Deep neural architectures for structured output problems

Soufiane Belharbi
soufiane.belharbi@litislab.fr

Clément Chatelain
clement.chatelain@insa-rouen.fr

LITIS - INSA de Rouen

Joint work with: J.Lerouge, R.Herault, S.Adam, R.Modzelewski, F.Jardin, B.Labbe

May 19, 2015

Plan

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 [3].

Traditional Machine Learning Problems

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

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

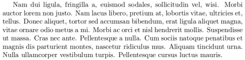
Machine Learning for Structured Output Problems

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

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

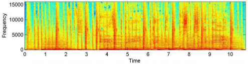
See C. Lampert slides [3].

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

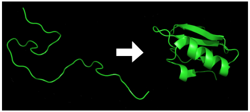


Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit 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.

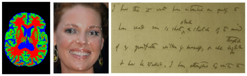
Text: part-of-speech tagging, translation



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

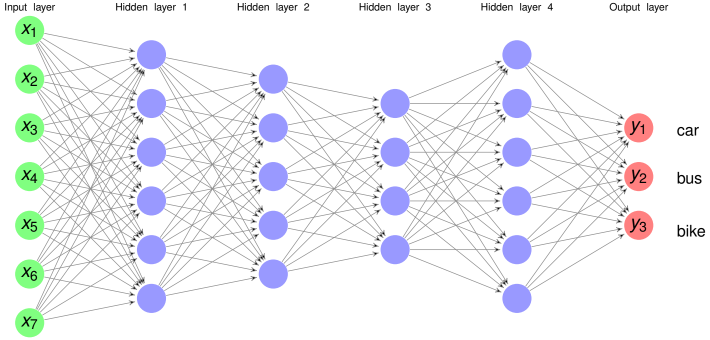
- Perform one single data transformation
- Difficult to deal with *high dimensional* data

Ideal approach

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

Deep neural networks?

Traditional Deep neural Network

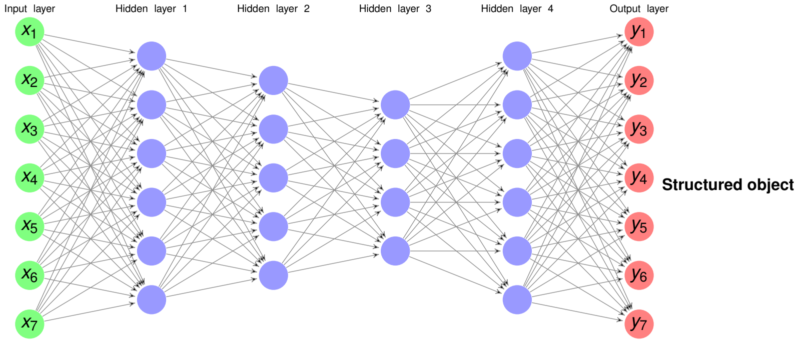


- High dimension data OK
- Multiple data transformation (complex mapping functions) OK
- Structured output problems NO

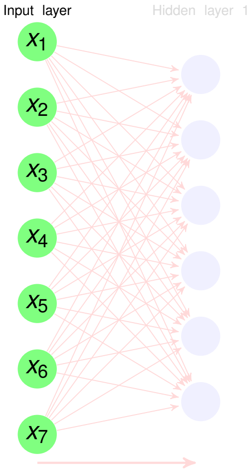
Plan

IODA:

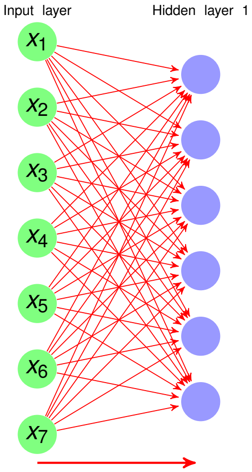
- Incorporate the output structure by learning
- Discover hidden dependencies in the outputs



