



**HAL**  
open science

# Deep neural networks regularization for structured output prediction

Soufiane Belharbi, Romain Hérault, Clément Chatelain, Sébastien Adam

► **To cite this version:**

Soufiane Belharbi, Romain Hérault, Clément Chatelain, Sébastien Adam. Deep neural networks regularization for structured output prediction. *Neurocomputing*, 2018, 281, pp.169-177. <10.1016/j.neucom.2017.12.002>. <hal-02094963>

**HAL Id: hal-02094963**

**<https://hal.science/hal-02094963v1>**

Submitted on 16 Feb 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

---

# Deep Neural Networks Regularization for Structured Output Prediction

---

**Soufiane Belharbi\***

Normandie Univ, UNIROUEN, UNIHAVRE,  
INSA Rouen, LITIS  
76000 Rouen, France  
soufiane.belharbi@insa-rouen.fr

**Romain Hérault**

Normandie Univ, UNIROUEN, UNIHAVRE,  
INSA Rouen, LITIS  
76000 Rouen, France  
romain.herault@insa-rouen.fr

**Clément Chatelain**

Normandie Univ, UNIROUEN, UNIHAVRE,  
INSA Rouen, LITIS  
76000 Rouen, France  
clement.chatelain@insa-rouen.fr

**Sébastien Adam**

Normandie Univ, UNIROUEN, UNIHAVRE,  
INSA Rouen, LITIS  
76000 Rouen, France  
sebastien.adam@univ-rouen.fr

## Abstract

A deep neural network model is a powerful framework for learning representations. Usually, it is used to learn the relation  $x \rightarrow y$  by exploiting the regularities in the input  $x$ . In structured output prediction problems,  $y$  is multi-dimensional and structural relations often exist between the dimensions. The motivation of this work is to learn the output dependencies that may lie in the output data in order to improve the prediction accuracy. Unfortunately, feedforward networks are unable to exploit the relations between the outputs. In order to overcome this issue, we propose in this paper a regularization scheme for training neural networks for these particular tasks using a multi-task framework. Our scheme aims at incorporating the learning of the output representation  $y$  in the training process in an unsupervised fashion while learning the supervised mapping function  $x \rightarrow y$ .

We evaluate our framework on a facial landmark detection problem which is a typical structured output task. We show over two public challenging datasets (LFPW and HELEN) that our regularization scheme improves the generalization of deep neural networks and accelerates their training. The use of unlabeled data and label-only data is also explored, showing an additional improvement of the results. We provide an opensource implementation<sup>2</sup> of our framework.

## 1 Introduction

In machine learning field, the main task usually consists in learning general regularities over the input space in order to provide a specific output. Most of machine learning applications aim at predicting a single value: a label for classification or a scalar value for regression. Many recent applications address challenging problems where the output lies in a multi-dimensional space describing discrete or continuous variables that are most of the time interdependent. A typical example is speech recognition, where the output label is a sequence of characters which are interdependent, following the statistics of the considered language. These dependencies generally constitute a regular structure such as a sequence, a string, a tree or a graph. As it provides constraints that may help the prediction,

---

\*<https://sbelharbi.github.io>

<sup>2</sup><https://github.com/sbelharbi/structured-output-ae>

this structure should be either discovered if unknown, or integrated in the learning algorithm using prior assumptions. The range of applications that deal with structured output data is large. One can cite, among others, image labeling [12, 26, 31, 35, 49, 16, 24, 39], statistical natural language processing (NLP) [17, 33, 38, 37], bioinformatics [18, 43], speech processing [34, 47] and handwriting recognition [15, 40]. Another example which is considered in the evaluation of our proposal in this paper is the facial landmark detection problem. The task consists in predicting the coordinates of a set of keypoints given the face image as input (Fig.1). The set of points are interdependent throughout geometric relations induced by the face structure. Therefore, facial landmark detection can be considered as a structured output prediction task.



Figure 1: Examples of facial landmarks from LFPW [4] training set.

One main difficulty in structured output prediction is the exponential number of possible configurations of the output space. From a statistical point of view, learning to predict accurately high dimensional vectors requires a large amount of data where in practice we usually have limited data. In this article we propose to consider structured output prediction as a representation learning problem, where the model must i) capture the discriminative relation between  $x$  (input) and  $y$  (output), and ii) capture the interdependencies laying between the variables of each space by efficiently modeling the input and output distributions. We address this modelization through a regularization scheme for training neural networks. Feedforward neural networks lack exploiting the structural information between the  $y$  components. Therefore, we incorporate in our framework an unsupervised task which aims at discovering this hidden structure. The advantage of doing so is there is no need to fix beforehand any prior structural information. The unsupervised task learns it on itself.

Our contributions is a multi-task framework dedicated to train feedforward neural networks models for structured output prediction. We propose to combine unsupervised tasks over the input and output data in parallel with the supervised task. This parallelism can be seen as a regularization of the supervised task which helps it to generalize better. Moreover, as a second contribution, we demonstrate experimentally the benefit of using the output labels  $y$  without their corresponding inputs  $x$ . In this work, the multi task framework is instantiated using auto-encoders [46, 5] for both representations learning and exploiting unlabeled data (input) and label-only data (output). We demonstrate the efficiency of our proposal over a real-world facial landmark detection problem.

The rest of the paper is organized as follows. Related works about structured output prediction is proposed in section 2. Section 3 presents the proposed formulation and its optimization details. Section 4 describes the instantiation of the formulation using a deep neural network. Finally, section 5 details the conducted experiments including the datasets, the evaluation metrics and the general training setup. Two types of experiments are explored: with and without the use of unlabeled data. Results are presented and discussed for both cases.

## 2 Related work

We distinguish two main categories of methods for structured output prediction. For a long time, graphical models have showed a large success in different applications involving 1D and 2D signals. Recently, a new trend has emerged based on deep neural networks.

## 2.1 Graphical Models Approaches

Historically, graphical models are well known to be suitable for learning structures. One of their main strength is an easy integration of explicit structural constraints and prior knowledge directly into the model's structure. They have shown a large success in modeling structured data thanks to their capacity to capture dependencies among relevant random variables. For instance, Hidden Markov Models (HMM) framework has a large success in modeling sequence data. HMMs make an assumption that 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 frameworks are widely used to model structured output data represented as a 1-D sequence [11, 34, 6, 21]. Many approaches have also been proposed to deal with 2-D structured output data as an extension of HMM and CRF. [29] propose a Markov Random Field (MRF) for document image segmentation. [44] provide an adaptation of CRF to 2-D signals with hand drawn diagrams interpretation. Another extension of CRF to 3-D signal is presented in [45] for 3-D medical image segmentation. Despite the large success of graphical models in many domains, they still encounter some difficulties. For instance, due to their inference computational cost, graphical models are limited to low dimensional structured output problems. Furthermore, HMM and CRF models are generally used with discrete output data where few works address the regression problem [32, 13].

