Face Alignment with Deep Regression

Baoguang Shi, Xiang Bai, *Senior Member, IEEE,* Wenyu Liu, *Senior Member, IEEE,*
and Jingdong Wang*, *Member, IEEE*

***Abstract*—In this paper, we present a deep regression approach for face alignment. The deep regressor is a neural network that consists of a global layer and multi-stage local layers. The global layer estimates the initial face shape from the whole image, while the following local layers iteratively updates the shape with local image observations. Combining standard derivations and numerical approximations, we make all layers able to back-propagate error differentials, so that we can apply the standard back-propagation to jointly learn the parameters from all layers. We show that the resulting deep regressor gradually and evenly approaches the true facial landmarks stage by stage, avoiding the tendency that often occurs in the cascaded regression methods and deteriorates the overall performance: yielding early stage regressors with high alignment accuracy gains but later stage regressors with low alignment accuracy gains. Experimental results on standard benchmarks demonstrate that our approach brings significant improvements over previous cascaded regression algorithms.**

***Index Terms*—Face alignment, cascaded regression, deep learning, back-propagation**

I. Introduction

FACE alignment, a.k.a. facial landmark localization, is a fundamental problem in computer vision and key to many applications, such as face recognition [48], [19], [44], facial expression classification [25], [16] and 3D face modeling [24]. It aims to predict landmark positions given a 2D facial image. This problem has attracted a lot of research efforts [9], [8], [45], [22], [13], [4], [27], [11]. It remains challenging when face images are taken under uncontrolled conditions.

Early work on face alignment includes active shape models (ASM) [9], active appearance models (AAM) [8] and template matching [45]. Recently, cascaded regression has achieved the state-of-the-art performance. Cascaded pose regression [14] and the following work explicit shape regression [6], robust cascaded pose regression [5], supervised descent method [43] etc. sequentially learn a cascade of random fern or linear regressors using shape indexed features and progressively regress the shape stage by stage over the learned cascade.

Previous cascaded regression based methods learn the regressors sequentially, from the first stage to the last stage. Each regressor stage predicts based on the shape output of the preceding regressor stage, and is learned to minimize the prediction error. Therefore, the later stage regressors do not affect the learning process of the earlier stage regressors. This results in two issues. On the one hand, early stages in the cascade may produce predictions with high variances, harming the performance of later stages. From our observations illustrated in Figure 2, early stage regressors obtain higher accuracy gains but also introduce larger variance in prediction errors. The larger input variance leads to higher difficulties for prediction in the later stages. On the other hand, in cascaded regression, each stage is learned to move facial landmarks to the corresponding ground truth locations as closely as possible. This scheme can be considered as a greedy one. Although under this scheme, each regressor alone is optimally learned. The cascade of regressor, however, could be sub-optimal when viewed as a whole, since there are potentially better paths where landmarks are moved to some locations away from the ground truth locations at early stages, and moved to the ground truth locations at the final stage.

In this paper, we propose a deep regression approach that adopts the back-propagation algorithm to jointly optimize a deep structure. The structure, as illustrated in Figure 1.a, consists of two sub-networks: a global layer and multi-stage local layers. The latter sub-network is similar to the structure of supervised descent method [43] and each local layer contains a local feature extraction sub-layer and a local regressor. The former sub-network, i.e., the global layer, aims to provide an initial result regressed from the facial image as the input of the latter local regressors.

Deep regression differs from cascaded regression in that regressors are jointly learned, instead of learned layer-wisely. In deep regression, regressors are stacked as a deep network and they are jointly optimized by the back-propagation algorithm. The resulting deep regressor gradually and simultaneously reduces the bias and the variance of the estimation from the first regressor to the last regressor, thus yielding a better facial landmark locations. As we will show in Section V-F2, this reduces the possibilities that early regressors move landmarks to bad locations, at which later regressors tend to fail to give precise predictions.

Experimental results show that the deep regression approach consistently achieves superior or comparable results on several challenging face alignment datasets. The joint learning scheme is quite general and can be applied to a wide range of cascaded regression based methods, improving their accuracies without harming their computational efficiencies. In Section V-E1 we apply the joint learning to the supervised descent method [43] and observe a significant performance improvement.

The rest of this paper is organized as follows: in Section II we present related work on face alignment. The methodology is divided into two parts. The architecture of the proposed network is described in Section III and the optimization scheme is presented in Section IV. We evaluate our methods as well as conduct comparative analysis in Section V. In

Baoguang Shi, Xiang Bai and Wenyu Liu are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, Hubei, P.R. China, 430074. (e-mail: shibaoguang@gmail.com, xbai@hust.edu.cn, liuwy@hust.edu.cn)

Jindong Wang is with Microsoft Research, Beijing, P.R. China, 100080. (e-mail: jingdw@microsoft.com)

*Corresponding author

Section VI we draw conclusions.

II. Related Work

In this section, we present an overview of related work on face alignment. Apart from the early work ASM [9] AAM [1] and template matching [45], most recent work can be categorized into part-based and cascaded regression-based.

A. Part based methods

The part-based methods usually optimize a cost function taking both local parts confidence and shape constraints into account. Constraint Local Models (CLM) [10] models the face shapes with Procrustes analysis and principle component analysis and models the texture locally. Local Evidence Aggregated Regression (LEAR) [26] models the face shape with a probabilistic graphical model and combine it with a regression-based approach. Zhu and Ramanan [50] present a unified for face detection, pose estimation as well as face alignment. The model uses a tree-structured model to capture global elastic deformation and uses a mixture model to capture topological changes to the viewpoint. Saragih et al. [35] propose a regularized landmark mean-shift method to solve the CLM optimization problem. This work also proposes extensions to handle partial occlusions and reduce computational complexity. Bayesian AAM [1] presents a Bayesian formulation of AAMs, which introduces a novel cost function only depending on the shape parameters by marginalizing out the latent texture space. Gauss-Newton deformable part models [40] jointly optimizes a global shape model and a part-based, flexible appearance model. Part-based methods, however, tend to be less accurate and slower than regression-based or deep learning-based methods in practice.

B. Cascaded regression based methods

Cascaded regression-based methods learn a cascade of regressors to iteratively update the shape estimation, starting from the initial shape which is typically the mean shape estimated from the training samples. Cascaded regression directly applies regression functions to local, usually shape-indexed features to predict the shape increment. Typically the shape constraint is not explicitly addressed but handled implicitly within the regression. Cascaded Pose Regression (CPR) [14] trains a sequence of weak random fern regressors to progressively refine an initial guess. The features are extracted at the locations specified in the coordinate system formed using the landmark points and are called the pose-indexed features. Explicit Shape Regression (ESR) [6] extends CPR by introducing two-level boosted regression and correlation-based feature selection. Robust Cascaded Pose Regression (RCPR) [5] proposes several extensions to the original CPR, including explicit occlusion detection and robust shape-indexed features. Supervised Descent Method (SDM) [43] shows that superior performance is achieved simply with SIFT descriptors and linear regressors. Ren et al. introduce Local Binary Features (LBF) [32] to the face alignment task. LBF are binary features induced by random forests, which are learned from data. The features are computationally cheap and result in very fast regression.

Our deep regression approach is closely related to the cascaded regression category. In contrast to previous works that focus on feature description ([5], [43], [32]) or regression functions ([6], [43]) within each regression stage, we explore the optimization strategy across the stages. Our approach is also quite general in that it can be directly applied to the cascaded regression-based methods, as long as their regression functions are continuous and differentiable *almost everywhere* (see Section III-B). In Section V-E1 we apply the deep regression approach to SDM [43] and observe a significant performance gain.

C. Deep learning-based methods

Some recent works apply deep learning methods, particularly the convolutional neural networks (CNNs) to the face alignment problem. A cascade of three CNN regressors [37], each of which regresses the facial landmark positions, is developed. To reduce the prediction error, each cascade stage further averages results from several CNNs applied on different face parts. Another deep learning solution, coarse-to-fine CNN cascade [49] is developed for face alignment. Zhang et al. [47] use a multi-task scheme, optimizing a landmark detection model together with other correlated tasks. In [46], Zhang et al. proposes Coarse-to-Fine Auto-encoder Networks (CFAN) which extends SDM with deep network regressors. The deep network regressors are pre-trained by a stack auto-encoder. Different from these works that directly adopt a deep network as a component regressor of the cascade and train each regressor separately, our approach regards the whole cascade of regressors as a deep network and trains the regressors jointly. In essence, these deep learning-based methods can benefit from the joint learning scheme. As an example, in Section V-E2 we show that our approach outperforms CFAN, a model that has a similar structure as ours but uses different learning scheme.