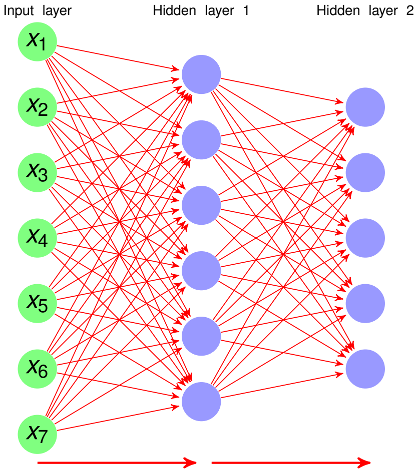
Training IODA



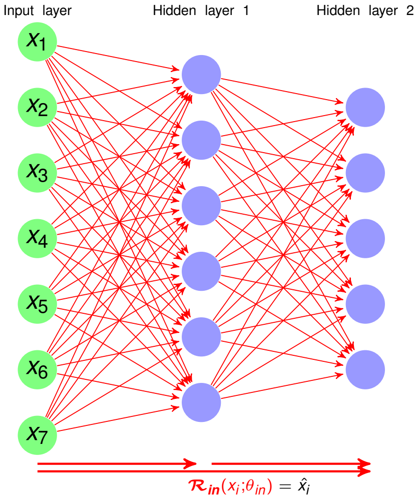
Training IODA



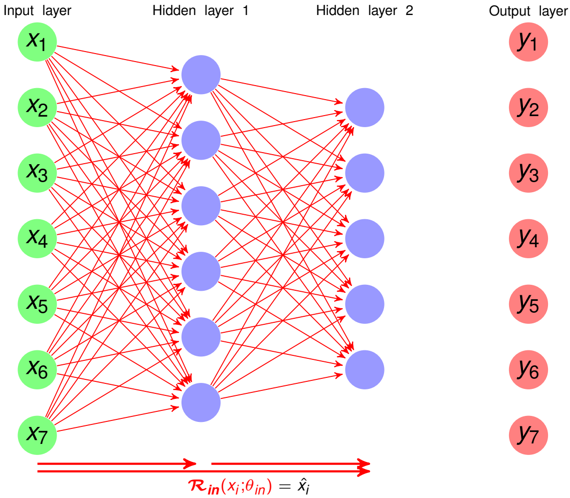
Training IODA



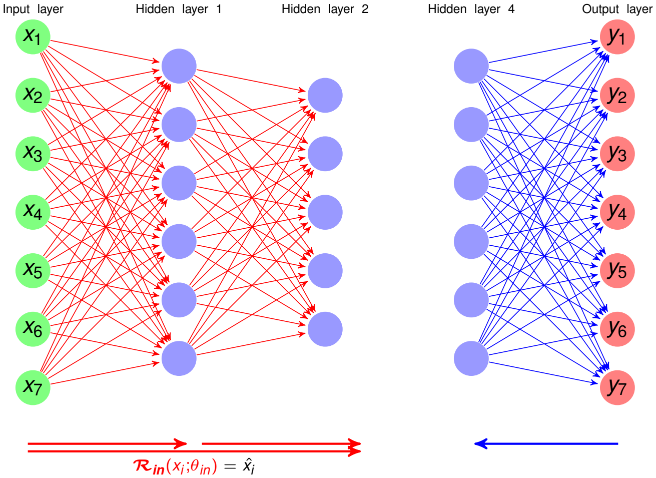
Training IODA



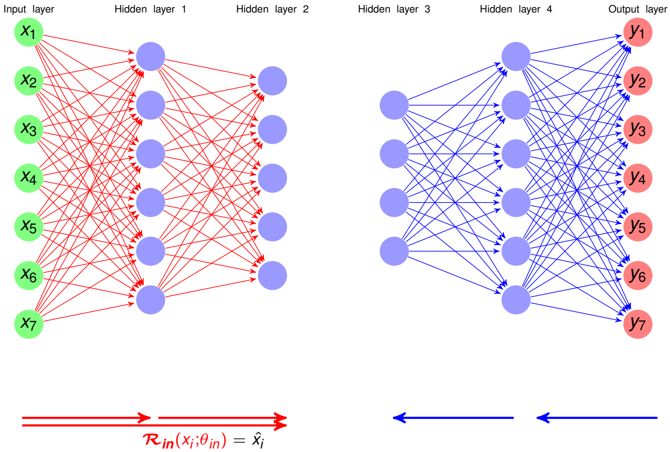
Training IODA



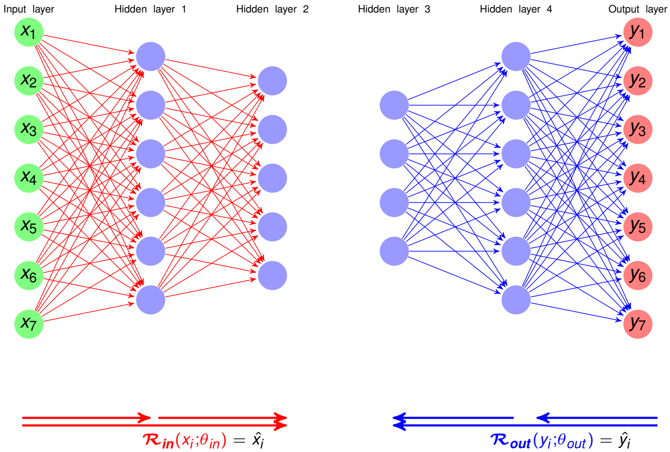
Training IODA



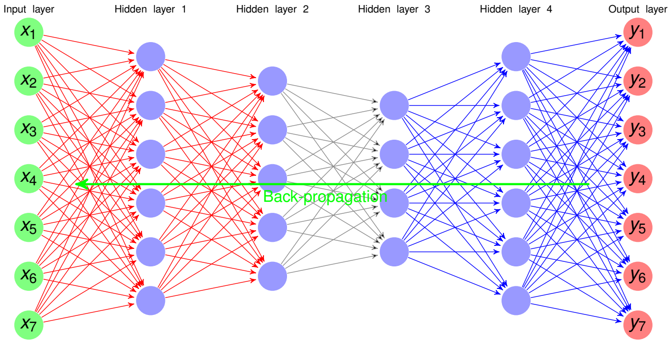
Training IODA



Training IODA



Training IODA



$\mathcal{R}_{in}(x_i;\theta_{in}) = \hat{x}_i$

$\mathcal{R}_{out}(y_i;\theta_{out}) = \hat{y}_i$

$\mathcal{M}(x_i;\theta,\theta_{in},\theta_{out}) = \hat{y}_i$

IODA framework: $\min\limits_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\theta}, \mathcal{D}(\mathbf{x}, \mathbf{y}))$

$$\mathfrak{L}(\boldsymbol{\theta}, \mathcal{D}(\mathbf{x}, \mathbf{y})) = \frac{1}{n}\sum_{i=1}^{n} \Bigg[\underbrace{\mathcal{C}(\mathcal{M}(x_i;\theta,\theta_{in},\theta_{out}), y_i)}_{\text{Learn } (\textit{input} \rightarrow \textit{output}) \text{ dependencies}} + \underbrace{\ell_{\mathbf{in}}(\mathcal{R}_{\boldsymbol{in}}(x_i;\theta_{in}), x_i)}_{\text{Learn input dependencies}} + \underbrace{\ell_{\mathbf{out}}(\mathcal{R}_{\boldsymbol{out}}(y_i;\theta_{out}), y_i)}_{\text{Learn output dependencies}} \Bigg]$$

$\mathcal{C}(.), \ell_{in}(.), \ell_{out}(.)$: defined costs.

$\min\limits_{\theta} \mathfrak{L}(\theta, \mathcal{D}(\mathbf{x}, \mathbf{y}))$ is hard to solve $\Rightarrow$ split $\mathfrak{L}(\theta, \mathcal{D}(\mathbf{x}, \mathbf{y}))$