## 2.2 Deep Neural Networks Approaches

More recently, deep learning based approaches have been widely used to solve structured output prediction, especially proposed for image labeling problems. Deep learning domain provides many different architectures. Therefore, different solutions were proposed depending on the application in hand and what is expected as a result.

In image labeling task (also known as semantic segmentation), one needs models able to adapt to the large variations in the input image. Given their large success in image processing related tasks [20], convolutional neural networks is a natural choice. Therefore, they have been used as the core model in image labeling problems in order to learn the relevant features. They have been used either combined with simple post-processing in order to calibrate the output [8] or with more sophisticated models in structure modeling such as CRF [12] or energy based models [30]. Recently, a new trend has emerged, based on the application of convolution [26, 35] or deconvolutional [31] layers in the output of the network which goes by the name of fully convolutional networks and showed successful results in image labeling. Despite this success, these models does not take in consideration the output representation.

In many applications, it is not enough to provide the output prediction, but also its probability. In this case, Conditional Restricted Boltzmann Machines, a particular case of neural networks and probabilistic graphical models have been used with different training algorithms according to the size of the plausible output configurations [28]. Training and inferring using such models remains a difficult task. In this same direction, [2] tackle structured output problems as an energy minimization through two feed-forward networks. The first is used for feature extraction over the input. The second is used for estimating an energy by taking as input the extracted features and the current state of the output labels. This allows learning the interdependencies within the output labels. The prediction is performed using an iterative backpropagation-based method with respect to the labels through the second network which remains computationally expensive. Similarly, Recurrent Neural Networks (RNN) are a particular architecture of neural networks. They have shown a great success in modeling sequence data and outputting sequence probability for applications such as Natural Language Processing (NLP) tasks [25, 42, 1] and speech recognition [14]. It has also been used for image captioning [19]. However, RNN models do not consider explicitly the output dependencies.

In [23], our team proposed the use of auto-encoders in order to learn the output distribution in a pre-training fashion with application to image labeling with promising success. The approach consists in two sequential steps. First, an input and output pre-training is performed in an unsupervised way using autoencoders. Then, a finetune is applied on the whole network using supervised data. While this approach allows incorporating prior knowledge about the output distribution, it has two main issues. First, the alteration of a network output layer is critical and must be performed carefully. Moreover, one needs to perform multiple trial-error loops in order to set the autoencoder's training

hyper-parameters. The second issue is overfitting. When pre-training the output auto-encoder, there is actually no information that indicates if the pre-training is helping the supervised task, nor when to stop the pre-training.

The present work proposes a general and easy to use multi-task training framework for structured output prediction models. The input and the output unsupervised tasks are embedded into a regularization scheme and learned in parallel with the supervised task. The rationale behind is that the unsupervised tasks should provide a *generalization* aspect to the main supervised task and should limit overfitting. This parallel transfer learning which includes an output reconstruction task constitutes the main contribution of this work. In structured output context, the role of the output task is to learn the hidden structure within the original output data, in an unsupervised way. This can be very helpful in models that do not consider the relations between the components of the output representation such as feedforward neural networks. We also show that the proposed framework enables to use labels without input in an unsupervised fashion and its effect on the generalization of the model. This can be very useful in applications where the output data is abundant such as in a speech recognition task where the output is ascii text which can be easily gathered from Internet. In this article, we validate our proposal on a facial landmark prediction problem over two challenging public datasets (LFPW and HELEN). The performed experiments show an improvement of the generalization of deep neural networks and an acceleration of their training.

### 3 Multi-task Training Framework for Structured Output Prediction

Let us consider a training set  $\mathcal{D}$  containing examples with both features and targets  $(x, y)$ , features without target  $(x, \_)$ , and targets without features  $(\_, y)$ . Let us consider a set  $\mathcal{F}$  which is the subset of  $\mathcal{D}$  containing examples with at least features  $x$ , a set  $\mathcal{L}$  which is the subset of  $\mathcal{D}$  containing examples with at least targets  $y$ , and a set  $\mathcal{S}$  which is the subset of  $\mathcal{D}$  containing examples with both features  $x$  and targets  $y$ . One can note that all examples in  $\mathcal{S}$  are also in  $\mathcal{F}$  and in  $\mathcal{L}$ .

#### Input task

The input task  $\mathcal{R}_{in}$  is an unsupervised reconstruction task which aims at learning global and more robust input representation based on the original input data  $\mathbf{x}$ . This task projects the input data  $\mathbf{x}$  into an intermediate representation space  $\tilde{\mathbf{x}}$  through a coding function  $P_{in}$ , known as encoder. Then, it attempts to recover the original input by reconstructing  $\hat{\mathbf{x}}$  from  $\tilde{\mathbf{x}}$  through a decoding function  $P'_{in}$ , known as decoder:

$$\hat{\mathbf{x}} = \mathcal{R}_{in}(\mathbf{x}; \mathbf{w}_{in}) = P'_{in}(\tilde{\mathbf{x}} = P_{in}(\mathbf{x}; \mathbf{w}_{cin}); \mathbf{w}_{din}), \quad (1)$$

where  $\mathbf{w}_{in} = \{\mathbf{w}_{cin}, \mathbf{w}_{din}\}$ . The decoder parameters  $\mathbf{w}_{din}$  are proper to this task however the encoder parameters  $\mathbf{w}_{cin}$  are shared with the main task (see Fig.2). This multi-task aspect will attract, hopefully, the shared parameters in the parameters space toward regions that build more general and robust input representations and avoid getting stuck in local minima. Therefore, it promotes generalization. This can be useful to start the training process of the main task.

The training criterion for this task is given by :

$$\mathcal{J}_{in}(\mathcal{F}; \mathbf{w}_{in}) = \frac{1}{\text{card } \mathcal{F}} \sum_{x \in \mathcal{F}} \mathcal{C}_{in}(\mathcal{R}_{in}(\mathbf{x}; \mathbf{w}_{in}), \mathbf{x}), \quad (2)$$

where  $\mathcal{C}_{in}$  is an unsupervised learning cost which can be computed on all the samples with features (i.e. on  $\mathcal{F}$ ). Practically, it can be the mean squared error.

