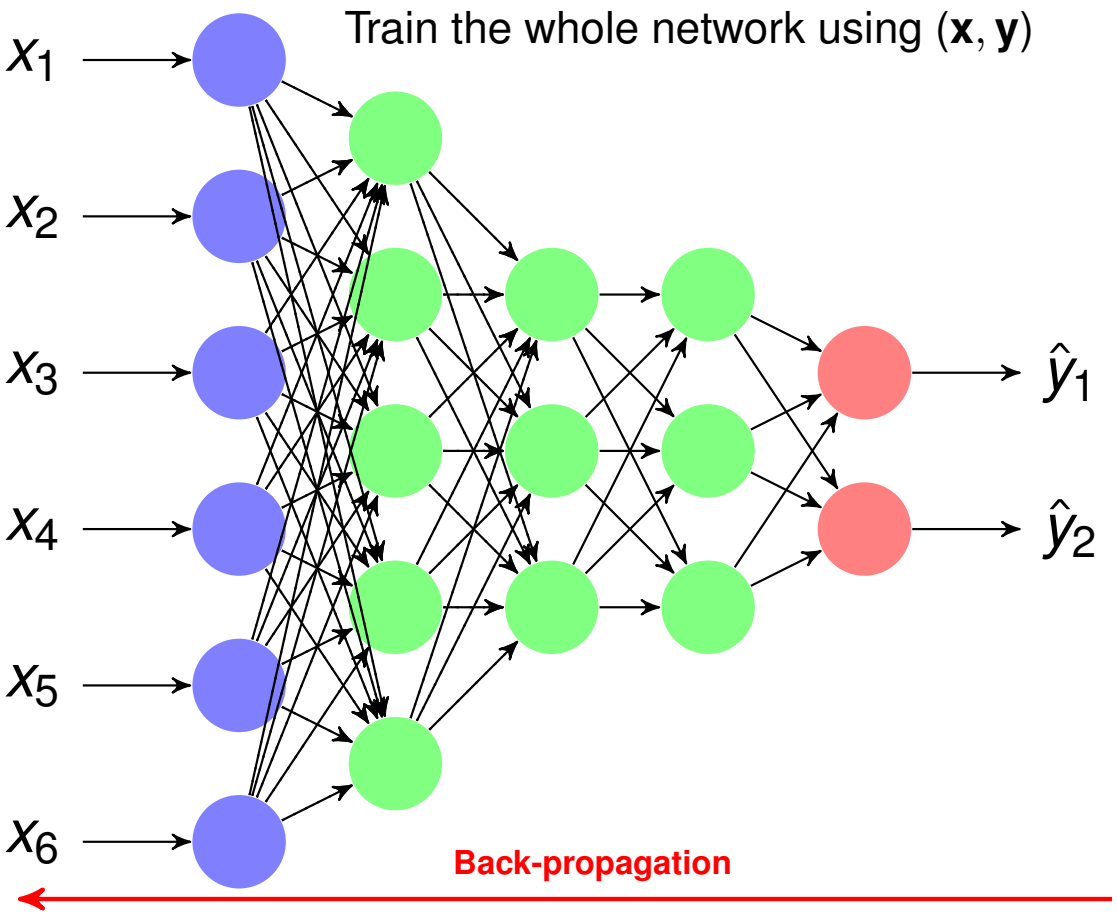

**At each layer**:

⇒ What hyper-parameters to use? When to stop training?

⇒ How to make sure that the pre-training improves the supervised task?

# Layer-wise pre-training: auto-encoders

2) Step 2: **Supervised training**



# Pre-training technique: Pros and cons

## Pros

- Improve generalization
- Can exploit unlabeled data
- Provide better initialization than random
- Train deep networks
  $\Rightarrow$ Circumvent the vanishing gradient problem

## Cons

- Add more hyper-parameters
- No good stopping criterion during pre-training phase
  Good criterion for the unsupervised task
  But
  May not be good for the supervised task

# Pre-training technique: Pros and cons

## Pros

- Improve generalization
- Can exploit unlabeled data
- Provide better initialization than random
- Train deep networks
  $\Rightarrow$ Circumvent the vanishing gradient problem

## Cons

- Add more hyper-parameters
- No good stopping criterion during pre-training phase

  Good criterion for the unsupervised task

  But

  May not be good for the supervised task

# Proposed solution

Why is it difficult in practice?

