A Unified Neural Based Model for Structured Output Problems

Soufiane Belharbi*[1], Clément Chatelain*[1], Romain Hérault*[1], and Sébastien Adam*[2]

[1]LITIS EA 4108, INSA de Rouen, Saint Étienne du Rouvray 76800, France
[2]LITIS EA 4108, UFR des Sciences, Université de Rouen, France.

April 13, 2015

Abstract

Structured output problems are characterized by structural dependencies between the outputs (e.g. the classes distribution in image labeling problem, the words positions in sequence tagging in natural language processing). Traditionally, graphical models such as HMM and CRF are used to capture the interdependencies of the outputs. In this article, we propose a unified framework to deal with this problem where we combine learning the hidden interdependencies of the inputs and the outputs in the same optimization. In our framework, we extend the input pre-training layer technique for deep neural networks to pre-train the output layers aiming at learning the outputs structure. We propose a neural based model, called Input/Output Deep Architecture (IODA) to solve the optimization. Facial landmark detection is a real-world application where the output key points of the face shape have an obvious geometric structure dependencies. We perform an evaluation of IODA on this task over two challenging datasets: LFPW and HELEN. We demonstrate that IODA outperforms a deep network with the traditional pre-training technique.

Keywords: structured output data, deep learning, stacked auto-encoders, facial landmark detection.

1 Introduction

In machine learning field, the main task consists in building systems able to generalize from previously seen data. Most of machine learning applications aim at predicting a single value: a label for classification or a scalar value for regression. For these applications, existing algorithms learn the mapping function from the input to the output space by focusing on the *dependencies* in the *input* data.

Many recent applications address challenging problems where the output is in *high dimension* (discrete or continuous values) and where dependencies lie between these outputs. These dependencies constitute a structure (sequences, strings, trees, graphs ...) which should be either discovered if unknown, or integrated in the learning algorithm.

The range of applications that deal with structured output data is large. Statistical Natural Language Processing (NLP) is an application where the output has a specified structure, such as in (i) machine translation[Och03] where the output is a sentence, (ii) sentence parsing[ST95] where the output is a parse tree, or (iii) part of speech tagging [Sch94] where the output is a sequence of tags. Bioinformatics also manipulates structured output data, such as in secondary structure prediction of proteins[Jon99] where the output is a sequence modeled by a bipartite graph, or in enzyme function prediction[SY09] where the output is a path in a tree. In speech processing we find speech recognition (speech-to-text) [Rab89] where the prediction is a sentence, and text-to-speech synthesis [ZTB09] where the output is an audio signal.

Many approaches have been proposed in the literature to solve structured output problems. We can divide these approaches into three categories: kernel based methods, discriminative methods and graphical methods. The two first categories aim at discovering the hidden output structure, whereas the latter integrates an priori knowledge about the structure.

Kernel based approaches have been used in structured data output problems where kernel functions are used to learn the dependencies in inputs $\mathbf{x}$ and outputs $\mathbf{y}$. By doing this, the unconditional distributions of the inputs $p(\mathbf{x})$ and the outputs $p(\mathbf{y})$ are estimated. Then, the conditional probability of the target $p(\mathbf{y}|\mathbf{x})$ is learned in the new spaces induced by the kernel functions. [WCV+02] uses this scheme under the name

*surname.lastname@litislab.eu

Kernel Dependency Estimation (KDE) where two kernel functions are used for $\mathbf{x}$ and $\mathbf{y}$ separately. Then, a multivariate regression is applied from the new input space into the new output space. Additional constraint is added which is the reconstruction output to get back to the original space of $\mathbf{y}$.

Discriminative approaches are based on *margin maximization* property using an appropriate loss function over the structured output data to solve input-output task problems as a regression. This is an extension of the large-margin classification scheme to the regression case using the joint kernel functions. Support Vector Machine (SVM) and its variants for structured output, such as Structure output SVM (SSVM) is the main technique used in this approach. [BL08] uses SSVM for object localization as a regression task.

For both kernel based and discriminative approaches, the conditional probability $p(\mathbf{y}|\mathbf{x})$ is learned in the new spaces. Thus, an inverse function is needed to return to the output original space. This problem generally called the pre-image problem is known to be difficult.

Graphical models have shown a large success in modeling structured output due to its capability to capture dependencies among relevant random variables. Hidden Markov models (HMM) is a model based on this approach where the interdependencies between the output are learned. In the HMM framework, the output random variables are supposed to be independent which is not the case in many real-world applications, where strong relations are present. Conditional Random Fields (CRF) have been proposed to overcome this issue, thanks to its capability to learn large dependencies of the observed output data. These two models are widely used to model structured output data represented as a 1-D sequence[Rab89, BSW99, LMP01]. Thus, many approaches have been proposed to deal with 2-D structured output data as an extension of HMM and CRF. [NPH06] proposes a Markov Random Field (MRF) for document image segmentation. [SQ04] provides an adaptation of CRF to 2-D signals with hand drawn diagrams interpretation. Another extension of CRF to 3-D signal is presented in [TWMM07] for 3-D medical image segmentation. Mainly, HMM and CRF models are used with discrete output data. Due to the inference computational cost, graphical models are limited to low dimensional structured output problems. Furthermore, few works address the regression problem using graphical models [NC12, Fri93].