#### Output task

The output task  $\mathcal{R}_{out}$  is an unsupervised reconstruction task which has the same goal as the input task. Similarly, this task projects the output data  $\mathbf{y}$  into an intermediate representation space  $\tilde{\mathbf{y}}$  through a coding function  $P_{out}$ , i.e. a coder. Then, it attempts to recover the original output data by reconstructing  $\hat{\mathbf{y}}$  based on  $\tilde{\mathbf{y}}$  through a decoding function  $P'_{out}$ , i.e. a decoder. In structured output data,  $\tilde{\mathbf{y}}$  can be seen as a code that contains many aspect of the original output data  $\mathbf{y}$ , most importantly, its hidden structure that describes the global relation between the components of  $\mathbf{y}$ . This hidden structure is discovered in an unsupervised way without priors fixed beforehand which makes it simple to use. Moreover, it allows using

labels only (without input  $\mathbf{x}$ ) which can be helpful in tasks with abundant output data such as in speech recognition task (Sec.2):

$$\hat{\mathbf{y}} = \mathcal{R}_{out}(\mathbf{y}; \mathbf{w}_{out}) = P'_{out}(\tilde{\mathbf{y}} = P_{out}(\mathbf{y}; \mathbf{w}_{cout}); \mathbf{w}_{dout}) \quad . \quad (3)$$

where  $\mathbf{w}_{out} = \{\mathbf{w}_{cout}, \mathbf{w}_{dout}\}$ . In the opposite of the input task, the encoder parameters  $\mathbf{w}_{cout}$  are proper to this task while the decoder parameters  $\mathbf{w}_{dout}$  are shared with the main task (see Fig.2).

The training criterion for this task is given by :

$$\mathcal{J}_{out}(\mathcal{L}; \mathbf{w}_{out}) = \frac{1}{\text{card } \mathcal{L}} \sum_{y \in \mathcal{L}} C_{out}(\mathcal{R}_{out}(\mathbf{y}; \mathbf{w}_{out}), \mathbf{y}) \quad , \quad (4)$$

where  $C_{out}$  is an unsupervised learning cost which can be computed on all the samples with labels (i.e. on  $\mathcal{L}$ ), typically, the mean squared error.

### Main task

The main task is a supervised task that attempts to learn the mapping function  $\mathcal{M}$  between features  $\mathbf{x}$  and labels  $\mathbf{y}$ . In order to do so, the first part of the mapping function is shared with the encoding part  $P_{in}$  of the input task and the last part is shared with the decoding part  $P'_{out}$  of the output task. The middle part  $m$  of the mapping function  $\mathcal{M}$  is specific to this task:

$$\hat{\mathbf{y}} = \mathcal{M}(\mathbf{x}; \mathbf{w}_{sup}) = P'_{out}(m(P_{in}(\mathbf{x}; \mathbf{w}_{cin}); \mathbf{w}_s); \mathbf{w}_{dout}) \quad . \quad (5)$$

where  $\mathbf{w}_{sup} = \{\mathbf{w}_{cin}, \mathbf{w}_s, \mathbf{w}_{dout}\}$ . Accordingly,  $\mathbf{w}_{cin}$  and  $\mathbf{w}_{dout}$  parameters are respectively shared with the input and output tasks.

Learning this task consists in minimizing its learning criterion  $\mathcal{J}_s$ ,

$$\mathcal{J}_s(\mathcal{S}; \mathbf{w}_{sup}) = \frac{1}{\text{card } \mathcal{S}} \sum_{(x,y) \in \mathcal{S}} C_s(\mathcal{M}(x; \mathbf{w}_{sup}), y) \quad , \quad (6)$$

where  $C_s(\cdot, \cdot)$  can be the mean squared error.

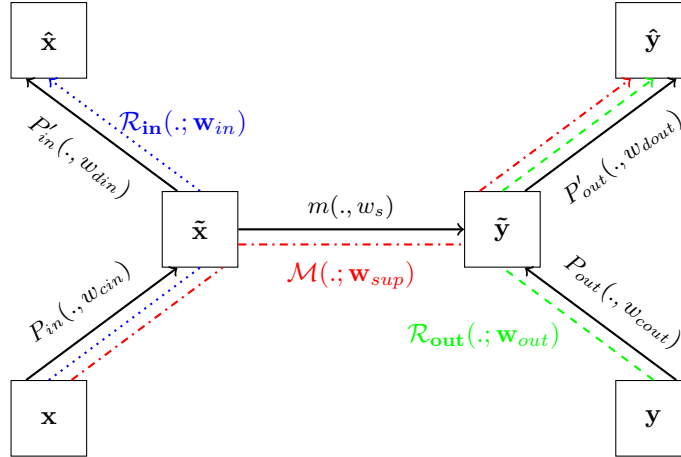


Figure 2: Proposed MTL framework. Black plain arrows stand for intermediate functions, blue dotted arrow for input auxiliary task  $\mathcal{R}_{in}$ , green dashed arrow for output auxiliary task  $\mathcal{R}_{out}$ , and red dash-dotted arrow for the main supervised task  $\mathcal{M}$ .

As a synthesis, our proposal is formulated as a multi-task learning framework (MTL) [7], which gathers a main task and two secondary tasks. This framework is illustrated in Fig. 2.

Learning the three tasks is performed in parallel. This can be translated in terms of training cost as the sum of the corresponding costs. Given that the tasks have different importance, we weight each

cost using a corresponding importance weight  $\lambda_{sup}$ ,  $\lambda_{in}$  and  $\lambda_{out}$  respectively for the supervised, the input and output tasks. Therefore, the full objective of our framework can be written as:

$$\mathcal{J}(\mathcal{D}; \mathbf{w}) = \lambda_{sup} \cdot \mathcal{J}_s(\mathcal{S}; \mathbf{w}_{sup}) + \lambda_{in} \cdot \mathcal{J}_{in}(\mathcal{F}; \mathbf{w}_{in}) + \lambda_{out} \cdot \mathcal{J}_{out}(\mathcal{L}; \mathbf{w}_{out}), \quad (7)$$

where  $\mathbf{w} = \{\mathbf{w}_{cin}, \mathbf{w}_{din}, \mathbf{w}_s, \mathbf{w}_{cout}, \mathbf{w}_{dout}\}$  is the complete set of parameters of the framework.