$\Rightarrow$ **Sequential** transfer learning

Possible solution:

$\Rightarrow$ **Parallel** transfer learning

Why in parallel?

- Interaction between tasks
- Reduce the number of hyper-parameters to tune
- Provide **one stopping criterion**

# Proposed solution

Why is it difficult in practice?

$\Rightarrow$ **Sequential** transfer learning

Possible solution:

$\Rightarrow$ **Parallel** transfer learning

Why in parallel?

- Interaction between tasks
- Reduce the number of hyper-parameters to tune
- Provide **one stopping criterion**

# Proposed solution

Why is it difficult in practice?

$\Rightarrow$ **Sequential** transfer learning

Possible solution:

$\Rightarrow$ **Parallel** transfer learning

Why in parallel?

- Interaction between tasks
- Reduce the number of hyper-parameters to tune
- Provide **one stopping criterion**

# Parallel transfer learning: Tasks combination

$$\text{Train cost} = \textbf{supervised task} + \underbrace{\textbf{unsupervised task}}_{\text{reconstruction}}$$

$l$ labeled samples, $u$ unlabeled samples, $\mathbf{w}_{sh}$: shared parameters.

**Reconstruction (auto-encoder) task:**

$$\mathcal{J}_r(\mathcal{D}; \mathbf{w}' = \{\mathbf{w}_{sh}, \mathbf{w}_r\}) = \sum_{i=1}^{l+u} \mathcal{C}_r(\mathcal{R}(\mathbf{x}_i; \mathbf{w}'), \mathbf{x}_i) \ .$$

**Supervised task:**

$$\mathcal{J}_s(\mathcal{D}; \mathbf{w} = \{\mathbf{w}_{sh}, \mathbf{w}_s\}) = \sum_{i=1}^{l} \mathcal{C}_s(\mathcal{M}(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) \ .$$

**Weighted tasks combination**

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$\lambda_s, \ \lambda_r \in [0, 1]$: importance weight, $\lambda_s + \lambda_r = 1$.

# Parallel transfer learning: Tasks combination

$$\text{Train cost} = \textbf{supervised task} + \underbrace{\textbf{unsupervised task}}_{\text{reconstruction}}$$

$l$ labeled samples, $u$ unlabeled samples, $\mathbf{w}_{sh}$: shared parameters.

**Reconstruction (auto-encoder) task**:

$$\mathcal{J}_r(\mathcal{D}; \mathbf{w}' = \{\mathbf{w}_{sh}, \mathbf{w}_r\}) = \sum_{i=1}^{l+u} \mathcal{C}_r(\mathcal{R}(\mathbf{x}_i; \mathbf{w}'), \mathbf{x}_i) \ .$$

**Supervised task**:

$$\mathcal{J}_s(\mathcal{D}; \mathbf{w} = \{\mathbf{w}_{sh}, \mathbf{w}_s\}) = \sum_{i=1}^{l} \mathcal{C}_s(\mathcal{M}(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) \ .$$

**Weighted tasks combination**

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$\lambda_s, \ \lambda_r \in [0, 1]$: importance weight, $\lambda_s + \lambda_r = 1$.

# Parallel transfer learning: Tasks combination

$$\text{Train cost} = \textbf{supervised task} + \underbrace{\textbf{unsupervised task}}_{\text{reconstruction}}$$

$l$ labeled samples, $u$ unlabeled samples, $\mathbf{w}_{sh}$: shared parameters.

**Reconstruction (auto-encoder) task**:

$$\mathcal{J}_r(\mathcal{D}; \mathbf{w}' = \{\mathbf{w}_{sh}, \mathbf{w}_r\}) = \sum_{i=1}^{l+u} \mathcal{C}_r(\mathcal{R}(\mathbf{x}_i; \mathbf{w}'), \mathbf{x}_i) \ .$$

**Supervised task**:

$$\mathcal{J}_s(\mathcal{D}; \mathbf{w} = \{\mathbf{w}_{sh}, \mathbf{w}_s\}) = \sum_{i=1}^{l} \mathcal{C}_s(\mathcal{M}(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) \ .$$

**Weighted tasks combination**

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$\lambda_s, \ \lambda_r \in [0, 1]$: importance weight, $\lambda_s + \lambda_r = 1$.

