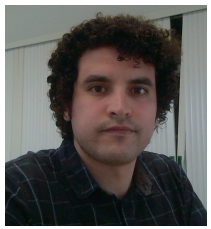
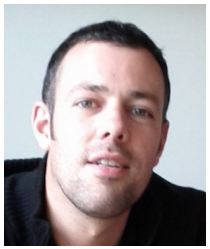
Neural networks regularization through representation learning

- presentation of my PhD work -

Japanese-French workshop on optimization for machine learning (Riken & LITIS),
INSA de Rouen. Sept.25th, 2017






Soufiane Belharbi Romain Hérault Clément Chatelain Sébastien Adam

soufiane.belharbi@insa-rouen.fr (https://sbelharbi.github.io)

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

My PhD work

Key words: *neural networks, regularization, representation learning.*

Selected work:

1. **A regularization framework for training neural networks for structured output problems.**

   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Multi-task Learning for Structured Output Prediction***. Under review, Neurocomputing. ArXiv: arxiv.org/abs/1504.07550. 2017.

2. **A regularization framework for training neural networks for classification.**

   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Neural Networks Regularization Through Class-wise Invariant Representation Learning***. In preparation for IEEE TNNLS. ArXiv: arxiv.org/abs/1709.01867. 2017.

3. **Transfer learning in neural networks: an application to medical domain.**

   S. Belharbi, R.Hérault, C. Chatelain, R. Modzelewski, S. Adam, M. Chastan, S. Thureau, ***Spotting L3 slice in CT scans using deep convolutional network and transfer learning***. In Medical Image Analysis journal (MIA). 2017.

Machine Learning

What is Machine Learning (ML)?

ML is programming computers (algorithms) to optimize a performance criterion using **example data or past experience**.

Learning a task

Learn general models from data to perform a specific task f.

$$f_{\mathbf{w}} : \mathbf{x} \longrightarrow \mathbf{y}$$

x: input, **y**: output (target, label)
w: parameters of $f(\cdot)$, $f(\mathbf{x}; \mathbf{w}) = \mathbf{y}$.
Find **w**: $\mathcal{E}_{train} = \mathbb{E}_{(x,y)\sim P_{data}}[l(f(x; \mathbf{w}), y)]$.

Learning is the capability to generalize

1. **Generalization:** $\mathcal{E}_{train} \approx \mathcal{E}_{test}$. (**challenge!!!**).
2. **Overfitting** (model capacity, maximum likelihood estimate).
3. The **no free lunch theorem**: your training algorithm can not be the best at every task: focus on the task in hand.
4. **Regularization**: to better generalize use **prior knowledge** about the task.

My PhD work

Key words: *neural networks, regularization, representation learning.*

Selected work:

1. **A regularization framework for training neural networks for structured output problems.**

   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Multi-task Learning for Structured Output Prediction***. Under review, Neurocomputing. ArXiv: arxiv.org/abs/1504.07550. 2017.

2. **A regularization framework for training neural networks for classification.**

   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Neural Networks Regularization Through Class-wise Invariant Representation Learning***. In preparation for IEEE TNNLS. ArXiv: arxiv.org/abs/1709.01867. 2017.

3. **Transfer learning in neural networks: an application to medical domain.**

   S. Belharbi, R.Hérault, C. Chatelain, R. Modzelewski, S. Adam, M. Chastan, S. Thureau, ***Spotting L3 slice in CT scans using deep convolutional network and transfer learning***. In Medical Image Analysis journal (MIA). 2017.

Traditional Machine Learning Problems

$$f : \mathcal{X} \rightarrow y$$

- Inputs $\mathcal{X} \in \mathbb{R}^d$: any type of input
- Outputs $y \in \mathbb{R}$ for the task: classification, regression, ...

Machine Learning for Structured Output Problems

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

- Inputs $\mathcal{X} \in \mathbb{R}^d$: any type of input
- Outputs $\mathcal{Y} \in \mathbb{R}^{d'}, d' > 1$ a structured object (*dependencies*)

See C. Lampert slides.

Traditional Machine Learning Problems

$$f : \mathcal{X} \rightarrow y$$

- Inputs $\mathcal{X} \in \mathbb{R}^d$: any type of input
- Outputs $y \in \mathbb{R}$ for the task: classification, regression, ...

Machine Learning for Structured Output Problems

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

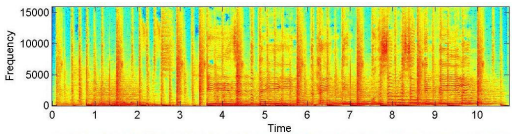
- Inputs $\mathcal{X} \in \mathbb{R}^d$: any type of input
- Outputs $\mathcal{Y} \in \mathbb{R}^{d'}, d' > 1$ a structured object (*dependencies*)

See C. Lampert slides.

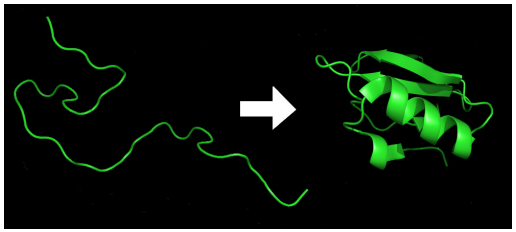
Data $=$ *representation* (*values*) $+$ *structure* (*dependencies*)

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

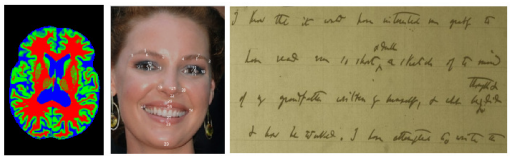
Text: part-of-speech tagging, translation



speech $\rightleftarrows$ *text*



Protein folding



Image

Structured data

Approaches that Deal with Structured Output Data

- Kernel based methods: Kernel Density Estimation (KDE)
- Discriminative methods: Structure output SVM
- Graphical methods: HMM, CRF, MRF, ...

Drawbacks

- Perform one single data transformation
- Difficult to deal with *high dimensional* data

Ideal approach

- Structured output problems
- High dimension data
- Multiple data transformation (complex mapping functions)

Deep neural networks?

Approaches that Deal with Structured Output Data

- Kernel based methods: Kernel Density Estimation (KDE)
- Discriminative methods: Structure output SVM
- Graphical methods: HMM, CRF, MRF, ...

Drawbacks

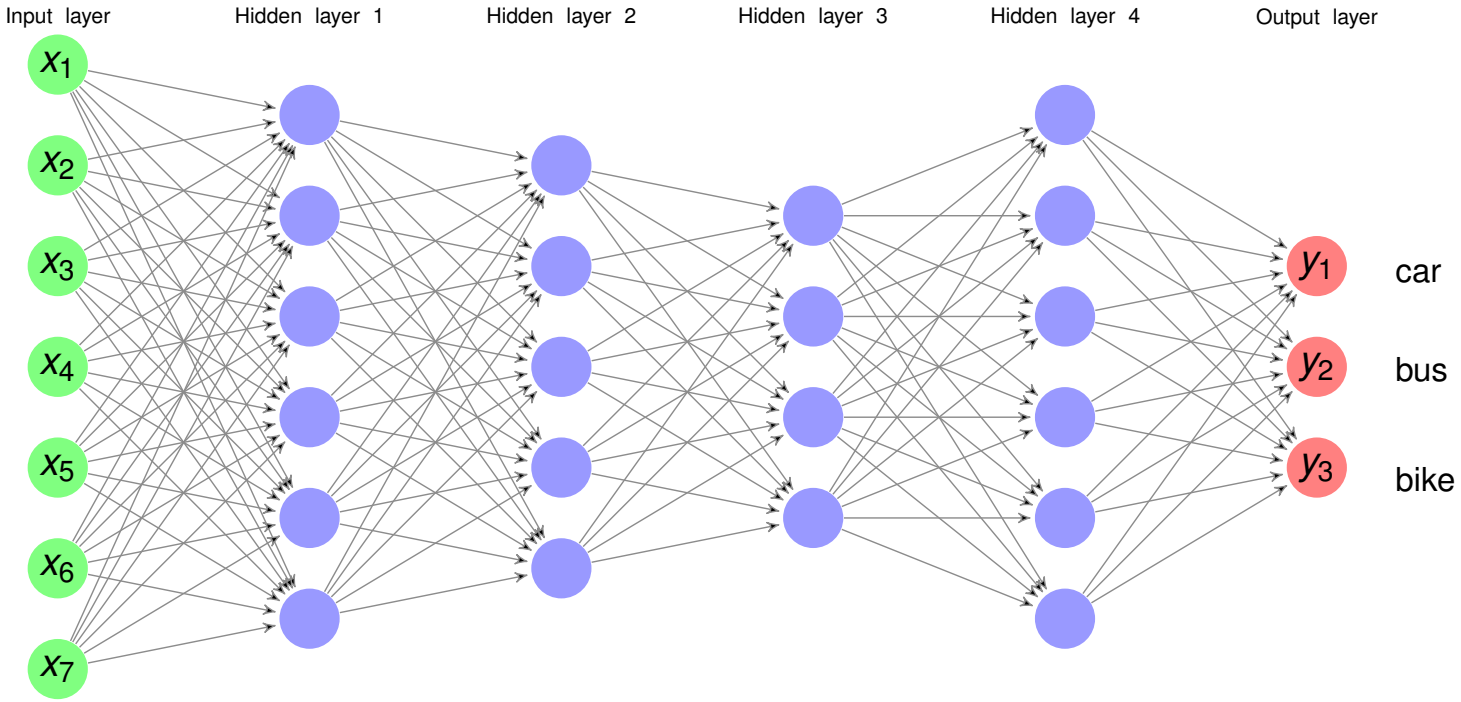
- Perform one single data transformation
- Difficult to deal with *high dimensional* data

Ideal approach

- Structured output problems
- High dimension data
- Multiple data transformation (complex mapping functions)

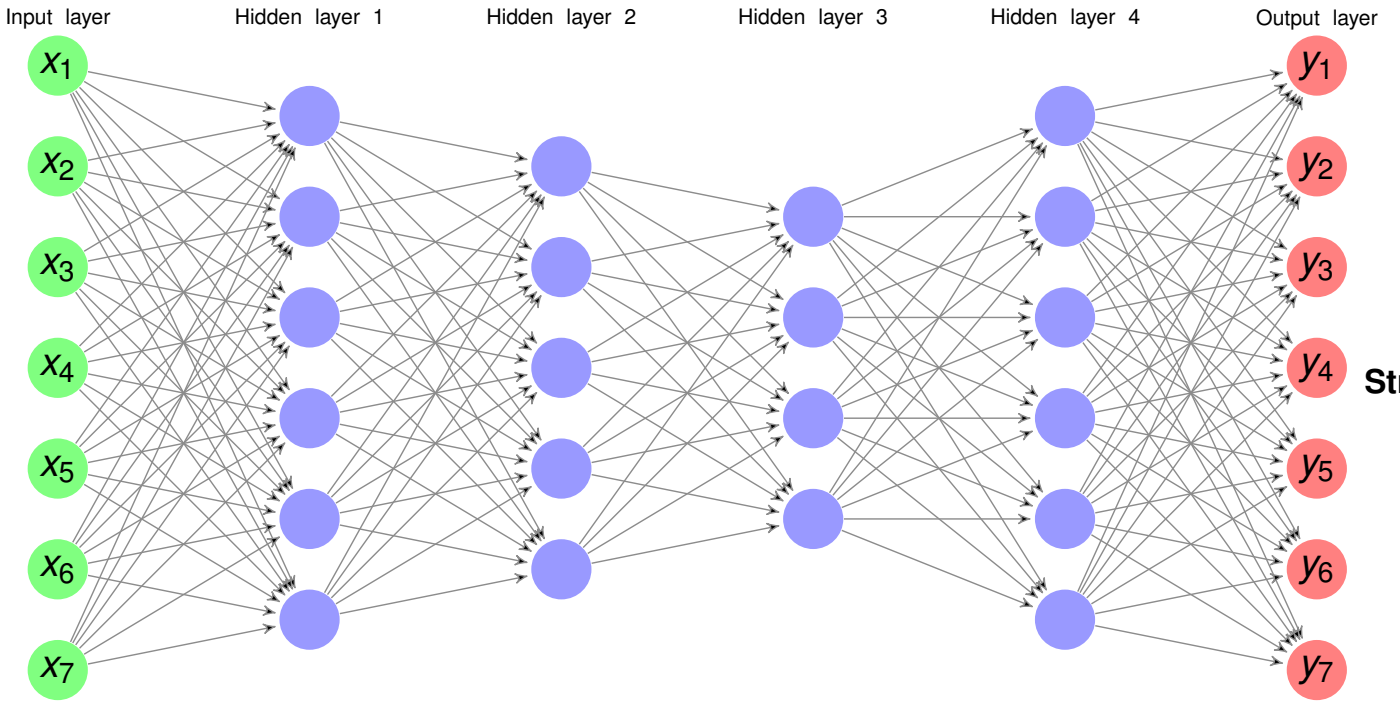
Deep neural networks?

Traditional Deep neural Network

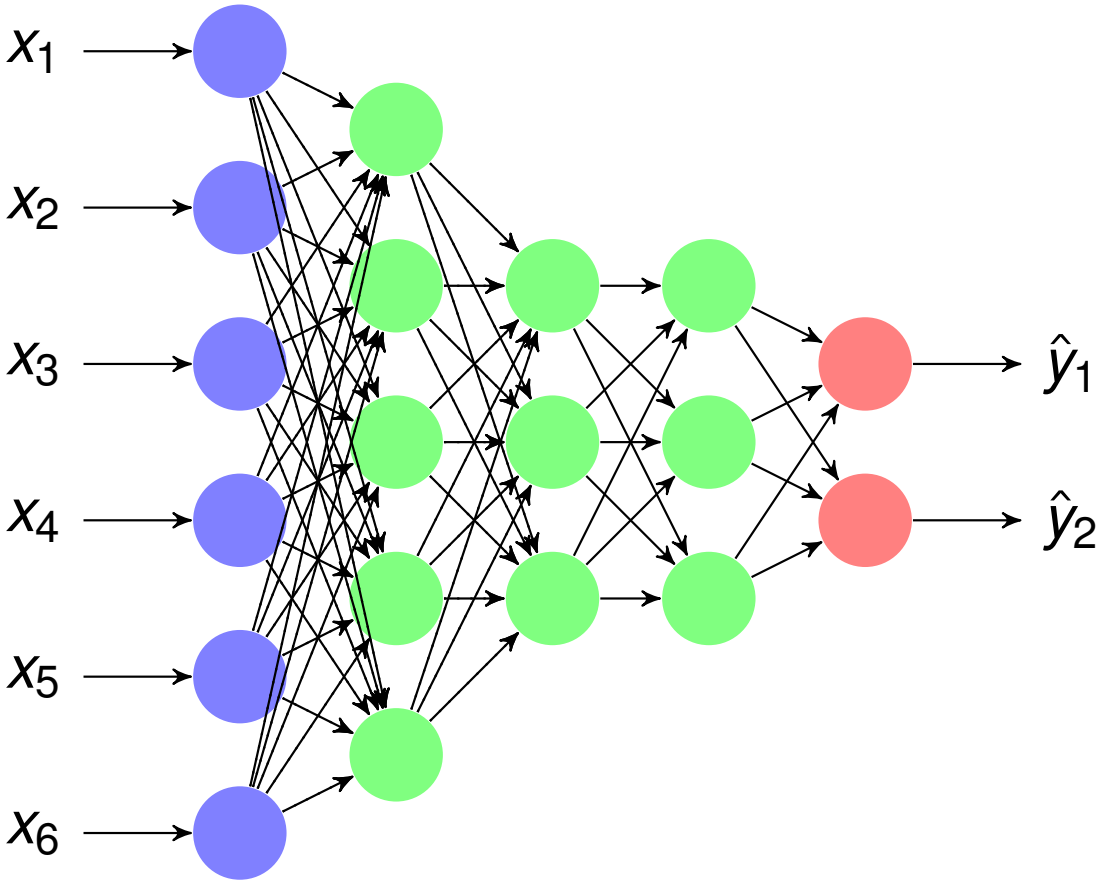


- High dimension data OK
- Multiple data transformation (complex mapping functions) OK
- Structured output problems NO

High dimensional output:



Unsupervised learning: Layer-wise pre-training, auto-encoders



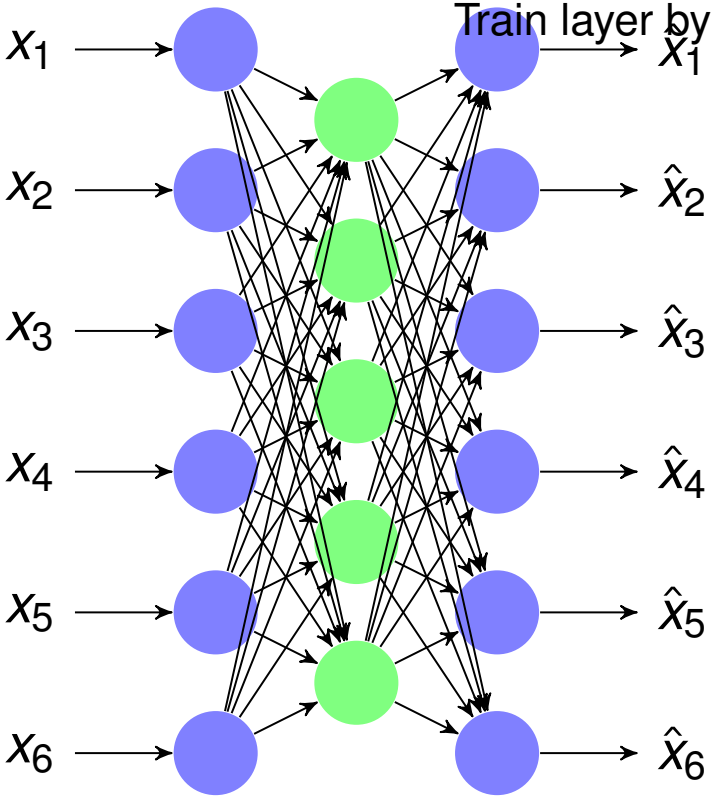
A DNN to train:

1. Use unsupervised training to initialize the network.
2. Finetune the network using supervised data.

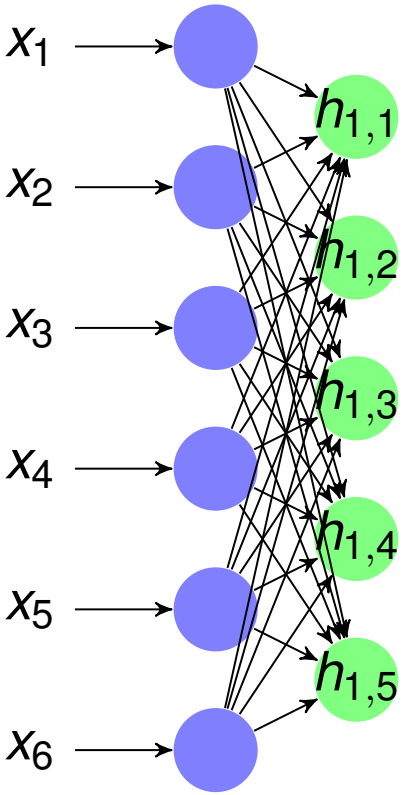
Unsupervised learning: Layer-wise pre-training, auto-encoders

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

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



Unsupervised learning: Layer-wise pre-training, auto-encoders



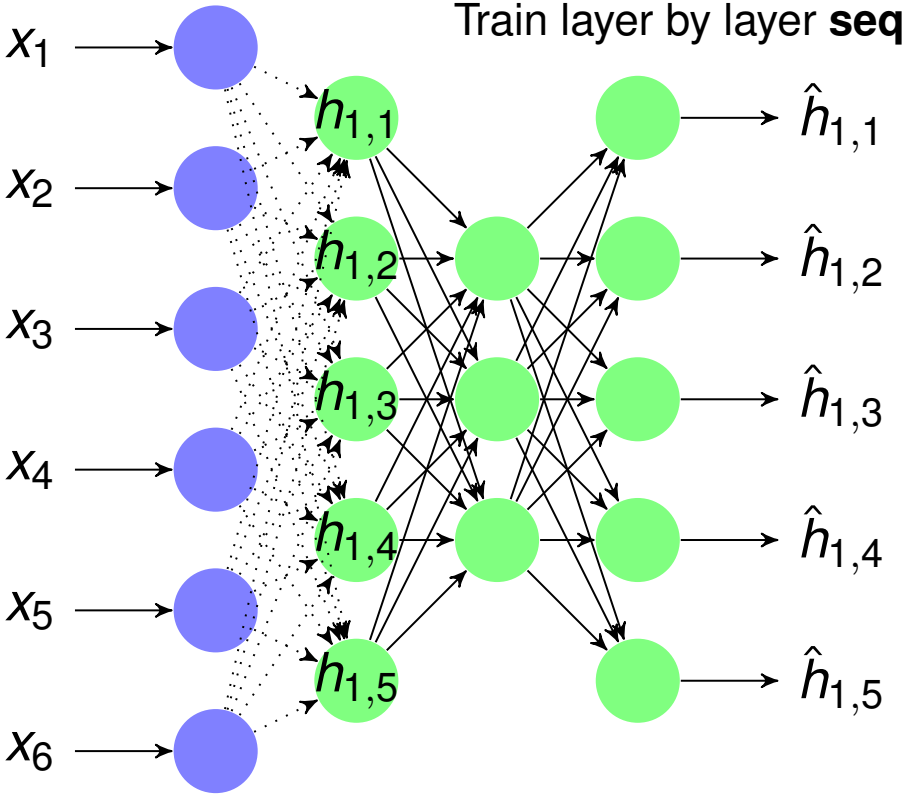
1) Step 1: **Unsupervised layer-wise pre-training**

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

Unsupervised learning: Layer-wise pre-training, auto-encoders

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

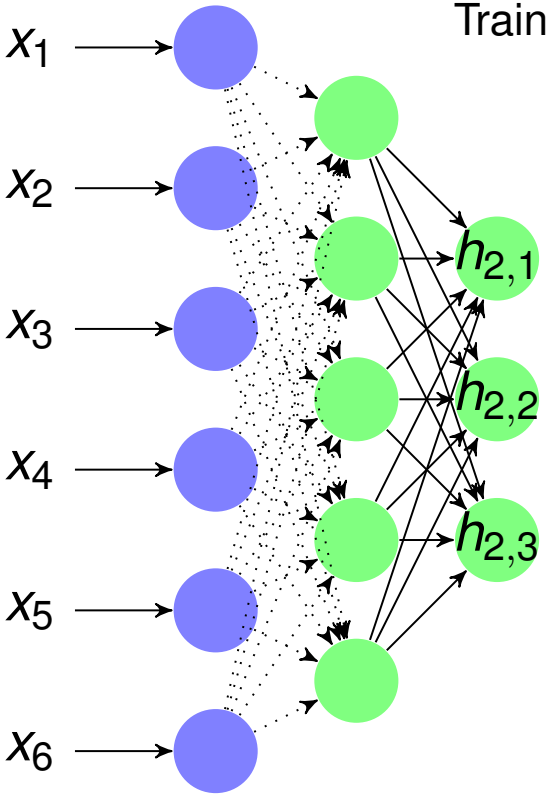
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



Unsupervised learning: Layer-wise pre-training, auto-encoders

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

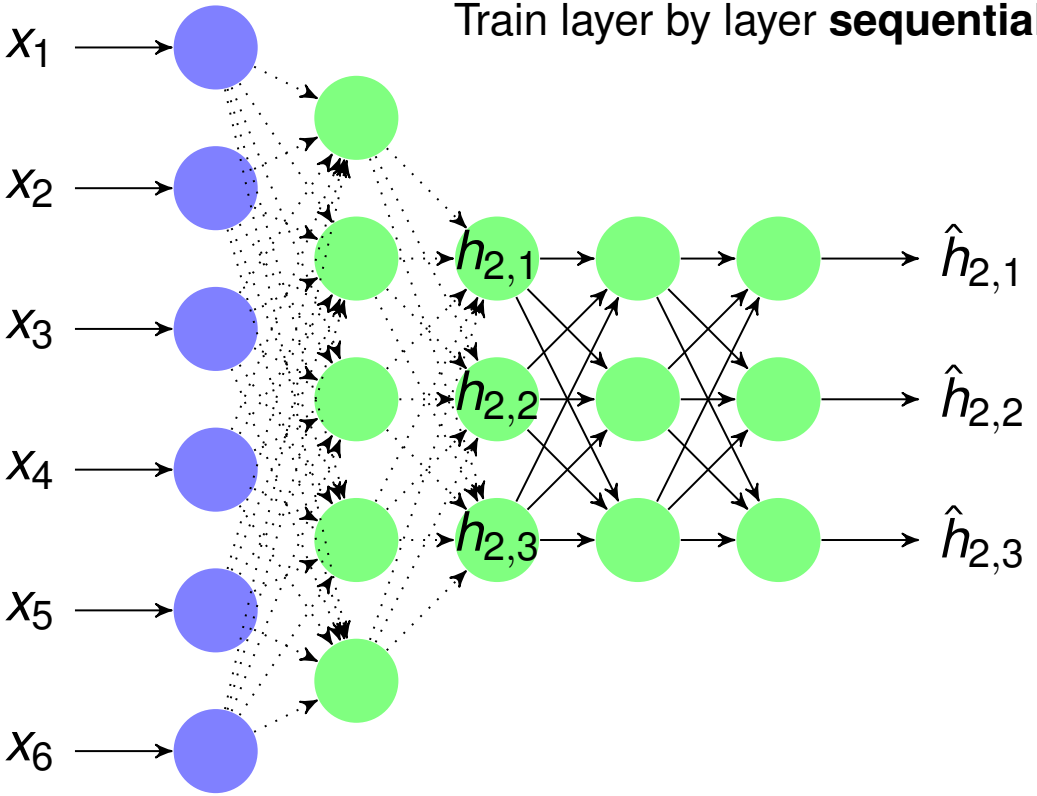
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



Unsupervised learning: Layer-wise pre-training, auto-encoders

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

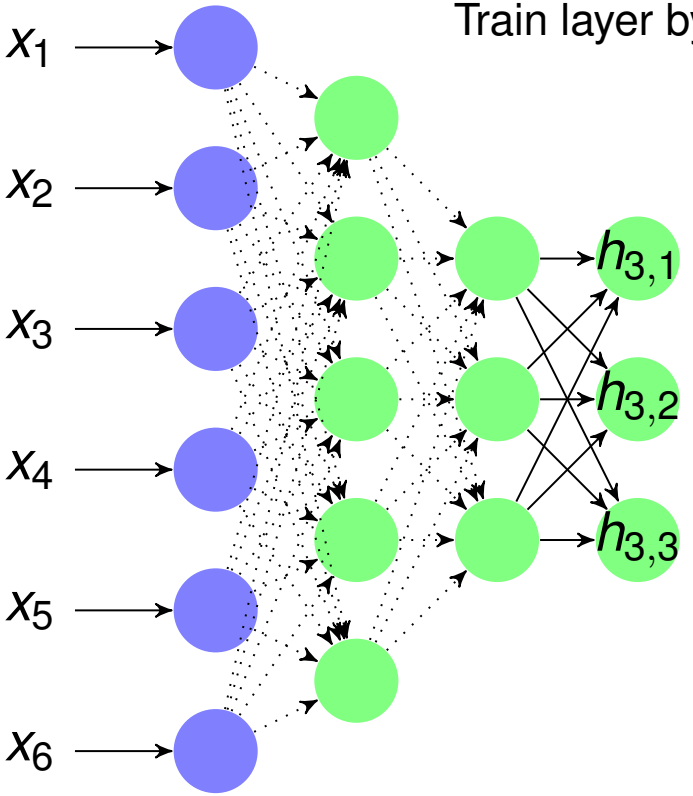
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



Unsupervised learning: Layer-wise pre-training, auto-encoders

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

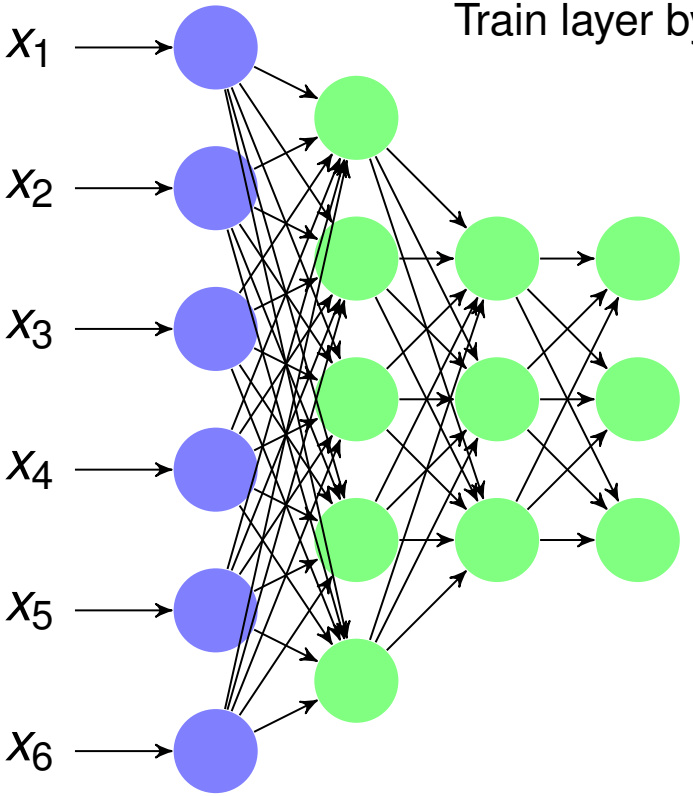
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



Unsupervised learning: Layer-wise pre-training, auto-encoders

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

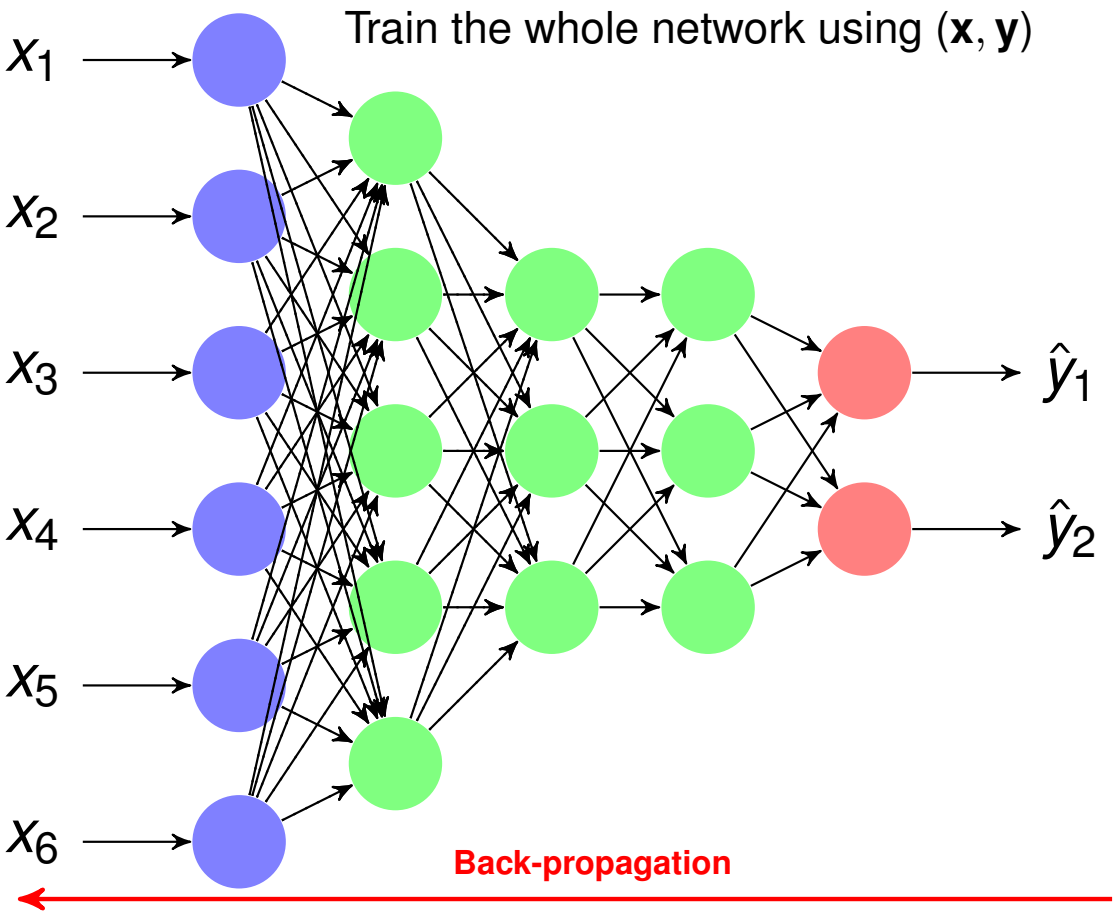
Train layer by layer **sequentially** using **only x** (labeled or unlabeled)



$\Rightarrow$ Unsupervisedly pre-trained network.