Instead of using fixed importance weights that can be difficult to optimally set, we evolve them through the learning epochs. In this context, Eq. 7 is modified as follows :

$$\mathcal{J}(\mathcal{D}; \mathbf{w}) = \lambda_{sup}(t) \cdot \mathcal{J}_s(\mathcal{S}; \mathbf{w}_{sup}) + \lambda_{in}(t) \cdot \mathcal{J}_{in}(\mathcal{F}; \mathbf{w}_{in}) + \lambda_{out}(t) \cdot \mathcal{J}_{out}(\mathcal{L}; \mathbf{w}_{out}), \quad (8)$$

where  $t \geq 0$  indicates the learning epochs. Our motivation to evolve the importance weights is that we want to use the secondary tasks to start the training and avoid the main task to get stuck in local minima early in the beginning of the training by moving the parameters towards regions that generalize better. Then, toward the end of the training, we drop the secondary tasks by annealing their importance toward zero because they are no longer necessary for the main task. The early stopping of the secondary tasks is important in this context of multi-tasking as shown in [50] otherwise, they will overfit, therefore, they will harm the main task. The main advantage of Eq.8 is that it allows an interaction between the main supervised task and the secondary tasks. Our hope is that this interaction will promote the generalization aspect of the main task and prevent it from overfitting.

## 4 Implementation

In this work, we implement our framework throughout a deep neural network. The main supervised task is performed using a deep neural network (DNN) with  $K$  layers. Secondary reconstruction tasks are carried out by auto-encoders (AE): the input task is achieved using an AE that has  $K_{in}$  layers in its encoding part, with an encoded representation of the same dimension as  $\tilde{\mathbf{x}}$ . Similarly, the output task is achieved using an AE that has  $K_{out}$  layers in its decoding part, with an encoded representation of the same dimension as  $\tilde{\mathbf{y}}$ . At least one layer must be dedicated in the DNN to link  $\tilde{\mathbf{x}}$  and  $\tilde{\mathbf{y}}$  in the intermediate spaces. Therefore,  $K_{in} + K_{out} < K$ .

Parameters  $\mathbf{w}_{in}$  are the parameters of the whole input AE,  $\mathbf{w}_{out}$  are the parameters of the whole output AE and  $\mathbf{w}_{sup}$  are the parameters of the main neural network (NN). The encoding layers of the input AE are tied to the first layers of the main NN, and the decoding layers of the output AE are in turn tied to the last layers of the main NN. If  $\mathbf{w}_i$  are the parameters of layer  $i$  of a neural network, then  $\mathbf{w}_1$  to  $\mathbf{w}_{K_{in}}$  parameters of the input AE are shared with  $\mathbf{w}_1$  to  $\mathbf{w}_{K_{in}}$  parameters of the main NN. Moreover, if  $\mathbf{w}_{-i}$  are the parameters of last minus  $i - 1$  layer of a neural network, then parameters  $\mathbf{w}_{-K_{out}}$  to  $\mathbf{w}_{-1}$  of the output AE are shared with the parameters  $\mathbf{w}_{-K_{out}}$  to  $\mathbf{w}_{-1}$  of the main NN.

During training, the loss function of the input AE is used as  $\mathcal{J}_{in}$ , the loss function of the output AE is used as  $\mathcal{J}_{out}$ , and the loss function of the main NN is used as  $\mathcal{J}_s$ .

Optimizing Eq.8 can be performed using Stochastic Gradient Descent. In the case of task combination, one way to perform the optimization is to alternate between the tasks when needed [9, 50]. In the case where the training set does not contain unlabeled data, the optimization of Eq.8 can be done in parallel over all the tasks. When using unlabeled data, the gradient for the whole cost can not be computed at once. Therefore, we need to split the gradient for each sub-cost according to the nature of the samples at each mini-batch. For the sake of clarity, we illustrate our optimization scheme in Algorithm 1 using on-line training (i.e. training one sample at a time). Mini-batch training can be performed in the same way.

## 5 Experiments

We evaluate our framework on a facial landmark detection problem which is typically a structured output problem since the facial landmarks are spatially inter-dependent. Facial landmarks are a set of key points on human face images as shown in Fig. 1. Each key point is defined by the coordinates  $(x, y)$  in the image  $(x, y \in \mathbb{R})$ . The number of landmarks is dataset or application dependent.

It must be emphasized here that the purpose of our experiments in this paper was not to outperform the state of the art in facial landmark detection but to show that learning the output dependencies helps improving the performance of DNN on that task. Thus, we will compare a model with/without

---

**Algorithm 1** Our training strategy for one epoch

---

```
1:  $\mathcal{D}$  is the shuffled training set.  $B$  a sample.  
2: for  $B$  in  $\mathcal{D}$  do  
3:   if  $B$  contains  $x$  then  
4:     Update  $w_{in}$ : Make a gradient step toward  $\lambda_{in} \times \mathcal{J}_{in}$  using  $B$  (Eq.2).  
5:   end if  
6:   if  $B$  contains  $y$  then  
7:     Update  $w_{out}$ : Make a gradient step toward  $\lambda_{out} \times \mathcal{J}_{out}$  using  $B$  (Eq.4).  
8:   end if  
9:   # parallel parameters update  
10:  if  $B$  contains  $x$  and  $y$  then  
11:    Update  $w$ : Make a gradient step toward  $\mathcal{J}$  using  $B$  (Eq.8).  
12:  end if  
13:  Update  $\lambda_{sup}$ ,  $\lambda_{in}$  and  $\lambda_{out}$ .  
14: end for
```

---

input and output training. [48] use a cascade of neural networks. In their work, they provide the performance of their first global network. Therefore, we will use it as a reference to compare our performance (both networks has close architectures) except they use larger training dataset.

We first describe the datasets followed by a description of the evaluation metrics used in facial landmark problems. Then, we present the general setup of our experiments followed by two types of experiments: without and with unlabeled data. An opensource implementation of our MTL deep instantiation is available online<sup>3</sup>.

## 5.1 Datasets

We have carried out our evaluation over two challenging public datasets for facial landmark detection problem: LFPW [4] and HELEN [22].

**LFPW dataset** 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 [4], we use only the 811 training images and the 224 test images provided by the ibug website<sup>4</sup>. Ground truth annotations of 68 facial points are provided by [36]. We divide the available training samples into two sets: validation set (135 samples) and training set (676 samples).

**HELEN dataset** is similar to LFPW dataset, where the images have been taken under unconstrained conditions with high resolution and collected from Flickr using text queries. It contains 2000 images for training, and 330 images for test. Images and face bounding boxes are provided by the same site as for LFPW. The ground truth annotations are provided by [36]. Examples of dataset are shown in Fig.3.



Figure 3: Samples from HELEN [22] dataset.

All faces are cropped into the same size ( $50 \times 50$ ) and pixels are normalized in  $[0,1]$ . The facial landmarks are normalized into  $[-1,1]$ .

---

<sup>3</sup><https://github.com/sbelharbi/structured-output-ae>

<sup>4</sup>300 faces in-the-wild challenge <http://ibug.doc.ic.ac.uk/resources/300-w/>

## 5.2 Metrics

In order to evaluate the prediction of the model, we use the standard metrics used in facial landmark detection problems.