# Parallel transfer learning: Tasks combination

$$\text{Train cost} = \textbf{supervised task} + \underbrace{\textbf{unsupervised task}}_{\text{reconstruction}}$$

$l$ labeled samples, $u$ unlabeled samples, $\mathbf{w}_{sh}$: shared parameters.

**Reconstruction (auto-encoder) task**:

$$\mathcal{J}_r(\mathcal{D}; \mathbf{w}' = \{\mathbf{w}_{sh}, \mathbf{w}_r\}) = \sum_{i=1}^{l+u} \mathcal{C}_r(\mathcal{R}(\mathbf{x}_i; \mathbf{w}'), \mathbf{x}_i) \ .$$

**Supervised task**:

$$\mathcal{J}_s(\mathcal{D}; \mathbf{w} = \{\mathbf{w}_{sh}, \mathbf{w}_s\}) = \sum_{i=1}^{l} \mathcal{C}_s(\mathcal{M}(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) \ .$$

## Weighted tasks combination

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$\lambda_s, \ \lambda_r \in [0, 1]$: importance weight, $\lambda_s + \lambda_r = 1$.

# Tasks combination with evolving weights

**Weighted tasks combination**:

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$\lambda_s,\ \lambda_r \in [0, 1]$: importance weight, $\lambda_s + \lambda_r = 1$.

**Problem**

How to **fix** $\lambda_s, \lambda_r$?

**Intuition**

At the end of the training, only $\mathcal{J}_s$ should matters

**Tasks combination with evolving weights (our contribution)**

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s(t) \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r(t) \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$t$: learning epochs, $\lambda_s(t),\ \lambda_r(t) \in [0, 1]$: importance weight, $\lambda_s(t) + \lambda_r(t) = 1$.

# Tasks combination with evolving weights

**Weighted tasks combination**:

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$\lambda_s,\ \lambda_r \in [0, 1]$: importance weight, $\lambda_s + \lambda_r = 1$.

**Problem**

How to **fix** $\lambda_s, \lambda_r$?

**Intuition**

At the end of the training, only $\mathcal{J}_s$ should matters

**Tasks combination with evolving weights (our contribution)**

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s(t) \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r(t) \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$t$: learning epochs, $\lambda_s(t),\ \lambda_r(t) \in [0, 1]$: importance weight, $\lambda_s(t) + \lambda_r(t) = 1$.

# Tasks combination with evolving weights

**Weighted tasks combination**:

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$\lambda_s,\ \lambda_r \in [0, 1]$: importance weight, $\lambda_s + \lambda_r = 1$.

**Problem**

How to **fix** $\lambda_s, \lambda_r$?