Unsupervised learning: Layer-wise pre-training, auto-encoders

2) Step 2: **Supervised training**



Unsupervised learning: Layer-wise pre-training, auto-encoders

This Why is it difficult in practice?

$\Rightarrow$ **Sequential** transfer learning

Possible solution:

$\Rightarrow$ **Parallel** transfer learning (1)

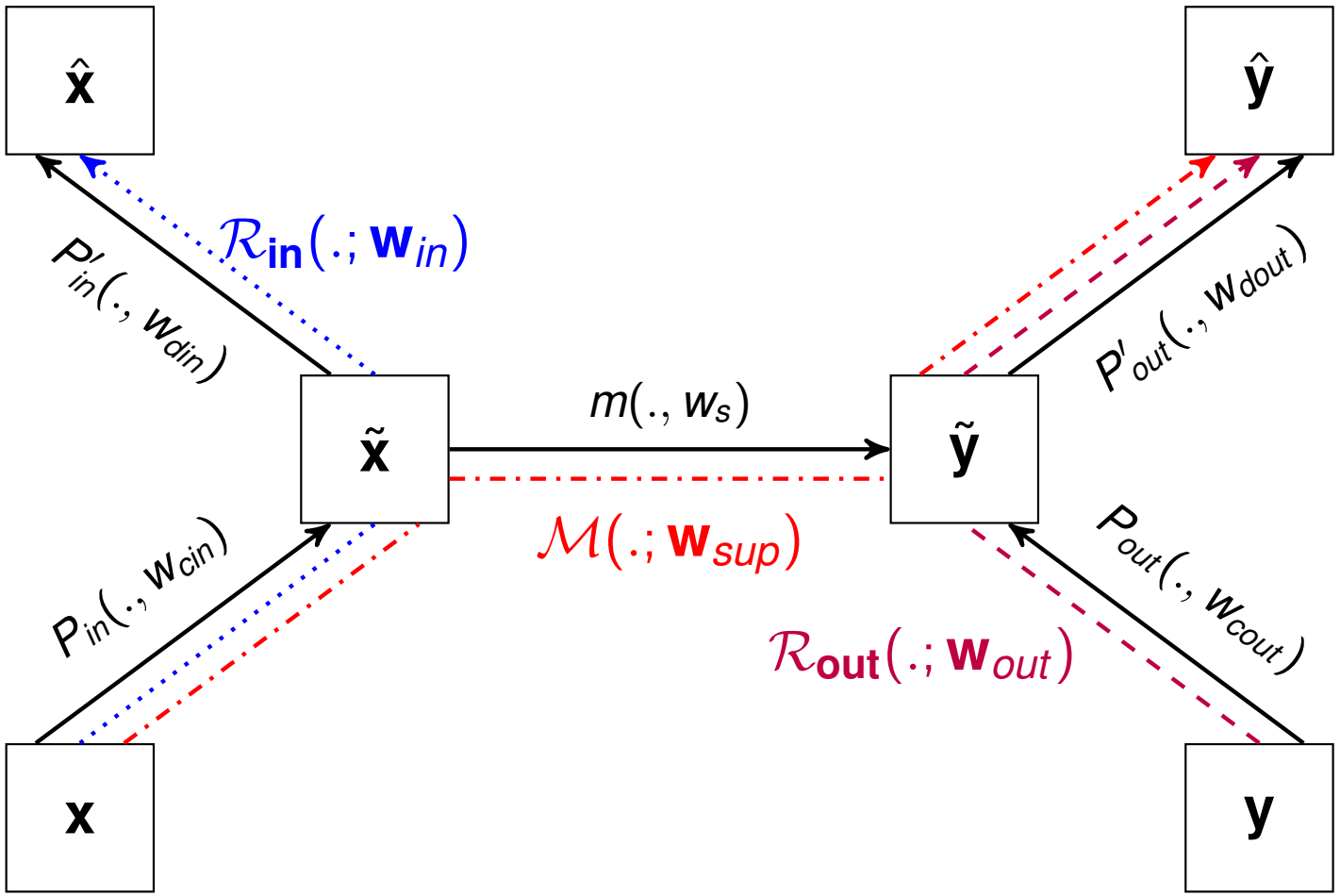
Why in parallel?

- Interaction between tasks
- Reduce the number of hyper-parameters to tune
- Provide **one stopping criterion**
- **Prevent overfitting of the unsupervised task.**

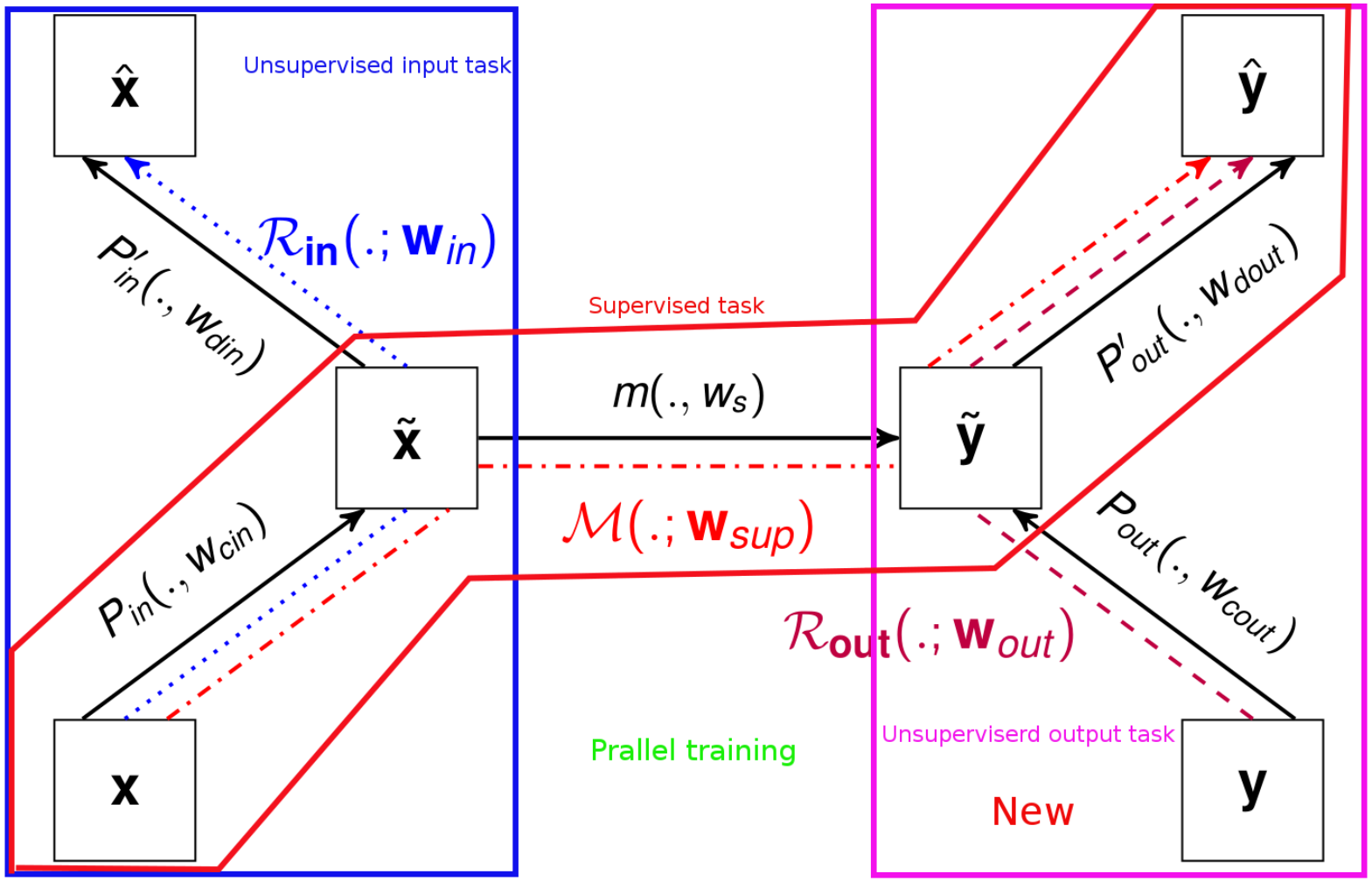
See our work:

(1): S. Belharbi, R.Hérault, C. Chatelain, S. Adam, ***Deep multi-task learning with evolving weights***, in conference: European Symposium on Artificial Neural Networks (ESANN), 2016.

Proposed framework

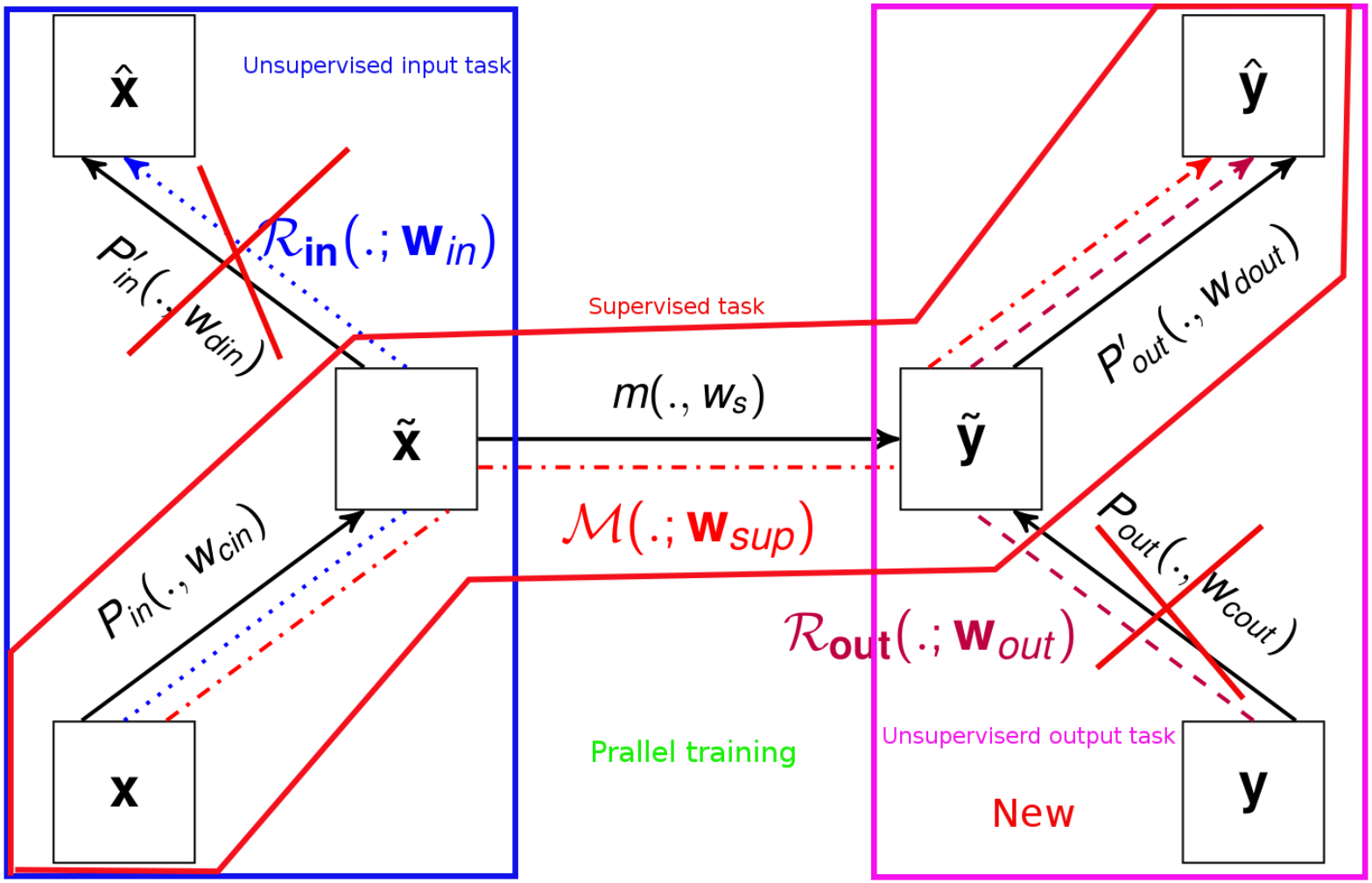


Proposed framework



Training.

Proposed framework



End of training.

Proposed framework

$\mathcal{F}$: all the $\mathbf{x}$, $\mathcal{L}$: all the $\mathbf{y}$, $\mathcal{S}$: all supervised data

Input task

- $$\hat{\mathbf{x}} = \mathcal{R}_{in}\left(\mathbf{x}; \mathbf{w}_{in}\right) = P'_{in}\left(\tilde{\mathbf{x}} = P_{in}\left(\mathbf{x}; \mathbf{w}_{cin}\right); \mathbf{w}_{din}\right) \ ,$$
- $$\mathcal{J}_{in}(\mathcal{F}; \mathbf{w}_{in}) = \frac{1}{\text{card}\,\mathcal{F}} \sum_{x \in \mathcal{F}} \mathcal{C}_{in}(\mathcal{R}_{in}(\mathbf{x}; \mathbf{w}_{in}), \mathbf{x}) \ .$$

Output task

- $$\hat{\mathbf{y}} = \mathcal{R}_{out}\left(\mathbf{y}; \mathbf{w}_{out}\right) = P'_{out}\left(\tilde{\mathbf{y}} = P_{out}\left(\mathbf{y}; \mathbf{w}_{cout}\right); \mathbf{w}_{dout}\right) \ ,$$
- $$\mathcal{J}_{out}(\mathcal{L}; \mathbf{w}_{out}) = \frac{1}{\text{card}\,\mathcal{L}} \sum_{y \in \mathcal{L}} \mathcal{C}_{out}(\mathcal{R}_{out}(\mathbf{y}; \mathbf{w}_{out}), \mathbf{y}) \ .$$

Main task

- $$\hat{\mathbf{y}} = \mathcal{M}\left(\mathbf{x}; \mathbf{w}_{sup}\right) = P'_{out}\left(m\left(P_{in}\left(\mathbf{x}; \mathbf{w}_{cin}\right); \mathbf{w}_{s}\right); \mathbf{w}_{dout}\right) \ ,$$
- $$\mathcal{J}_{s}(\mathcal{S}; \mathbf{w}_{sup}) = \frac{1}{\text{card}\,\mathcal{S}} \sum_{(x,y) \in \mathcal{S}} \mathcal{C}_{s}(\mathcal{M}(x; \mathbf{w}_{sup}), y) \ .$$

Tasks combination

$$\mathcal{J}(\mathcal{D};\mathbf{w}) = \lambda_{sup}(t)\cdot\mathcal{J}_s(\mathcal{S};\mathbf{w}_{sup}) + \lambda_{in}(t)\cdot\mathcal{J}_{in}(\mathcal{F};\mathbf{w}_{in}) + \lambda_{out}(t)\cdot\mathcal{J}_{out}(\mathcal{L};\mathbf{w}_{out})\ ,$$

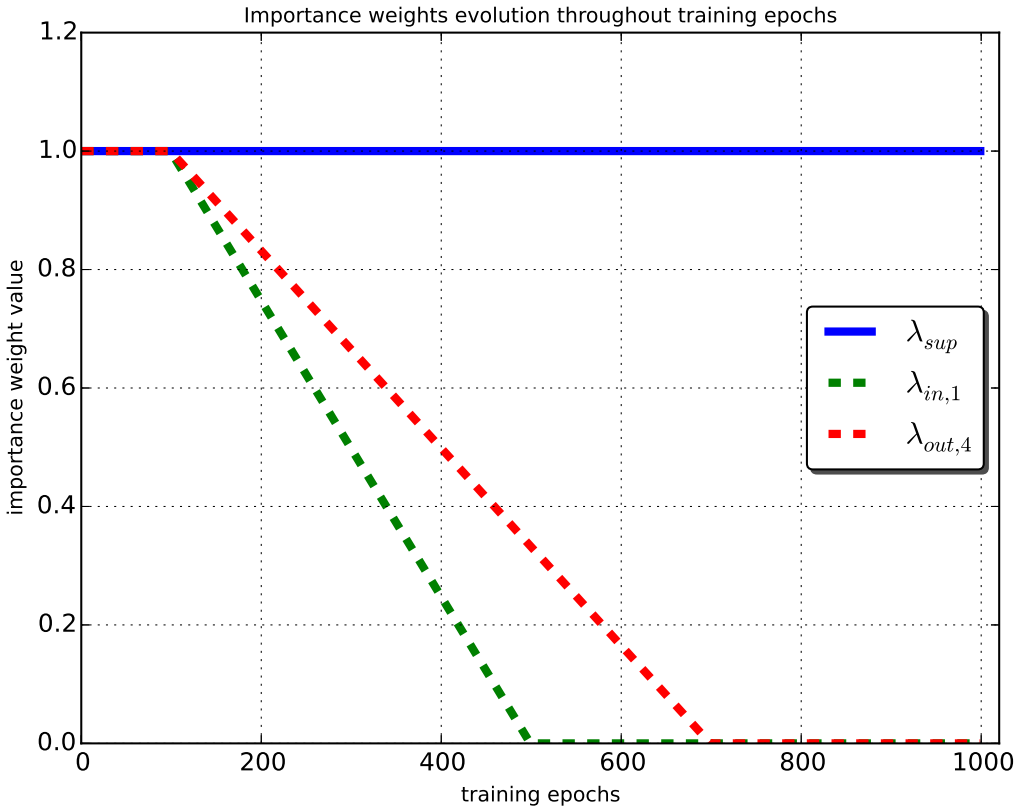


Figure 3 : Linear evolution of the importance weights during training.