The Normalized Root Mean Squared Error (NRMSE)[10] (Eq.9) 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),

$$NRMSE(s_p, s_g) = \frac{1}{N * D} \sum_{i=1}^N \|s_{pi} - s_{gi}\|_2, \quad (9)$$

where  $s_p$  and  $s_g$  are the predicted and the ground truth shapes, respectively. Both shapes have the same number of points  $N$ .  $D$  is the inter-ocular distance of the shape  $s_g$ .

Using the NRMSE, we can calculate the Cumulative Distribution Function for a specific NRMSE ( $CDF_{NRMSE}$ ) value (Eq.10) overall the database,

$$CDF_x = \frac{CARD(NRMSE \leq x)}{n}, \quad (10)$$

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 problem. 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  $10^{-3}$ .

## 5.3 General training setup

To implement our framework, we use: - a DNN with four layers  $K = 4$  for the main task; - an input AE with one encoding layer  $K_{in} = 1$  and one decoding layer; - an output AE with one encoding layer and one decoding layer  $K_{out} = 1$ . Referring to Fig.2, the size of the input representation  $\mathbf{x}$  and estimation  $\hat{\mathbf{x}}$  is  $2500 = 50 \times 50$ ; the size of the output representation  $\mathbf{y}$  and estimation  $\hat{\mathbf{y}}$  is  $136 = 68 \times 2$ , given the 68 landmarks in a 2D plane; the dimension of intermediate spaces  $\tilde{\mathbf{x}}$  and  $\tilde{\mathbf{y}}$  have been set to 1025 and 64 respectively; finally, the hidden layer in the  $m$  link between  $\tilde{\mathbf{x}}$  and  $\tilde{\mathbf{y}}$  is composed of 512 units. The size of each layer has been set using a validation procedure on the LFPW validation set.

Sigmoid activation functions are used everywhere in the main NN and in the two AEs, except for the last layer of the main NN and the tied last layer of output AE which use a hyperbolic tangent activation function to suite the range  $[-1, 1]$  for the output  $\mathbf{y}$ .

We use the same architecture through all the experiments for the different training configurations. To distinguish between the multiple configurations we set the following notations:

1. **MLP**, a DNN for the main task with no concomitant training;
2. **MLP + in**, a DNN with input AE parallel training;
3. **MLP + out**, a DNN with output AE parallel training;
4. **MLP + in + out**, a DNN with both input and output reconstruction secondary tasks.

We recall that the auto-encoders are used only during the training phase. In the test phase, they are dropped. Therefore, the final test networks have the same architecture in all the different configurations.

Beside these configurations, we consider the mean shape (the average of the  $\mathbf{y}$  in the training data) as a simple predictive model. For each test image, we predict the same estimated mean shape over the train set.

To clarify the benefit of our approach, all the configurations must start from the same initial weights to make sure that the obtained improvement is due to the training algorithm, not to the random initialization.

For the input reconstruction tasks, we use a denoising auto-encoder with a corruption level of 20% for the first hidden layer. For the output reconstruction task, we use a simple auto-encoder. To avoid overfitting, the auto-encoders are trained using  $L_2$  regularization with a weight decay of  $10^{-2}$ .

In all the configurations, the update of the parameters of each task (supervised and unsupervised) is performed using Stochastic Gradient Descent with momentum [41] with a constant momentum coefficient of 0.9. We use mini-batch size of 10. The training is performed for 1000 epochs with a learning rate of  $10^{-3}$ .

In these experiments, we propose to use a simple linear evolution scheme for the importance weights  $\lambda_{sup}$  (supervised task),  $\lambda_{in}$  (input task) and  $\lambda_{out}$  (output task). We retain the evolution proposed in [3], and presented in Fig.4.

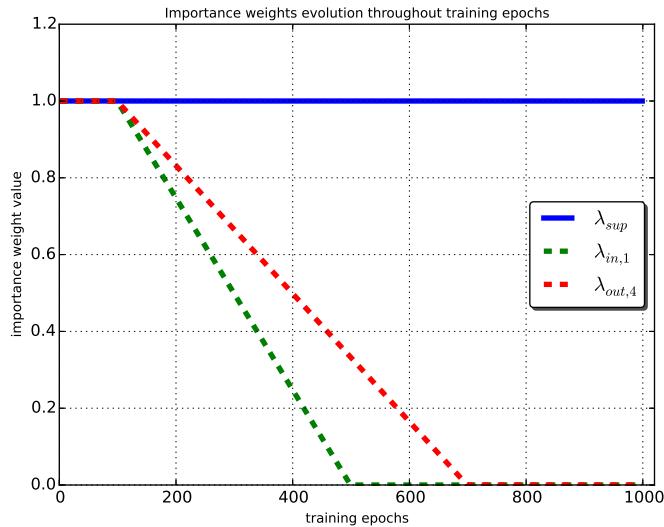


Figure 4: Linear evolution of the importance weights during training.

The hyper-parameters (learning rate, batch size, momentum coefficient, weight decay, the importance weights) have been optimized on the LFPW validation set. We apply the same optimized hyper-parameters for HELEN dataset.

Using these configurations, we perform two types of experiments: with and without unlabeled data. We present in the next sections the obtained results.

### 5.3.1 Experiments with fully labeled data

In this setup, we use the provided labeled data from each set in a classical way. For LFPW set, we use the 676 available samples for training and 135 samples for validation. For HELEN set, we use 1800 samples for training and 200 samples for validation.

In order to evaluate the different configurations, we first calculate the Mean Squared Error (MSE) of the best models found using the validation during the training. Column 1 (no unlabeled data) of Tab.1, 2 shows the MSE over the train and valid sets of LFPW and HELEN datasets, respectively. Compared to an MLP alone, adding the input training of the first hidden layer slightly reduces the train and validation error in both datasets. Training the output layer also reduces the train and validation error, with a more important factor. Combining the input train of the first hidden layer and output train of the last layer gives the best performance. We plot the tracked MSE over the train and valid sets of HELEN dataset in Fig.7(a), 7(b). One can see that the input training reduces slightly the validation MSE. The output training has a major impact over the training speed and the generalization of the model which suggests that output training is useful in the case of structured output problems.

Combining the input and the output training improves even more the generalization. Similar behavior was found on LFPW dataset.