**Intuition**

At the end of the training, only $\mathcal{J}_s$ should matters

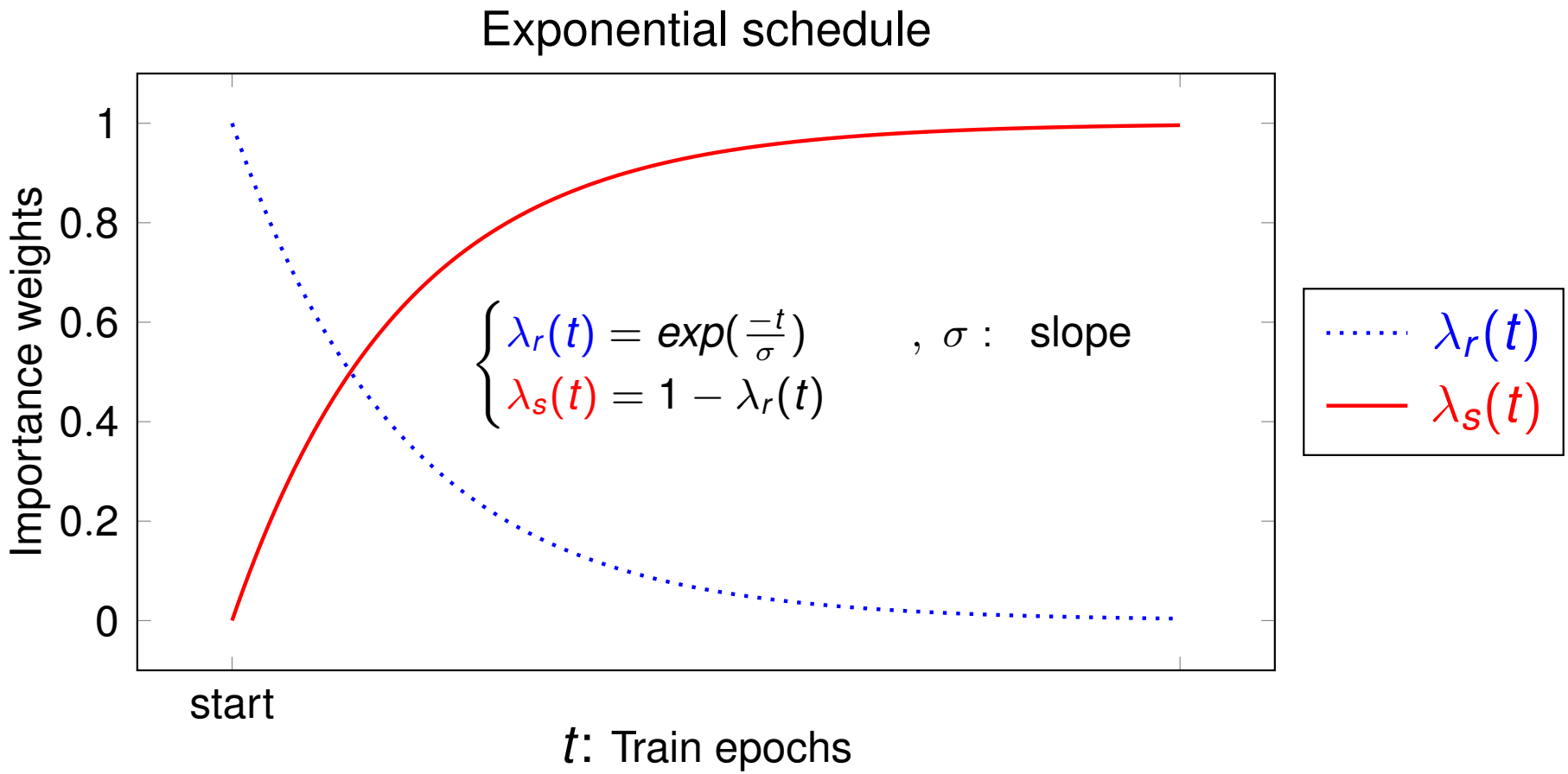
**Tasks combination with evolving weights (our contribution)**

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s(t) \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r(t) \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$t$: learning epochs, $\lambda_s(t),\ \lambda_r(t) \in [0, 1]$: importance weight, $\lambda_s(t) + \lambda_r(t) = 1$.

# Tasks combination with evolving weights

$$\mathcal{J}(\mathcal{D};\{\mathbf{w}_{sh},\mathbf{w}_s,\mathbf{w}_r\}) = \lambda_s(t)\cdot\mathcal{J}_s(\mathcal{D};\{\mathbf{w}_{sh},\mathbf{w}_s\})+\lambda_r(t)\cdot\mathcal{J}_r(\mathcal{D};\{\mathbf{w}_{sh},\mathbf{w}_r\})$$



# Tasks combination with evolving weights: Optimization

## Tasks combination with evolving weights (our contribution)

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s(t) \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r(t) \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ .$$

$t$: learning epochs, $\lambda_s(t),\ \lambda_r(t) \in [0, 1]$: importance weight, $\lambda_s(t) + \lambda_r(t) = 1$.

**Algorithm 1** Training our model for one epoch

1: $\mathcal{D}$ is the *shuffled* training set. $B$ a mini-batch.
2: **for** $B$ in $\mathcal{D}$ **do**
3: Make a gradient step toward $\mathcal{J}_r$ using $B$ (update $\mathbf{w}'$)
4: $B_s \Leftarrow$ labeled examples of $B$,
5: Make a gradient step toward $\mathcal{J}_s$ using $B_s$ (update $\mathbf{w}$)
6: **end for**