Framework training

Algorithm 1 Training our framework for one epoch

$\mathcal{D}$ is the *shuffled* training set. B a mini-batch.
for B in $\mathcal{D}$ **do**
    $B_{\mathcal{S}} \Leftarrow$ examples of B that contain both $(\mathbf{x}, \mathbf{y})$.
    $B_{\mathcal{F}} \Leftarrow$ all the $\mathbf{x}$ samples of B.
    $B_{\mathcal{L}} \Leftarrow$ all the $\mathbf{y}$ samples of B.
    `Update` $\mathbf{w}_{in}$:
     $\rightarrow$ Make a gradient step toward $\mathcal{J}_{in}$ using $B_{\mathcal{F}}$.
    `Update` $\mathbf{w}_{out}$:
     $\rightarrow$ Make a gradient step toward $\mathcal{J}_{out}$ using $B_{\mathcal{L}}$.
    `Update` $\mathbf{w}_{sup}$:
     $\rightarrow$ Make a gradient step toward $\mathcal{J}_{s}$ using $B_{\mathcal{S}}$.
    `Update` λ_{sup}, λ_{in} `and` λ_{out}.
end for

Framework evaluation

Task: Facial landmark detection. Localize 68 points (x,y).

Experiments: setup

- Datasets: LFPW (1035 images), HELEN (2330 images)
- Each dataset: train, valid, test.
- Architecture: MLP with 4 hidden layers: 1025, 2500, 136, 64.
- In: 50x50. Output: 68x2
- **Data augmentation** (use extra **x** and/or **y** from other dataset), **no data augmentation** (use only the provided $(\mathbf{x}, \mathbf{y})$)

Experiments: Results (No data augmentation)

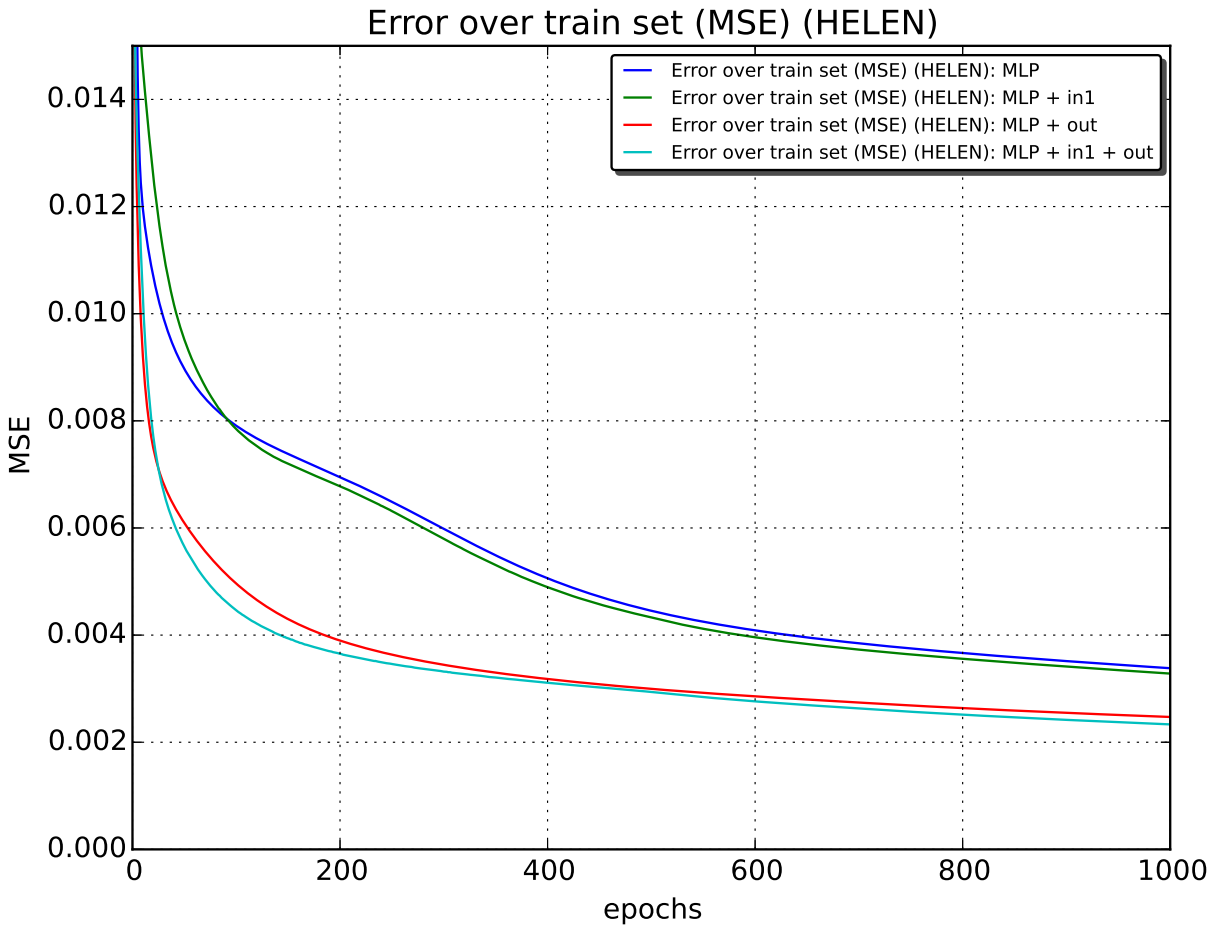


Figure 5 : MSE during training epochs over HELEN train set using different training setups for the MLP (no augmentation).

Experiments: Results (No data augmentation)

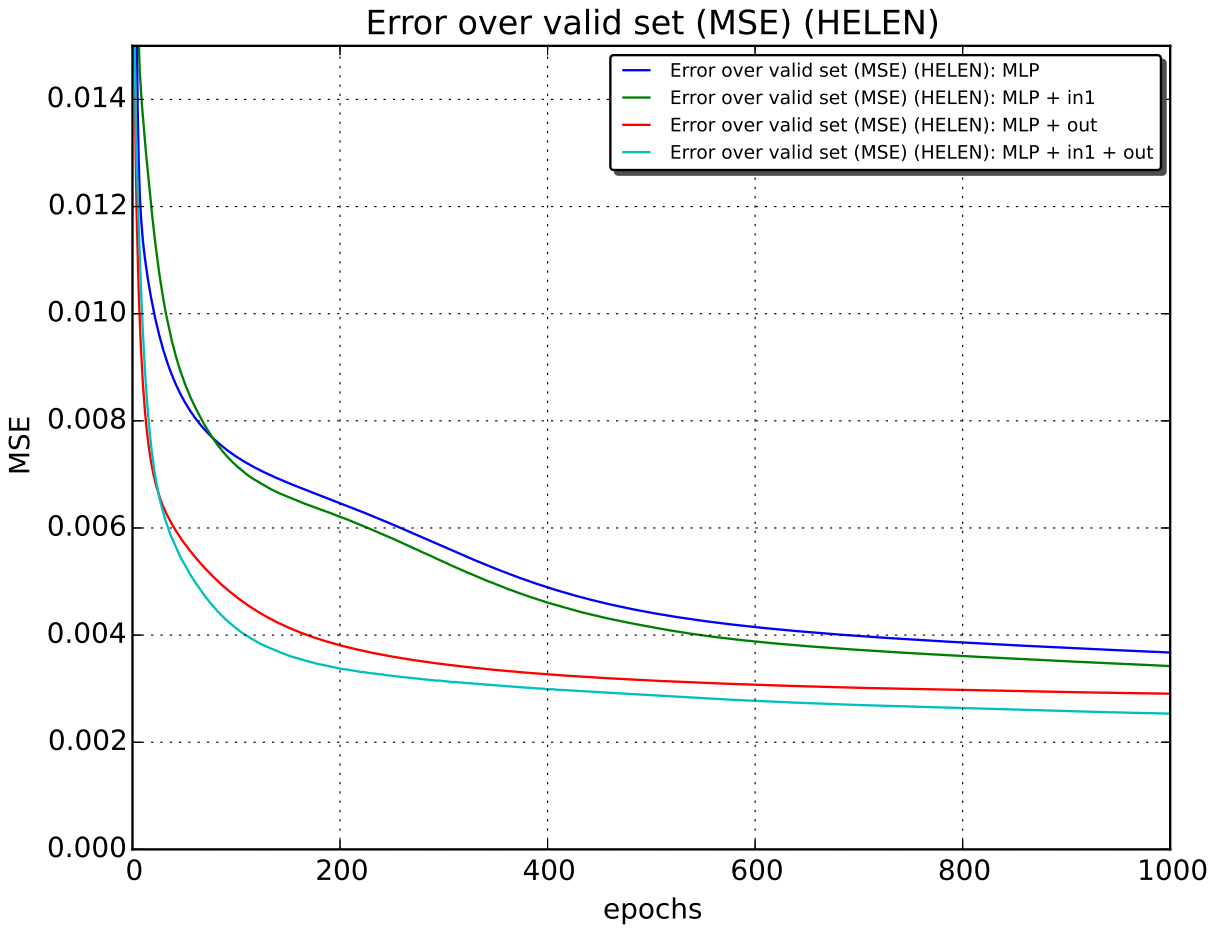


Figure 6 : MSE during training epochs over HELEN valid set using different training setups for the MLP (no augmentation).

Experiments: Results (No data augmentation)

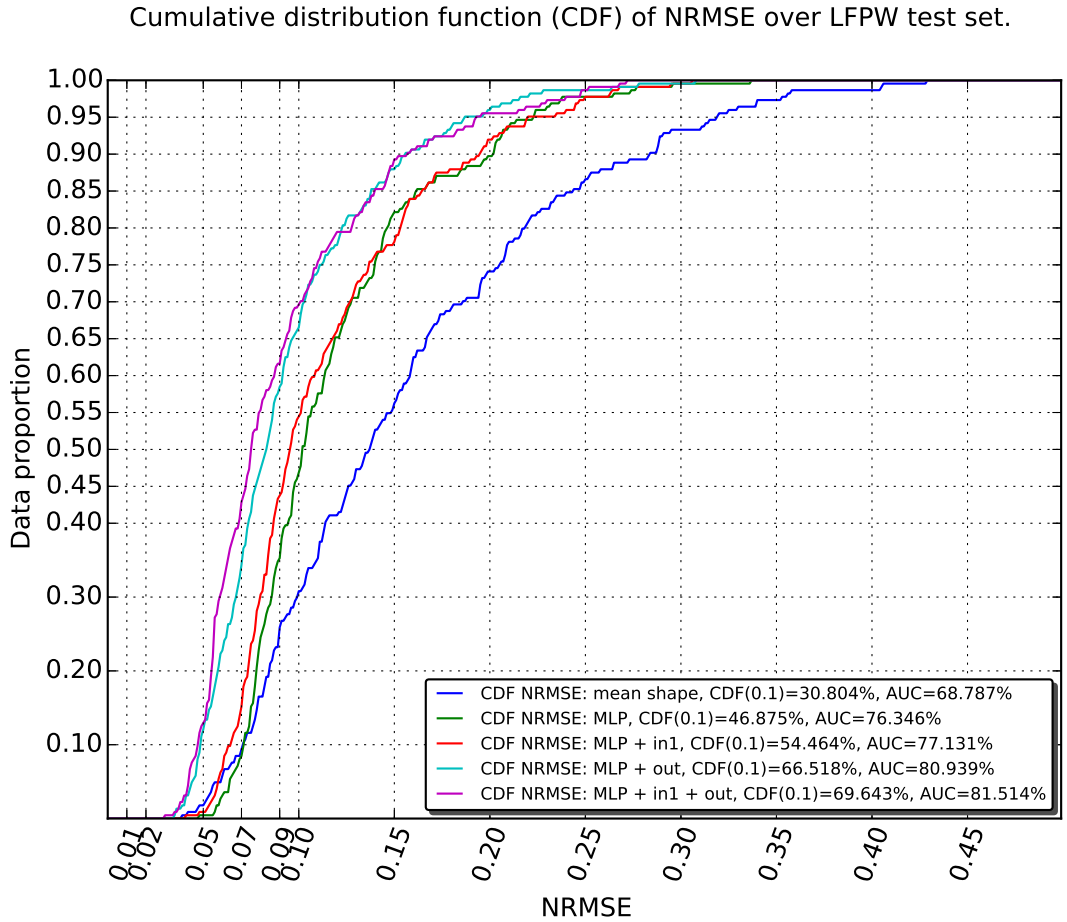


Figure 7 : CDF (Cumulative Distribution Function) curves of different configurations on LFPW.

Experiments: Results (No data augmentation)

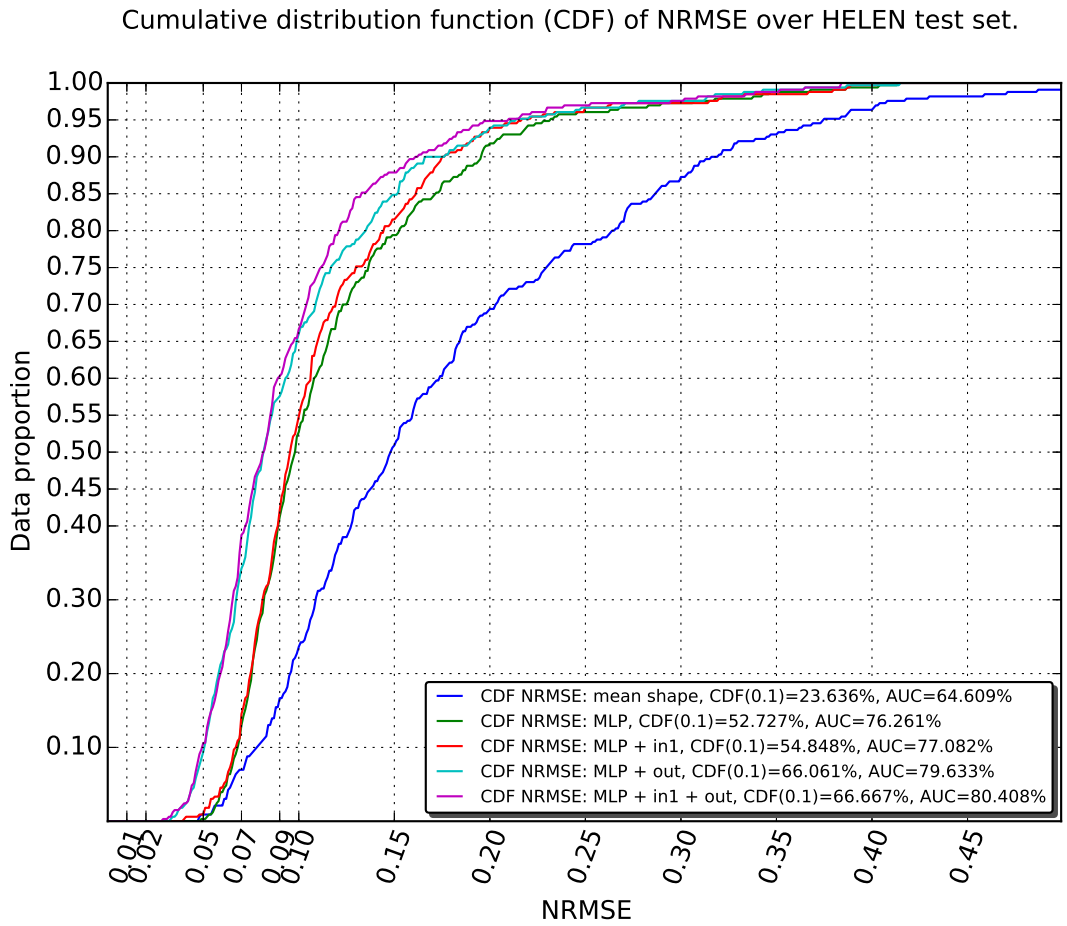


Figure 8 : CDF curves of different configurations on HELEN.