IODA framework: $\min_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\theta}, \mathcal{D}(\mathbf{x}, \mathbf{y}))$

$$\mathfrak{L}(\boldsymbol{\theta}, \mathcal{D}(\mathbf{x}, \mathbf{y})) = \frac{1}{n}\sum_{i=1}^{n} \Bigg[\underbrace{\mathcal{C}(\mathcal{M}(x_i;\theta,\theta_{in},\theta_{out}), y_i)}_{\text{Learn } (\textit{input} \rightarrow \textit{output}) \text{ dependencies})} + \underbrace{\ell_{\mathbf{in}}(\mathcal{R}_{\boldsymbol{in}}(x_i;\theta_{in}), x_i)}_{\text{Learn input dependencies}} + \underbrace{\ell_{\mathbf{out}}(\mathcal{R}_{\boldsymbol{out}}(y_i;\theta_{out}), y_i)}_{\text{Learn output dependencies}} \Bigg]$$

$\mathcal{C}(.), \ell_{in}(.), \ell_{out}(.)$: defined costs.

$\min_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\theta}, \mathcal{D}(\mathbf{x}, \mathbf{y}))$ is hard to solve $\Rightarrow$ split $\mathfrak{L}(\boldsymbol{\theta}, \mathcal{D}(\mathbf{x}, \mathbf{y}))$

Relaxed-simplified version of IODA

1. Unsupervised training:
   - → *Input* dependencies : $\min_{\theta_{in}} \frac{1}{n} \sum_{i=1}^{n} \ell_{\mathbf{in}}(\mathcal{R}_{\mathbf{in}}(x_i;\theta_{in}), x_i)$
   - → *output* dependencies: $\min_{\theta_{out}} \frac{1}{n} \sum_{i=1}^{n} \ell_{\mathbf{out}}(\mathcal{R}_{\mathbf{out}}(y_i;\theta_{out}), y_i)$
2. Standard supervised learning:
   $\min_{\theta,\theta_{in},\theta_{out}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{C}(\mathcal{M}(x_i;\theta,\theta_{in},\theta_{out}), y_i)$

