Recurrent Convolutional Face Alignment

Conference Paper · November 2016

CITATION
1

READS
369

3 authors:

Wei Wang
Università degli Studi di Trento
9 PUBLICATIONS **44** CITATIONS
SEE PROFILE

Sergey Tulyakov
Università degli Studi di Trento
14 PUBLICATIONS **43** CITATIONS
SEE PROFILE

Nicu Sebe
Università degli Studi di Trento
402 PUBLICATIONS **8,767** CITATIONS
SEE PROFILE

Recurrent Convolutional Face Alignment

Wei Wang, Sergey Tulyakov, Nicu Sebe

University of Trento, Italy

Abstract. Mainstream direction in face alignment is now dominated by cascaded regression methods. These methods start from an image with an initial shape and build a set of shape increments by computing features with respect to the current shape estimate. These shape increments move the initial shape to the desired location. Despite the advantages of the cascaded methods, they all share two major limitations: (i) shape increments are learned separately from each other in a cascaded manner, (ii) the use of standard generic computer vision features such SIFT, HOG, does not allow these methods to learn problem-specific features. In this work, we propose a novel Recurrent Convolutional Face Alignment method that overcomes these limitations. We frame the standard cascaded alignment problem as a recurrent process and learn all shape increments jointly, by using a recurrent neural network with the gated recurrent unit. Importantly, by combining a convolutional neural network with a recurrent one we alleviate hand-crafted features, widely adopted in the literature and thus allowing the model to learn task-specific features. Moreover, both the convolutional and the recurrent neural networks are learned jointly. Experimental evaluation shows that the proposed method has better performance than the state-of-the-art methods, and further support the importance of learning a single end-to-end model for face alignment.

1 Introduction

Face alignment methods trace their lineage from Active Shape Models [1, 2] and Active Appearance Models (AAM) [3], developed a couple of decades ago. These works first build a statistical shape and appearance models of the face, and during testing use numerical optimization techniques to find a set of parameters of the statistical model that could have generated the query face. Todays mainstream face alignment methods belong to Cascaded Regression Methods (CRM) group [4–9]. These methods operate in a cascaded fashion, *i.e.* starting from an initial shape and producing several shape increments that move the initial shape closer to the desired location. Shape increments are learned in a supervised manner during training stage. Formally CRMs operate in the following fashion:

$$\Delta\mathbf{S}_{t+1} = \mathbf{R}_t(\mathbf{F}_t(\mathbf{I}, \hat{\mathbf{S}}_t)), \tag{1}$$

$$\hat{\mathbf{S}}_{t+1} = \hat{\mathbf{S}}_t + \Delta\mathbf{S}_{t+1}, \tag{2}$$

where $\mathbf{I}$ denotes a 2D image, $\mathbf{F}_t(\mathbf{I}, \hat{\mathbf{S}}_t)$ represents the feature values extracted using the previous shape estimate $\hat{\mathbf{S}}_t$, $\Delta\mathbf{S}_{t+1}$ is a shape update produced by the

t-th regressor $\mathbf{R}_t$ in the cascade. To initialize the pipeline the average face shape over all images in the training set $\bar{\mathbf{S}}$ is taken. The feature extraction function ($\mathbf{F}_t(\cdot,\cdot)$) and a set of regressors ($\mathbf{R}_t(\cdot)$) constitute the main ingredients of a CRM framework. The final outcome of the CRMs writes as:

$$\hat{\mathbf{S}}(T) = \bar{\mathbf{S}} + \sum_{t=1}^{T} \Delta\mathbf{S}_t, \tag{3}$$

where T is the total number of layers in the cascade. In order to frame a task at hand as a cascaded regression problem, one has to decide upon the feature extraction function ($\mathbf{F}_t(\cdot,\cdot)$), as well as to select a proper regression function ($\mathbf{R}_t(\cdot)$). Various features have been explored by the community *e.g.* HoG [9], SIFT [5, 7], pixel differences [10–12], local binary features [13], as well as different regression functions have been tried: linear regression [5, 12], random ferns [14], regression trees [10, 11]. This brings to light two major limitations of the CRMs, that we are going to remove in this work: (i) manually designed features and (ii) relative independence of the regressors at the different layers in the cascade.