In this paper we propose a new generic unified formulation for regression with structured output data in high dimension. Through this formulation, we *explicitly* incorporate the inputs dependencies, the outputs dependencies and the final regression task. This formulation is solved using the Input Output Deep Architecture (IODA) [LHC$^+$15]. IODA is DNN trained using both input and output pre-training, making it capable of learning the dependencies of the inputs and the outputs. It has been applied with success for image labeling where the outputs represent a 2D structured data in high dimension. In this paper, the proposed formulation is instantiated on a real-world multivariate regression problem, namely facial landmark detection.

The rest of the paper is organized as follows. Section 2 describes the proposed formulation and its solving using IODA. Section 3 presents our experiments on two facial landmark detection datasets.

Figure 1: Definition of 68 facial landmarks from LFPW [BJKK11] training set.

2 New Statement for Structured Output problems

When dealing with structured output data in high dimension in machine learning, we add an additional constraint on the model related to the structure of the outputs, in the opposite of the traditional case where the output is a single value. The previous cited works in this domain such as kernel based methods and graphical methods have shown that when dealing with structured output data in a learning task, it is important to learn the outputs structure [WCV$^+$02, BL08, EYGB02, Rab89, BSW99, LMP01, NPH06, SQ04, TWMM07, NC12, Fri93], beside learning the inputs structure. Learning the outputs structure allows discovering the hidden dependencies between the outputs dimensions. By doing this, we reduce the complexity of the output space which helps to alleviate the final learning task.

2.1 Proposed Formulation

Starting from these results, we propose a generic framework where we incorporate learning the inputs-inputs, the outputs-outputs dependencies and the fi-

nal task inputs-outputs in the same optimization function. Learning both the outputs and the inputs dependencies in *unsupervised* way is *explicitly* formulated. In our framework, we formulate the cost function of learning the parameters $\boldsymbol{\theta}$ of a model over a labeled dataset $\boldsymbol{\mathcal{D}} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, as in the equation Eq.1. We call $\mathcal{M}(x)$ the mapping function from the input space to the output space.

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

The cost function is split into three parts . $\boldsymbol{\ell_{\mathbf{in}}}(\boldsymbol{\mathcal{R}_{in}}(\mathbf{x};\theta_{in}), \mathbf{x})$, which is the cost function $\boldsymbol{\ell_{\mathbf{in}}}(.,.)$ applied on the *reconstruction* of the input data $\boldsymbol{\mathcal{R}_{in}}(\mathbf{x};\theta_{out})$ where we learn the inputs dependencies. The second part consists of $\boldsymbol{\ell_{\mathbf{out}}}(\boldsymbol{\mathcal{R}_{out}}(\mathbf{y};\theta_{out}), \mathbf{y})$ which is the cost function $\boldsymbol{\ell_{\mathbf{out}}}(.,.)$ applied on the *reconstruction* of the output data $\boldsymbol{\mathcal{R}_{out}}(\mathbf{y};\theta_{out})$ where we learn the outputs dependencies. The first part, $\boldsymbol{\mathcal{C}}(\mathcal{M}(\mathbf{x};\theta,\theta_{in},\theta_{out}), \mathbf{y})$ is the cost of the final supervised learning task, which is the regression in our case. Note that the task parameters θ_{in} and θ_{out} are shared with the previous reconstruction tasks. We learn the parameters $\boldsymbol{\theta}$ by minimizing the cost $\mathfrak{L}(\boldsymbol{\theta}, \boldsymbol{\mathcal{D}}(\mathbf{x}, \mathbf{y}))$.

IODA is a neural based model that has been proposed in [LHC+15] in the purpose of solving image labeling problem which is a structured output data problem. In IODA, a pre-training of the inputs is firstly performed using a stacked auto-encoders. This allows to build a hierarchical features representation of the input data where the interdependencies of the data are learned at each stage. Next, using the same trick, a pre-training is applied on the outputs in order to learn a hierarchical representation of the output data that allows to discover the hidden structural topology of the outputs. After pre-training the input and the output, a supervised finetune is performed.

In the next paragraphs, we present briefly the framework IODA and how we can use it to solve Eq.1.

2.2 IODA for Structured Output Data

IODA[LHC+15] was designed to address two problems. The first problem is related to training deep network architectures to deal with the vanishing gradient problem using the pre-training trick [HOT06, BLPL07, VLL+10, BCV13, RPCL07]. The pre-training technique allows to initialize the network layers in a better way. The second problem is related to the structured output data in high dimensional space. This is done by pre-training the output layers in a backward fashion starting from the original outputs. Pre-training the output layers allows IODA to learn the output structure and discover the hidden dependencies of the outputs. It also allows to incorporate the structure of the output data into the network. The final learning is applied on the whole network in a supervised fashion. We define in the next paragraphs the tools that we will use through out the paper.

An auto-encoder (AE) is a 2-layers MLP. The purpose of the AE is to reconstruct an input signal $\mathbf{x}$ in the output $\hat{\mathbf{x}}$ [BK87, BLPL07].