III. Architecture

Let the vector $\mathbf{s} = [x_1, y_1, \ldots, x_P, y_P]^\intercal \in \Re^{2P}$ be the shape of the face, where x_p, y_p are the coordinate of the pth landmark in the image I. P is the number of landmarks. The task of face alignment is to predict the positions of all the P landmarks, i.e., the shape $\mathbf{s}$ of the face in the image I. To make the prediction less affected by the location and scale of the face, a face detector is run on the image at first to detect the face bounding box. The coordinate system is then chosen such that the center of the bounding box is the origin and that the longer side of the box is of the unit length. Thereafter, the values in $\mathbf{s}$ are within range $[-0.5, 0.5]$. Besides, I is taken as the face image cropped from the detected bounding box.

Our proposed model is a multi-layered network. Each layer in the network outputs a shape estimation $\mathbf{s}_t$, either by predicting from the image I directly or by predicting from the image I as well as the shape $\mathbf{s}_{t-1}$ estimated from the previous layer. These layers are stacked to be a deep network, which predicts the shape stage-by-stage.

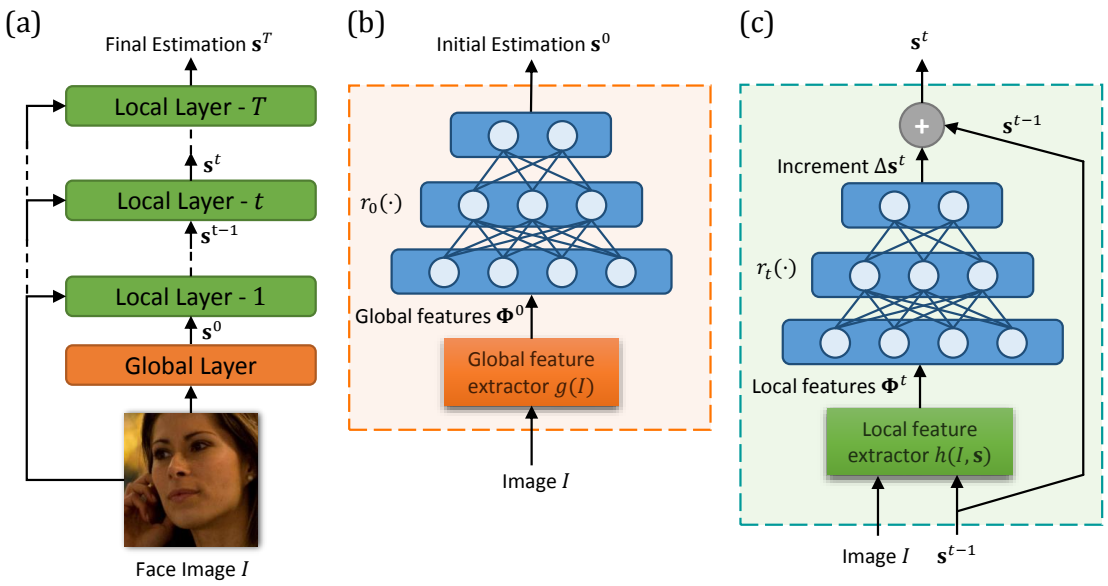

Fig. 1. (a) Overview of the proposed learning architecture. The network takes the face image I as the input and outputs the shape estimation $\mathbf{s}^T$. The global layer estimates an initial shape $\mathbf{s}^0$ and the rest local layers refine the estimation iteratively. (b) The inner structure of the global layer. It consists of a global feature extractor and an MLP regressor, see Section III-A for details. (c) The inner structure of the tth local layer, which consists of a local feature extractor and an MLP regressor, see Section III-B for details.

As depicted in Figure 1.a, the network consists of $T+1$ layers. The first layer is a *global layer*, which consists of a global feature extractor and a global regressor. The rest are T *local layers*, each of which is composed of a local feature extractor and a local regressor.

A. Global layer

The architecture of a global layer is illustrated in Figure 1.b. The global layer predicts the initial shape estimation $\mathbf{s}_0$ from the global image feature. The layer function $\mathrm{GR}(\cdot)$ is:

$$\mathbf{s}_0 = \mathrm{GR}(I) = r_0(\mathbf{\Phi}_0; \theta_0), \quad \mathbf{\Phi}_0 = g(I). \tag{1}$$

Here, $g : I \rightarrow \mathbf{\Phi}_0 \in \Re^{d_0}$ is the global feature extraction function and extracts a d_0-dimensional global features from the image I. r_0 is the regression function with parameter θ_0 and takes $\mathbf{\Phi}_0$ as the input and outputs a $2P$-dimensional shape vector $\mathbf{s}_0$.

The global layer gives a coarse estimation of the shape, which provides a good start for the later local layers.

B. Local layer

Local layers take both the image and the shape as the input. Instead of directly estimating the shape from the image appearance, it refines the shape estimation using local features. The architecture of the tth local layer is depicted in Figure 1.c. A local layer extracts the shape-indexed local feature $\mathbf{\Phi}_t$ and uses it to predict the shape increment $\Delta\mathbf{s}_t$ with the regressor r_t. The increment is added to $\mathbf{s}_{t-1}$ estimated from the previous layer to produce the refined shape estimation $\mathbf{s}_t = \mathbf{s}_{t-1} + \Delta\mathbf{s}_t$. The layer function $\mathrm{LR}^t(\cdot,\cdot)(t = 1 \dots T)$ is defined as:

$$\mathbf{s}_t = \mathrm{LR}^t(I, \mathbf{s}_{t-1}) = \mathbf{s}_{t-1} + r_t(\mathbf{\Phi}_t; \theta_t), \tag{2}$$

where $\mathbf{\Phi}_t = h(I, \mathbf{s}_{t-1})$. r_t is the tth regression function. $h : (I, \mathbf{s}_{t-1}) \rightarrow \mathbf{\Phi}_t \in \Re^{d_t}$ is the local feature extraction function which extracts the local feature by concatenating local image descriptors on all the landmarks:

$$\mathbf{\Phi}_t = [(\mathbf{\Phi}_t^1)^\intercal \ \dots \ (\mathbf{\Phi}_t^P)^\intercal]^\intercal, \tag{3}$$

where $\mathbf{\Phi}_t^p$, $p = 1 \dots P$, is the local descriptor extracted from the patch on the pth landmark in the shape $\mathbf{s}_t$.

Throughout the experiments, we use the HOG descriptor [12] extracted from a down-sampled version of the input image as the global feature and the SIFT [23] as the local descriptor.

The choice of regression functions $\{r_t\}_{t=0,\dots,T}$ is also arbitrary, as long as it is continuous and differentiable *almost everywhere* with respected to its parameters θ_t and its input $\mathbf{s}_{t-1}$ (except $t = 0$). That is to say, regression functions support back-propagation [21]. In our experiments, we use the multi-layered perceptron (MLP) regressor with the ReLU rectifier [30] as the regression function:

$$r_t(\mathbf{\Phi}; \theta_t) = \mathbf{W}_2^t \sigma(\mathbf{W}_1^t \mathbf{\Phi}_t + \mathbf{b}_1^t) + \mathbf{b}_2^t, \tag{4}$$

where $\sigma(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x})$ is the element-wise thresholding function. Regression parameter θ_t is the concatenation of $\mathbf{W}_1^t$, $\mathbf{b}_1^t$, $\mathbf{W}_2^t$ and $\mathbf{b}_2^t$.

IV. Optimization

Denote the training set as $\mathcal{X} = \{(I^{(i)}, \hat{\mathbf{s}}^{(i)})\}_{i=1}^n$ where $I^{(i)}$ is the ith face image and $\hat{\mathbf{s}}^{(i)}$ is the corresponding ground truth shape vector. The network parameters are learned by minimizing the empirical loss over the training set:

$$E(\theta) = \frac{1}{2} \sum_{i=1}^{n} \|\mathrm{DR}^T(I^{(i)}) - \hat{\mathbf{s}}^{(i)}\|_2^2, \tag{5}$$

where θ is the network parameter vector, the concatenation of the layer parameters: $\theta = [\theta_0^\intercal, \dots, \theta_T^\intercal]^\intercal$. $\mathrm{DR}^T(I^{(i)})$

represents the output of the network and is defined recursively by its sub-networks as:

$$\mathrm{DR}^t(I^{(i)}) = \begin{cases} \mathrm{LR}^t(I^{(i)}, \mathrm{DR}^{t-1}(I^{(i)})), & t = 1, \ldots, T; \\ \mathrm{GR}(I^{(i)}), & t = 0. \end{cases} \tag{6}$$

To solve the problem of minimizing $E(\boldsymbol{\theta})$, we first introduce the layer-wise learning approach that is used in the cascaded regression algorithm [43] and empirically show its drawbacks. Then, we propose the joint learning algorithm based on back-propagation.