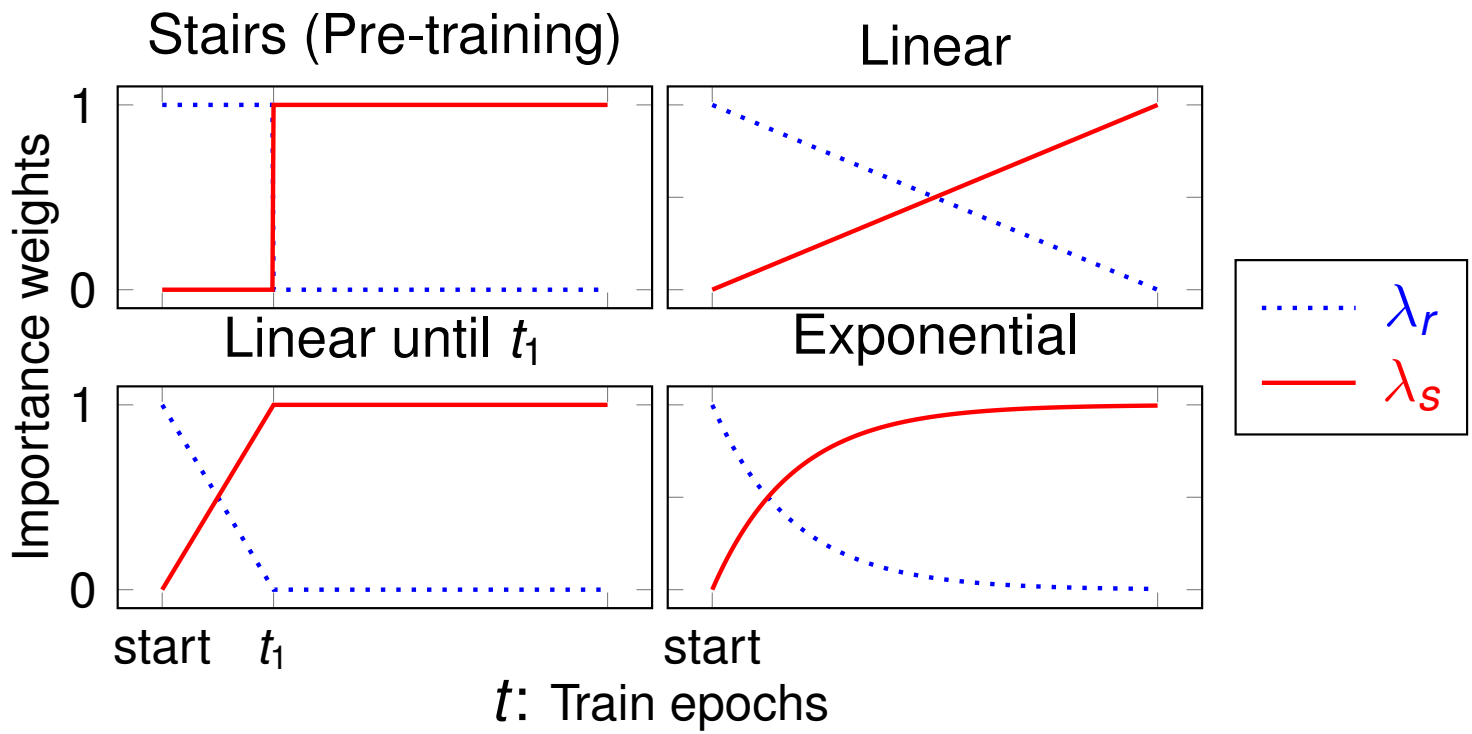
[R.Caruana 97, J.Weston 08, R.Collobert 08, Z.Zhang 15]

# Experimental protocol

**Objective**: Compare Training DNN using different approaches:

- No pre-training (base-line)
- With pre-training (Stairs schedule)
- Parallel transfer learning (proposed approach)