Hand-crafted computer vision features, such as HoG features for pedestrian detection [15], SIFT features for object recognition [16], attribute detection [17, 18] have played an important role in many application domains for a long time since they offer illumination, rotation and scaling invariance. These features, however, represent a generic image transformation that lacks any domain specific knowledge. Many works, have tackled this problem by *selecting* best features out of an overcomplete set [10, 11, 13]. However, this feature selection is suboptimal, since it is still performed on a generated set. Recently, it has been shown for object detection [19], tracking [20], image labeling [21] and other fields that features learned for a specific problem using deep convolutional neural networks show much better performance. Moreover, features learned for image classification often generalize well for different tasks, showing the ability of CNNs, such as AlexNet [19], VGGNet [22] and GoogleNet [23], to learn a generic image representation.

The second limitation of the CRMs is the independence of the regressors at every level of the cascade. One can argue the regressor at time t is learned by using the output of the previous regressor at time $t-1$, with the final prediction given by Eq. 3. This however, affects only the feature computation (see Eq. 1), while the regressors themselves are learned independently. It has been shown in [5] that a single regressor is not capable of arriving at the desired location in a single step. As shown in Eq. 3 the final prediction of the cascade $\hat{\mathbf{S}}(T)$ is a function of the number of layers in the cascade T. One can think of $\hat{\mathbf{S}}(T)$ as a sequence of measurements of some stochastic process. It has been recently shown that Recurrent Neural Network are extremely powerful in modeling the sequential inputs and outputs [24]. In order to model long time-varying sequences, various RNN units have been proposed. In particular, long-short term memory cells and later Gated Recurrent Units have proven to be efficient in modeling time-varying processes and sequence-to-sequence learning [25]. Additionally, it

has been shown that using a CNN for feature extraction and an RNN for classification brings extra advantages [26–28].

This discussion naturally brings us to the main contribution of this work. We present a unified face alignment framework that features end-to-end learning starting from raw pixel values. We replace the manually hand-crafted features $\mathbf{F}_t(\cdot,\cdot)$ by learning a patch-based CNN. In contrast with boosted regression methods, where one has a sequence of regressors $\{\mathbf{R}_1(\cdot), \mathbf{R}_2(\cdot), ..., \mathbf{R}_T(\cdot)\}$, our method learns a single recurrent module trained jointly with the CNN which can generate the regressor $\mathbf{R}(\cdot)$ recursively based on the input data and the memory of the recurrent module. We would like to highlight for the reader, that the parameters of both the CNN module and the RNN module are learned jointly. Additionally, we show that our model is capable of generalizing beyond the learned number of recurrent iterations, being able to automatically decide when to stop iterating. The experimental evaluation we detail in Section 4 proves that learning a task-specific end-to-end model brings higher accuracy than that of the available state-of-the-art.

2 Related work

In this section we review relevant works in face alignment as well as discuss recent advances in the neural-network learning important to formulate our Recurrent Convolutional Face Alignment (RFCA) method.

2.1 Face alignment

According to the widely accepted classification, methods for face alignment can be grouped into three broad categories [29]: Active Appearance Models (AAM), Constrained Local Models (CLM) [30–32], and Cascaded Regression Methods (CRM). Initial works on face alignment such as ASMs [1, 2] and AAMs [3, 33], build a parametric statistical shape and appearance models from a set of training faces. These methods show reasonable accuracy when the testing image is close to the training distribution. However, they fail to generalize to an unseen subject [34]. Although such methods still attract the attention of researchers [35, 36], the more recent Cascaded Regression Methods have shown higher accuracy at impressive frame rates [13, 11]. In the following we will mostly detail this latter group of works.