Stacked Auto-encoders for Building DNN and IODA:

A pre-trained DNN is build using a stacked input AEs. In the case of IODA, a stack of input AEs is linked to a stack of output layers. Optimizing Eq.1 using IODA is done step by step. Firstly, IODA minimizes the reconstruction of the input data, then the reconstruction of the output data and finally, the supervised cost function. The IODA training involves the following steps:

1. Unsupervised pre-training of the layers close to the input layer to learn the inputs dependencies
2. Unsupervised pre-training of the layers close the output layer to learn the outputs dependencies
3. Final supervised back-propagation for the whole model to learn the mapping function from the inputs-outputs

IODA allows the creation of a third part between the input and the output layers which consists of a simple MLP. We extend our framework Eq.1, by adding a new term describing the the cost of learning a mapping function from the learned input space to the learned output space. Our extended framework is formulated in Eq.2. We call $\mathcal{M}_{link}(x;\theta_{link})$ the mapping function from the input space to the output space of the link. We call $\mathbf{r}_{in}(\mathbf{x})$ the last representation in the input stack AEs, $\mathbf{r}_{out}(\mathbf{y})$ the last representation in the output stack AEs.

$$\mathfrak{L}(\boldsymbol{\theta}, \boldsymbol{\mathcal{D}}(\mathbf{x}, \mathbf{y})) = \frac{1}{n} \sum_{i=1}^{n} \Big[\boldsymbol{\mathcal{C}}(\mathcal{M}(x_i;\theta,\theta_{in},\theta_{link},\theta_{out}), y_i) + \boldsymbol{\ell_{\mathbf{in}}}(\boldsymbol{\mathcal{R}_{in0}}(x_i;\theta_{in}), x_i) + \boldsymbol{\ell_{\mathbf{out0}}}(\boldsymbol{\mathcal{R}_{out0}}(y_i;\theta_{out}), y_i) + \boldsymbol{\ell_{\mathbf{link}}}(\mathcal{M}_{link}(\mathbf{r_{in}}(x_i);\theta_{link}), \mathbf{r_{out}}(y_i)) \Big] \quad (2)$$

Let us consider a network with 3 layers with its mapping function Eq.3:

$$\mathcal{M}(\mathbf{x};\theta) = g(\mathbf{W_3} \times \mathbf{h}(\mathbf{W_2} \times \mathbf{h}(\mathbf{W1} \times \mathbf{x} + \mathbf{b_1}) + \mathbf{b_2}) + \mathbf{b_3}) \quad (3)$$

Where $\theta = \{\mathbf{W_1}, \mathbf{b_1}, \mathbf{W_2}, \mathbf{b_2}, \mathbf{W_3}, \mathbf{b_3}\}$, $\mathbf{g}(.)$ and $\mathbf{h}(.)$ are activation functions. We consider the first layer as the input layer, the second layer as the link layer and the third layer as the output layer.

Input pre-training
Let's replace the reconstruction function $\boldsymbol{\mathcal{R}_{in}}$ by an AE and optimize it:

$$\begin{aligned}\mathfrak{L}_{in} &= \min_{\theta_{in}} \sum_i \boldsymbol{\ell_{in}}(\boldsymbol{\mathcal{R}_{in}}(x_i;\theta_{in}), x_i) \\ &= \min_{\mathbf{W_{in}}, \mathbf{b}_{in}, \mathbf{b}'_{in}} \sum_i \boldsymbol{\ell_{in}}(\mathbf{g}(\mathbf{W_{in}}^\intercal \times \mathbf{h}(\mathbf{W_{in}} \times x_i \\ &\quad + \mathbf{b_{in}}) + \mathbf{b}'_{in}), x_i) \ .\end{aligned}$$

Parameters $\mathbf{W_{in}}$ and $\mathbf{b}_{in}$ are kept to initialize $\mathbf{W_1}$ and $\mathbf{b_1}$ respectively.

Output pre-training
Let's replace the reconstruction function $\boldsymbol{\mathcal{R}_{out}}$ by an AE and optimize it:

$$\begin{aligned}\mathfrak{L}_{out} &= \min_{\theta_{out}} \sum_i \boldsymbol{\ell_{out}}(\boldsymbol{\mathcal{R}_{out}}(y_i;\theta_{out}), y_i) \\ &= \min_{\mathbf{W_{out}}, \mathbf{b}_{out}, \mathbf{b}'_{out}} \sum_i \boldsymbol{\ell_{out}}(\mathbf{g}(\mathbf{W_{out}}^\intercal \times \mathbf{h}(\mathbf{W_{out}} \times y_i \\ &\quad + \mathbf{b_{out}}) + \mathbf{b}'_{out}), y_i) \ .\end{aligned}$$

Parameters $\mathbf{W_{out}}^\intercal$ and $\mathbf{b}'_{out}$ are kept to initialize $\mathbf{W_3}$ and $\mathbf{b_3}$ respectively.

Link pre-training[optional]
Learn $\{\mathbf{W}_{link}, \mathbf{b}_{link}\}$ by the following optimization problem:

$$\begin{aligned}\mathfrak{L}_{link} &= \min_{\mathbf{W}_{link}, \mathbf{b}_{link}} \sum_i \boldsymbol{\ell_{link}}(\mathbf{h}(\mathbf{W}_{out} \times y_i + \mathbf{b}_{out}) \\ &\quad , \mathbf{h}(\mathbf{W}_{link} \times \mathbf{h}(\mathbf{W}_{in} \times x_i + \mathbf{b}_{in}) + \mathbf{b}_{link})) \ .\end{aligned}$$

Note that $\{\mathbf{W}_{in}, \mathbf{b_{in}}, \mathbf{W}_{out}, \mathbf{b}_{out}\}$ are fixed. Parameters $\mathbf{W}_{link}$ and $\mathbf{b}_{link}$ are kept to initialize $\mathbf{W_2}$ and $\mathbf{b_2}$ respectively.