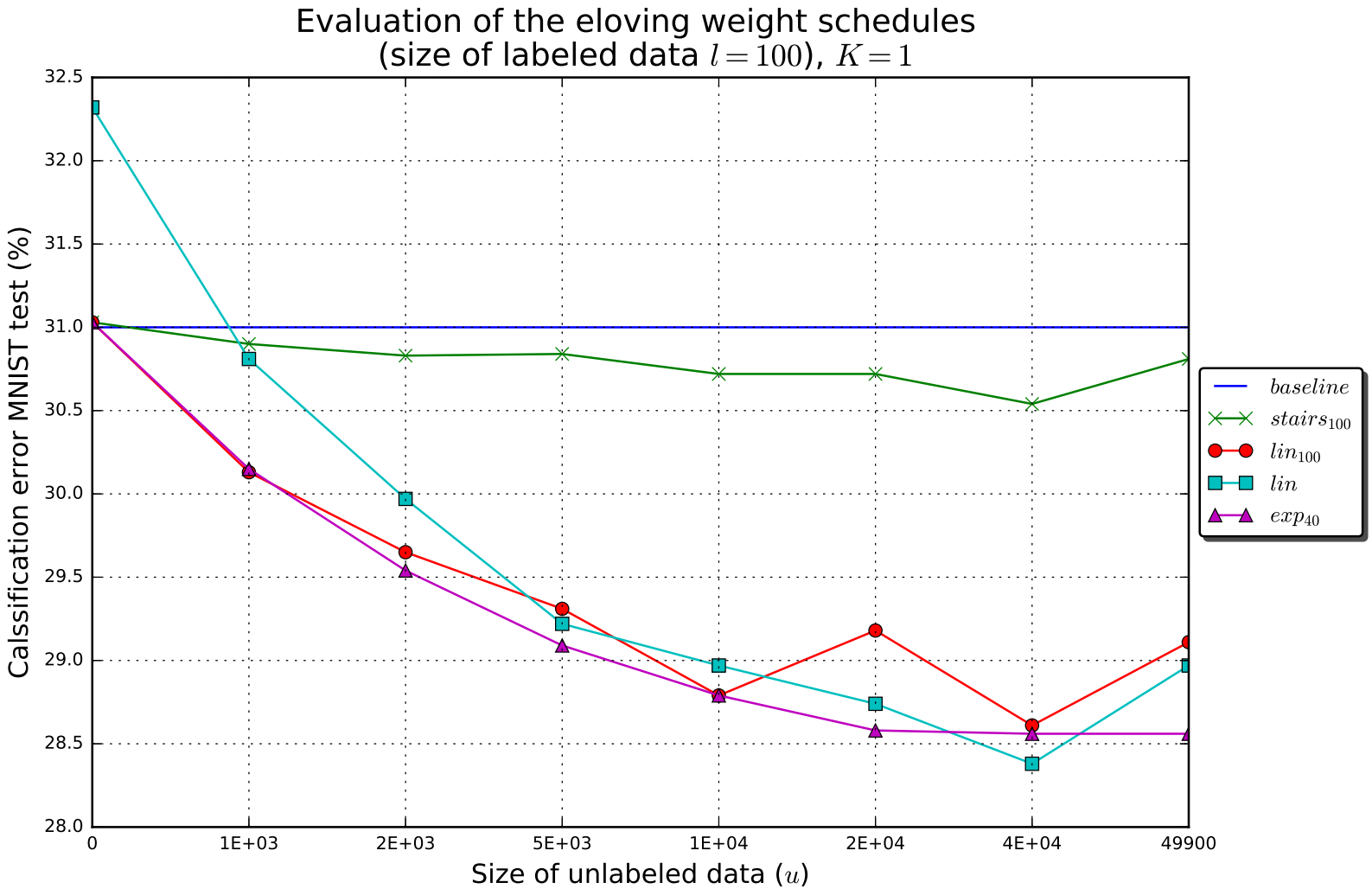
**Studied evolving weights schedules**:



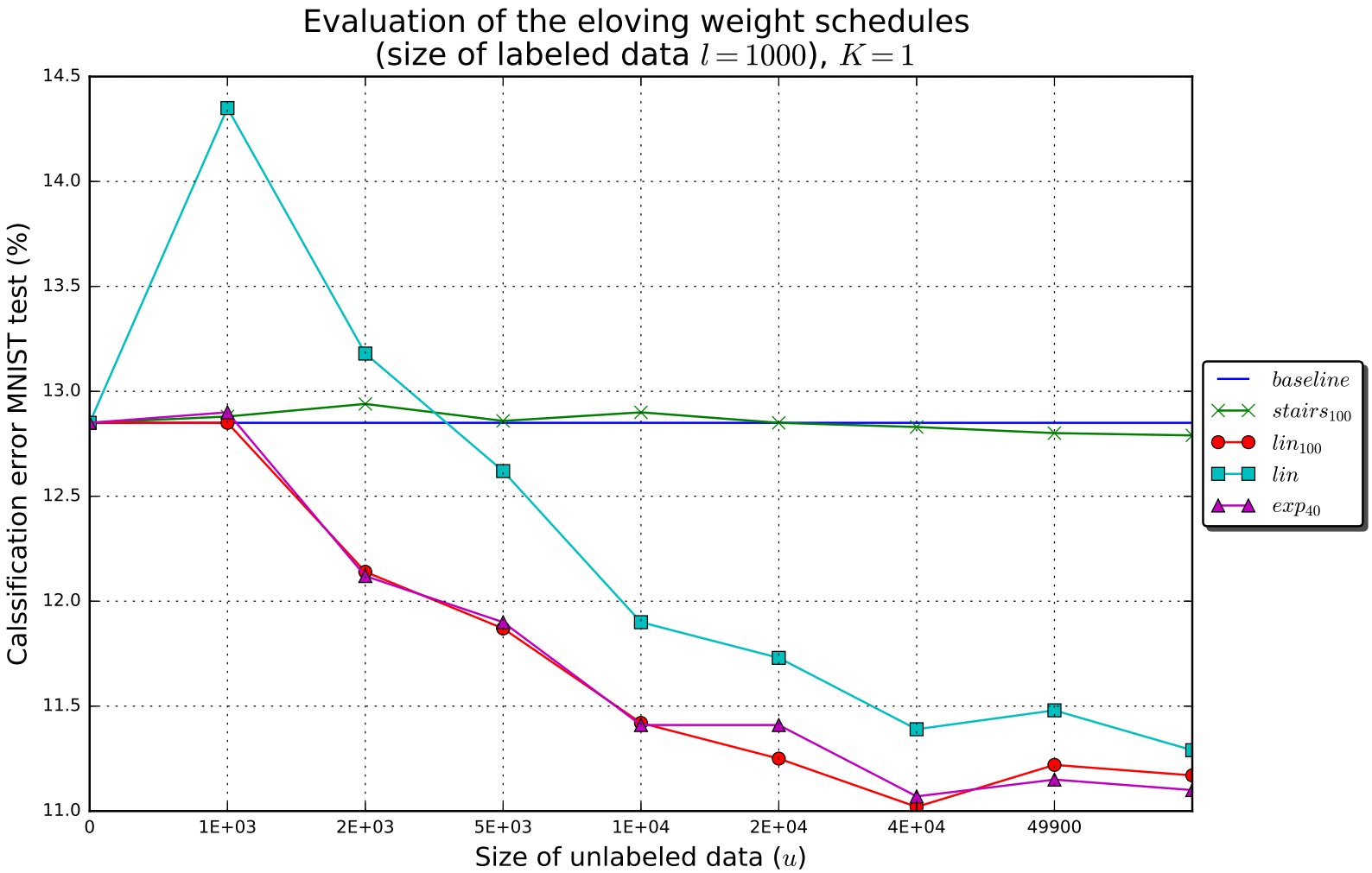
# Experimental protocol

- **Task**: Classification (MNIST)
- **Number of hidden layers** $K$: 1, 2, 3, 4.
- **Optimization**:
  - **Epochs**: 5000
  - **Batch size**: 600
  - **Options**: **No** regularization, **No** adaptive learning rate
- **Hyper-parameters of the evolving schedules**:
  - $t_1$: 100
  - $\sigma$: 40