A. Layer-wise learning

Layer-wise learning finds the parameters of the regressors sequentially, from $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_T$, to approximately minimize the objective function $E(\boldsymbol{\theta})$. The parameter of each regressor is optimized, by fixing the trained parameters of the regressors preceding it and minimizing the difference of its predicted shape from the true shape. Mathematically, the parameter $\boldsymbol{\theta}_0$ of r_0 in the global layer is found as:

$$\boldsymbol{\theta}_0 = \arg\min_{\boldsymbol{\theta}_0} \frac{1}{2n} \sum_{i=1}^{n} \|r_0(\boldsymbol{\phi}_0^{(i)}; \boldsymbol{\theta}_0) - \hat{\mathbf{s}}^{(i)}\|_2^2. \tag{7}$$

The parameter $\boldsymbol{\theta}_t$ of the tth ($t = 1, \ldots, T$) local layer is computed as:

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}_t} \frac{1}{2n} \sum_{i=1}^{n} \|\mathbf{s}_{t-1}^{(i)} + r_t(\boldsymbol{\phi}_t^{(i)}; \boldsymbol{\theta}_t) - \hat{\mathbf{s}}^{(i)}\|_2^2, \tag{8}$$

where $\boldsymbol{\phi}_t^{(i)} = h(I^{(i)}, \mathbf{s}_{t-1}^{(i)})$ and $\mathbf{s}_{t-1}^{(i)}$ are constants given that the parameters of the first t regressors are already found. Both Equations 7 and 8 are MLP regression problems and are solved by the stochastic gradient descent [21].

Layer-wise learning is closely linked with previously proposed cascaded regression-based face alignment algorithms. In particular, SDM [43] takes the linear regressor as $r_t(\cdot)$ and finds its parameter by iteratively solving linear regression problems in the cascade. The layer-wise learning scheme, however, may not result in optimal solution, as the regressor parameters estimation of each regressor does not exploit the later regressors. Empirically, we observe that the first few regressors make greater paces to approach the true shape, i.e., smaller bias of the shape estimation from those regressors, while the later regressors make smaller paces. Importantly, we find that the shape estimation from the first regressors has larger estimation variances. This results in that the variance of the local (shape-indexed) features is also larger. As a consequence, it is harder for the later regressors to make a good shape estimation.

In the following, we propose the joint learning algorithm using the back-propagation to directly optimize the objective function such that the optimization of the parameters of all the stage regressors helps each other. The empirical results show that joint learning is able to make a balanced optimization of the bias and the variance of the shape estimation from the regressors: both the bias and the variance gradually decrease from the early regressors to the later ones. This results in that the variances of the local features, the inputs of the later stage regressors, are small and thus that the learned later regressors are more capable of yielding a better alignment performance. Consequently, joint learning yields a better shape estimation. Figure 2 illustrates the comparison of the bias and the variance of the prediction from each regressor optimized with layer-wise learning and joint learning.

B. Joint learning

We adopt the gradient descent method to jointly estimate the regression parameters by minimizing the global error function $E(\boldsymbol{\theta})$. We apply the back-propagation algorithm [33] to efficiently evaluate the derivatives of the error function with respect to the parameters and the inputs.

1) Partial derivatives of the loss function: Starting from the loss function, the error differentials is back-propagated layer-by-layer from the top to the bottom. On the top, the partial derivatives are calculated by $\partial E/\partial \mathbf{s}_T = \mathbf{s}_T - \hat{\mathbf{s}}$. Then, the derivatives of the local layers and global layer are calculated as follows.

The derivatives of local layers. The partial derivatives of the error function $E(\boldsymbol{\theta})$ with respect to regression parameters $\boldsymbol{\theta}_t$ and the input $\mathbf{s}_{t-1}$ are calculated using the backward recurrence as:

$$\frac{\partial E}{\partial \boldsymbol{\theta}_t} = \frac{\partial E}{\partial \mathbf{s}_t} \frac{\partial \mathrm{LR}^t}{\partial \boldsymbol{\theta}_t}, \tag{9}$$

$$\frac{\partial E}{\partial \mathbf{s}_{t-1}} = \frac{\partial E}{\partial \mathbf{s}_t} \frac{\partial \mathrm{LR}^t}{\partial \mathbf{s}_{t-1}}. \tag{10}$$

According to Equation 2, the partial derivative with respect to the regression parameter $\boldsymbol{\theta}_t$ is calculated as:

$$\frac{\partial E}{\partial \boldsymbol{\theta}_t} = \frac{\partial E}{\partial \mathbf{s}_t} \frac{\partial r_t}{\partial \boldsymbol{\theta}_t}. \tag{11}$$

In Equation 10, $\frac{\partial \mathrm{LR}^t}{\partial \mathbf{s}_{t-1}}$ is computed by taking partial derivatives with respect to $\mathbf{s}_{t-1}$ on both sides of Equation 2:

$$\frac{\partial \mathrm{LR}^t}{\partial \mathbf{s}_{t-1}} = \mathbf{I} + \frac{\partial r_t}{\partial \boldsymbol{\phi}_t} \frac{\partial h}{\partial \mathbf{s}_{t-1}}, \tag{12}$$

where $\mathbf{I} \in \Re^{2P \times 2P}$ is the identity matrix, and $\frac{\partial h}{\partial \mathbf{s}_{t-1}}$ is the partial derivatives of the local feature extractor with respect to its input shape vector $\mathbf{s}_{t-1}$. Both $\frac{\partial r_t}{\partial \boldsymbol{\theta}_t}$ in Equation 11 and $\frac{\partial r_t}{\partial \boldsymbol{\phi}_t}$ in Equation 12 are computed by doing back-propagation in the MLP [21].

The derivatives of h. The Jacobian matrix of the feature extraction function $h(I, \mathbf{s})$ with respect to the shape vector $\mathbf{s}$ in Equation 12 is denoted by

$$\boldsymbol{\psi} = \frac{\partial h}{\partial \mathbf{s}} \in \Re^{d \times 2P}, \tag{13}$$

where $d = \sum_{i=1}^{P} d_p$ is the number of dimensions of the local feature. d_p is the number of dimensions in the pth local descriptor. For simplicity, we drop the subscript t.

According to Equation 3, h involves local descriptor extraction operations. Its partial derivatives cannot be calculated analytically. We numerically approximate $\boldsymbol{\psi}$ by computing the second-order approximation:

$$\psi_{(ij)} = \frac{\partial h_{(i)}}{\partial s_{(j)}} \approx \frac{h(I, \mathbf{s}^{j+})_{(i)} - h(I, \mathbf{s}^{j-})_{(i)}}{2\epsilon}, \tag{14}$$

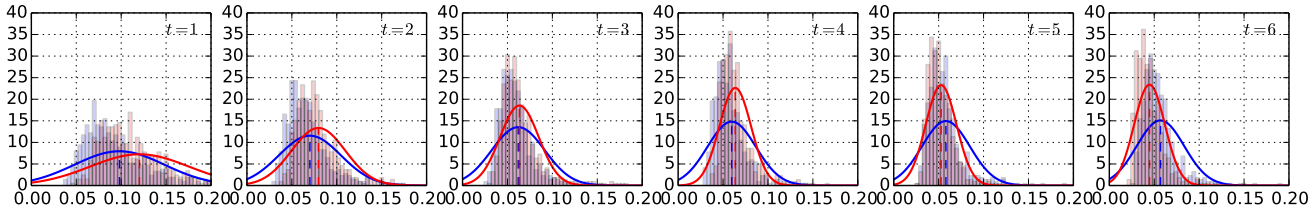

Fig. 2. The bias and variance comparison of the shape estimation error at each stage. The error is fitted into a Gaussian distribution, in which the mean corresponds to the bias and the width corresponds to the variance. Layer-wise learning (in blue color) over-strongly reduces the bias early and results in a larger variance, which makes later stages weak. Joint learning (in red color) balances between bias and variance and makes them gradually and simultaneously decrease, resulting in lower error eventually. The bias and variance are estimated on the 300-W Common test set and plotted as normal distributions. The horizontal axis represents the normalized shape estimation error (Section V-A).

where $\psi_{(ij)}$ denotes the element of $\boldsymbol{\psi}$ on the ith row and jth column. $h_{(i)}$ is the ith dimension in the output vector of h and $s_{(j)}$ is jth dimension in $\mathbf{s}$. $\mathbf{s}^{j+}$ and $\mathbf{s}^{j-}$ are vectors that are equal to $\mathbf{s}$ except the jth dimension: $s^{j+}_{(j)} = s_{(j)} + \epsilon$, and $s^{j-}_{(j)} = s_{(j)} - \epsilon$. ϵ is a small value that corresponds to a few pixels in the image.