Initially CRMs were introduced in the medical image processing community for anatomic structure prediction [37]. Since then they have been extensively exploited by the computer vision community with many seminal works proposed in the literature. Currently this avenue of research represents the mainstream direction of the deformable shape fitting. In [14] a method for cascaded pose regression was introduced. The authors used a *pose-indexed features* and learned a sequence of weak-regressors (random ferns in their case) to regress a deformable shape from an image. In order to compute pose-indexed features one has to provide the current belief regarding the shape. This naturally brings some form

of pose invariance to the framework. Later, these ideas were extended to regress the whole face shape [38]. Importantly, it was shown that regressing the whole shape imposes the result to lie in the space constructed by all the training images.

The supervised descent method (SDM) [5] further extends the cascaded framework to generic non-linear optimization problems: face alignment, template tracking and camera calibration. SDM learns a sequence of descent directions that applied sequentially solve the optimization problem. The authors replace the feature extraction part with SIFT [39] and achieve impressive results by using linear regressors in the layers of the cascade. A downside of SDM, is its inability to generalize well to non-frontal poses, requiring to train separate regressors depending on the detected head pose. This constraint is relaxed in [9] by introducing a global SDM to automatically learn several descent maps at every level of the cascade to handle complex cost-functions. These ideas were extended in [7], where the authors learn both the Jacobian and the Hessian matrices, in a manner inspired by the Gauss-Newton optimization method. Similarly to the original SDM, the authors use hand-crafted SIFT features extracted around the keypoints locations. SDM-based methods have become popular in various applications of face analysis [40] and are used in several commercially available face alignment systems[1]. A different strategy for feature extraction is presented in [10, 13, 11]. Instead of employing hand-crafted features (*e.g.*, HoG, SIFT), they perform feature selection using a framework of regression trees. Alleviating the need to compute hand-crafted features, these works reach impressive processing speed.

Multiple CRM-based 3D methods have been proposed. In [41], an extension of [38] is introduced to fit a 2D-3D parametric shape model. Similar ideas were explored in [42], where a cascaded coupled regressor is introduced to obtain the camera projection matrix and the 3D landmarks of the face. The work in [10] proposes to include the third dimension directly into the learning pipeline. They used a large generated set of training faces and showed that considering a face shape as a 3D object gives better results. An interesting work in [12] uses binary features to track a large number of points on the face, with subsequent 3D deformable model fitting to obtain a 3D mesh of the face. Notably, this work shows impressive frame rates for the whole pipeline as well as the tracking accuracy comparable to purely 3D methods [43].

From a higher perspective the aforementioned methods have two independent steps: (i) feature extraction and (ii) applying a sequence of regressors. Typically the first step is performed by using some hand-crafted features such as SIFT [5, 7], HOG [9]. Some form of feature learning is employed in [13, 10, 11], while the levels of the cascade in the second step still remain independent. This requires a researcher to use a trial-and-fail approach in selecting which features and which regressors work the best.

In contrast, the method presented in our study is end-to-end. By learning convolutional filters, RCFA does not require manual supervision in defining fea-

[1] http://www.humansensing.cs.cmu.edu/intraface/index.php
http://www.zface.org/

ture extraction functions. Additionally, our method replaces a cascade of independent regressors by a single recurrent model, where all iterations are leaned jointly. This formulation merges the two steps of the typical CRM pipeline into a single unified framework, simultaneously trained using the available data.

2.2 Recurrent and convolutional neural networks

Recurrent Neural Networks (RNN) have become increasingly popular to learn complex dynamic systems, because of their impressive capability to recurrently operate with sequential input. During each recurrence of the traditional RNN [44], an input signal is mapped to the hidden state, which is passed forward to the next recurrence. This way, the information of the previous states is memorized and persists during the whole process. Therefore, RNNs have proven to have an advantage in modeling sequences with long-term dependencies. During the last decade, we have seen a lot of success in applying RNNs to various application domains, such as generating text description of videos [45], image caption generation [46], face aging [47], machine translation [24] and speech recognition [48].