Open source implementation

***Implemented** using our library: **Crino** [1] [**Python-Theano** based].*

Relaxed-simplified version of IODA

1. Unsupervised training:
   - $\rightarrow$ *Input* dependencies : $\min\limits_{\theta_{in}} \frac{1}{n} \sum_{i=1}^{n} \ell_{\mathbf{in}}(\mathcal{R}_{\mathbf{in}}(x_i;\theta_{in}), x_i)$
   - $\rightarrow$ *output* dependencies: $\min\limits_{\theta_{out}} \frac{1}{n} \sum_{i=1}^{n} \ell_{\mathbf{out}}(\mathcal{R}_{\mathbf{out}}(y_i;\theta_{out}), y_i)$
2. Standard supervised learning:
   $\min\limits_{\theta,\theta_{in},\theta_{out}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{C}(\mathcal{M}(x_i;\theta,\theta_{in},\theta_{out}), y_i)$

Open source implementation

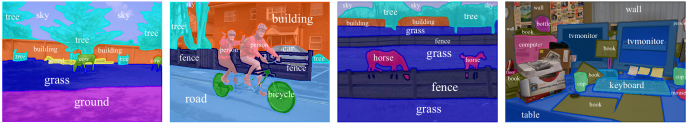
***Implemented** using our library: **Crino** [1] [**Python-Theano** based].*

Plan

Image labeling problems

Definition

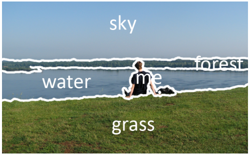
Assigning a label to each pixel of an image
(AKA "semantic segmentation")

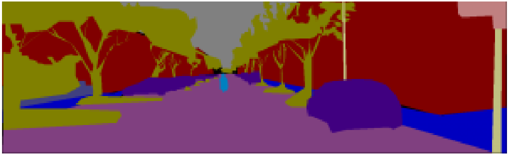


Various applications in:

- Document image analysis (text, image, tables, etc.)
- Computer vision (road safety, natural scene understanding)
- Medical imaging (organ, tumour segmentation)

Image labeling problems





Output dependencies

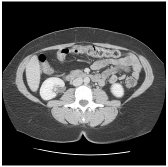
- Local dependencies (neigbouring labels are correlated)
- Structural dependencies (sky is generally above grass)

$\rightarrow$ Image labeling can be considered as a structured output problems

Application of IODA on a medical Image labeling problem

Collaboration with the Henri Becquerel Center (Quantif team)

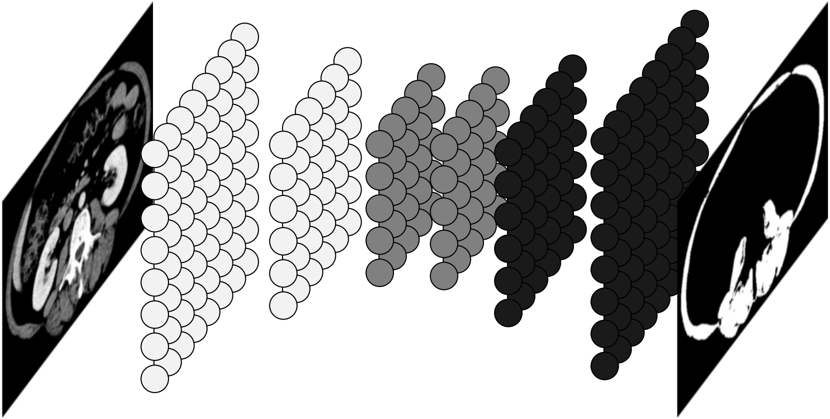
- Sarcopenia is a critical indication for lymphoma treatment
- Can be measured on scanner images by labeling squeletal muscle at L3 (third vertebra)
- 4 min/patient for a senior radiologist



Dataset

- 128 labeled L3 scanner images 512*512 pix
- Reference method from Chung (based on registration)

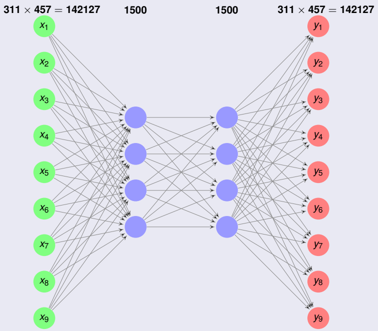
Input/Output Deep Architecture (IODA) for Image Labeling



IODA architecture for squeletal muscle segmentation

Implementation

Architecture (optimized on validation set)



A few figures:

- 428 M parameters (!!)
- Less than an hour for training (GPU, 4Go)
- 201.2 ms for decision

Qualitative results 1/2

(a) CT image

(b) Ground truth

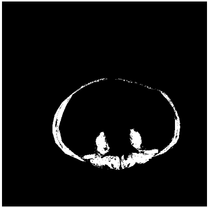
(c) Chung

(d) IODA

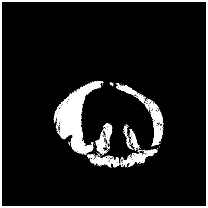
Non-sarcopenic patient

Qualitative results 2/2

(a) CT image



(b) Ground truth



(c) Chung

(d) IODA

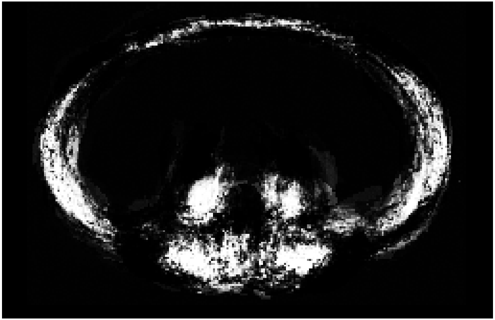
Sarcopenic patient

Quantitative results

Method	Diff. (%)	Jaccard (%)
Chung (reference method)	-10.6	60.3
No pre-train DA	0.12	85.88
Input pre-train DA	0.15	85.91
Input/Output pre-train DA (IODA)	3.37	**88.47**

The "blank test image"

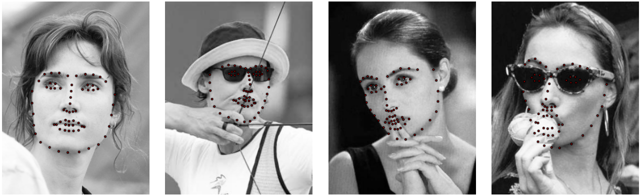
Feed the network with a blank image



Published in *pattern recognition* [4]

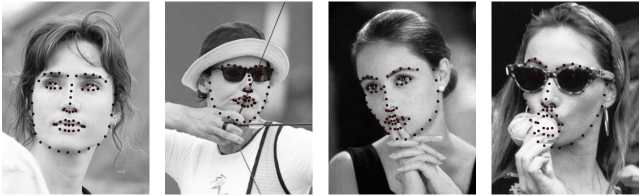
Plan

- ***Facial landmarks***:
  set of **facial key points** with **coordinates** (x,y)
  $\overset{\text{Task}}{\Longrightarrow}$ predict the ***shape****(set of points)* given a facial image



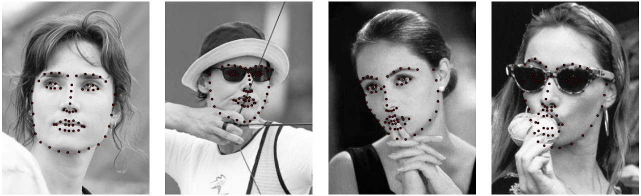
$\Rightarrow$ **Geometric dependencies** $\Rightarrow$ **structured output problem**
$\Rightarrow$ Apply **IODA** (**regression** task)

- ***Facial landmarks***:
  set of **facial key points** with **coordinates** (x,y)
  $\xRightarrow{\text{Task}}$ predict the ***shape****(set of points)* given a facial image



$\Rightarrow$ **Geometric dependencies** $\Rightarrow$ **structured output problem**
$\Rightarrow$ Apply **IODA** (**regression** task)

- ***Facial landmarks***:
  set of **facial key points** with **coordinates** (x,y)
  $\xrightarrow{\textit{Task}}$ predict the ***shape****(set of points)* given a facial image



$\Rightarrow$ **Geometric dependencies** $\Rightarrow$ **structured output problem**
$\Rightarrow$ Apply **IODA** (**regression** task)

Datasets & Performance Measures

- Datasets: LFPW(∼1000 samples), HELEN(∼2300 samples)
- Performance Measure:
  - Normalized Root Mean Square Error (**NRMSE**)
  - Cumulative Distribution Function: $\mathbf{CDF_{NRMSE}}$
  - Area Under the CDF Curve (**AUC**) **new**

Architecture (optimized on validation set)

⇒ Total **training** on **GPU** takes less than **30mins**.