Experiments: Results (With data augmentation)

Table 1 : MSE over LFPW: train and valid sets, at the end of training with and without data augmentation.

	No augmentation		**With augmentation**	
	MSE train	MSE valid	MSE train	MSE valid
Mean shape	7.74×10^{-3}	8.07×10^{-3}	7.78×10^{-3}	8.14×10^{-3}
MLP	3.96×10^{-3}	4.28×10^{-3}	-	-
MLP + in	3.64×10^{-3}	3.80×10^{-3}	1.44×10^{-3}	2.62×10^{-3}
MLP + out	2.31×10^{-3}	2.99×10^{-3}	1.51×10^{-3}	2.79×10^{-3}
MLP + in + out	$\mathbf{2.12 \times 10^{-3}}$	$\mathbf{2.56 \times 10^{-3}}$	$\mathbf{1.10 \times 10^{-3}}$	$\mathbf{2.23 \times 10^{-3}}$

Experiments: Results (With data augmentation)

Table 2 : **AUC** and **$CDF_{0.1}$** performance over LFPW test dataset with and without data augmentation.

	No augmentation		with augmentation	
	AUC	**$CDF_{0.1}$**	**AUC**	**$CDF_{0.1}$**
Mean shape	68.78%	30.80%	77.81%	22.33%
MLP	76.34%	46.87%	-	-
MLP + in	77.13%	54.46%	80.78%	67.85%
MLP + out	80.93%	66.51%	81.77%	67.85%
MLP + in + out	**81.51%**	**69.64%**	**82.48%**	**71.87%**

Table 3 : **AUC** and **$CDF_{0.1}$** performance over HELEN test dataset with and without data augmentation.

	No augmentation		With augmentation	
	AUC	**$CDF_{0.1}$**	**AUC**	**$CDF_{0.1}$**
Mean shape	64.60%	23.63%	64.76%	23.23%
MLP	76.26%	52.72%	-	-
MLP + in	77.08%	54.84%	79.25%	63.33%
MLP + out	79.63%	66.60%	80.48%	65.15%
MLP + in + out	**80.40%**	**66.66%**	**81.27%**	**71.51%**

Experiments: Visual results

Figure 9 : Examples of prediction on LFPW test set. For visualizing errors, red segments have been drawn between ground truth and predicted landmark. Top row: MLP. Bottom row: MLP+in+out. (no data augmentation)

Experiments: Visual results

Figure 10 : Examples of prediction on HELEN test set. Top row: MLP. Bottom row: MLP+in+out. (no data augmentation)

Conclusion

- Generic regularization scheme for structured output problems based on transfer learning
- Exploit input/output unlabeled data
- Speedup convergence and improve generalization
- Code at github:
  https://github.com/sbelharbi/structured-output-ae

Perspectives

- Evolve the importance weight according to the train/validation error.
- Explore other evolving schedules (toward automatic schedule)

My PhD work

Key words: *neural networks, regularization, representation learning.*

Selected work:

1. **A regularization framework for training neural networks for structured output problems.**

   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Multi-task Learning for Structured Output Prediction***. Under review, Neurocomputing. ArXiv: arxiv.org/abs/1504.07550. 2017.

2. **A regularization framework for training neural networks for classification.**

   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Neural Networks Regularization Through Class-wise Invariant Representation Learning***. In preparation for IEEE TNNLS. ArXiv: arxiv.org/abs/1709.01867. 2017.

3. **Transfer learning in neural networks: an application to medical domain.**

   S. Belharbi, R.Hérault, C. Chatelain, R. Modzelewski, S. Adam, M. Chastan, S. Thureau, ***Spotting L3 slice in CT scans using deep convolutional network and transfer learning***. In Medical Image Analysis journal (MIA). 2017.

Intuition

Task: classification.
Objective: learn invariant representations (class-wise).

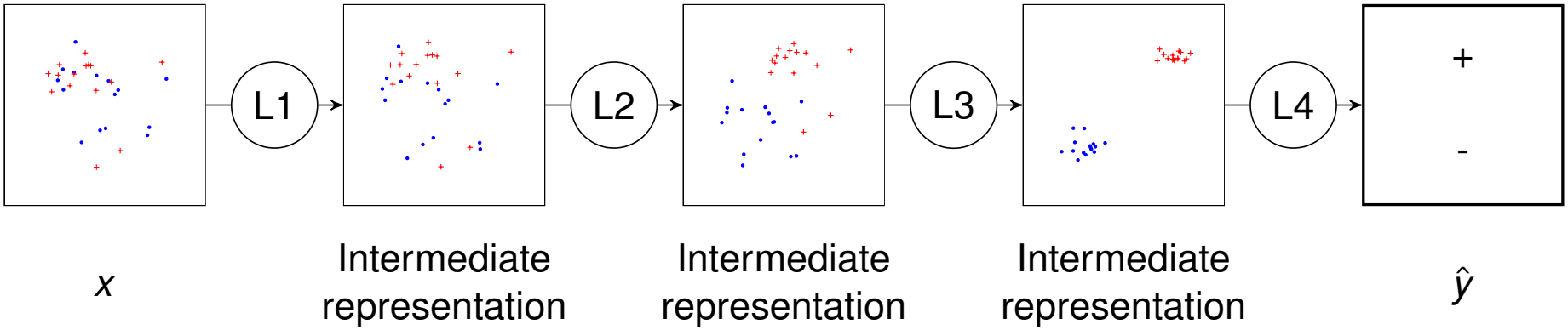


Figure 11 : Expected behavior of representation distributions within neural network for classification task.

Idea:
Promote this behavior by: **constraining samples within the same class to have the same representation within the network.**

Formulation

Network Decomposition

Let:

- $\mathcal{M}(\cdot;\theta) : \mathbf{X} \rightarrow \mathbf{Y}$ a mapping function, represented by a neural network.
- $\Gamma(\cdot;\theta_\Gamma) : \mathbf{X} \rightarrow \mathbf{Z}$ a **representation function**. **Z** is a representation space.
- $\Psi(\cdot;\theta_\Psi) : \mathbf{Z} \rightarrow \mathbf{Y}$ a **decision function** of **Z**.
- $\theta = \{\theta_\Gamma, \theta_\Psi\}$.

The network decision can be written as: $\mathcal{M}(x_i;\theta) = \Psi(\Gamma(x_i;\theta_\Gamma);\theta_\Psi)$.

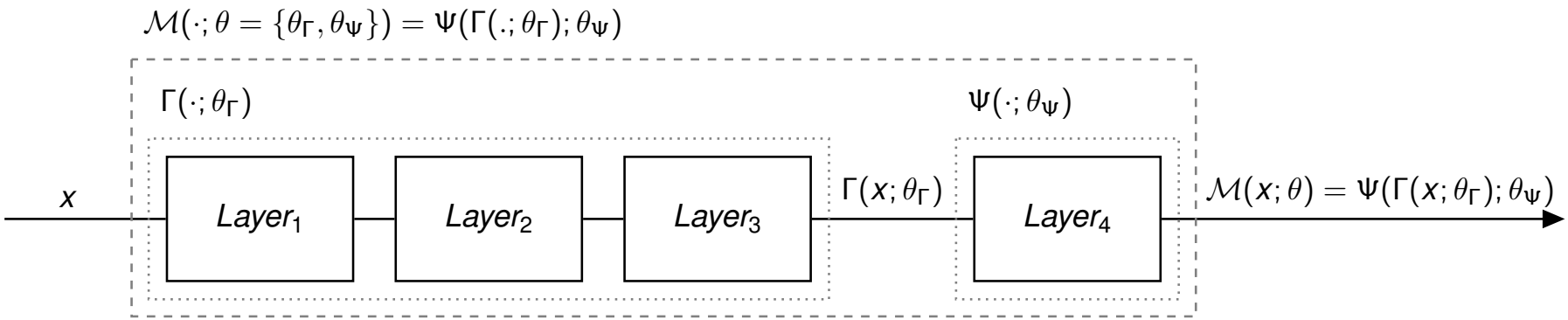


Figure 12 : A decomposition of a network with 4 layers.

Proposed Penalty

Dissimilarity Measure

Let:

- $\mathcal{D} = \{(x_i, y_i)\}$, training set with N samples and S classes.
- $\mathcal{D}_{\mathbf{x}} = \{\bigcup_{i=1}^{N} x_i\}$ set of all inputs.
- $\mathcal{L}(x_i)$ is a function that retrieves y_i the label of input sample x_i from $\mathcal{D}$.
- Partition $\mathcal{D}_{\mathbf{x}}$ into S sets:

  $$\mathcal{D}_{\mathbf{x}} = \{\bigcup_{s=1}^{S} \mathcal{D}_s, \quad \forall x_i \in \mathcal{D}_s \,|\, \mathcal{L}(x_i) = s\}.$$

For one samples x_i, we would like to minimize:

$$J_r(x_i; \theta_\Gamma) = \frac{1}{|\mathcal{D}_s| - 1} \sum_{\substack{x_j \in \mathcal{D}_s \\ j \neq i}} \mathcal{C}_r(\Gamma(x_i; \theta_\Gamma), \Gamma(x_j; \theta_\Gamma)) \tag{1}$$

$\mathcal{C}_r(\cdot, \cdot)$ is a loss function that measures how much two projections in $\mathbf{Z}$ are dissimilar and $|\mathcal{D}_s|$ is the number of samples in $\mathcal{D}_s$.

Proposed Penalty

Total training cost:

$$J(\mathcal{D};\theta) = \underbrace{\frac{\gamma}{N}\sum_{(x_i,y_i)\in\mathcal{D}} \mathcal{C}_{sup}(\Psi(\Gamma(x_i;\theta_\Gamma);\theta_\Psi),y_i)}_{\text{Supervised loss } \boldsymbol{J_{sup}}} + \underbrace{\frac{\lambda}{S}\sum_{s=1}^{S}\frac{1}{|\mathcal{D}_s|}\sum_{x_i\in\mathcal{D}_s} J_r(x_i;\theta_\Gamma)}_{\text{Dissimilarity loss } \boldsymbol{J_r}} \quad (2)$$

γ and λ are regularization weights, $\mathcal{C}_{sup}(\cdot,\cdot)$ the classification loss function.

Proposed Penalty

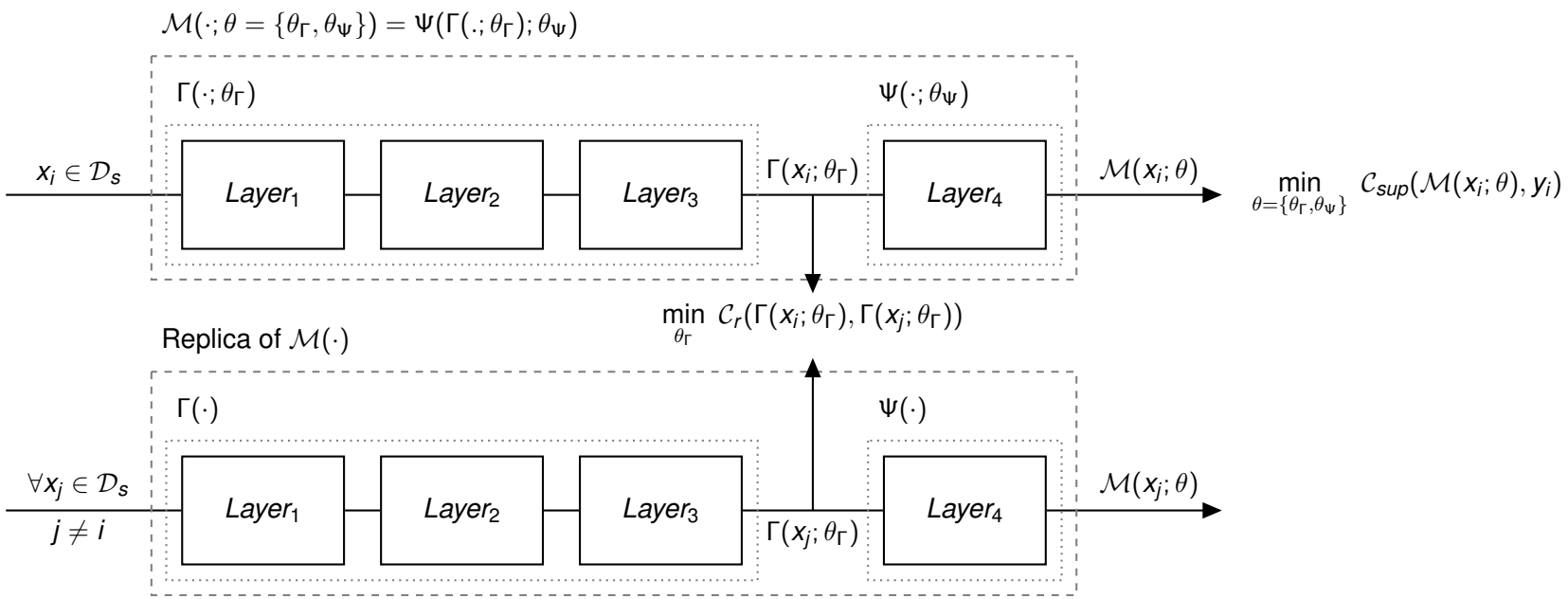


Figure 13 : Constraining the intermediate learned representations to be similar over a decomposed network $\mathcal{M}(\cdot)$ during the *training phase*.

Dissimilarity Measures

- The squared Euclidean distance (*SED*):

$$\mathcal{C}_h(a,b) = \|a-b\|_2^2 = \sum_{v=1}^{V}(a_v - b_v)^2 \quad , \tag{3}$$

- The normalized Manhattan distance (*NMD*):

$$\mathcal{C}_h(a,b) = \frac{1}{V}\sum_{v=1}^{V}|a_v - b_v| \quad , \tag{4}$$

- The angular similarity (*AS*):

$$\mathcal{C}_h(a,b) = \arccos\left(\frac{\langle a,b\rangle}{\|a\|_2\ \|b\|_2}\right) \quad . \tag{5}$$

Optimization

Algorithm 2 Our training strategy

$\mathcal{D}$ is the training set. B_s a mini-batch. B_r a mini-batch of all the possible pairs in B_s (Eq.2). OP_s an optimizer of the supervised term. OP_r an optimizer of the dissimilarity term. `max_epochs`: maximum epochs. γ, λ are regularization weights.

for i=1..`max_epoch` **do**
    Shuffle $\mathcal{D}$. Then, split it into mini-batches.
    for (B_s, B_r) in $\mathcal{D}$ **do**
        Make a gradient step toward $\boldsymbol{J_{sup}}$ using B_s and OP_s. (Eq.2)
        Make a gradient step toward $\boldsymbol{J_r}$ using B_h and OP_r. (Eq.2)
    end for
end for

Experiments

Benchmarks: 10 classes.

Figure 14 : Samples from training set of each benchmark. *Top row*: **mnist-std** benchmark. *Middle row*: **mnist-noise** benchmark. *Bottom row*: **mnist-img** benchmark.

Study the effect of the size of train set: *1k*, *3k*, *5k*, *50k* and *100k*.

Experiments

Models: two each one has 4 layers.

- **Multilayer perceptron** with 3 hidden layers followed by a classification output layer. Architecture(1): $1200 - 1200 - 200$. This model is referred to as ***mlp***.
- **LeNet convolutional network** (2) (Lenet-4) with 2 convolution layers with 20 and 50 filters of size 5×5, followed by a dense layer of size 500, followed by a classification output layer. This model is referred to as ***lenet***.

Reference to layers (from input to output): $h_1, h2, h3, h4$.