Given the success of RNNs, a lot of RNN variants have been explored, such as the Long Short Term Memory (LSTM) [49] networks, Gated Recurrent Unit (GRU) [25], and Clockwork RNN [50]. All these architectures consist of a chain of repeated modules, where each module contains several gates, controlling the information flow in the network and states, memorizing necessary information for future recurrences. Although the combination of gates/states varies depending on the selected architectures, each subsequent iteration is performed similarly, by processing a new input using the memory of the current state. These architectures show varying performances for different tasks. In [51] it was shown that, in general, GRU-based models feature superior performance compared with other architectures.

Convolutional Neural Networks (CNN) have recently demonstrated notable success in multiple tasks, such as image classification [19], super-resolution [52], as well as image segmentation [53]. One of the main advantages of CNNs, is that they do not require human supervision to design feature transformation. Their feature representations have shown to provide significantly higher performance, compared to commonly adopted hard-crafted features, in numerous application domains. Thus, it is very promising to combine the RNN architecture together with the CNN architecture into a hybrid architecture. This hybrid architecture has been successfully applied to many tasks, such as scene labeling [26], object recognition [27], and text classification [28].

3 Method

The overview of the proposed Recurrent Convolutional Face Alignment method is given in Fig. 1. The framework mainly consists of two parts, the *recurrent module* and the *convolutional module*. During each recurrent iteration t the current shape estimate $\mathbf{h}_t$ is imposed onto the image and the convolutional neural

network is applied to the patches extracted around the points of the shape. The output of the last layer of the CNN is passed to the RNN as an input. During the first iteration, the average shape of all the images in the training set is set to the initial shape estimate: $\widehat{\mathbf{h}}_0 = \bar{\mathbf{S}}$.

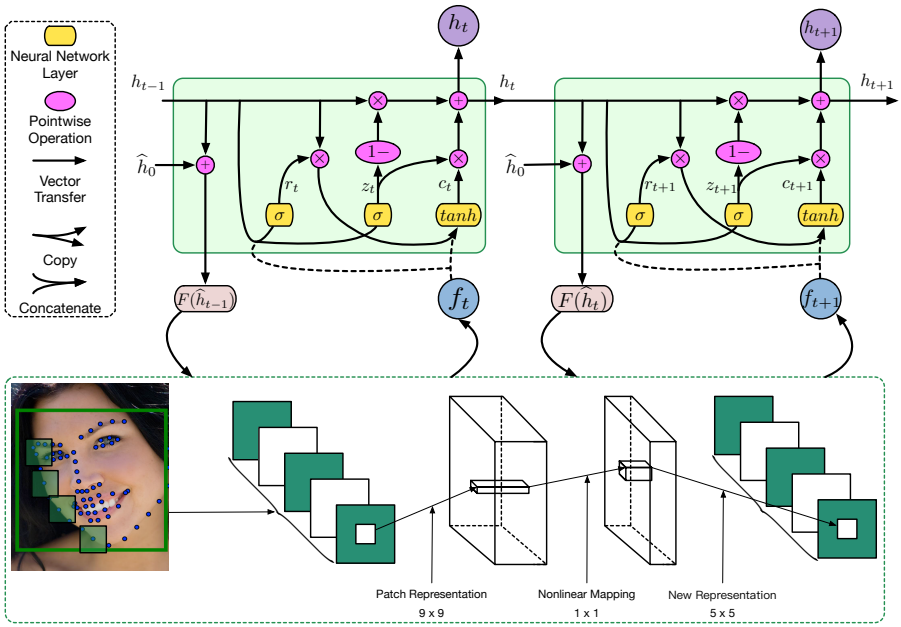

Fig. 1. The overview of the proposed approach. Top: the RNN with gated recurrent units unrolled in time. Bottom: the CNN architecture used for feature extraction. Note that feature extraction is performed at every recurrent iteration.

3.1 Recurrent module

In the current study we use an RNN with GRU module for its simplicity and superior performance as compared to other RNN types [51]. The structure of two recurrent iterations is given in the top row of Fig. 1. A GRU contains two gates and one state. The gates are the *reset* gate and the *update* gate. The *hidden* state $\mathbf{h}_t$ represents the relative movement or increment of the landmark positions after the adjustment in t-th iteration. Then the predicted position after t iterations is $\widehat{\mathbf{h}}_t = \widehat{\mathbf{h}}_0 + \mathbf{h}_t$.