Datasets & Performance Measures

- Datasets: LFPW(∼1000 samples), HELEN(∼2300 samples)
- Performance Measure:
  - Normalized Root Mean Square Error (**NRMSE**)
  - Cumulative Distribution Function: $\mathbf{CDF_{NRMSE}}$
  - Area Under the CDF Curve (**AUC**) **new**

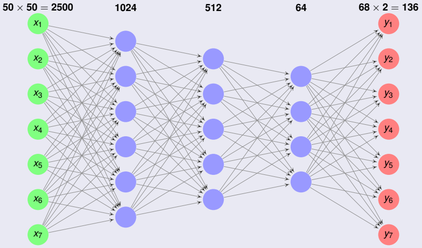
Architecture (optimized on validation set)

⇒ Total **training** on **GPU** takes less than **30mins**.

Datasets & Performance Measures

- Datasets: LFPW($\sim$1000 samples), HELEN($\sim$2300 samples)
- Performance Measure:
  - Normalized Root Mean Square Error (**NRMSE**)
  - Cumulative Distribution Function: $\mathbf{CDF_{NRMSE}}$
  - Area Under the CDF Curve (**AUC**) **new**

Architecture (optimized on validation set)



$\Rightarrow$ Total **training** on **GPU** takes less than **30mins**.

No pre-train DA

Input pre-train DA

Input/Output pre-train

DA (IODA)

Visual results LFPW

No pre-train DA

Input pre-train DA

Input/Output pre-train DA (IODA)

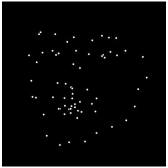
Visual results HELEN

	LFPW		HELEN	
	AUC	$CDF_{0.1}$	AUC	$CDF_{0.1}$
Mean shape	66.15%	18.30%	63.30%	16.97%
No pre-train DA 0-0-0	77.60%	50.89%	80.91%	69.69%
Input pre-train DA 1-0-0	79.25%	62.94%	82.13%	76.36%
2-0-0	79.10%	58.48%	82.39%	75.75%
3-0-0	79.51%	65.62%	82.25%	77.27%
Input/Output pre-train DA 1-0-1	80.66%	68.30%	83.95%	83.03%
1-1-1	81.50%	72.32%	83.51%	80.90%
1-0-2	81.00%	71.42%	83.91%	82.42%
1-1-2	81.06%	70.98%	83.81%	83.03%
1-0-3	**81.91%**	**74.55%**	83.72%	80.30%
2-0-1	81.32%	72.76%	83.61%	80.00%
2-1-1	81.47%	70.08%	**84.11%**	**83.33%**
2-0-2	81.35%	71.87%	83.88%	82.12%
3-0-1	81.62%	72.76%	83.38%	78.48%

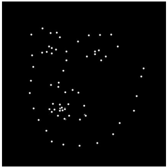
Performance of mean shape, NDA, IDA and IODA on LFPW and HELEN.

Blank Image Test

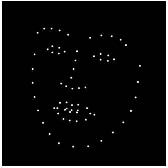
Feed a blank image to a trained network
$\Rightarrow$ what is the output?



No pre-train
DA 0-0-0



Input pre-train DA
3-0-0



Input/Output pre-train
DA 1-0-3

The outputs on LFPW

Paper submitted to ECML 2015 (arXiv [2]).

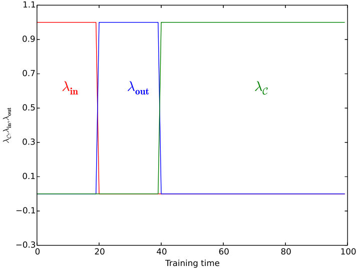
Plan

- Fully **neural** based approach
- Able to learn the **output dependencies** in **high dimension**
- Efficient on two real world problems

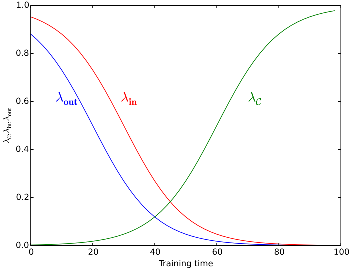
Plan

1. **Embedded Pre-training** (draft on arXiv):

$$\mathfrak{L}(\theta, \mathcal{D}(\mathbf{x}, \mathbf{y})) = \frac{1}{n} \sum_{i=1}^{n} \Big[\lambda_{\mathcal{C}}\ \mathcal{C}(\mathcal{M}(x_i;\theta,\theta_{in},\theta_{out}), y_i) + \lambda_{in}\ \ell_{\mathbf{in}}(\mathcal{R}_{in}(x_i;\theta_{in}), x_i) + \lambda_{out}\ \ell_{\mathbf{out}}(\mathcal{R}_{out}(y_i;\theta_{out}), y_i) \Big]$$