The function h concatenates local descriptors $\boldsymbol{\phi}_{p,p=1...P}$ according to Equation 3. Each local descriptor $\boldsymbol{\phi}_p$ is determined by the input image I and landmark position (x_p, y_p), which are two elements in the shape vector $\mathbf{s}$. Therefore, the Jacobian matrix $\boldsymbol{\psi}$ is a block-sparse matrix:

$$\boldsymbol{\psi} = \begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \ddots & \\ & & \boldsymbol{\psi}_P \end{bmatrix}, \tag{15}$$

where each block $\boldsymbol{\psi}_p = [\boldsymbol{\psi}_{px}\ \boldsymbol{\psi}_{py}] \in \Re^{d_p \times 2}$. $\boldsymbol{\psi}_{px}, \boldsymbol{\psi}_{py} \in \Re^{d_p \times 1}$ are respectively partial derivatives of the pth local descriptor with respect to its x and y coordinates, calculated as:

$$\boldsymbol{\psi}_{px} = \frac{h_p(I, x_p + \epsilon, y_p) - h_p(I, x_p - \epsilon, y_p)}{2\epsilon}, \tag{16}$$

$$\boldsymbol{\psi}_{py} = \frac{h_p(I, x_p, y_p + \epsilon) - h_p(I, x_p, y_p - \epsilon)}{2\epsilon}. \tag{17}$$

Here, $h_p(I, x_p, y_p)$ extracts the local descriptor from patch centered on x_p, y_p. Throughout our experiments, we use the same extraction function h_p for all landmarks in all local layers. Utilizing the block-sparse property the computation of $\boldsymbol{\psi}$ requires computing local descriptors on $4P$ number of locations, which is computationally cheap.

The derivatives of the global layer. For the global layer, the partial derivative with respect to $\boldsymbol{\theta}_0$ is also computed by Equation 11. Since the global layer is the first layer in the network, there is no need to compute the partial derivative with respect to its input.

2) Initialization and Regularization: We apply some regularization strategies in order to successfully train the network. To obtain a good initialization, we pre-train the network using the layer-wise learning algorithm. After pre-training, we jointly optimize the network. The overall process of the network training is summarized in Algorithm 1.

Another important strategy for regularization is the dropout algorithm [15], which we find critical for avoiding over-fitting. We apply the dropout strategy to the hidden layers of the MLP regressors, adding a dropout layer after the feature extractor layer, in all the regressors in the network, during both layer-wise learning initialization and joint learning. The dropout rate is fixed to $p = 0.5$ throughout our experiments.

Algorithm 1 The overall training process of deep regression

1: **Input:** The training set $\mathcal{X} = \{(I^{(i)}, \hat{\mathbf{s}}^{(i)})\}_{i=1}^{n}$, the number of local layers T
2: **Output:** The learned network parameter $\boldsymbol{\theta}$
3: Calculate the parameter $\boldsymbol{\theta}_0$ for r_0 in GR by Eq. 7
4: $\mathbf{s}_0^{(i)} = r_0(I^{(i)})$
5: **for** $t = 1$ **to** T **do**
6: Calculate the parameters $\boldsymbol{\theta}_t$ for r_t in LR^t by Eq. 8
7: Update $\mathbf{s}_t$ by: $\mathbf{s}_t = \mathbf{s}_{t-1} + r_t(h(I^{(i)}, \mathbf{s}_{t-1}); \boldsymbol{\theta}_t)$
8: **end for**
9: Perform joint optimization for all layers, according to Section IV-B
10: Output the network parameter $\boldsymbol{\theta}$

V. Experiments

In this section, we evaluate the efficacy of the Deep Regression as well as its variants. The experiments are in two parts. The first part evaluates our methods on public datasets and performs comparisons with previous works. The second part presents some interesting phenomenons, demonstrating the superiorities of joint learning over layer-wise learning both quantitatively and qualitatively.

A. Datasets and evaluation metric

We evaluate our approach on the commonly-used public datasets, including the LFPW dataset [4], the Helen dataset [20] and the 300-W dataset [34].

The *LFPW* dataset is annotated by 35 landmarks. Following [4], 29 landmarks are used for training and testing. This dataset provides URLs only, some of which are no longer valid, and we are only able to use 717 of the 1100 images for training and 249 of the 300 images for testing. As the preprocessing step, faces are detected by a face detector [32].

The *Helen* dataset is annotated by 194 landmarks. It contains 2000 images for training and 330 images for testing. We use the face bounding boxes provided by the dataset.

The *300-W* dataset is created from several re-annotated datasets including the LFPW [4], AFW [50], Helen [20]

and XM2VTS [28] datasets. The number of landmarks is 68 and face bounding boxes are provided in the dataset. Since the official testing set in 300-W is not publicly available, we follow [32] and build the training set using the AFW dataset, the training set in LFPW and the training set in Helen, resulting in 3148 training images in total. The testing set consists of the IBUG dataset, the testing set in LFPW and the testing set in Helen, with 689 testing images altogether. Following [32], we evaluate the performance on 1) all the images in the testing set, called the Fullset 2) the testing sets in Helen and LFPW, called the Common Subset and 3) the IBUG dataset, called the Challenging Subset.

Following [4], we evaluate the performance by the average landmark error normalized by the inter-pupil distance:

$$\text{error} = \frac{1}{N}\sum_{i=1}^{N}\frac{\frac{1}{P}\sum_{p=1}^{P}\sqrt{(x_p^{(i)}-\hat{x}_p^{(i)})^2+(y_p^{(i)}-\hat{y}_p^{(i)})^2}}{d_{\text{pupils}}^{(i)}}, \tag{18}$$

where $\hat{x}_p^{(i)}, \hat{y}_p^{(i)}$ are ground truth coordinates of the pth landmark in the ith sample, $d_{\text{pupils}}^{(i)}$ is the inter-pupil distance of the ith sample. For the Helen and 300-W datasets, pupils are not annotated. They are replaced by the mean landmarks of the landmarks around each eye.

B. Implementation details

For the global feature extractor $g(I)$, we use the HOG [12] descriptor computed on images down-sampled to sizes of 64×64. The block size, block stride, cell size and number of bins are chosen as 16×16, 8×8, 8×8 and 9 respectively. This results in feature vectors with 1764 dimensions.

We use SIFT [23] for the local feature extractor $h(I, \mathbf{s})$. However, the numerical approximation of $\boldsymbol{\psi}$ requires extracting multiple SIFT descriptors in every training iteration, thus very time-consuming. A solution is to store dense SIFT descriptors on all images, but this requires too much memory. We propose to use a modified version of SIFT which can be computed on-line very efficiently. For each image, the responses for 8 orientation bins on all locations are computed as 8 response maps. Then, the spatial bin interpolation operation in SIFT is approximated by blurring the response maps using a Gaussian kernel, which is inspired by the DAISY descriptor [39]. Finally, the response maps are converted to integral maps, where SIFT histograms can be computed via only a few addition and subtraction operations [31]. The integral maps of all training images are stored in memory throughout the training process, so that descriptors can be efficiently extracted on-line.

The number of local layers is set to $T = 5$. The SIFT patch sizes for local layers are to $64, 48, 32, 16, 16$ respectively. We augment training samples by rotation, translation and horizontal flipping (landmarks are re-indexed). A validation set of 200 samples is split out from the training set for monitoring the training process.

The ϵ in Equation 14 is set to $\frac{2}{w_I}$ throughout our experiments, where w_I is the image width in pixels that is used for shape normalization. Other small values have also been tried but have no significant impact on performance. Network parameters are updated by the stochastic gradient descent [21] with the momentum [38] set to 0.9. The mini-batch size set to 128. During training, the learning rate is set to 10^{-2} at first and decreased with a factor of 0.1 when the validation error stops to decrease [18]. The training process is terminated when the learning rate is less than 10^{-4}.

Our approach is implemented with the Torch7 framework [7], with some custom C++ implementations, including the computation of SIFT and HOG. Experiments are carried out on a workstation with a 2.50 GHz Intel(R) Xeon(R) E5-2609 CPU and 64GB RAM. The time taken by the training process varies depending on the number of landmarks. It takes about 5 hours on the LFPW dataset, 2 days on 300-W, and 4 days on Helen. The optimization converges after around 50k training iterations. The testing process is very efficient. It takes ∼20ms on LFPW (29 landmarks), ∼30ms on 300-W (68 landmarks), and ∼50ms on Helen (194 landmarks). Both the training and the testing processes are expected to be further accelerated with a GPU-accelerated implementation, which is not covered in this paper.