The feature extraction function $f(t) = \mathbf{F}(\mathbf{I}, \widehat{\mathbf{h}}_t)$ is performed using a super resolution convolutional neural network (SRCNN), described in Section 3.

The *reset* gate $\mathbf{r}_t$ controls whether the adjustment from the previous recurrence should be ignored, *i.e.* if $\mathbf{r}_t$ is close to 0, the information of the previous adjustment operation will be forced to be discarded. Then the unit will focus on

its current features without referring to the previous operation. To sum up, the reset gate allows the unit to remember or drop the adjustment operation from the previous operation.

The *update* gate $\mathbf{z}_t$ has two functions. The first one is to control what to forget from the previous operation which is implemented by the term $\mathbf{z}_t$, and the second one is to control the acceptance of the new input operation which is implemented by the term $1-\mathbf{z}_t$. If $\mathbf{z}_t$ is set to 0, $1-\mathbf{z}_t$ will be 1. This means that all the information from the previous operation will be kept and the new input will be totally discarded. Thus, the new adjustment operation will be exactly the same as the previous operation. However, if $\mathbf{z}_t$ is set to 1, $1-\mathbf{z}_t$ will be 0, and the next operation will be based only on the new input operation without referring to the previous adjustment operation.

The described process is schematically presented in Fig. 1, where a single recurrent iteration is governed by the following equations:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_{zh}\mathbf{h}_{t-1} + \mathbf{W}_{zf}f_t + \mathbf{b}_z) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_{rh}\mathbf{h}_{t-1} + \mathbf{W}_{rf}f_t + \mathbf{b}_r) \\
\mathbf{c}_t &= tanh(\mathbf{W}_{ch}\mathbf{r}_t \odot \mathbf{h}_{t-1} + \mathbf{W}_{cf}f_t + \mathbf{b}_c) \\
\mathbf{h}_t &= (1-\mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \mathbf{c}_t
\end{aligned} \tag{4}$$

where $\odot$ represents element-wise multiplication, and $\mathbf{c}_t$ is the new increment candidate created by the *tanh* layer that could be added to the current shape increment using the following rule:

$$\mathbf{h}_t{=}(1{-}\mathbf{z}_t){\odot}\mathbf{h}_{t-1}{+}\mathbf{z}_t{\odot}\mathbf{c}_t. \tag{5}$$

If the reset gate is always activated, the system will have only the short-term memory, since the calculation of the new increment candidate ignores the previous increments and focuses on the current input only. If the update gate is not activated, the system can have the long-term memory and the previous increments will be memorized.

Within this framework, the RNN acts as a refinement process which tries to find the optimal shape increment by *gradually* changing the previous shape. We use T recurrent steps to train RCFA. In order for the RNN to focus on the later iterations we define a series of weights $\mathbf{w}{=}[w_1, w_2, ...w_T]$ each one for a single recurrent iteration. These weights increase monotonically, therefore forcing the recurrent network to adjust the shape slowly and penalizing the model more for the error during the later recurrent steps. Formally, the loss writes as:

$$J = \sum_{i=1}^{n}\sum_{t=1}^{T} w_t \|(\widehat{\mathbf{h}}_0 + \mathbf{h}_t^i) - \mathbf{h}_*^i\|_F^2, \tag{6}$$

where $\widehat{\mathbf{h}}_0$ is the initial shape estimate, *i.e.* the average shape, $\mathbf{h}_t^i$ is the predicted shape increment after t iterations, $\mathbf{h}_*^i$ is the target shape, the superscript i defines the ith image in the mini-batch of n images. The final shape after t steps is obtained as $\widehat{\mathbf{h}}_0 + \mathbf{h}_t^i$. During training, for each face image $\mathbf{I}^i$, the initial shape $\widehat{\mathbf{h}}_0$ is sampled several times by adding noise to the mean shape.