At a second time, we evaluate each configuration over the test set of each datasets using the  $CDF_{0.1}$  metric. The results are depicted in Tab.3, 4 in the first column for LFPW and HELEN datasets, respectively. Similarly to the results previously found over the train and validation set, one can see that the joint training (supervised, input, output) outperforms all the other configurations in terms of  $CDF_{0.1}$  and AUC. The CDF curves in Fig.8 also confirms this result. Compared to the global DNN in [48] over LFPW test set, our joint trained MLP performs better ([48]:  $CDF_{0.1} = 65\%$ , ours:  $CDF_{0.1} = 69.64\%$ ), despite the fact that their model was trained using larger supervised dataset (combination of multiple supervised datasets beside LFPW).

An illustrative result of our method is presented in Fig.5, 6 for LFPW and HELEN using an MLP and MLP with input and output training.



Figure 5: 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 unlabeled data)



Figure 6: Examples of prediction on HELEN test set. Top row: MLP. Bottom row: MLP+in+out. (no unlabeled data)

### 5.3.2 Data augmentation using unlabeled data or label-only data

In this section, we experiment our approach when adding unlabeled data (input and output). Unlabeled data (i.e. image faces without the landmarks annotation) are abundant and can be found easily for

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

	No unlabeled data		With unlabeled data	
	MSE train	MSE valid	MSE train	MSE valid
<b>Mean shape</b>	$7.74 \times 10^{-3}$	$8.07 \times 10^{-3}$	$7.78 \times 10^{-3}$	$8.14 \times 10^{-3}$
<b>MLP</b>	$3.96 \times 10^{-3}$	$4.28 \times 10^{-3}$	-	-
<b>MLP + in</b>	$3.64 \times 10^{-3}$	$3.80 \times 10^{-3}$	$1.44 \times 10^{-3}$	$2.62 \times 10^{-3}$
<b>MLP + out</b>	$2.31 \times 10^{-3}$	$2.99 \times 10^{-3}$	$1.51 \times 10^{-3}$	$2.79 \times 10^{-3}$
<b>MLP + in + out</b>	<b><math>2.12 \times 10^{-3}</math></b>	<b><math>2.56 \times 10^{-3}</math></b>	<b><math>1.10 \times 10^{-3}</math></b>	<b><math>2.23 \times 10^{-3}</math></b>

Table 2: MSE over HELEN: train and valid sets, at the end of training with and without data augmentation.

	Fully labeled data only		Adding unlabeled or label-only data	
	MSE train	MSE valid	MSE train	MSE valid
<b>Mean shape</b>	$7.59 \times 10^{-3}$	$6.95 \times 10^{-3}$	$7.60 \times 10^{-3}$	$0.95 \times 10^{-3}$
<b>MLP</b>	$3.39 \times 10^{-3}$	$3.67 \times 10^{-3}$	-	-
<b>MLP + in</b>	$3.28 \times 10^{-3}$	$3.42 \times 10^{-3}$	$2.31 \times 10^{-3}$	$2.81 \times 10^{-3}$
<b>MLP + out</b>	$2.48 \times 10^{-3}$	$2.90 \times 10^{-3}$	$2.00 \times 10^{-3}$	$2.74 \times 10^{-3}$
<b>MLP + in + out</b>	<b><math>2.34 \times 10^{-3}</math></b>	<b><math>2.53 \times 10^{-3}</math></b>	<b><math>1.92 \times 10^{-3}</math></b>	<b><math>2.40 \times 10^{-3}</math></b>

Table 3: AUC and  $CDF_{0.1}$  performance over LFPW test dataset with and without unlabeled data.

	Fully labeled data only		Adding unlabeled or label-only data	
	AUC	$CDF_{0.1}$	AUC	$CDF_{0.1}$
<b>Mean shape</b>	68.78%	30.80%	77.81%	22.33%
<b>MLP</b>	76.34%	46.87%	-	-
<b>MLP + in</b>	77.13%	54.46%	80.78%	67.85%
<b>MLP + out</b>	80.93%	66.51%	81.77%	67.85%
<b>MLP + in + out</b>	<b>81.51%</b>	<b>69.64%</b>	<b>82.48%</b>	<b>71.87%</b>

Table 4: AUC and  $CDF_{0.1}$  performance over HELEN test dataset with and without unlabeled data.

	Fully labeled data only		Adding unlabeled or label-only data	
	AUC	$CDF_{0.1}$	AUC	$CDF_{0.1}$
<b>Mean shape</b>	64.60%	23.63%	64.76%	23.23%
<b>MLP</b>	76.26%	52.72%	-	-
<b>MLP + in</b>	77.08%	54.84%	79.25%	63.33%
<b>MLP + out</b>	79.63%	66.60%	80.48%	65.15%
<b>MLP + in + out</b>	<b>80.40%</b>	<b>66.66%</b>	<b>81.27%</b>	<b>71.51%</b>

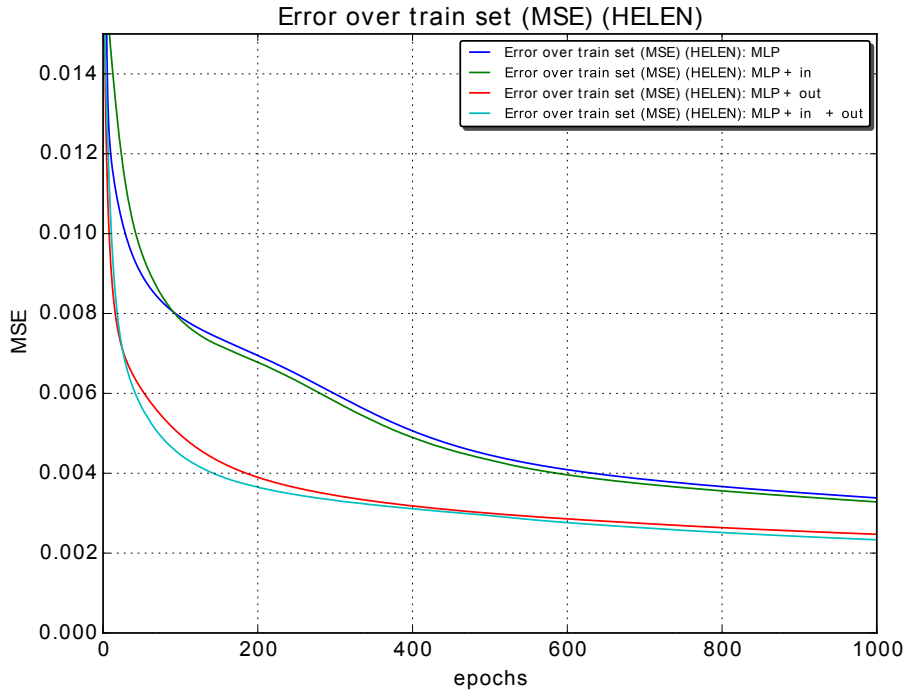
example from other datasets or from the Internet which makes it practical and realistic. In our case, we use image faces from another dataset.