Supervised finetuning
A standard backprobagation is undertaken on the DNN. More details on training IODA can be found in [LHC+15].

3 Facial Landmark Detection

In this paper, our formulation and its instantiation with IODA is evaluated on a facial landmark detection problem. We first describe the application and briefly present the related works. The datasets and the experimental protocol are then detailed and our implementation is described before presenting the results. All the experiments are done using a custom version of the library Crino [LHC+15].

3.1 Facial Landmark Detection

Facial landmark detection is an example of structured output problem which aims at predicting a geometric shape induced by an input face image. It plays an important role in face recognition and analysis. Therefore, it has been studied extensively in the recent years. However, this task remains a challenging problem due to the complex variations in the face appearance caused by the high variation in the poses, expressions, illuminations and by partial occlusions.

Facial landmarks are a set of key points on human face images. These points are defined by their real coordinates (x,y) on the image as shown in Fig. 1. The number of landmarks is dataset or application dependent. As the positions of the points in the face shape are dependent (spatial dependencies), facial landmark detection task falls naturally into the structured output regression problem.

Successful approaches used in facial landmark detection integrate in the learning process the face shape constraints for the prediction which is the key success of these techniques. Point Distribution Model (PDM) is widely used as a statistical model to capture the geometric distribution of the landmarks [CTCG95, SLC11, ZBL13]. [ZSCC13] uses SSVM to constrain the face shape based on a tree shape. Pictorial image [FH05] has been used to model the facial landmarks relations as a graph. Many approaches use the pictorial image method as a face shape constraint [ZR12, YHZ+13]. 3D face model has been also used in a graph matching problem [YHZ+13, TYRW14]. In [SWT13], the face shape constraints are learned *implicitly* by the carefully designed convolutional networks. [DGFG12] uses Markov Random Field to model the relation between the key points.

Whereas most approaches consist in defining the face shape constraints explicitly, we propose to *learn* the structure of the face shape by discovering the hidden dependencies between the landmarks. For that, our formulation is implemented using IODA.

3.2 Datasets

We have carried out our evaluation over two public datasets: LFPW[BJKK11] and HELEN[LBL+12].

LFPW dataset: It consists of 1132 training images and 300 test images taken under unconstrained conditions (in the wild) with large variations in the pose, expression, illumination and with partial occlusions (Fig.1). This makes the facial point detection a challenging task on this dataset. From the initial dataset described in LFPW[BJKK11], we use only the 811 training images and the 224 test images provided by the ibug website [fac]. The ground truth annotation of 68 facial points is provided by [STZP13]. We divide the available training samples into two sets: validation set (135 samples) and training set (676 samples).

HELEN dataset: It is similar to LFPW dataset, where the images have been taken under unconstrained conditions with high resolution and collected from Fliker using text queries. It contains 2000 images for training, and 330 images for test. The images and the face bounding boxes are provided by [fac]. The ground truth annotations are provided by [STZP13]. Examples of dataset are shown in Fig.2.

Figure 2: Samples from HELEN [LBL$^+$12] dataset.

For both datasets, due to the variation of the input images size, faces are cropped into the same size (50×50) then normalized in [0,1]. The face shapes are normalized into [-1,1].

3.3 IODA Implementation

We use a deep neural network with three hidden layers the size of which has been set to 1024, 512, 64 through a validation procedure on the LFPW validation set. The input representation size is: $50 \times 50 = 2500$, and the output representation size is: $68 \times 2 = 136$.

We use the same architecture through all the experiments with different configurations. To distinguish between the multiple configurations we set this notation: **DNN in-link-out** where:

- **in**: The number of pre-trained input layers.
- **link**: Indicates if the link has been trained (1) or not (0). The link may contain more than one layer.
- **out**: The number of pre-trained output layers. The configuration is called IODA if **out** > 0.

We use the same initial parameters in all the configurations. All the AEs are trained using a standard back-propagation algorithm. The classical momentum[SMDH13], with a constant momentum coefficient of 0.5 is used in training the output AEs.

To control the overfitting in the auto-encoders, we use in the training of each AE the l_1 regularization over the weights of the coding and the decoding layers with weight decay values of: $10^{-5}, 10^{-4}, 10^{-5}$ for the input AEs, the link and the output AEs respectively.

Two termination criteria are used for all pre-training and training procedures: the validation error and a maximum number of epochs. The maximum number of epochs is 300 for the supervised training. Less epochs are used in the pre-training (less than 40), except the first input layer where it needs 200 epochs.

The hyper-parameters (learning rate, batch size, momentum coefficient, weight decay) have been optimized on the LFPW validation set ($\frac{1}{6}$ of available training set, selected randomly). We use the same optimized hyper-parameters for HELEN dataset.