Relaxed IODA



Embedded IODA

2. **Use of unlabeled data:**

$$\mathfrak{L}(\boldsymbol{\theta}, \mathcal{D}(\mathbf{x}, \mathbf{y})) = \frac{1}{n}\sum_{i=1}^{n} \quad \lambda_{\mathcal{C}}\, \mathcal{C}(\mathcal{M}(x_i;\theta,\theta_{in},\theta_{out}), y_i)$$
$$+\frac{1}{n+n_{in}}\sum_{i=1}^{n+n_{in}} \quad \boldsymbol{\lambda_{in}}\, \ell_{\mathbf{in}}(\boldsymbol{\mathcal{R}_{in}}(x_i;\theta_{in}), x_i)$$
$$+\frac{1}{n+n_{out}}\sum_{i=1}^{n+n_{out}} \quad \boldsymbol{\lambda_{out}}\, \ell_{\mathbf{out}}(\boldsymbol{\mathcal{R}_{out}}(y_i;\theta_{out}), y_i)$$

n_{in}, n_{out} potentially **huge unlabeled** input, output data.

3. **Convolutional IODA**:
   Convolutional layers are efficient in feature extraction

   $\Rightarrow$ Use convolutional layers instead of auto-encoders in the input-layers

Bibliography I

[1] Crino, a neural-network library based on Theano. https://github.com/jlerouge/crino, 2014.

[2] S. Belharbi, C. Chatelain, R. Hérault, and S. Adam.
Input/Output Deep Architecture for Structured Output Problems.
ECML, 2015.

[3] CH. Lampert.
Slides: Learning with Structured Inputs and Ouputs, http://www.di.ens.fr/willow/events/cvml2010/materials/INRIA_summer_school_2010_Christoph.pdf, 2010.

[4] J. Lerouge, R. Herault, C. Chatelain, F. Jardin, and R. Modzelewski.
Ioda: An input output deep architecture for image labeling.
Pattern Recognition, 48(9):2847–2858, 2015.

Thank you for your attention.

Sets	Train samples	Test samples
LFPW	811	224
HELEN	2000	330

Number of samples in datasets.

Normalized Root Mean Square Error (NRMSE)

$NRMSE(s_p, s_g) = \frac{1}{n*D} \sum_{i=1}^{n} ||s_{pi} - s_{gi}||_2,$

s_p, s_g predicted, ground truth shape. D inter-ocular distance of s_g

Cumulative Distribution Function: CDF_{NRMSE}

$CDF_x = \frac{CARD(NRMSE \leq x)}{N}$

$CARD(.)$ cardinal of a set. N number of images.

e.g. $CDF_{0.1} = 0.4$ means that 40% of images have an NRMSE error less or equal than 0.1

Area Under the CDF Curve (AUC) **new**: more numerical precision

- Plot a CDF_{NRMSE} curve by varying NRMSE in [0, 0.5].
- Calculate the area under this curve.

Normalized Root Mean Square Error (NRMSE)

$NRMSE(s_p, s_g) = \frac{1}{n*D} \sum_{i=1}^{n} ||s_{pi} - s_{gi}||_2,$
s_p, s_g predicted, ground truth shape. D inter-ocular distance of s_g

Cumulative Distribution Function: CDF_{NRMSE}

$CDF_x = \frac{CARD(NRMSE \leq x)}{N}$
$CARD(.)$ cardinal of a set. N number of images.
e.g. $CDF_{0.1} = 0.4$ means that 40% of images have an NRMSE error less or equal than 0.1

Area Under the CDF Curve (AUC) **new**: more numerical precision

- Plot a CDF_{NRMSE} curve by varying NRMSE in [0, 0.5].
- Calculate the area under this curve.

Normalized Root Mean Square Error (NRMSE)

$NRMSE(s_p, s_g) = \frac{1}{n*D} \sum_{i=1}^{n} ||s_{pi} - s_{gi}||_2$,
s_p, s_g predicted, ground truth shape. D inter-ocular distance of s_g

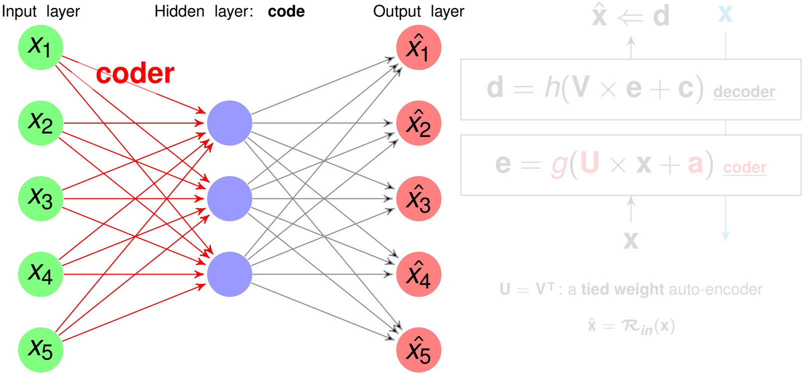
Cumulative Distribution Function: CDF_{NRMSE}