3.2 Convolutional module

We employ the super resolution convolutional neural network (SRCNN) for feature extraction [52]. We apply the SRCNN to the pixel values around the landmarks position Fig. 1. We denote the patch around a landmark location as $\mathbf{Y}$, and use it as an input for the SRCNN. The SRCNN consists of three convolution layers, formulated as the following operations:

$$\begin{aligned} F_1(\mathbf{Y}) &= \max(0, \mathbf{W}_1 * \mathbf{Y} + \mathbf{B}_1) \\ F_2(\mathbf{Y}) &= \max(0, \mathbf{W}_2 * F_1(\mathbf{Y}) + \mathbf{B}_2) \\ F_3(\mathbf{Y}) &= \mathbf{W}_3 * F_2(\mathbf{Y}) + \mathbf{B}_3 \end{aligned} \tag{7}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$ and $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ represent the filters and biases respectively. The Rectified Linear Unit (ReLU) is employed as the activation function for the first two convolution layers. The dimensions of $\mathbf{W}_1$ are set to $c \times f_1 \times f_1 \times n_1 = [1 \times 9 \times 9 \times 64]$, where c is the number of channels of the input image, f_1 is the filter size, and n_1 is the number of filters which also corresponds to the number of feature maps. $\mathbf{W}_2$ is of the size $n_1 \times 1 \times 1 \times n_2 = [64 \times 1 \times 1 \times 32]$ and $\mathbf{W}_3$ has the size of $n_2 \times f_3 \times f_3 \times c = [32 \times 5 \times 5 \times 3]$. The first layer can be regarded as PCA where each filter works as a basis and projects the input $\mathbf{Y}$ to a high-dimension vector. The second layer has the filter size of 1×1, and this layer can be understood as a non-linear mapping operation which maps an $n-1$ dimensional vector to a n_2 dimensional vector. Originally, the last layer in the SRCNN works as an averaging filter which projects the n_2 dimensional vector to a high-resolution patch, and take the average of the overlapping high-resolution patches. However, instead of projecting the n_2 dimensional vector to a high-resolution patch, the last layer in our network will project the n_2 dimensional vector to a feature space which can is then passed to the recurrent module.

3.3 Supervised descent method as GRU

In this section we show that the proposed RCFA method is a generalization of the widely adopted Supervised Descent Method [5]. Given a set of images $[\mathbf{I}^1, \mathbf{I}^2, ...\mathbf{I}^i, ..., \mathbf{I}^n]$, $\mathbf{h}^i$ denotes the positions of the landmarks in image $\mathbf{I}^i$. $\mathbf{F}$ is a feature extraction function, and $\mathbf{F}(\mathbf{h}^i)$ represent the extracted features. Let $\mathbf{y}^i_* = \mathbf{F}(\mathbf{h}^i_*)$ be the ground-truth features extracted at the manually labeled landmark positions $\mathbf{h}^i_*$. Then we have the following objective function for face alignment with respect to image $\mathbf{I}^i$,

$$\min \|\mathbf{F}(\mathbf{h}^i) - \mathbf{y}^i_*\|_2^2. \tag{8}$$

SDM applies the gradient descent rule to Eq. 8, and yields the following discrete update equation:

$$\begin{aligned} \mathbf{h}^i_t &= \mathbf{h}^i_{t-1} - \mathbf{R}_{t-1}(\mathbf{F}(\mathbf{h}^i_{t-1}) - \mathbf{y}^i_*) \\ &= \mathbf{h}^i_{t-1} - \mathbf{R}_{t-1}\mathbf{F}(\mathbf{h}^i_{t-1}) + \mathbf{R}_{t-1}\mathbf{y}^i_*, \end{aligned} \tag{9}$$

where $\mathbf{R}_{t-1}{=}\alpha\mathbf{F}'(\mathbf{x}^i_{t-1})$, and $\mathbf{R}_{t-1}$ is regarded as a regressor. Thus, instead of calculating the derivatives, the SDM learns a descend direction from the available training data.