C. Method variants

We term our approach as DR, our approach (the network structure) with layer-wise learning as DR-Seq. In addition, we propose a variant of the network, termed as DR-SDM, in which 1) the MLP regressors are replaced by linear regressors; 2) the global layer is dropped and the initial shape estimation $\mathbf{s}_0$ is given by the mean shape $\bar{\mathbf{s}}$ calculated from the training set, as adopted in the cascaded regression [43], [5], [32]. We set the number of stages $T = 5$, adopt linear regressors and use SIFT as the local descriptors, consequently DR-SDM behaves the same as SDM [43] during prediction. However, the regression parameters of DR-SDM and SDM are learned in different ways. In DR-SDM, parameters are firstly initialized by layer-wise learning (Section IV-A) and then optimized by joint learning (Section IV-B). In SDM, parameters are learned by layer-wise learning.

D. Benchmark results

First, we compare the results of DR with two baseline algorithms: DR-Seq and DR-SDM. The results are listed in Table I. As can be seen, DR outperforms both DR-Seq and DR-SDM on all datasets. The superiority of DR over DR-Seq stems from joint optimization, which is able to balance the biases and the variances of all the regressors. The superiority over DR-SDM is because the global regressor is helpful to generate a robust initialization, and the MLP has better regression capability than linear regression.

Second, in comparison with the closely-related regression algorithm, supervised descent method (SDM) [43], DR-SDM performs better, which also demonstrate that joint optimization helps improve performance. In particular, the relative performance gains of DR over SDM are 4.6% on the LFPW dataset, 13.0% on the Helen dataset, 18.9%, 22.3% and 13.6% on the 300-W Fullset, Common, and challenging datasets.

Third, we report the comparison results with other state-of-the-art algorithms, including the algorithm using a consensus

TABLE I
RESULTS ON THE LFPW, HELEN AND 300-W DATASETS, MEASURED BY THE SHAPE ERROR NORMALIZED BY THE INTER-PUPIL DISTANCE ($\times 10^{-2}$). THE TOP 2 RESULTS FOR EACH DATASET ARE MARKED IN BOLD. THE SDM RESULTS ON THE HELEN AND 300-W DATASETS ARE REPORTED BY [32] THE CFAN [46], DRMF [2] AND RCPR [5] RESULTS ARE PRODUCED BY THE CODE/BINARY AUTHOR HAVE RELEASED.

LFPW		**Helen**		**300-W**			
Method	Error	Method	Error	Method	Fullset	Common	Challenging
CoE [4]	3.90	-	-	-	-	-	-
-	-	STASM [29]	11.1	-	-	-	-
-	-	CompASM [20]	9.10	-	-	-	-
ESR [6]	3.47	ESR [6]	5.7	ESR [6]	7.58	5.28	17.00
RCPR [5]	3.50	RCPR [5]	6.5	RCPR [5]	8.35	6.18	17.26
-	-	-	-	DRMF [2]	9.22	6.65	19.79
SDM [43]	3.47	SDM [43]	5.85	SDM [43]	7.52	5.60	15.40
ERT [17]	3.80	ERT [17]	4.9	ERT [17]	6.4	-	-
-	-	-	-	CFAN [46]	7.69	5.50	16.78
OSRD [42]	**3.19**	-	-	-	-	-	-
LBF [32]	3.35	LBF [32]	5.41	LBF [32]	6.32	4.95	**11.98**
TCDCN [47]	-	TCDCN [47]	**4.63**	TCDCN [47]	**5.54**	4.80	**8.60**
DR-Seq	3.90[2]	DR-Seq	6.47	DR-Seq	7.82	5.44	17.6
DR-SDM	3.40[1]	DR-SDM	5.40	DR-SDM	6.57	**4.67**	14.30
DR[2]	**3.31**	DR	**5.09**	DR	**6.10**	**4.35**	13.3

[1] Compared with our earlier pre-print [36], this result is slightly lower because we have re-run the training procedure, and get a slightly better result. Our optimization algorithm is stochastic so results may vary among different runs.

[2] Notice that both DR-SDM and DR have different structures with those in [36], therefore the results are also different.

of exemplars (CoE) [4], explicit shape regression (ESR) [6], extended active shape model (STASM) [29], component-based active shape model (CompASM) [20], discriminative response map fitting (DRMF) [2], robust cascaded pose regression (RCPR) [5], ensemble of regression trees (ERT) [17], coarse-to-fine auto-encoder network (CFAN) [46], occlusion handling stagewise relational dictionary (OSRD) [42], local binary features (LBF) [32], and task-contrained deep convolutional network (TCDCN) [47].

As shown in Table I, our results are consistently very competitive, some surpassing the state-of-the-art. Specifically, our approach performs the best in 300-W Fullset and 300-W Common Subset, even outperforming TCDCN [47] which is based on convolutional neural network and uses extra labeled data. On 300-W Challenging Subset, the result is still comparable, but lower than LBF [32] and TCDCN [47]. The reason is that both LBF and TCDCN performs an extra feature learning step that is essential for good performance. We adopt SIFT for simplicity, but more sophisticated features, e.g., LBF, can also benefit our approach.

Our approach performs the second best and slightly poorer than OSRD [42] in LFPW. The performance of our approach is 3.31 and the relative performance loss is only 3.7% compared with OSRD. We found that the failure cases from our approach in LFPW are the face images with occlusions or large rotations, which OSRD focuses on solving. In the future, we plan to extend our approach to address multi-view and partially-occluded faces. On Helen, our approach is also competitive. It is a little poorer than ERT [17], which, however, performs very unsatisfactorily in both LFPW and 300-W.

Compared with the state-of-the-art approach LBF [32] that exploits complicated features, our approach gets the relative improvements: 1.19% on the LFPW dataset, 5.91% on the Helen dataset, 3.48% on the 300-W Fullset, 12.12% on the 300-W Common dataset, and the relative loss: 1.1% on the 300-W challenging dataset. Given the performances on these datasets are already very high, the improvements from our approach are actually significant.

On both Helen and 300-W, TCDCN [47] achieves excellent results and outperforms our approach on most subsets. However, TCDCN [47] is based on relatively expensive convolutional neural networks, which requires much more memory to store and more data to train. Moreover, TCDCN uses an external dataset, which is annotated with landmarks and attribute labels, to learn an initial model. With more training data, the performance of our model is also expected to further increase.

Last, in Figures 3, 4 and 5, we show some example images together with their shapes predicted by DR on the LFPW, Helen and 300W datasets respectively. We also show the comparison with some other state-of-the-art methods in Figure 6.

E. Comparisons with SDM and CFAN

In this section, we give some detailed comparisons and analysis with two closely related methods, namely SDM [43] and CFAN [46], particularly to demonstrate the effectiveness of the proposed joint learning scheme.

1) Comparisons with SDM: The Supervised Descent Method (SDM [43]) uses the SIFT descriptors as local descriptors and do regression with simple linear regressors. As have shown in Section V-C, the variant DR-SDM shares the same model as the SDM [43] The difference lies in the model learning approach, where SDM is optimized by layer-wise learning while DR-SDM is optimized by joint learning.

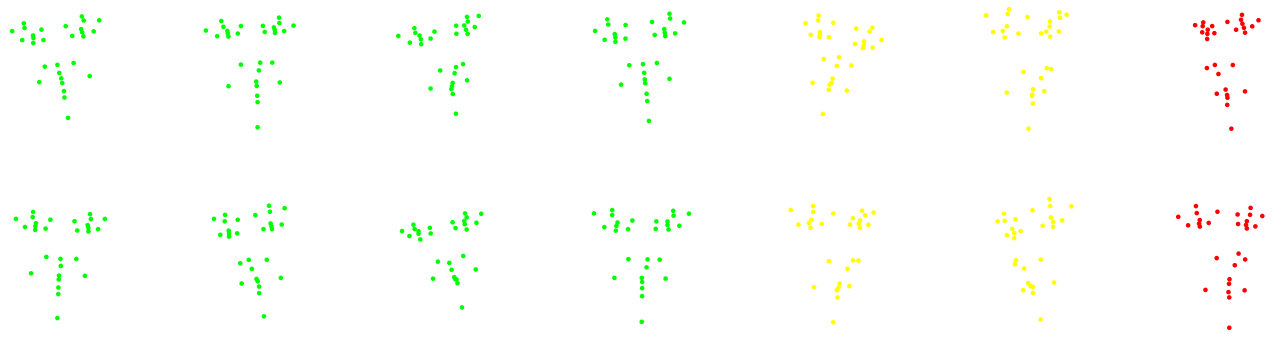

Fig. 3. Example results on the LFPW dataset. Samples with green, yellow and red landmark colors are respectively simple cases, hard but successfully predicted cases and failure cases. The two samples in the last column are fail cases mainly due to that the chin landmarks are not correctly detected.