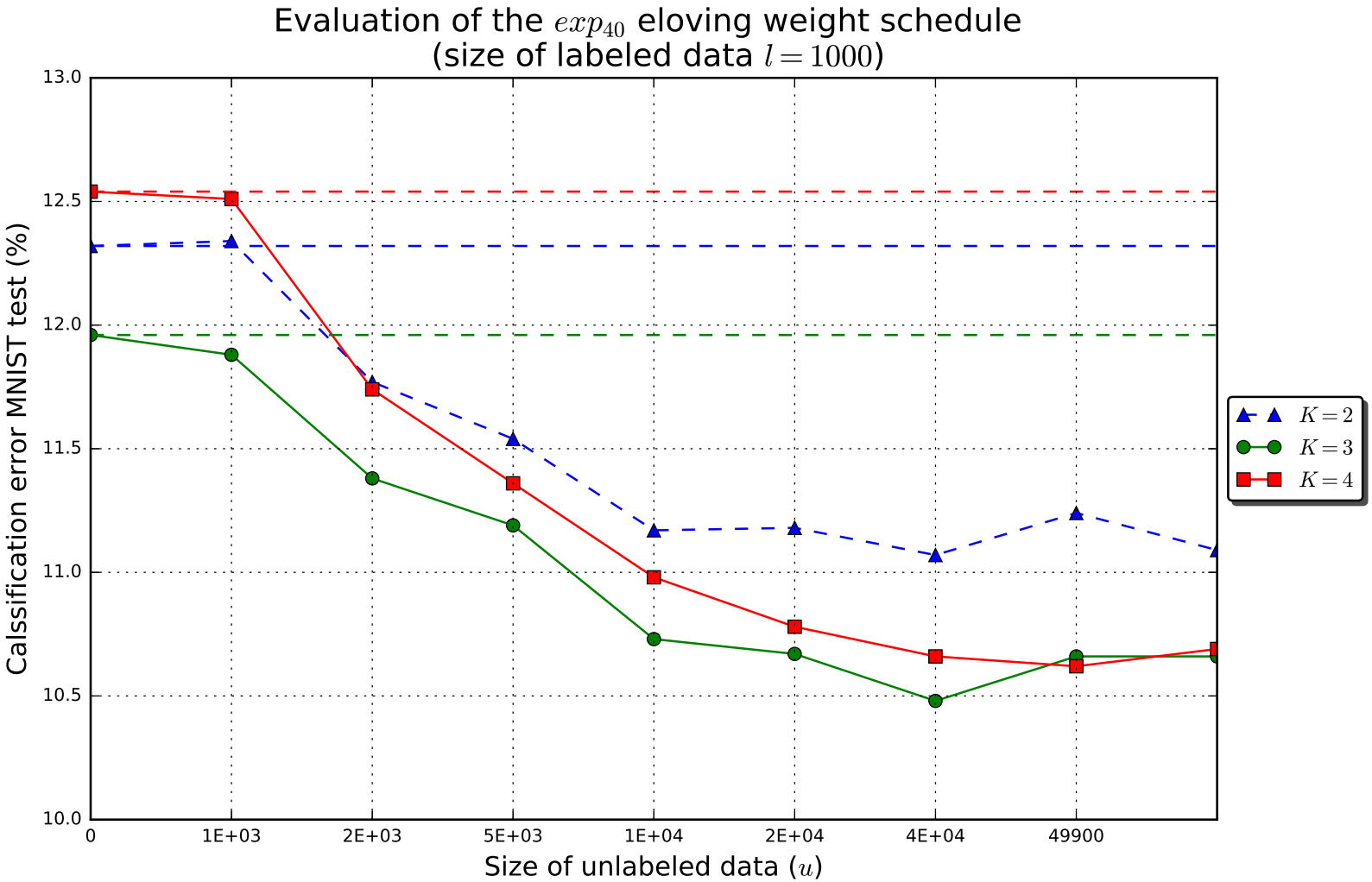
# Shallow networks: ($K = 1$, $l = 1E2$)



# Shallow networks: ($K = 1,\ l = 1E3$)



# Deep networks: exponential schedule ($l = 1E3$)



# Conclusion

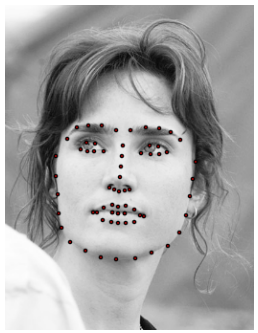
- An alternative method to the pre-training.

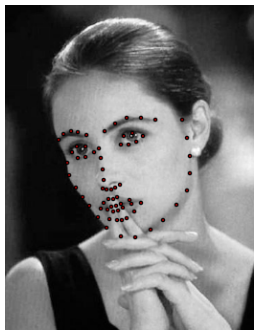
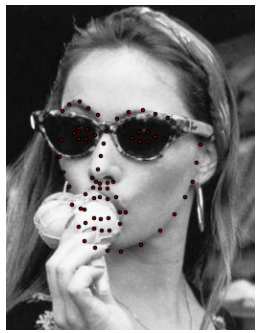
  Parallel transfer learning with evolving weights
- Improve generalization easily.
- Reduce the number of hyper-parameters ($t_1$, $\sigma$)

# Perspectives

- Evolve the importance weight according to the train/validation error.
- Explore other evolving schedules (toward automatic schedule)
- Optimization
- **Extension to structured output problems**

$$\text{Train cost} = \textbf{supervised task} + \textbf{Input unsupervised task} + \textbf{Output unsupervised task}$$



# Questions

Thank you for your attention,

Questions?