A sigmoid function is used everywhere except for the decoding layer of the first output AE which is a tangent hyperbolic function to be adequate to the range of the output [-1, 1]. We use the mean squared loss function in every pre-training and training procedure.

3.4 Metrics

The Normalized Root Mean Square Error (NRMSE)[CC06] (Eq.4) is the Euclidean distance between the predicted shape and the ground truth normalized by the product of the number of points in the shape and the inter-ocular distance D (distance between the eyes pupils of the ground truth) :

$$NRMSE(s_p, s_g) = \frac{1}{n * D} \sum_{i=1}^{n} ||s_{pi} - s_{gi}||_2 \quad (4)$$

where s_p and s_g are the predicted and the ground truth shapes, respectively. D is the inter-ocular distance of the shape s_g.

Using the NMRSE, we can calculate the Cumulative Distribution Function for a specific NRMSE (CDF_{NRMSE}) value (Eq.5) overall the database.

$$CDF_x = \frac{CARD(NRMSE \leq x)}{N} \quad (5)$$

where $CARD(.)$ is the cardinal of a set. N is the total number of images.

The CDF_{NRMSE} represents the percentage of images with error less or equal than the specified NRMSE value.

For example a $CDF_{0.1} = 0.4$ over a test set means that 40% of the test set images have an error less or equal than 0.1. A CDF curve can be plotted according to these CDF_{NRMSE} values by varying the value of $NRMSE$.

These are the usual evaluation criteria used in facial landmark detection. To have more numerical precision in the comparison in our experiments, we calculate the Area Under the CDF Curve (AUC), using only the NRMSE range [0,0.5] with a step of 10^{-3}.

3.5 Results

Beside the multiple configurations based on the DNN, we consider also the case where we predict the mean shape for every test image estimated only from the training set.

The total training time (unsupervised and supervised) of the DNN takes less than 20 mins for LFPW and less than 30 mins for HELEN on a GPU. The results of the studied configurations are reported in Tab. 1. This table presents the performance of the different configurations (mean shape, input pre-trained DNN, IODA) on LFPW and HELEN datasets. We report the area under the CDF curve (AUC), and the CDF value at NRMSE=0.1.

From Tab.1, one can see that pre-training the input improves the performance of the DNN compared to non pre-trained DNN. This confirms the interest of the input pre-training in deep architectures. On the other hand, pre-training many input layers does not improve too much the performance. It may decrease it, as in DNN 2-0-0 for LFPW. This may be explained by the fact that pre-training one input layer is enough to learn the relevant features for the final task. Pre-training the second input layer did not provide a good features based on the previous extracted features in the first layer. This may be caused by the pre-training that harms the parameters (i.e. the weights and the biases) of second layer and make it difficult for the final supervised training to correct this damage because of the vanishing gradient. This may also be seen as an overfitting during the pre-training where the auto-encoder achieve a low error on the training and the validation set but it is not in favor of the final supervised training. We recall that when pre-training a stack of auto-encoders, the supervised criterion is not used. Thus, the learned intermediate features are *independent* from the supervised task. In the final supervised training, the network *corrects* the learned intermediate features in the layers, in a way that they are straightforward useful for the task. This is one of the main problems in the pre-training technique.

One can also see that in the proposed method where an input and output pre-training is performed, the DNN achieves the best results. This is due to the capability of IODA to learn the structural dependencies between the key points of the face shape in the output space. This alleviates the difficulty of learning the mapping function from the input to the output space. Pre-training the output also faces the same difficulty as the input, which is finding the adequate number of layers to pre-train. In LFPW, pre-training 3 output layers provides the best results whereas 1 layer is enough in HELEN.

Pre-training the link also helps improving the results as in DNN 1-0-1 and DNN 1-1-1 on LFPW where we obtained an AUC of 80.66% and 81.50%, respectively. Another experiment exhibits the ability of IODA setups

	LFPW		**HELEN**	
	AUC	**$CDF_{0.1}$**	**AUC**	**$CDF_{0.1}$**
Mean shape	66.15%	18.30%	63.30%	16.97%
DNN 0-0-0	77.60%	50.89%	80.91%	69.69%
DNN 1-0-0	79.25%	62.94%	82.13%	76.36%
DNN 2-0-0	79.10%	58.48%	82.39%	75.75%
DNN 3-0-0	79.51%	65.62%	82.25%	77.27%
DNN 1-0-1	80.66%	68.30%	83.95%	83.03%
DNN 1-1-1	81.50%	72.32%	83.51%	80.90%
DNN 1-0-2	81.00%	71.42%	83.91%	82.42%
DNN 1-1-2	81.06%	70.98%	83.81%	83.03%
DNN 1-0-3	**81.91%**	**74.55%**	83.72%	80.30%
DNN 0-0-4	81.60%	70.98%	83.51%	81.21%
DNN 2-0-1	81.32%	72.76%	83.61%	80.00%
DNN 2-1-1	81.47%	70.08%	**84.11%**	**83.33%**
DNN 2-0-2	81.35%	71.87%	83.88%	82.12%
DNN 3-0-1	81.62%	72.76%	83.38%	78.48%