(1): Harm De Vries, R Memisevic, and A Courville. Deep learning vector quantization. In European Symposium on Artificial Neural Networks (ESANN) 2016, 2016.

(2): Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient- based learning applied to document recognition. In Proceedings of the IEEE, pages 2278–2324, 1998.

Experiments

At which layer to apply our regularization?

Model/train data size	*1k*		*3k*		*5k*		*50k*	
	vl	tst	vl	tst	vl	tst	vl	tst
	mlp							
	10.49 ± 0.031	11.24 ± 0.050	6.69 ± 0.039	7.17 ± 0.010	5.262 ± 0.030	5.63 ± 0.126	$\mathbf{1.574 \pm 0.016}$	$\mathbf{1.66 \pm 0.016}$
	mlp + reg.							
h_3	$\mathbf{8.80 \pm 0.093}$	$\mathbf{9.50 \pm 0.093}$	$\mathbf{5.81 \pm 0.104}$	$\mathbf{6.24 \pm 0.069}$	$\mathbf{4.74 \pm 0.065}$	$\mathbf{5.05 \pm 0.035}$	1.67 ± 0.043	1.73 ± 0.080
h_2	11.48 ± 0.081	12.32 ± 0.090	6.72 ± 0.031	7.29 ± 0.038	5.33 ± 0.031	5.84 ± 0.030	1.88 ± 0.043	1.97 ± 0.071
h_1	12.15 ± 0.043	12.74 ± 0.189	6.75 ± 0.041	7.26 ± 0.049	5.35 ± 0.028	5.87 ± 0.050	1.83 ± 0.033	1.95 ± 0.025

Table 4 : Mean $\pm$ standard deviation error over validation and test set of the benchmark *mnist-std* using the model *mlp* and the SED as dissimilarity measure over the different hidden layers: h_1, h_2, h_3. (**bold font indicates lowest error.**)

$\Rightarrow$ Last hidden layer.

Experiments

At which layer to apply our regularization?

Model/train data size	*1k*		*3k*		*5k*		*50k*	
	vl	tst	vl	tst	vl	tst	vl	tst
	mlp							
	10.49 ± 0.031	11.24 ± 0.050	6.69 ± 0.039	7.17 ± 0.010	5.262 ± 0.030	5.63 ± 0.126	$\mathbf{1.574 \pm 0.016}$	$\mathbf{1.66 \pm 0.016}$
	mlp + reg.							
h_3	$\mathbf{8.80 \pm 0.093}$	$\mathbf{9.50 \pm 0.093}$	$\mathbf{5.81 \pm 0.104}$	$\mathbf{6.24 \pm 0.069}$	$\mathbf{4.74 \pm 0.065}$	$\mathbf{5.05 \pm 0.035}$	1.67 ± 0.043	1.73 ± 0.080
h_2	11.48 ± 0.081	12.32 ± 0.090	6.72 ± 0.031	7.29 ± 0.038	5.33 ± 0.031	5.84 ± 0.030	1.88 ± 0.043	1.97 ± 0.071
h_1	12.15 ± 0.043	12.74 ± 0.189	6.75 ± 0.041	7.26 ± 0.049	5.35 ± 0.028	5.87 ± 0.050	1.83 ± 0.033	1.95 ± 0.025

Table 4 : Mean $\pm$ standard deviation error over validation and test set of the benchmark *mnist-std* using the model *mlp* and the SED as dissimilarity measure over the different hidden layers: h_1, h_2, h_3. (**bold font indicates lowest error.**)

$\Rightarrow$ Last hidden layer.

Experiments

What is the best dissimilarity measure?

Model/train data size	*1k*		*3k*		*5k*		*50K*	
	vl	tst	vl	tst	vl	tst	vl	tst
	MLP							
2-9 *mlp*	10.49 ± 0.031	11.24 ± 0.050	6.69 ± 0.039	7.17 ± 0.010	5.262 ± 0.030	5.63 ± 0.126	1.574 ± 0.016	1.66 ± 0.016
mlp + reg. (SED)	$\mathbf{8.80 \pm 0.093}$	$\mathbf{9.50 \pm 0.093}$	$\mathbf{5.81 \pm 0.104}$	$\mathbf{6.24 \pm 0.069}$	$\mathbf{4.74 \pm 0.065}$	$\mathbf{5.05 \pm 0.035}$	1.67 ± 0.043	1.73 ± 0.080
mlp + reg. (NMD)	10.32 ± 0.028	10.92 ± 0.094	6.69 ± 0.075	7.22 ± 0.059	5.34 ± 0.035	5.79 ± 0.045	1.44 ± 0.020	1.47 ± 0.020
mlp + reg. (AS)	10.27 ± 0.068	10.71 ± 0.123	6.52 ± 0.044	6.89 ± 0.013	4.96 ± 0.041	5.25 ± 0.051	$\mathbf{1.37 \pm 0.023}$	$\mathbf{1.37 \pm 0.025}$
	Lenet							
lenet	6.25 ± 0.016	7.27 ± 0.033	3.65 ± 0.085	4.02 ± 0.073	2.62 ± 0.031	2.90 ± 0.058	1.31 ± 0.028	1.23 ± 0.024
lenet + reg. (SED)	$\mathbf{4.54 \pm 0.150}$	5.05 ± 0.115	$\mathbf{2.70 \pm 0.124}$	$\mathbf{2.85 \pm 0.082}$	$\mathbf{2.06 \pm 0.113}$	$\mathbf{2.37 \pm 0.105}$	$\mathbf{0.97 \pm 0.087}$	$\mathbf{1.04 \pm 0.060}$
lenet + reg. (NMD)	6.70 ± 0.040	$\mathbf{4.60 \pm 0.065}$	3.85 ± 0.032	4.30 ± 0.036	2.87 ± 0.045	3.14 ± 0.035	1.99 ± 0.043	2.075 ± 0.079
lenet + reg. (AS)	6.72 ± 0.024	7.66 ± 0.024	3.86 ± 0.049	4.26 ± 0.049	2.80 ± 0.033	3.12 ± 0.021	1.75 ± 0.123	1.97 ± 0.063

Table 5 : Mean $\pm$ standard deviation error over validation and test set of the benchmark *mnist-std* using different dissimilarity measures (*SED*, *NMD*, *AS*) over the layer h_3. (**bold font indicates lowest error.**)

$\Rightarrow$ The squared Euclidean distance (SED).

Experiments

What is the best dissimilarity measure?

Model/train data size	*1k*		*3k*		*5k*		*50K*	
	vl	tst	vl	tst	vl	tst	vl	tst
	MLP							
2-9 *mlp*	10.49 ± 0.031	11.24 ± 0.050	6.69 ± 0.039	7.17 ± 0.010	5.262 ± 0.030	5.63 ± 0.126	1.574 ± 0.016	1.66 ± 0.016
mlp + reg. (SED)	$\mathbf{8.80 \pm 0.093}$	$\mathbf{9.50 \pm 0.093}$	$\mathbf{5.81 \pm 0.104}$	$\mathbf{6.24 \pm 0.069}$	$\mathbf{4.74 \pm 0.065}$	$\mathbf{5.05 \pm 0.035}$	1.67 ± 0.043	1.73 ± 0.080
mlp + reg. (NMD)	10.32 ± 0.028	10.92 ± 0.094	6.69 ± 0.075	7.22 ± 0.059	5.34 ± 0.035	5.79 ± 0.045	1.44 ± 0.020	1.47 ± 0.020
mlp + reg. (AS)	10.27 ± 0.068	10.71 ± 0.123	6.52 ± 0.044	6.89 ± 0.013	4.96 ± 0.041	5.25 ± 0.051	$\mathbf{1.37 \pm 0.023}$	$\mathbf{1.37 \pm 0.025}$
	Lenet							
lenet	6.25 ± 0.016	7.27 ± 0.033	3.65 ± 0.085	4.02 ± 0.073	2.62 ± 0.031	2.90 ± 0.058	1.31 ± 0.028	1.23 ± 0.024
lenet + reg. (SED)	$\mathbf{4.54 \pm 0.150}$	5.05 ± 0.115	$\mathbf{2.70 \pm 0.124}$	$\mathbf{2.85 \pm 0.082}$	$\mathbf{2.06 \pm 0.113}$	$\mathbf{2.37 \pm 0.105}$	$\mathbf{0.97 \pm 0.087}$	$\mathbf{1.04 \pm 0.060}$
lenet + reg. (NMD)	6.70 ± 0.040	$\mathbf{4.60 \pm 0.065}$	3.85 ± 0.032	4.30 ± 0.036	2.87 ± 0.045	3.14 ± 0.035	1.99 ± 0.043	2.075 ± 0.079
lenet + reg. (AS)	6.72 ± 0.024	7.66 ± 0.024	3.86 ± 0.049	4.26 ± 0.049	2.80 ± 0.033	3.12 ± 0.021	1.75 ± 0.123	1.97 ± 0.063

Table 5 : Mean $\pm$ standard deviation error over validation and test set of the benchmark *mnist-std* using different dissimilarity measures (*SED*, *NMD*, *AS*) over the layer h_3. (**bold font indicates lowest error.**)

$\Rightarrow$ The squared Euclidean distance (SED).

Experiments

Results on **mnist-img** using *lenet*:

Model/train data size	*1k*		*3k*		*5k*		*100k*	
	vl	tst	vl	tst	vl	tst	vl	tst
	mnist-noise							
lenet	9.62 ± 0.123	10.72 ± 0.116	5.95 ± 0.059	6.39 ± 0.032	4.92 ± 0.036	5.11 ± 0.012	1.90 ± 0.020	2.011 ± 0.018
lenet + reg.	$\mathbf{7.12 \pm 0.200}$	$\mathbf{7.74 \pm 0.148}$	$\mathbf{4.09 \pm 0.130}$	$\mathbf{4.62 \pm 0.059}$	$\mathbf{3.53 \pm 0.117}$	$\mathbf{3.98 \pm 0.167}$	$\mathbf{1.60 \pm 0.107}$	$\mathbf{1.64 \pm 0.116}$
	mnist-img							
lenet	13.88 ± 0.114	15.34 ± 0.124	8.34 ± 0.030	8.66 ± 0.024	6.64 ± 0.057	6.46 ± 0.033	2.53 ± 0.080	2.55 ± 0.007
lenet + reg.	$\mathbf{10.30 \pm 0.425}$	$\mathbf{11.18 \pm 0.290}$	$\mathbf{6.19 \pm 0.281}$	$\mathbf{6.61 \pm 0.212}$	$\mathbf{5.37 \pm 0.358}$	$\mathbf{5.65 \pm 0.310}$	$\mathbf{2.15 \pm 0.105}$	$\mathbf{2.21 \pm 0.032}$

Table 6 : Mean $\pm$ standard deviation error over validation and test set of the benchmarks *mnist-noise* and *mnist-img* using *lenet* model (regularization applied over the layer h_3). (**bold font indicates lowest error.**)

Experiments: Observation

Observation: On Learning Invariance within Neural Networks.

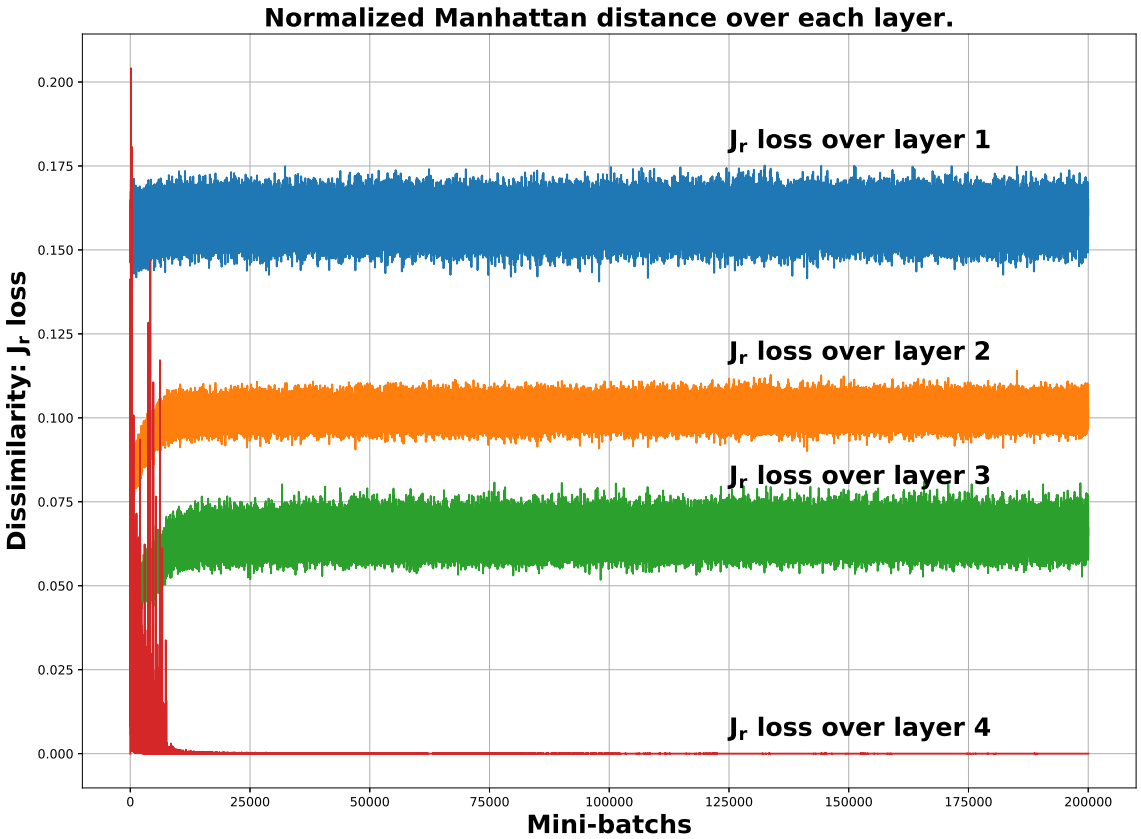


Figure 15 : Measuring the dissimilarity J_r of Eq.2 over the training set within each layer (simultaneously) of the *mlp* over the train set of *mnist-std* benchmark for a binary classification task: the digit “1” against the digit “7”.

Conclusion & Perspectives

- Our proposal helps improving the network generalization (small train set).
- Neural networks improve the representations invariance through depth. However, at each layer, it does not seem to improve through learning.
- Toward more explicit constraints (fix prior distributions).

My PhD work

Key words: *neural networks, regularization, representation learning.*

Selected work:

1. **A regularization framework for training neural networks for structured output problems.**
   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Multi-task Learning for Structured Output Prediction***. Under review, Neurocomputing. ArXiv: arxiv.org/abs/1504.07550. 2017.
2. **A regularization framework for training neural networks for classification.** S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Neural Networks Regularization Through Class-wise Invariant Representation Learning***. In preparation for IEEE TNNLS. ArXiv: arxiv.org/abs/1709.01867. 2017.
3. **Transfer learning in neural networks: an application to medical domain.** S. Belharbi, R.Hérault, C. Chatelain, R. Modzelewski, S. Adam, M. Chastan, S. Thureau, ***Spotting L3 slice in CT scans using deep convolutional network and transfer learning***. In Medical Image Analysis journal (MIA). 2017.