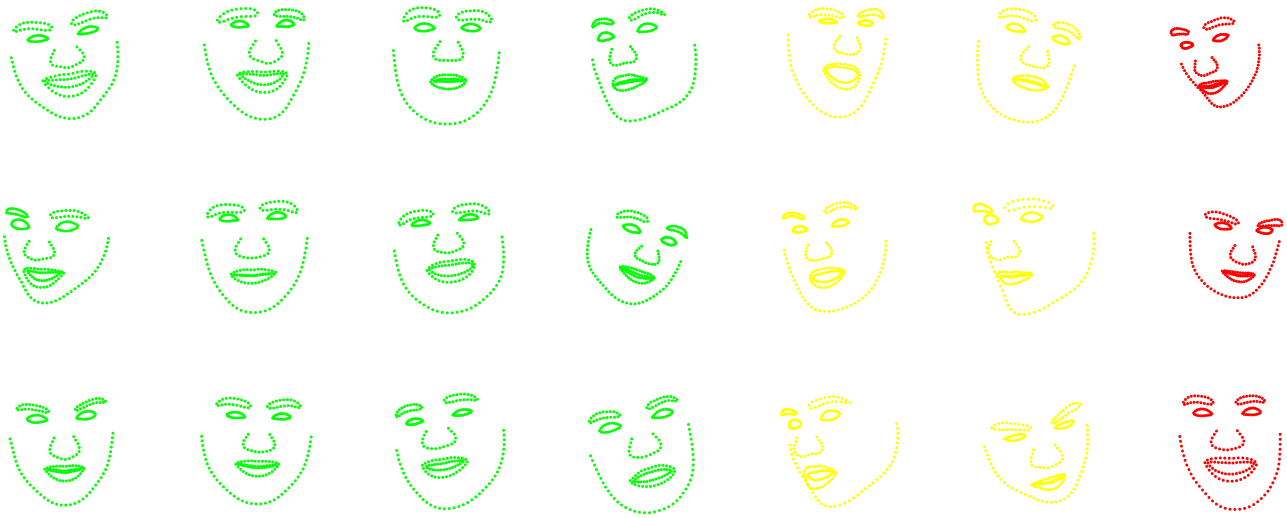

Fig. 4. Example results on the Helen dataset. See Figure 3 for the meanings of the colors.

From Table I we can see that DR-SDM performs significantly better than SDM on all datasets we compare, especially on the challenging 300-W dataset where a larger performance gain can be observed. Apart from the error reported in Table I, we give more extensive comparisons with SDM, focusing on the efficacy of joint learning. To rule out the effects of other algorithm details, we implement SDM with the number of stages $T = 5$ and the SIFT patch sizes are set to $68, 48, 32, 16, 16$. We also used the modified SIFT descriptors as described in Section V-B. In this way, our implementation of SDM behaves *exactly the same* as DR-SDM during prediction and the only difference is the learning method.

In Figure 7 we report the cumulative distribution function (CDF) with respect to the normalized root mean squared error (NRMSE). From it, we can see that DR-SDM surpass SDM on the full shape error. Furthermore, we compare the errors on the outer points and inner points respectively. The outer points and inner points are also illustrated in Figure 7. It can be observed that generally outer points are much harder to predict than the inner points. The reasons could be that outer points tend to be more flexible with respect to the face shape and face pose. Our DR-SDM performs better on both the outer points and the inner points. Besides, on the outer points, our approach achieves slightly larger improvements compared to the inner points.

We also demonstrate that DR-SDM better handles shapes with larger variations. In Figure 8 we plot the trend of prediction errors with respect to the shape variations (normalized differences from the mean shape). Naturally, the prediction error grows with shape variations for both SDM and DR-SDM. DR-SDM constantly surpass SDM on all shape variations, and the gap widens when the shape variation becomes larger. This indicates that DR-SDM better handles the profiles, pose variations as well as expression changes. Several representative shapes corresponding to their shape variations are also presented in Figure 8.

2) Comparisons with CFAN: We also compare our deep regression approach with a representative deep learning

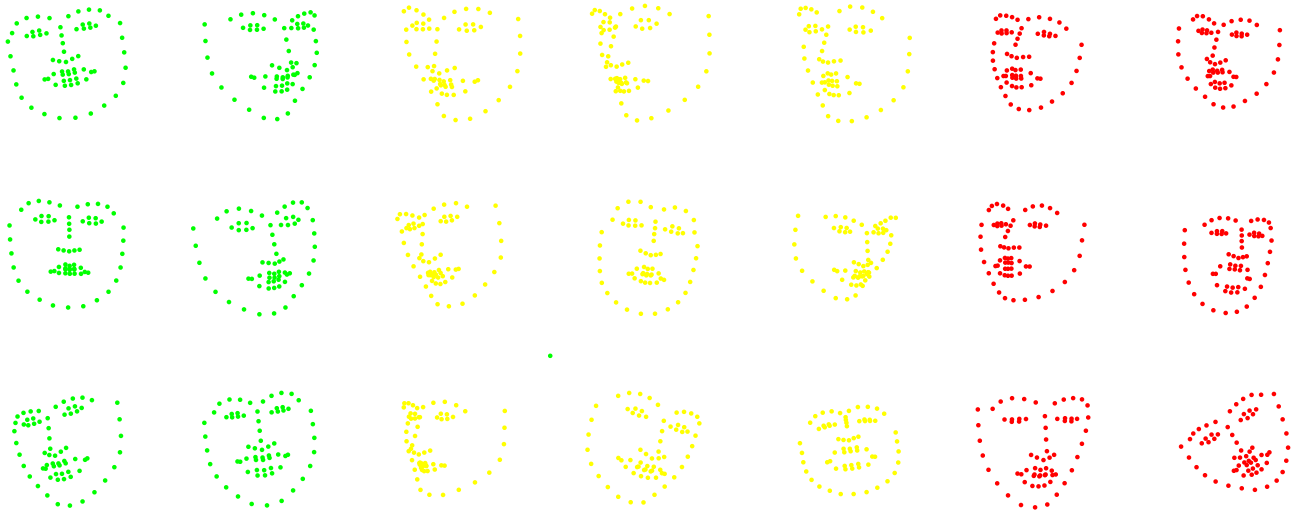

Fig. 5. Example results on the LFPW dataset. See Figure 3 for the meanings of the colors.

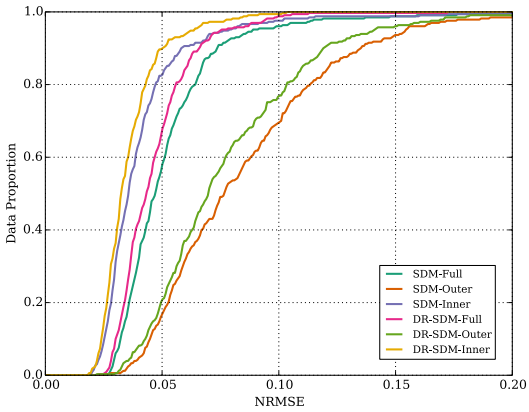

Fig. 7. Cumulative error distribution curves with respect to NRMSE for SDM and DR-SDM, evaluated on the full shape, inner points and the outer points respectively. The full shape (Full) represents the whole shape with 68 annotated landmarks; the outer shape (Outer) represents the shape formed by the 17 landmarks that lie on the face boundary; the inner shape (Inner) represents the shape formed by the rest 51 landmarks.