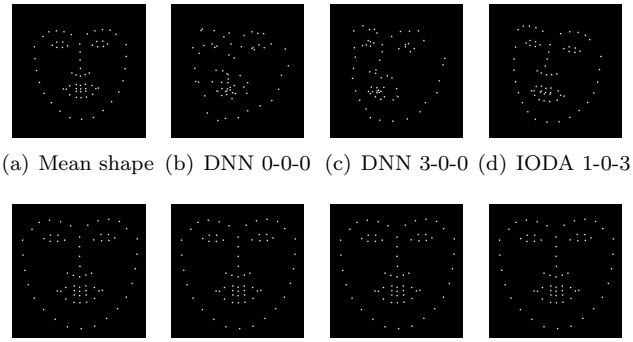
Table 1: Performance of mean shape, non pre-trained DNN, multiple configurations of input pre-trained DNN and IODA on LFPW and HELEN.

to learn the output dependencies. It consists in feeding the trained network a blank image (black) and see what it outputs. Fig.3 shows the shape output for each dataset and each best configuration.

From Fig.3(b), we see that the non pre-trained DNN learned a shape close the face shape. Pre-training the input, Fig. 3(c), improves a little the shape, but still struggle with the details. IODA, Fig.3(d), outputs a better face shape since the structure of the face (i.e. the relations between the landmarks) is better modeled.

In the case of HELEN, all the configurations succeeded to learn a face shape with a little difficulties in the non pre-trained DNN. This is due to the fact that a large part of HELEN dataset images are frontal, in the opposite of LFPW that presents a high variations in the face pose. Moreover, HELEN dataset contains

more samples than LFPW, which allows the network to learn better the face shape.



(a) Mean shape (b) DNN 0-0-0 (c) DNN 3-0-0 (d) IODA 1-0-3

(e) Mean shape (f) DNN 0-0-0 (g) DNN 2-0-0 (h) IODA 2-1-1

Figure 3: The shape response of the network for a black image for the three best configurations and the mean shape in LFPW (up) and HELEN (bottom) dataset.

We plot the CDF curves of the three best configurations in figure Fig. 4. Our results on LFPW using DNN 3-0-0 are in the same range as in [ZSKC14] using only the first global DNN, where they use a network with three hidden layers: 1600,900,400 with the same input and output dimensions as ours:(2500,136), respectively. Our $CDF_{0.1} = 65.62\%$ is close to the $CDF_{0.1}$ obtained in [ZSKC14]. Let us emphasize that we use only one dataset for training whereas in [ZSKC14], the system is trained using multiple datasets. Because of the variability in the datasets distributions, the mean shape obtained is far from the test set, thus, we obtain better results than [ZSKC14] on LFPW using the mean shape. Figures 5 and 6 show some examples of the prediction for the best DNN configurations on LFPW and HELEN respectively. We plot a segment between every predicted key point and ground truth. The longer the segment is, the higher the error is.

4 Conclusions

In this article we proposed a generic framework for structured output problems where we incorporate the structure of the output data into the model by learning the structural topology of the output space. To optimize our framework we use IODA, a neural based model. Based on the pre-training trick, IODA deals with two problems. The first is to find a good initialization to the deep network. The second is learning a better representation of the input space, and incorporating the structure of the output data into the network by learning the hidden relations in the output space. This is more appropriate when dealing with problems with structured output in a high dimensional space. We compared our method with the traditional approach to pre-train DNN, where only input pre-training is performed. To validate our approach, we tested IODA on the facial landmark detection problem on two challenging datasets: LFPW and HELEN. Our model outperformed the classical input pre-training. This demonstrated the interest of pre-training the output when there is a structured dependencies in the output space. Our work also demonstrated the capability of a DNN to perform a regression of a large vector for facial landmark detection based simply on the raw image. Cascading many IODAs' models may give better results than [ZSKC14].

By the application of the pre-training trick, it showed one of its problems which is the non-cooperation between the unsupervised and the supervised training. As a future work, we plan to build a framework where the finale supervised training guides the unsupervised pre-training in a direction that helps *explicitly* the final supervised task.