The problem: L3 slice localization

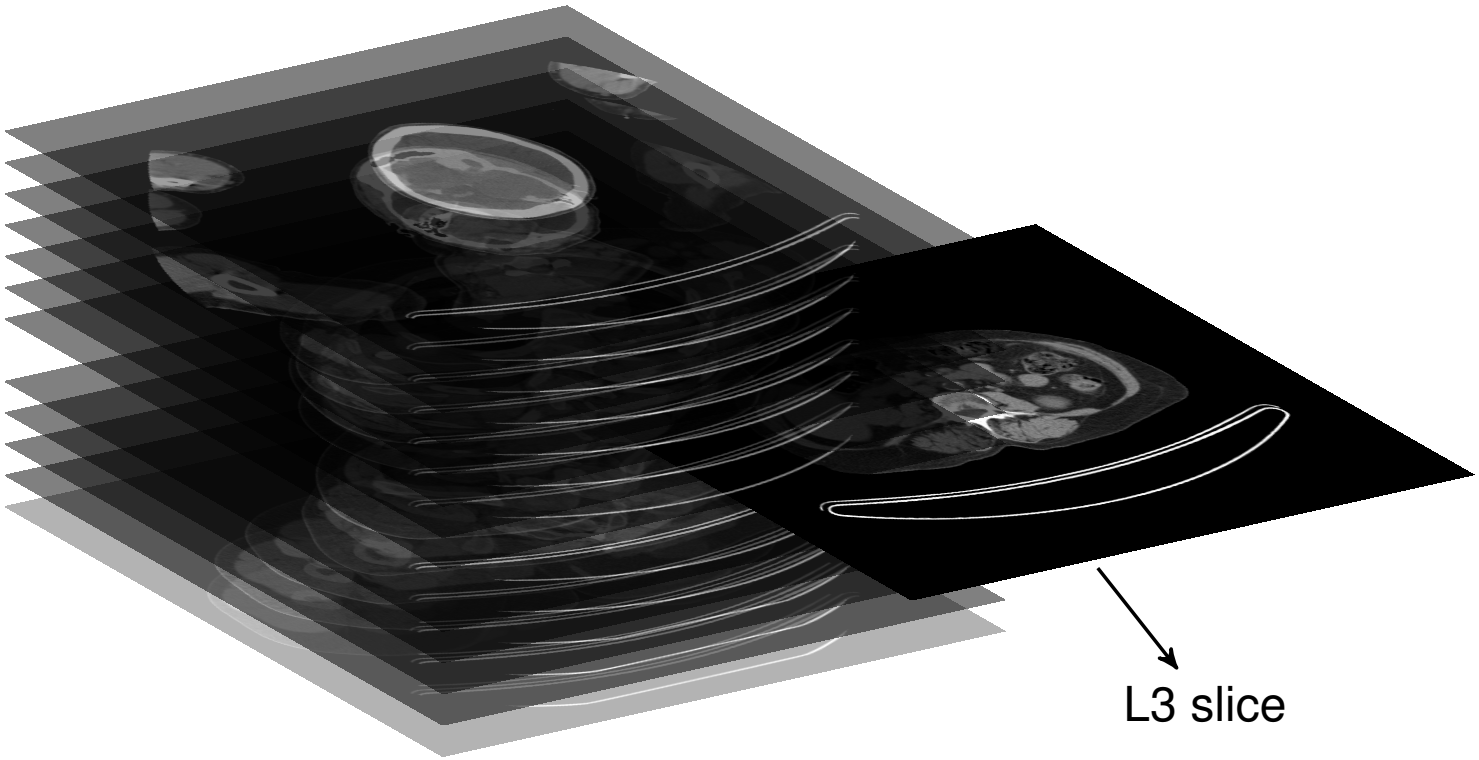


Figure 16 : Finding the L3 slice within a whole CT scan.

$\rightarrow$ Over a dataset of **642 CT scans**, we obtained an **average localization error** of **1.82 slice** ($<$ **5*mm***).

The problem: L3 slice localization

Informal statement

Given a CT scan of a part of a body, find the slice which corresponds to the L3 slice from thousands of slices.
The L3 slice contains the 3^{rd} lumbar vertebra.

Difficulties

- Inter-patients **variability**.
- Visual **similarity** of the L3 slice.
- The need to use the **context** to localize the L3 slice.

$\Longrightarrow$ **Machine Learning**

Possible approaches

Classification (discrete value)

Classify each slice for: “L3” or “Not L3”:

- Simple, ☺
- No context, ☹

Sequence labeling

Label all the slices (vertebrae): L1, L2, L3, . . . :

- Global analysis: context, ☺
- Existing work with promising results, ☺
- Requires labeling every slice, ☹

Regression (real value)

Predict the height (position) of the L3 slice inside the CT scan:

- Global analysis: context, ☺
- Requires labeling only the L3 slice position, ☺

Possible approaches: Difficulties

Figure 17 : Two slices from the same patient: a L3 (up) and a non L3 (L2) (down). The similar shapes of both vertebraes prevent from taking a robust decision given a single slice.

Regression for L3 detection

Which model?

- Deep learning, Convolutional neural network (CNN).
- No manual feature extraction.
- State of the art in vision.
- Requires fixed input size (when using dense layers).

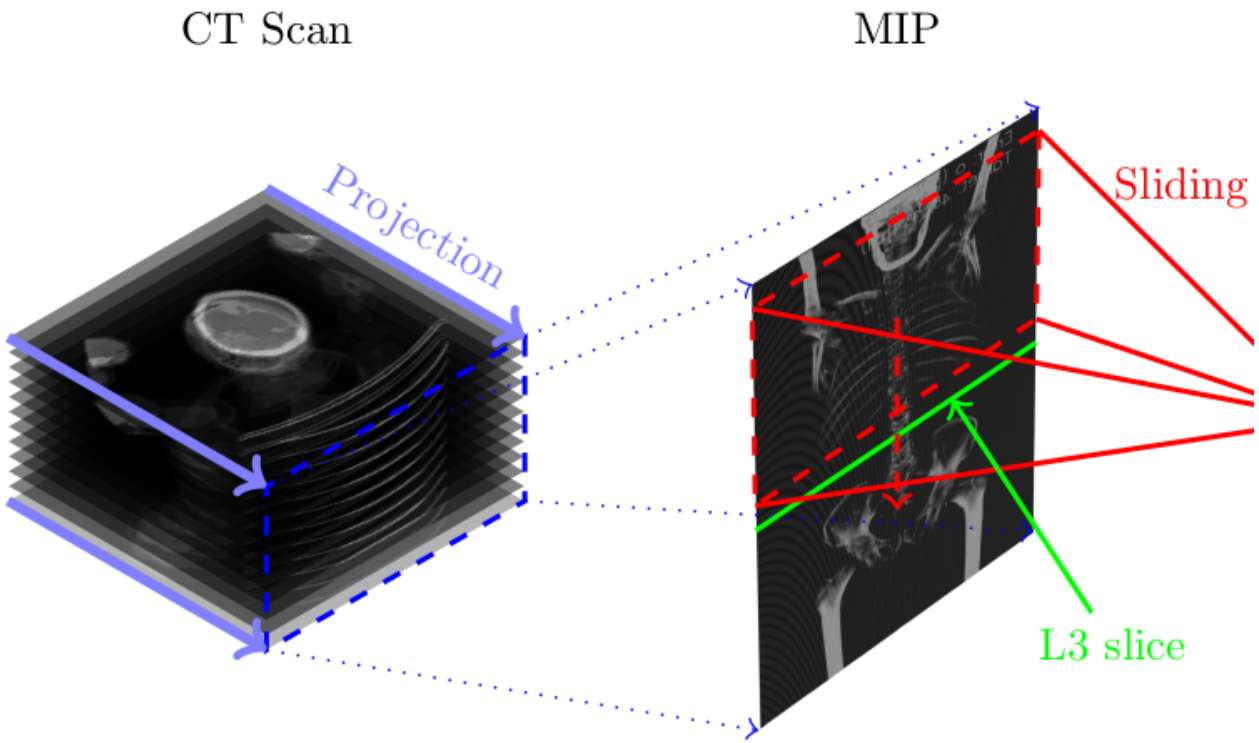
Some numbers . . .

- Input space: $\underbrace{1 \text{ scan} = N \times 512 \times 512}_{\text{Problem 1: large input space}}$, with $400 < N < 1200$.
- Dataset with annotated L3 position: $\underbrace{642 \text{ patients}}_{\text{Problem 2: few data}}$. (L3CT1 dataset)
- $\underbrace{\textit{Variability}}_{\text{Problem 3: Different input size}}$ of the height of each scan.

Regression for L3 detection

Problem 1: Input dimension space

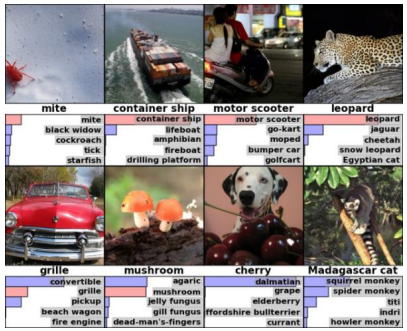
- $131M$ inputs for one example (large input dimension):
  $\Longrightarrow$ Frontal or lateral **Maximum Intensity Projection (MIP)**.
- $512 \times 512 \times N \Longrightarrow 512 \times N$.
- Conserves pertinent information (skeletal structure)



Regression for L3 detection

Problem 2: Few data (642 patients) [1]

- Train CNN from scratch $\rightarrow$ poor results.
  $\Longrightarrow$ Use pre-trained CNNs over **large datasets**
- Alexnet, GoogleNet, VGG16, VGG19, ... for **classification**
- Pre-trained models over ImageNet: 14 millions of natural images [Fei-Fei and Russakovsky 2013].



Regression for L3 detection

Problem 2: Few data (642 patients) [2]

$\Longrightarrow$ **Transfer learning**

Exploit **pre-trained filters** over natural images, Next, **refine** them over L3 detection task.

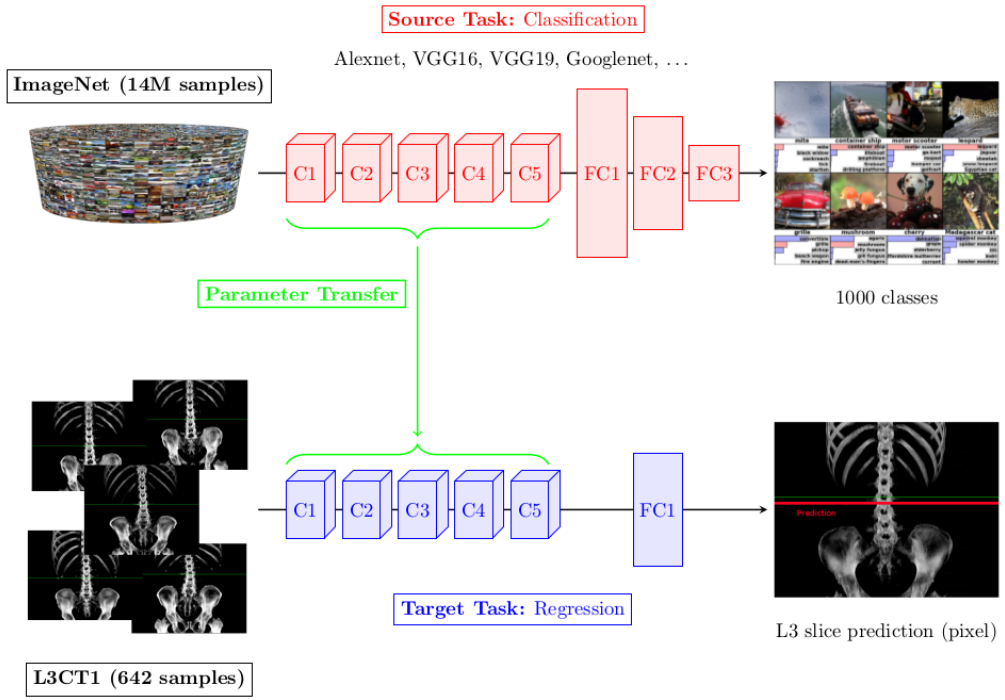


Figure 18 : System overview. Layers C_i are Convolutionnal layers, while FC_i denote Full Connected layers. Convolution parameters of previously learnt ImageNet classifier are used as initial values of corresponding L3 regressor layers to overcome the lack of CT examples

Regression for L3 detection

Problem 3: Different input size

- Classical problem
- Use sliding window technique
- Use post-processing

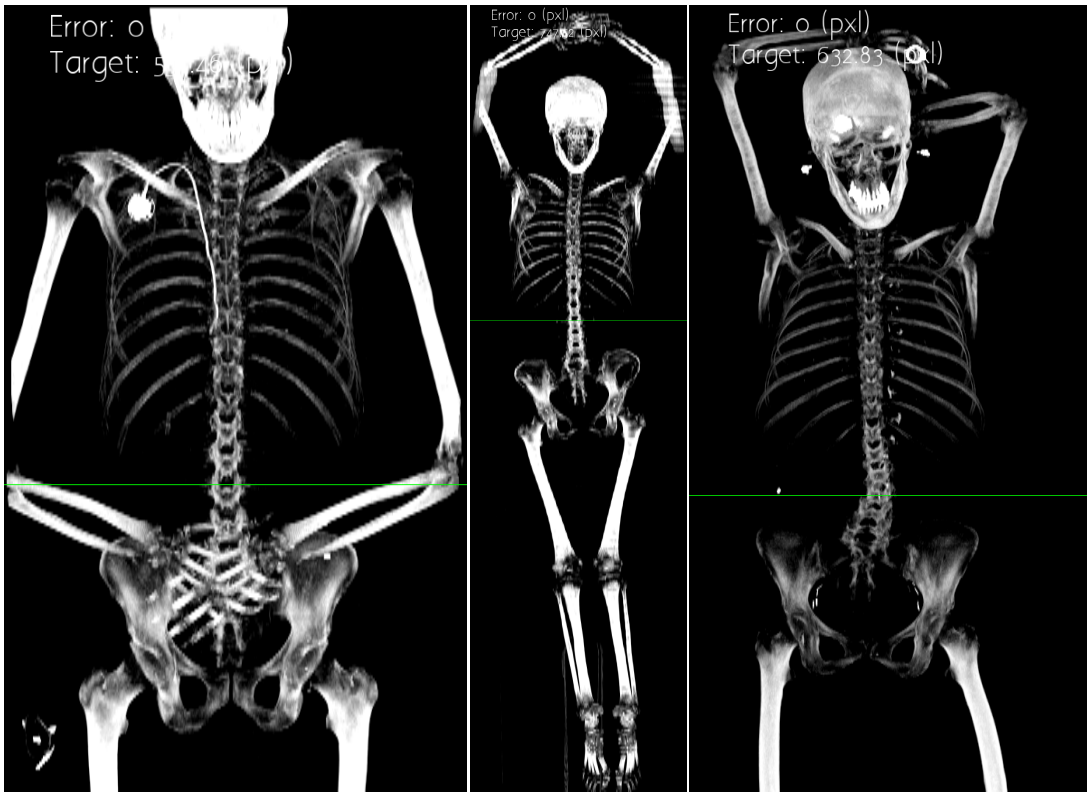


Figure 19 : Examples of normalized frontal MIP images with the L3 slice position.

Regression for L3 detection

Problem 3: Different input size

- Classical problem
- Use sliding window technique
- Use post-processing

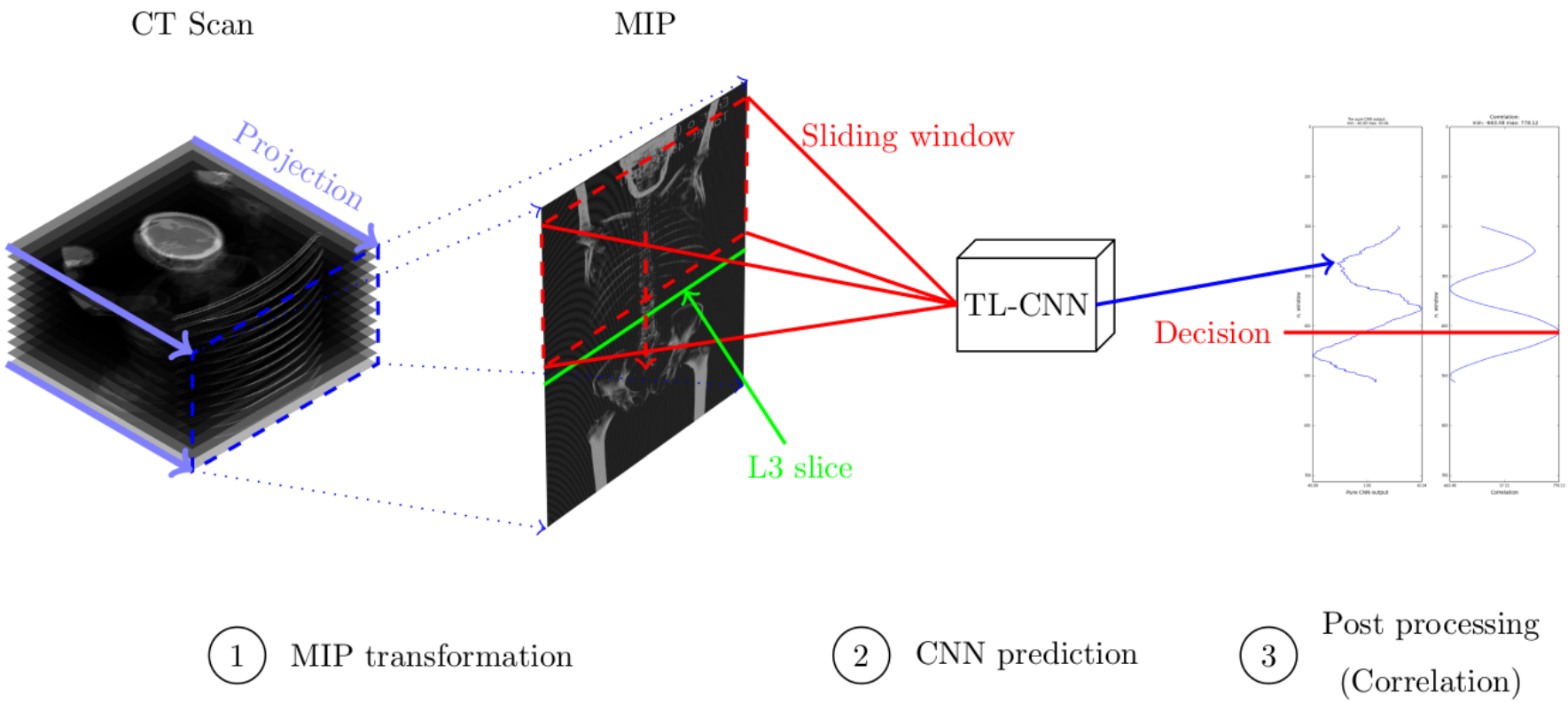


Figure 20 : System overview describing the three important stage of our approach : MIP transformation, TL-CNN prediction, and post processing.