In the other hand, label-only data (i.e. the landmarks annotation without image faces) are more difficult to obtain because we usually have the annotation based on the image faces. One way to obtain accurate and realistic facial landmarks without image faces is to use a 3D face model as a generator. We use an easier way to obtain facial landmarks annotation by taking them from another dataset.

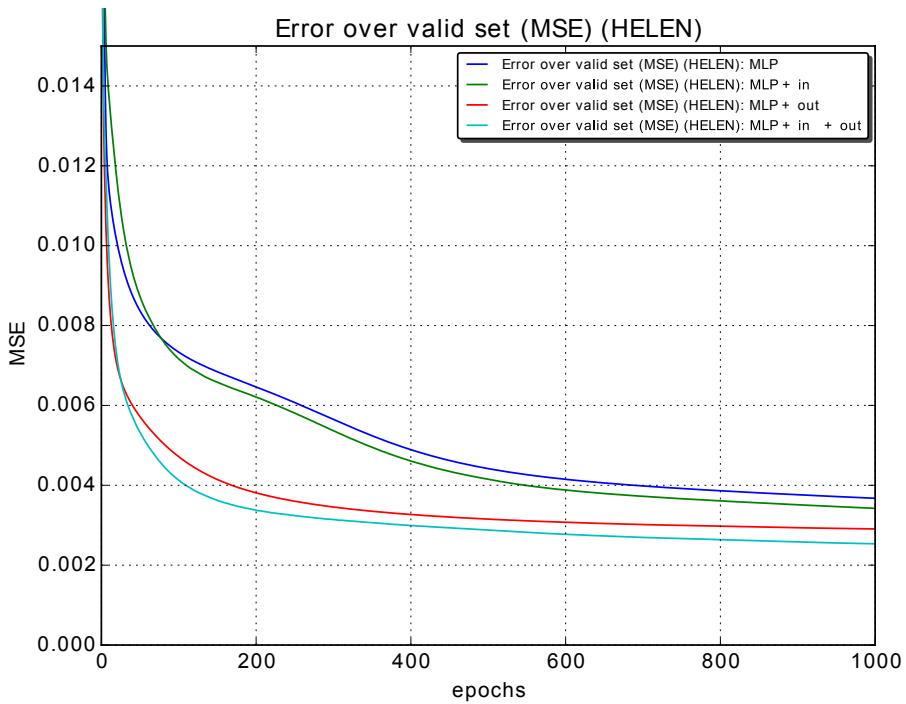
In this experiment, in order to add unlabeled data for LFPW dataset, we take all the image faces of HELEN dataset (train, valid and test) and vice versa for HELEN dataset by taking all LFPW image faces as unlabeled data. The same experiment is performed for the label-only data using the facial landmarks annotation. We summarize the size of each train set in Tab.5..

Table 5: Size of augmented LFPW and HELEN train sets.

Train set / size of	Supervised data	Unsupervised input x	Unsupervised output y
LFPW	676	2330	2330
HELEN	1800	1035	1035



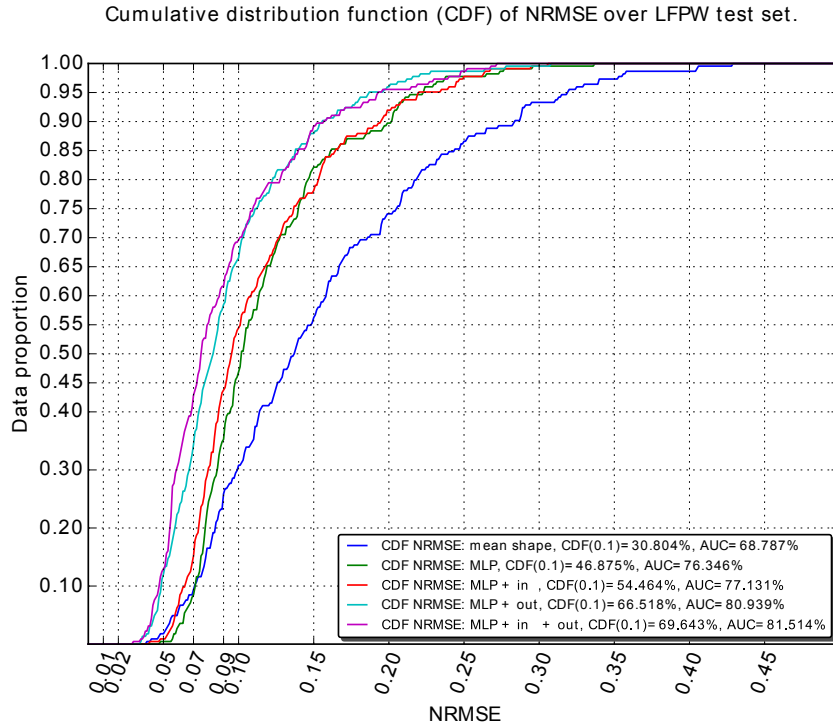
(a)



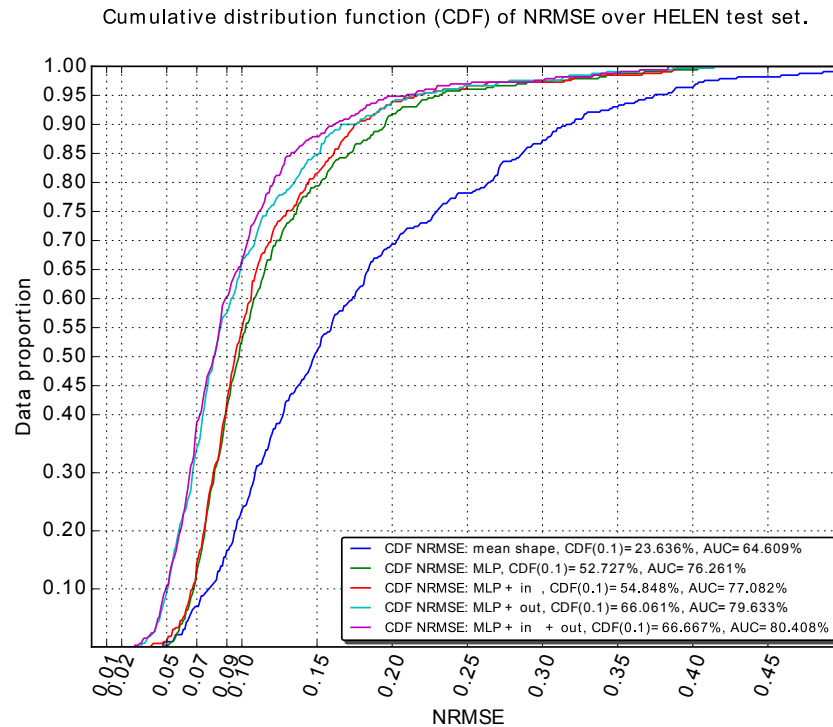
(b)

Figure 7: MSE during training epochs over HELEN train (a) and valid (b) sets using different training setups for the MLP.