However, Eq. 9 has an inconsistency problem, *i.e.* $\mathbf{y}^i_*$ is only available in the training phase and it is unknown in the testing phase. Therefore, Eq. 9 could not be used to calculate the position of the landmarks. To solve this inconsistency problem, $\mathbf{y}^i_*$ is replaced by $\overline{\mathbf{y}}_* = (\sum_i \mathbf{y}^i_*)/n$. By defining $\mathbf{b}_{t-1} = \mathbf{R}_{t-1}\overline{\mathbf{y}}_*$ we obtain the new update equation:

$$\mathbf{h}^i_t = \mathbf{h}^i_{t-1} - \mathbf{R}_{t-1}\mathbf{F}(\mathbf{h}^i_{t-1}) + \mathbf{b}_{t-1}, \tag{10}$$

which solves the inconsistency problem. During the training phase, $\mathbf{h}^i_t$ is set to $\mathbf{h}^i_*$ as our goal is to make $\mathbf{h}^i_t$ equal to the target $\mathbf{h}^i_*$. The loss is defined as:

$$\sum_i \|\mathbf{h}^i_* - \mathbf{h}^i_{t-1} + \mathbf{R}_{t-1}\mathbf{F}(\mathbf{h}^i_{t-1}) - \mathbf{b}_{t-1}\|^2 \tag{11}$$

where $\mathbf{h}^i_0$ is obtained using Monte Carlo integration.

Thus, Eq. 11 can be considered as a special case of Eq. 6, making the SDM a special case of our GRU network. As shown in Fig. 2 (b), the traditional linear regressor is equivalent to GRU if the *update* gate and *reset* gate are removed. Finally, if we replace the *tanh* layer with the regressor $\mathbf{R}$, we obtain the formula for the shape increment $\mathbf{h}_t$ for the image $\mathbf{I}^i$, as follows:

$$\mathbf{h}^i_t = \mathbf{h}^i_{t-1} - \mathbf{R}_{t-1}\mathbf{F}(\mathbf{h}^i_{t-1}) \tag{12}$$

Eq. 12 is a recurrent version of Eq. 10 except for the term $\mathbf{b}_{t-1}$ which can be implemented by expanding the feature space by several columns set to 1.

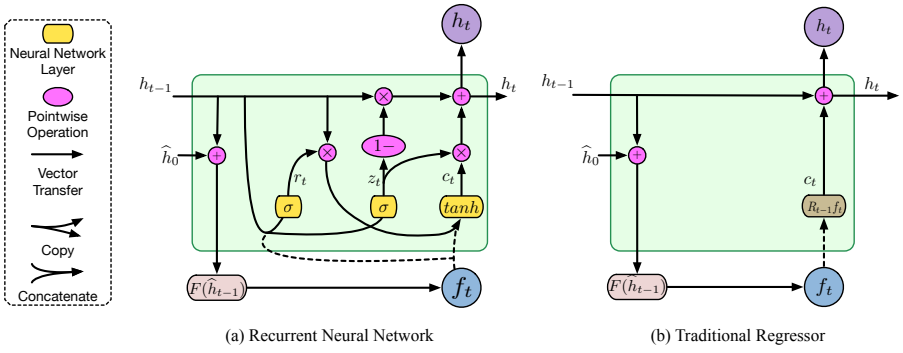

Fig. 2. Differences in the architecture of the proposed recurrent regressor (a) compared to the traditional regressor (b).

As shown in Fig. 2, the traditional regressors at different time steps ($\mathbf{R}_1, \mathbf{R}_2, ...$) are trained independently, relying only on the input features while totally lacking

memory regarding previous states, since as it is shown in Eq. 11, every step has a separate loss function. In contrast, for our model the overall loss over all the recurrent steps is defined and learned jointly by summing the steps up (Eq. 6). Another way of thinking about our recurrent module, is to treat it not as a regressor, but rather as a way of generating unique regressors at every recurrent step with respect to the memory and the input features.