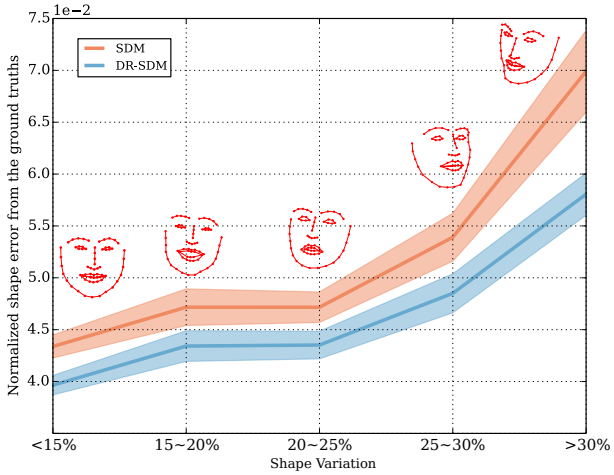

Fig. 8. Means and deviations of shape prediction errors, with respect to the ground truth shape variations. Shape variations are measured by the normalized shape differences from the mean ground truth shapes. Shape variations are divided into 5 ranges and for each range the mean and deviation of the predictions are calculated. For each range of the shape variations, we show a representative shape.

based method, a coarse-to-fine auto-encoder network approach (CFAN) [46], in which each stage is a stacked auto-encoder network. The CFAN approach is similar to our Deep Regression in that each cascade stage is a multi-layered regressor. For each stage, CFAN uses a deep network with two hidden layers and they are initialized by a pre-trained stacked auto-encoder. In DR, each regression stage contains one hidden layer. The key difference lies in that CFAN uses the layer-wise learning scheme while DR adopts joint learning.

For fair comparisons with CFAN, we train our DR on the training set of Helen, the training set of LFPW and the AFW dataset. The annotations are from the 300-W dataset. Following CFAN [46], we evaluate the normalized root mean squared error (NRMSE) on the 66 facial points among the 68 points annotations. The CDFs are plotted and compared in Figure 9. As can be seen, our approach outperforms CFAN significantly on both datasets. Note that our DR has a similar architecture with CFAN, but the way of training differs. This shows the superiority of the joint learning over layer-wise learning.

F. Discussions

In this section, we empirically compare the layer-wise learning and the joint learning, and show the advantages of joint learning both quantitatively and quantitatively.

1) Bias and variance: In contrast to the heuristic shrinkage strategy that is adopted in cascaded regression learning [17],

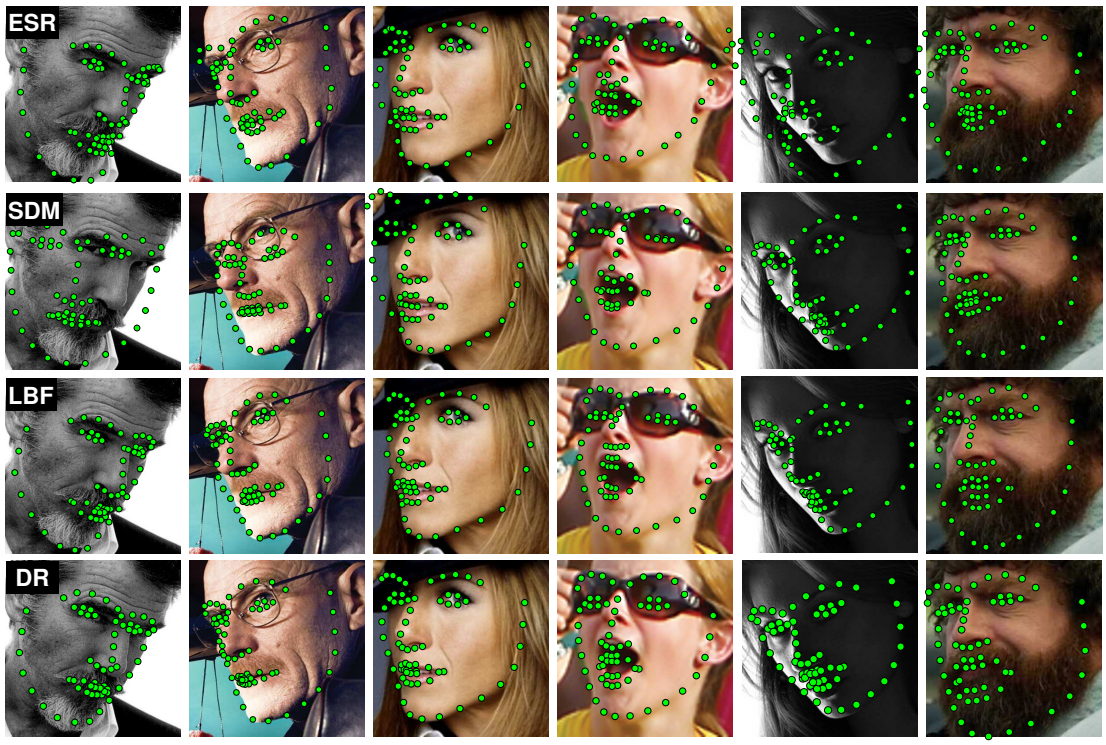

Fig. 6. Comparisons between ESR [6], SDM [43], LBF [32], and the proposed DR on the Challenging Subset of 300-W.

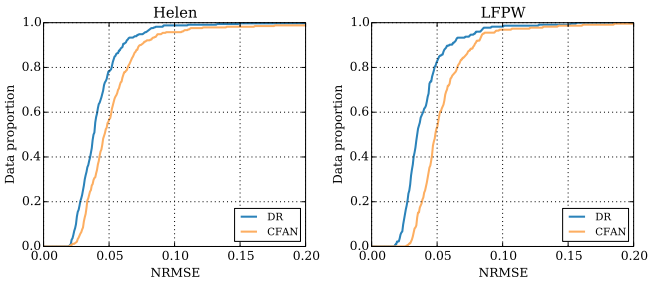

Fig. 9. Cumulative distribution functions with respect to NRMSE evaluated on the Helen (left) and LFPW (right) datasets. Our approach DR performs better than CFAN [46] on both the datasets.

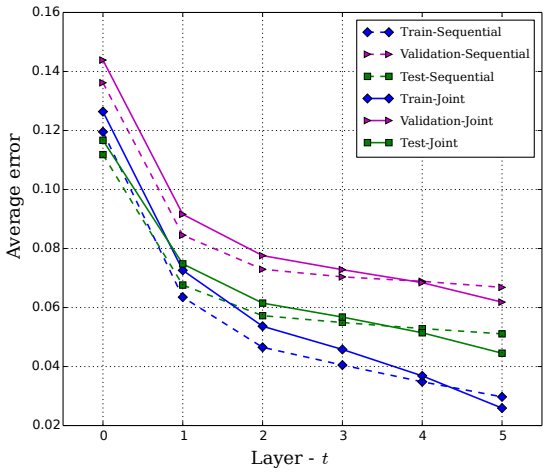

Fig. 10. The average estimation errors of all layers in the network, calculated on the training, validation and testing sets of the 300-W Common subset. The dashed lines represent the sequentially trained layers and the solid lines represent jointly trained layers.

the joint learning strategy implicitly create the balance between the bias and the variances. Figure 10 plots the estimation errors of all the stages on the training and testing sets. One can see that layer-wise learning tends to result in strong early stages which eliminate most of the error. The later stages, however, are weaker. Joint learning mitigates this and the estimation gradually and evenly approaches the ground truth, resulting in a flattened error curve and better estimation eventually. Furthermore, as shown in Figure 2, joint learning balances between the bias and the variance and makes them gradually and simultaneously decrease, while in layer-wise learning the variance decreases much slower.

2) Path analysis: In layer-wise learning, the regressors are learned to minimize the difference between the current shape estimation $\mathbf{s}_t$ and the true shape $\hat{\mathbf{s}}$. In joint learning, the intermediate shape estimations, $\mathbf{s}_{t,t=1...T-1}$, are not directly optimized to minimize the difference. This allows the shape estimation $\mathbf{s}_t$ to go from $\mathbf{s}_0$ to $\mathbf{s}_T$ in a better way that brings some advantages over layer-wise learning.

For demonstration, we investigate two face boundary landmarks, the #1 and the #17, in the 300-W Common testset. As illustrated in Figure 11.a, the two landmarks are respectively on the left and right sides of the face outline. To visualize landmark predictions, we calcualte the x, y differences between groundtruth positions, respectively $\Delta x_p^t = x_p^t - \hat{x}_p$ and $\Delta y_p^t = y_p^t - \hat{y}_p$, estimate Gaussian distributions of the differences, and plot the distributions as ellipses in Figure 11.b. Paths that connect the centers of these distributions reflect the

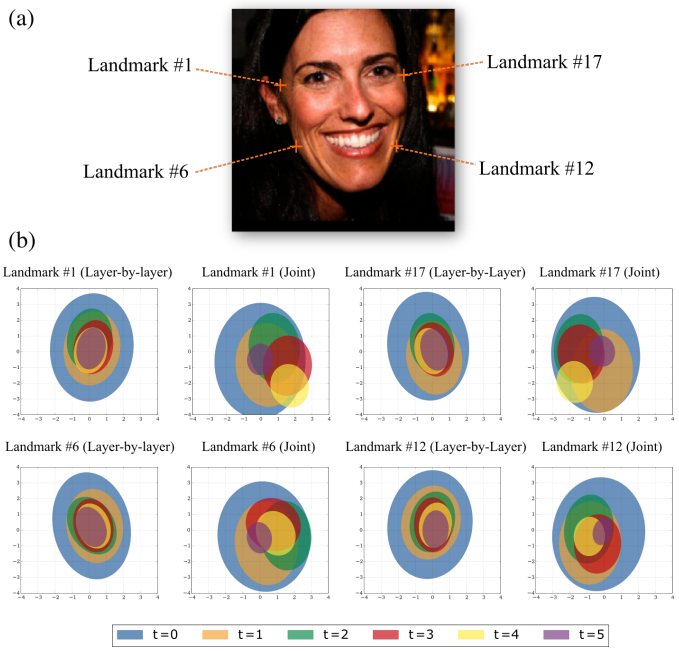

Fig. 11. (a) The illustration of the #1, #17, #6 and #12 landmarks in the 300-W Common subset. (b) Gaussian distributions of relative landmark positions predicted by layers $t = 0$ to 5, by both the layer-wise learning and the joint learning. The x and y coordinates are measured in pixels in the image.

way shape estimations $\mathbf{s}_{t,t=1...T}$ goes during prediction.