Regression for L3 detection

Problem 3: Different input size

- Classical problem
- Use sliding window technique
- Use post-processing: correlation

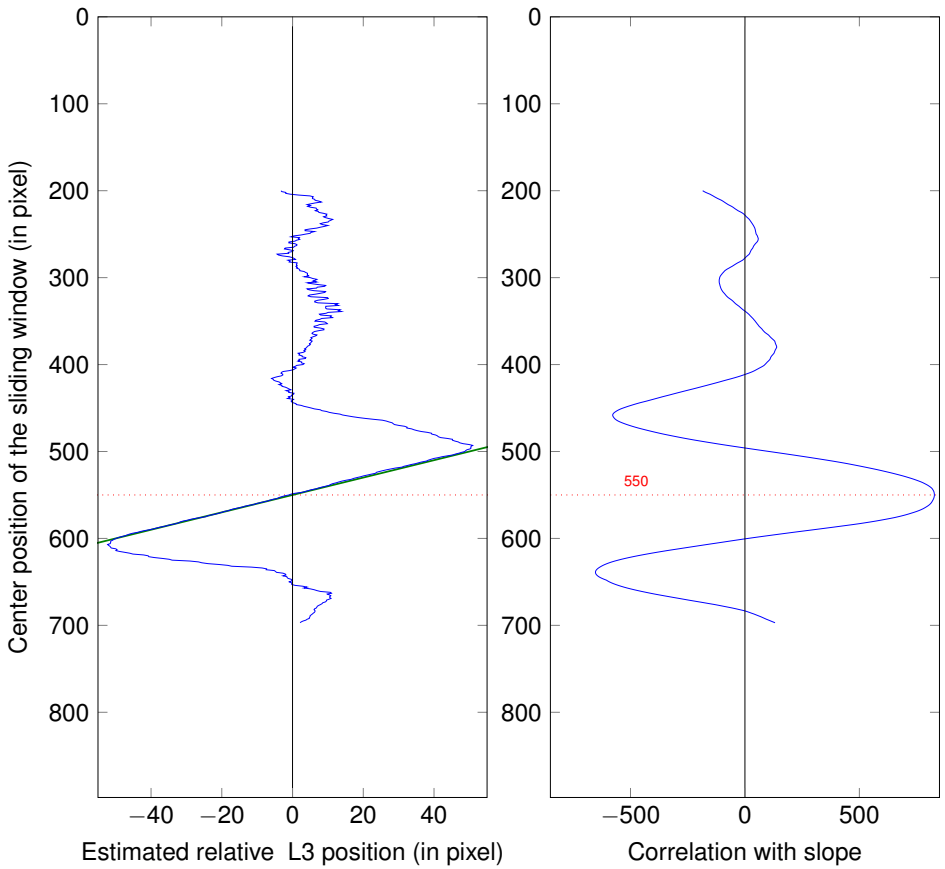


Figure 21 : [left]: CNN output sequence obtained using for $H = 400$ and $a = 50$ on a test CT scan. The sequence contains the typical straight line of slope -1 centered on the L3 (the theoretical line is plotted in green), surrounded by random values. [right]: correlation between the CNN output sequence and the theoretical. The maximum of correlation indicates the position of the L3.

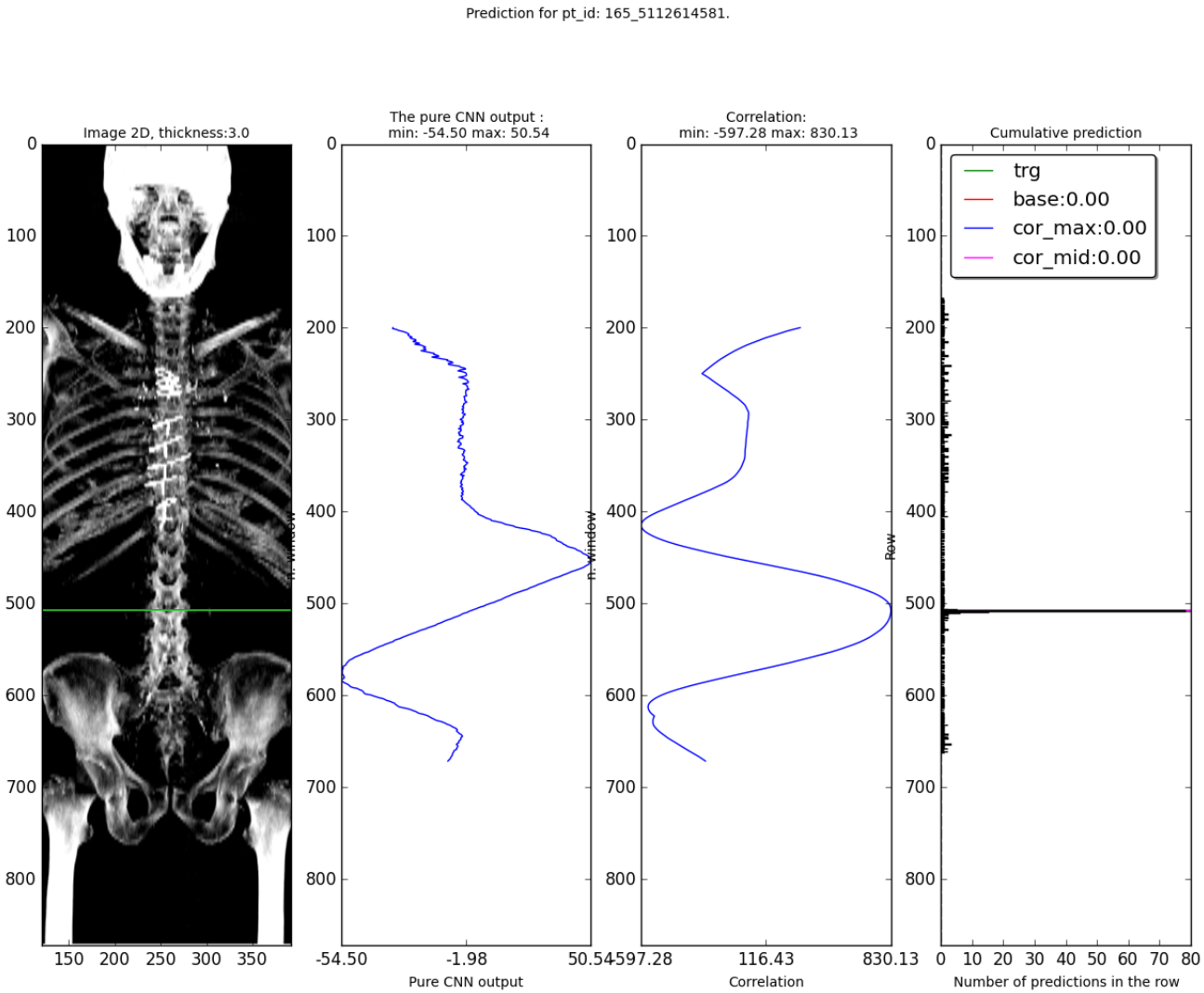
Regression for L3 detection: Quantitative results

Cross-validation:

	RF500	CNN4	Alexnet	VGG16	VGG19	Googlenet
fold 0	7.31 ± 6.52	2.85 ± 2.37	2.21 ± 2.11	2.06 ± 4.39	1.89 ± 1.77	1.81 ± 1.74
fold 1	11.07 ± 11.42	3.12 ± 2.90	2.44 ± 2.41	1.78 ± 2.09	1.96 ± 2.10	3.84 ± 12.86
fold 2	13.10 ± 13.90	3.12 ± 3.20	2.47 ± 2.38	1.54 ± 1.54	1.65 ± 1.73	2.62 ± 2.52
fold 3	12.03 ± 14.34	2.98 ± 2.38	2.42 ± 2.23	1.96 ± 1.62	1.76 ± 1.75	2.22 ± 1.79
fold 4	8.99 ± 7.83	1.87 ± 1.58	2.69 ± 2.41	1.74 ± 1.96	1.90 ± 1.83	2.20 ± 2.20
Average	10.50 ± 10.80	2.78 ± 2.48	2.45 ± 2.42	$\mathbf{1.82 \pm 2.32}$	1.83 ± 1.83	2.54 ± 4.22

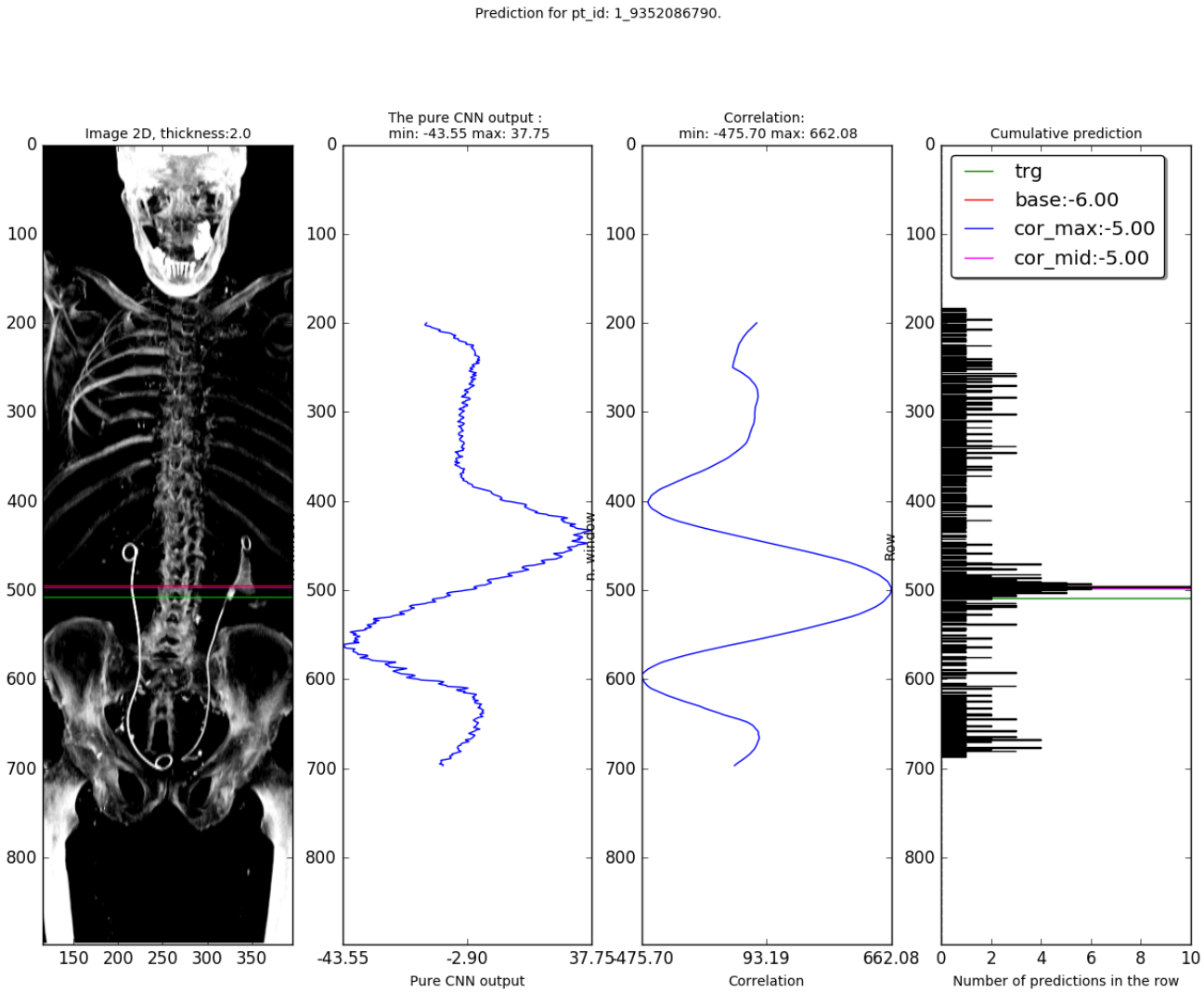
Table 7 : Error expressed in slice over all the folds using different models: RF500, CNN4 (Homemade model), and Alexnet/VGG16/VGG19/GoogleNet (Pre-trained models).

Regression for L3 detection: Qualitative results



Localization error: 0 coupes.

Regression for L3 detection: Qualitative results



Localization error: 6 coupes.

Regression for L3 detection: Evaluation time

	Number of parameters	Average processing time (seconds/CT scan)
CNN4	55 K	04.46
Alexnet	2 M	06.37
VGG16	14 M	13.28
VGG19	20 M	16.02
GoogleNet	6 M	17.75

Table 8 : Number of parameters vs. evaluation time over a GPU (K40).

Can be speedup more by increasing the window stride (without loosing in performance).

VGG16:

- **stride=1**: $\sim$ **13** ***seconds/CT scan*** with a an error of **1.82** $\pm$ **2.32**.
- **stride=4**: $\sim$ **02** ***seconds/CT scan*** with a an error of **1.91** $\pm$ **2.69**.

Regression for L3 detection: CNN vs. Radiologists

Setup

1. **New** evaluation set: **43 CT scans** annotated by the same **reference radiologist** (who annotated the L3CT1 dataset).
2. Ask **3 other radiologists** to localize the L3 slice.
3. Perform this experiment **twice**.

Errors (slices) / operator	CNN4	VGG16	Ragiologist #1	Radiologist #2	Radiologist #3
Review1	2.37 ± 2.30	1.70 ± 1.65	0.81 ± 0.97	0.72 ± 1.51	0.51 ± 0.62
Review2	2.53 ± 2.27	1.58 ± 1.83	0.77 ± 0.68	0.95 ± 1.61	0.86 ± 1.30

Table 9 : Comparison of the performance of both the automatic systems and radiologists. The L3 annotations given by the reference radiologist vary between the two reviews.

Regression for L3 detection: Conclusion

- Interesting results.
- Adapted pipeline: pre-processing, CNN, post-processing.
- Use of transfer learning alleviates the need of large training set.
- Generic framework: can be easily adapted for detecting other subjects given the required annotation.

My PhD work

Key words: *neural networks, regularization, representation learning.*

Selected work:

1. **A regularization framework for training neural networks for structured output problems.**

   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Multi-task Learning for Structured Output Prediction***. Under review, Neurocomputing. ArXiv: arxiv.org/abs/1504.07550. 2017.

2. **A regularization framework for training neural networks for classification.**

   S. Belharbi, C. Chatelain, R.Hérault, S. Adam, ***Neural Networks Regularization Through Class-wise Invariant Representation Learning***. In preparation for IEEE TNNLS. ArXiv: arxiv.org/abs/1709.01867. 2017.

3. **Transfer learning in neural networks: an application to medical domain.**

   S. Belharbi, R.Hérault, C. Chatelain, R. Modzelewski, S. Adam, M. Chastan, S. Thureau, ***Spotting L3 slice in CT scans using deep convolutional network and transfer learning***. In Medical Image Analysis journal (MIA). 2017.

Questions

Thank you for your attention,

Questions?

I am currently looking for a post-doc in deep learning.

Website: https://sbelharbi.github.io
Contact: soufiane.belharbi@insa-rouen.fr