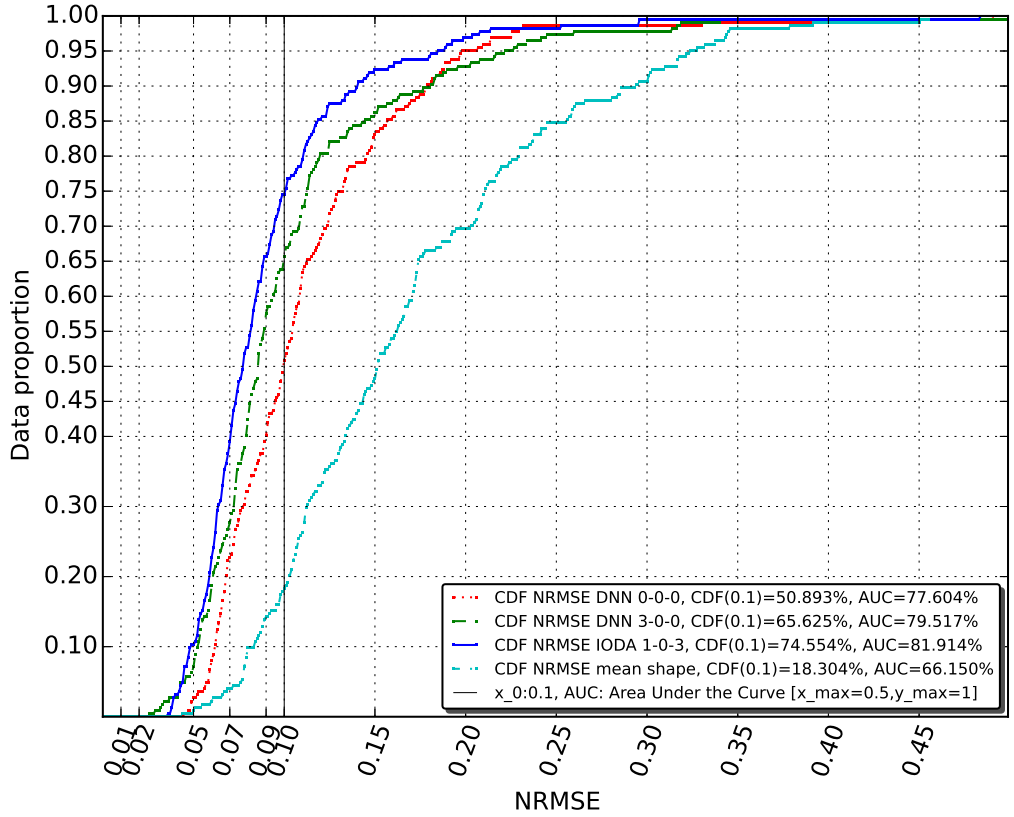
Acknowledgement

This work has been partly supported by the ANR-11-JS02-010 project LeMon.

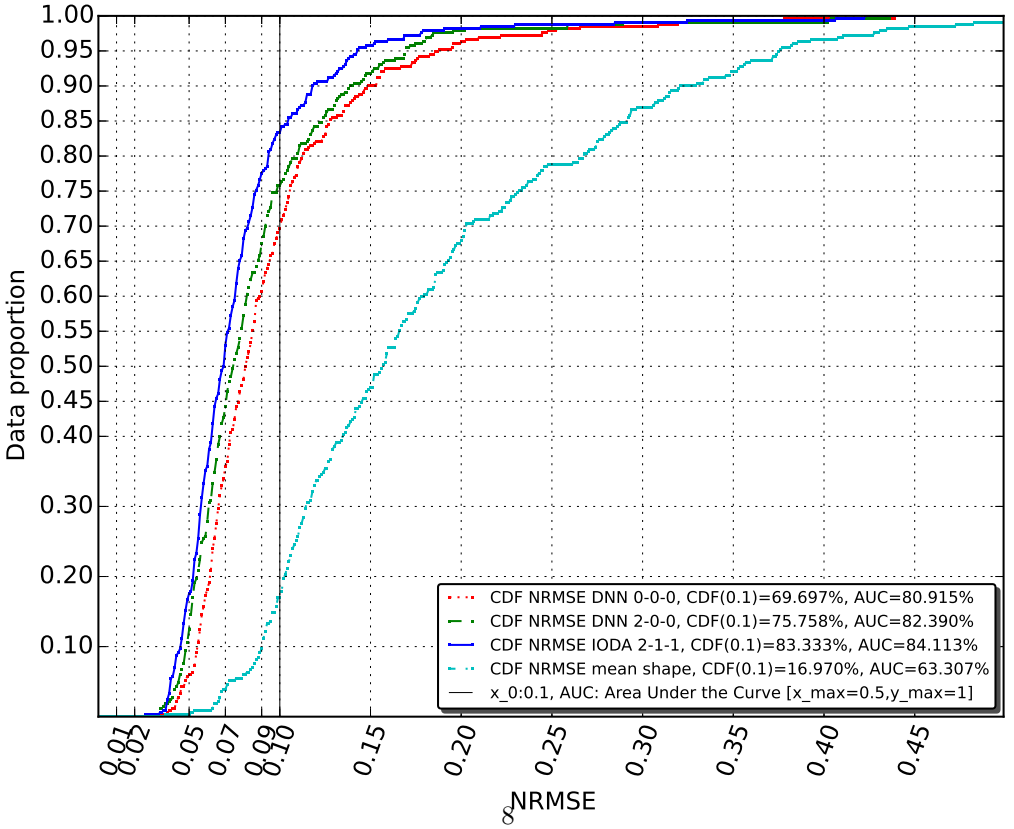
Figure 5: Prediction error on LFPW, each row: DNN 0-0-0, DNN 3-0-0, IODA 1-0-3.

References

[BCV13] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *IEEE PAMI*, 35(8):1798–1828, 2013.



(a) LFPW



(b) HELEN

Figure 4: CDF curves of best multiple configurations on: (a) LFPW, (b) HELEN.

Figure 6: Prediction error on HELEN, each row: DNN 0-0-0, DNN 2-0-0, IODA 2-1-1.

[BJKK11] Peter N. Belhumeur, David W. Jacobs, David J. Kriegman, and Neeraj Kumar. Localizing parts of faces using a consensus of exemplars. In *CVPR*, pages 545–552. IEEE, 2011.

[BK87] H. Bourlard and Y. Kamp. Auto-Association by Multilayer Perceptrons and Singular Value Decomposition. Technical Report M217, Philips Research Laboratory, Brussels, Belgium, 1987.

[BL08] MB. Blaschko and CH. Lampert. Learning to Localize Objects with Structured Output Regression. In *ECCV 2008*, pages 2–15, 2008.