$CDF_x = \frac{CARD(NRMSE \leq x)}{N}$
$CARD(.)$ cardinal of a set. N number of images.
e.g. $CDF_{0.1} = 0.4$ means that 40% of images have an NRMSE error less or equal than 0.1

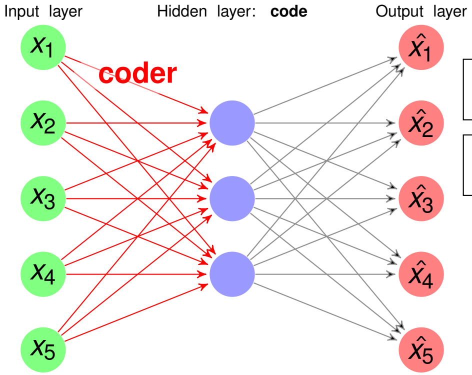
Area Under the CDF Curve (AUC) **new**: more numerical precision

- Plot a CDF_{NRMSE} curve by varying NRMSE in [0, 0.5].
- Calculate the area under this curve.

Input layer pre-training using auto-encoders (1)



Input layer pre-training using auto-encoders (1)



$$\hat{\mathbf{x}} \Leftarrow \mathbf{d} \qquad \mathbf{x}$$

$$\mathbf{d} = h(\mathbf{V} \times \mathbf{e} + \mathbf{c}) \quad \underline{\textbf{decoder}}$$

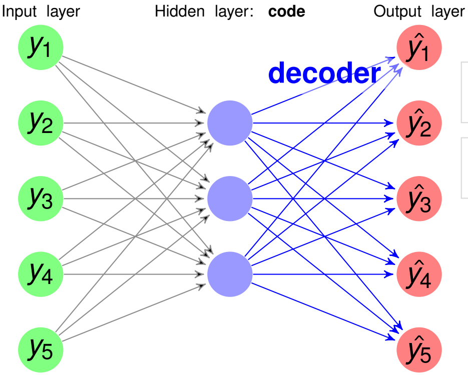
$$\mathbf{e} = g(\mathbf{U} \times \mathbf{x} + \mathbf{a}) \quad \underline{\textbf{coder}}$$

$$\mathbf{x}$$

$\mathbf{U} = \mathbf{V}^\top$: a **tied weight** auto-encoder

$\hat{\mathbf{x}} = \mathcal{R}_{in}(\mathbf{x})$

Output layer pre-training using auto-encoders (2)



$\hat{\mathbf{y}} \Leftarrow \mathbf{d}$ $\quad \mathbf{y}$

$\mathbf{d} = h(\mathbf{V} \times \mathbf{e} + \mathbf{c})$ decoder

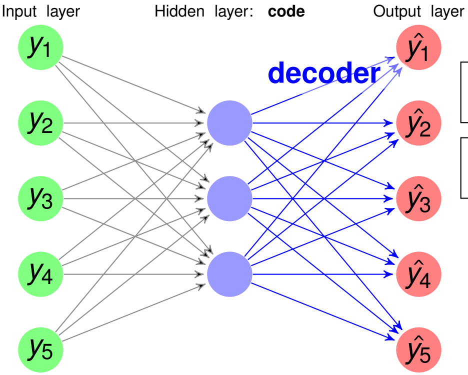
$\mathbf{e} = g(\mathbf{U} \times \mathbf{y} + \mathbf{a})$ coder

$\mathbf{y}$

$\mathbf{U} = \mathbf{V}^T$: a **tied weight** auto-encoder

$\hat{\mathbf{y}} = \mathcal{R}_{out}(\mathbf{y})$

Output layer pre-training using auto-encoders (2)



$\hat{\mathbf{y}} \Leftarrow \mathbf{d}$ $\qquad$ $\mathbf{y}$

$$\boxed{\mathbf{d} = h(\mathbf{V} \times \mathbf{e} + \mathbf{c})_{\text{decoder}}}$$

$$\boxed{\mathbf{e} = g(\mathbf{U} \times \mathbf{y} + \mathbf{a})_{\text{coder}}}$$

$\mathbf{y}$

$\mathbf{U} = \mathbf{V}^{\mathsf{T}}$: a **tied weight** auto-encoder

$$\hat{\mathbf{y}} = \mathcal{R}_{out}(\mathbf{y})$$