In layer-wise learning, regressors at all the stages are learned to make landmarks go towards true positions. As can be seen in the first and the third columns of Figure 11.b, all distributions centers are close to the ground truths. In contrary, the joint learning only encourages the last regressor to produce landmarks near true positions. The intermediate layers may make landmarks deviated from true positions. Seen from the second and the fourth columns of Figure 11.b, the landmarks produced at stages $t = 2, 3, 4$ are away from the ground truths, going toward the face interior.

We argue that the path produced by joint learning is a better one. Compared with layer-wise learning, the landmarks at intermediate stages are "pulled" towards the face interior. Consequently, more landmarks are brought to the interior, rather than falling out the face boundary to the background. Local features of such landmarks are less affected by cluttered backgrounds, thus better for later regressors. Therefore, compared with layer-wise learning, joint learning finds a better path for regression.

VI. Conclusion

In this paper, we present a deep regression approach to face alignment. We adopt the back-propagation with the dropout strategy to jointly optimize the regression parameters of a deep network, a sequence of one global layer and multi-stage local layers. The benefit of joint optimization lies in that the resulting regressors gradually and simultaneously decrease the bias and the variance of each shape estimator, yielding a superior shape predictor over the layer-wise learning algorithm as done in cascaded regression. Experimental results demonstrate the powerfulness of the proposed approach. Possible future work includes replacing hand-crafted SIFT/HOG features with richer deep features [41] and applying the joint optimization scheme to other algorithms, *e.g.* [3].

Acknowledgment

This work was primarily supported by National Natural Science Foundation of China (NSFC) (No. 61222308, No. 61573160 and No. 61503145), and Open Project Program of the State Key Laboratory of Digital Publishing Technology (No. F2016001).

References

[1] J. Alabort-i-Medina and S. Zafeiriou. Bayesian active appearance models. In *Proc. CVPR*, 2014.
[2] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic. Robust discriminative response map fitting with constrained local models. In *Proc. CVPR*, 2013.
[3] X. Bai, S. Bai, Z. Zhu, and L. J. Latecki. 3d shape matching via two layer coding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(12):2361–2373, 2015.
[4] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. In *Proc. CVPR*, 2011.
[5] X. P. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *Proc. ICCV*, 2013.
[6] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. *Int. J. Computer Vision*, 107(2):177–190, 2014.
[7] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.
[8] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *Trans. Pattern Anal. Mach. Intell.*, 23(6):681–685, 2001.
[9] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.
[10] D. Cristinacce and T. F. Cootes. Feature detection and tracking with constrained local models. In *Proc. BMVC*, 2006.
[11] D. Cristinacce and T. F. Cootes. Boosted regression active shape models. In *Proc. BMVC*, 2007.
[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005.
[13] L. Ding and A. M. Martínez. Precise detailed detection of faces and facial features. In *Proc. CVPR*, 2008.
[14] P. Dollár, P. Welinder, and P. Perona. Cascaded pose regression. In *Proc. CVPR*, 2010.
[15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.
[16] V. G. Kaburlasos, S. E. Papadakis, and G. A. Papakostas. Lattice computing extension of the FAM neural classifier for human facial expression recognition. *Trans. Neural Netw. Learning Syst.*, 24(10):1526–1538, 2013.
[17] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proc. CVPR*, 2014.
[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012.
[19] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: a convolutional neural-network approach. *Trans. Neural Networks*, 8(1):98–113, 1997.
[20] V. Le, J. Brandt, Z. Lin, L. D. Bourdev, and T. S. Huang. Interactive facial feature localization. In *Proc. ECCV*, 2012.
[21] Y. LeCun, L. Bottou, G. B. Orr, and K. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade - Second Edition*. 2012.
[22] L. Liang, R. Xiao, F. Wen, and J. Sun. Face alignment via component-based discriminative search. In *Proc. ECCV*, 2008.
[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision*, 60(2):91–110, 2004.
[24] X. Lu and A. K. Jain. Deformation modeling for robust 3d face matching. *Trans. Pattern Anal. Mach. Intell.*, 30(8):1346–1357, 2008.
[25] A. Martinez and S. Du. A model of the perception of facial expressions of emotion by humans: Research overview and perspectives. *J. Mach. Learning Research*, 13:1589–1608, 2012.
[26] B. Martínez, M. F. Valstar, X. Binefa, and M. Pantic. Local evidence aggregation for regression-based facial point detection. *Trans. Pattern Anal. Mach. Intell.*, 35(5):1149–1163, 2013.

[27] I. A. Matthews and S. Baker. Active appearance models revisited. *Int. J. Computer Vision*, 60(2):135–164, 2004.
[28] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. Xm2vtsdb: The extended m2vts database. In *Proc. Int. Conf. Audio and Video-based Biometric Person Authentication*, 1999.
[29] S. Milborrow and F. Nicolls. Locating facial features with an extended active shape model. In *Proc. ECCV*, 2008.
[30] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, 2010.
[31] F. M. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Proc. CVPR*, 2005.
[32] S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 FPS via regressing local binary features. In *Proc. CVPR*, 2014.
[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning representations by back-propagating errors*. MIT Press, Cambridge, MA, USA, 1988.
[34] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. A semi-automatic methodology for facial landmark annotation. In *Proc. CVPR*, 2013.
[35] J. M. Saragih, S. Lucey, and J. F. Cohn. Deformable model fitting by regularized landmark mean-shift. *Int. J. Computer Vision*, 91(2):200–215, 2011.
[36] B. Shi, X. Bai, W. Liu, and J. Wang. Deep regression for face alignment. *CoRR*, abs/1409.5230, 2014.
[37] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *Proc. CVPR*, 2013.
[38] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *Proc. ICML*, 2013.
[39] E. Tola, V. Lepetit, and P. Fua. DAISY: an efficient dense descriptor applied to wide-baseline stereo. *Trans. Pattern Anal. Mach. Intell.*, 32(5):815–830, 2010.
[40] G. Tzimiropoulos and M. Pantic. Gauss-newton deformable part models for face alignment in-the-wild. In *Proc. CVPR*, 2014.
[41] J. Wang, Z. Wei, T. Zhang, and W. Zeng. Deeply-fused nets. *CoRR*, abs/1605.07716, 2016.
[42] J. Xing, Z. Niu, J. Huang, W. Hu, and S. Yan. Towards multi-view and partially-occluded face alignment. In *Proc. CVPR*, 2014.
[43] X. Xiong and F. D. la Torre. Supervised descent method and its applications to face alignment. In *Proc. CVPR*, 2013.
[44] M. Yang, L. Zhang, S. C. Shiu, and D. Zhang. Robust kernel representation with statistical local features for face recognition. *Trans. Neural Netw. Learning Syst.*, 24(6):900–912, 2013.
[45] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *Int. J. Computer Vision*, 8(2):99–111, 1992.
[46] J. Zhang, S. Shan, M. Kan, and X. Chen. Coarse-to-fine auto-encoder networks (CFAN) for real-time face alignment. In *Proc. ECCV*, 2014.
[47] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Learning deep representation for face alignment with auxiliary attributes. *Trans. Pattern Anal. Mach. Intell.*, 38(5):918–930, 2016.
[48] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, 2003.
[49] E. Zhou, H. Fan, Z. Cao, Y. Jiang, and Q. Yin. Extensive facial landmark localization with coarse-to-fine convolutional network cascade. In *Proc. ICCV Workshops*, 2013.
[50] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Proc. CVPR*, 2012.