We use the same validation sets as in Sec.5.3.1 in order to have a fair comparison. The MSE are presented in the second column of Tab.1, 2 over LFPW and HELEN datasets. One can see that



(a)



(b)

Figure 8: CDF curves of different configurations on: (a) LFPW, (b) HELEN.

adding unlabeled data decreases the MSE over the train and validation sets. Similarly, we found that the input training along with the output training gives the best results. Identically, these results are translated in terms of  $CDF_{0.1}$  and AUC over the test sets (Tab.3, 4). All these results suggest that adding unlabeled input and output data can improve the generalization of our framework and the training speed.

## 6 Conclusion and Future Work

In this paper, we tackled structured output prediction problems as a representation learning problem. We have proposed a generic multi-task training framework as a regularization scheme for structured output prediction models. It has been instantiated through a deep neural network model which learns the input and output distributions using auto-encoders while learning the supervised task  $x \rightarrow y$ . Moreover, we explored the possibility of using the output labels  $y$  without their corresponding input data  $x$  which showed more improvement in the generalization. Using a parallel scheme allows an interaction between the main supervised task and the unsupervised tasks which helped preventing the overfitting of the main task.

We evaluated our training method on a facial landmark detection task over two public datasets. The obtained results showed that our proposed regularization scheme improves the generalization of neural networks model and speeds up their training. We believe that our approach provides an alternative for training deep architectures for structured output prediction where it allows the use of unlabeled input and label of the output data.

As a future work, we plan to evolve automatically the importance weights of the tasks. For that and in order to better guide their evolution, we can consider the use of different indicators based on the training and the validation errors instead of the learning epochs only. Furthermore, one may consider other kind of models instead of simple auto-encoders in order to learn the output distribution. More specifically, generative models such as variational and adversarial auto-encoders [27] could be explored.

### Acknowledgments

This work has been partly supported by the grant ANR-11-JS02-010 LeMon, the grant ANR-16-CE23-0006 “Deep in France” and has benefited from computational means from CRIANN, the contributions of which are greatly appreciated.

### References

- [1] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1044–1054, 2013.
- [2] David Belanger and Andrew McCallum. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 983–992, 2016.
- [3] S. Belharbi, R.Hérault, C. Chatelain, and S. Adam. Deep multi-task learning with evolving weights. In *European Symposium on Artificial Neural Networks (ESANN)*, 2016.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

- [8] Dan C. Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2852–2860, 2012.
- [9] R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.
- [10] D. Cristinacce and T. Cootes. Feature Detection and Tracking with Constrained Local Models. In *BMVC*, pages 95.1–95.10, 2006.
- [11] 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.
- [12] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning Hierarchical Features for Scene Labeling. *IEEE PAMI*, 35(8):1915–1929, 2013.
- [13] Moshe Fridman. *Hidden markov model regression*. PhD thesis, Graduate School of Arts and Sciences, University of Pennsylvania, 1993.
- [14] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1764–1772, 2014.
- [15] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.
- [16] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, abs/1612.02649, 2016.
- [17] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep structured output learning for unconstrained text recognition. *CoRR*, abs/1412.5903, 2014.
- [18] David T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202, 1999.
- [19] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3128–3137, 2015.
- [20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [21] 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.
- [22] 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.
- [23] J. Lerouge, R. Herault, C. Chatelain, F. Jardin, and R. Modzelewski. IODA: An Input Output Deep Architecture for image labeling. *Pattern Recognition*, 2015.
- [24] Xirong Li, Tiberio Uricchio, Lamberto Ballan, Marco Bertini, Cees G. M. Snoek, and Alberto Del Bimbo. Socializing the semantic gap: A comparative survey on image tag assignment, refinement, and retrieval. *ACM Comput. Surv.*, 49(1):14:1–14:39, 2016.
- [25] Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. A recursive recurrent neural network for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1491–1500, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
- [26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3431–3440, 2015.

- [27] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.
- [28] Volodymyr Mnih, Hugo Larochelle, and Geoffrey E. Hinton. Conditional restricted boltzmann machines for structured output prediction. In *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 514–522, 2011.
- [29] Stéphane Nicolas, Thierry Paquet, and Laurent Heutte. A Markovian Approach for Handwritten Document Segmentation. In *ICPR (3)*, pages 292–295, 2006.
- [30] F. Ning, D. Delhomme, Yann LeCun, F. Piano, Léon Bottou, and Paolo Emilio Barbano. Toward automatic phenotyping of developing embryos from videos. *IEEE Trans. Image Processing*, 14(9):1360–1371, 2005.
- [31] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1520–1528, 2015.
- [32] Keith Noto and Mark Craven. Learning Hidden Markov Models for Regression using Path Aggregation. *CoRR*, abs/1206.3275, 2012.
- [33] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the ACL*, volume 1, 2003.
- [34] Lawrence Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, pages 234–241, 2015.
- [36] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. A semi-automatic methodology for facial landmark annotation. In *CVPR Workshops*, pages 896–903, 2013.
- [37] H. Schmid. Part-of-speech tagging with neural networks. *conference on Computational linguistics*, 12:44–49, 1994.
- [38] Daniel Dominic Sleator and David Temperley. Parsing English with a Link Grammar. *CoRR*, 1995.
- [39] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS 2015*, pages 3483–3491, 2015.
- [40] Bruno Stuner, Clément Chatelain, and Thierry Paquet. Cohort of LSTM and lexicon verification for handwriting recognition with gigantic lexicon. *CoRR*, abs/1612.07528, 2016.
- [41] 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.
- [42] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.
- [43] U. Syed and G. Yona. Enzyme function prediction with interpretable models. *Computational Systems Biology. Humana press*, pages 373–420, 2009.
- [44] M. Szummer and Y. Qi. Contextual Recognition of Hand-drawn Diagrams with Conditional Random Fields. In *IWFHR*, pages 32–37, 2004.
- [45] 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.
- [46] 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.
- [47] H. Zen, K. Tokuda, and A. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064, 2009.

- [48] 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.
- [49] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. *CoRR*, abs/1707.09465, 2017.
- [50] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, pages 94–108, 2014.