[BLPL07] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In B. Schölkopf, J.C. Platt, and T. Hoffman, editors, *NIPS*, pages 153–160. 2007.

[BSW99] Daniel M Bikel, Richard Schwartz, and Ralph M Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1-3):211–231, 1999.

[CC06] D. Cristinacce and T. Cootes. Feature Detection and Tracking with Constrained Local Models. In *BMVC*, pages 95.1–95.10, 2006.

[CTCG95] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models-their training and application. *CVIU*, 61(1):38–59, 1995.

[DGFG12] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc J. Van Gool. Real-time facial feature detection using conditional regression forests. In *CVPR*, pages 2578–2585. IEEE, 2012.

[EYGB02] M. El-Yacoubi, M. Gilloux, and J-M Bertille. A statistical approach for phrase location and recognition within a text line: An application to street name recognition. *IEEE PAMI*, 24(2):172–188, 2002.

[fac] 300 faces in-the-wild challenge. `http://ibug.doc.ic.ac.uk/resources/300-W/`.

[FH05] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

[Fri93] Moshe Fridman. *Hidden markov model regression*. PhD thesis, Graduate School of Arts and Sciences, University of Pennsylvania, 1993.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006.

[Jon99] David T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202, 1999.

[LBL+12] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir D. Bourdev, and Thomas S. Huang. Interactive Facial Feature Localization. In *ECCV, 2012, Proceedings, Part III*, pages 679–692, 2012.

[LHC+15] J. Lerouge, R. Herault, C. Chatelain, F. Jardin, and R. Modzelewski. IODA: An Input Output Deep Architecture for image labeling. *Pattern Recognition*, 2015.

[LMP01] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, pages 282–289, 2001.

[NC12] Keith Noto and Mark Craven. Learning Hidden Markov Models for Regression using Path Aggregation. *CoRR*, abs/1206.3275, 2012.

[NPH06] Stéphane Nicolas, Thierry Paquet, and Laurent Heutte. A Markovian Approach for Handwritten Document Segmentation. In *ICPR (3)*, pages 292–295, 2006.

[Och03] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the ACL*, volume 1, 2003.

[Rab89] Lawrence Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RPCL07] A. Ranzato, C. Poultney, S. Chopra, and Y. Lecun. Efficient Learning of Sparse Representations with an Energy-Based Model. In *NIPS*, pages 1137–1144. 2007.

[Sch94] H. Schmid. Part-of-speech tagging with neural networks. *conference on Computational linguistics*, 12:44–49, 1994.

[SLC11] J. Saragih, S. Lucey, and J. Cohn. Deformable Model Fitting by Regularized Landmark Mean-Shift. *IJCV*, 2011.

[SMDH13] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, volume 28, pages 1139–1147, 2013.

[SQ04] M. Szummer and Y. Qi. Contextual Recognition of Hand-drawn Diagrams with Conditional Random Fields. In *IWFHR*, pages 32–37, 2004.

[ST95] Daniel Dominic Sleator and David Temperley. Parsing English with a Link Grammar. *CoRR*, 1995.

[STZP13] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. A semi-automatic methodology for facial landmark annotation. In *CVPR Workshops*, pages 896–903, 2013.

[SWT13] Y. Sun, X. Wang, and X. Tang. Deep Convolutional Network Cascade for Facial Point Detection. In *CVPR*, pages 3476–3483, 2013.

[SY09] U. Syed and G. Yona. Enzyme function prediction with interpretable models. *Computational Systems Biology. Humana press*, pages 373–420, 2009.

[TWMM07] G. Tsechpenakis, Jianhua Wang, B. Mayer, and D. Metaxas. Coupling CRFs and Deformable Models for 3D Medical Image Segmentation. In *ICCV*, pages 1–8, 2007.

[TYRW14] Y. Taigman, Ming Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *CVPR*, pages 1701–1708, 2014.

[VLL+10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *JMLR*, 11:3371–3408, 2010.

[WCV+02] J. Weston, O. Chapelle, V. Vapnik, A. Elisseeff, and B. Schölkopf. Kernel dependency estimation. In *NIPS*, pages 873–880, 2002.

[YHZ+13] X. Yu, J. Huang, S. Zhang, W. Yan, and D. Metaxas. Pose-Free Facial Landmark Fitting via Optimized Part Mixtures and Cascaded Deformable Shape Model. In *ICCV*, pages 1944–1951, 2013.

[ZBL13] F. Zhou, J. Brandt, and Z. Lin. Exemplar-Based Graph Matching for Robust Facial Landmark Localization. In *ICCV*, pages 1025–1032, 2013.

[ZR12] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, pages 2879–2886. IEEE, 2012.

[ZSCC13] X. Zhao, S. Shan, X. Chai, and X. Chen. Cascaded Shape Space Pruning for Robust Facial Landmark Detection. In *ICCV*, pages 1033–1040, 2013.

[ZSKC14] J. Zhang, S. Shan, M. Kan, and X. Chen. Coarse-to-Fine Auto-Encoder Networks (CFAN) for Real-Time Face Alignment. In *ECCV, Part II*, pages 1–16, 2014.

[ZTB09] H. Zen, K. Tokuda, and A. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064, 2009.