4 Experiments

Datasets We evaluate the performance of our algorithm using the widely adopted 300-W dataset [54]. This dataset is a combination of several in-the-wild datasets, including AFW [55], LFPW [56], HELEN [57] and XM2VTS [58], that are annotated with 68-point markup in a consistent manner. Similarly to previous works [8, 13], for training the model we use the training samples from LFPW, HELEN and the whole AFW dataset, which makes 3148 images in total. Testing is performed on three different sets of images: (i) the *common* set includes the testing images from the LPFW and HELEN, (ii) the *challenging* set includes recently released 135 images also known as the IBUG set, and (iii) the *full* set is a combination of the first two. We do not report the results for the original annotations for HELEN and LFPW, since the accuracy of the state-of-the-art methods has saturated.

Evaluation Metrics To evaluate the performance of our method, we follow the widely adopted evaluation metric [8, 10, 13], which is the **average error** of the point-to-point Euclidean distance, normalized by the distance between the outer corners of the eyes. This metric has been adopted for the 300-W challenge.

4.1 Implementation

For the CNN module, we follow the settings of the SRCNN framework in [52]. This module will extract features for all the image patches. After obtaining the features for each image patch, we concatenate the feature vectors of the 68 landmarks and feed the features to the RNN module. For the RNN module, we set the total number of recurrent iterations T to be 5. The weights in Eq. 6 are set to powers of 10: $\mathbf{w} = [10^{-2}, 10^{-1}, ..., 10^{2}]$. Powers of 2 and 5 showed slightly inferior performance to the powers of 10, while equal weights $[1, 1, 1, 1, 1]$ showed the worst performance.

To augment the size of the training data, we duplicate the images by adding the mirrored examples, and we also replicate the training data 3 times by adding noise to the bounding boxes. In the training phase, the batch size is set to 204 images. The learning rate is set to 0.01. The decay rate is set to 0.5, and the learning rate will be decayed after every 10 epochs and the training process is terminated after 200 epochs. After we obtain the model, we generate another three replicates in the same manner and fine tune the network with the new replicates for another 200 epochs.

4.2 Understanding when to stop iterating

One of the further advantages of our RCFA is that the model can be easily extended beyond the learned number of recurrent iterations without the need of retraining the whole pipeline. Importantly, there is no upper bound on the number of recurrent iterations one can perform. This, however, requires devising a strategy to stop iterating. During training the RNN performed 5 recurrent iterations. Intuitively, the model should require less iterations for a simple image, while difficult examples may need additional recurrences. In order to define a stopping strategy, we show the relationship between the average error and the recurrent steps as shown in Fig. 3.

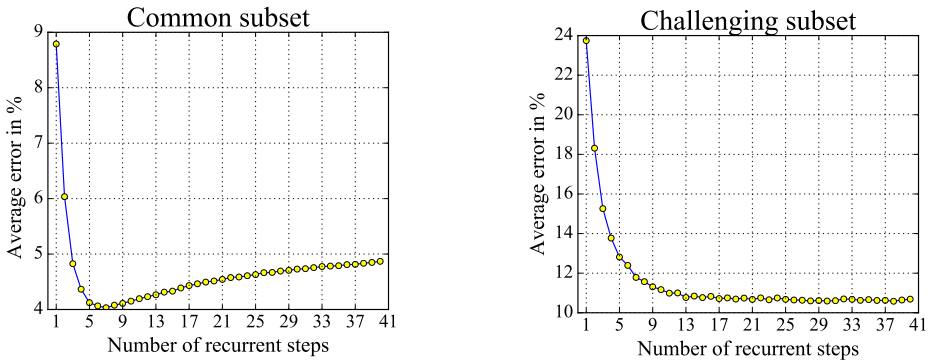

Fig. 3. Average error vs the number of recurrent iterations

As it is seen from the left graph in Fig. 3 for easy images from the common set additional iterations are redundant and do more harm, while the hard cases from the challenging set benefit from iterating further (see Fig. 3, right). Typically, for hard cases the error still continues decreasing when the number of iterations is more than 15, while for the easy ones it remains stable between 5-th and 9-th iterations and then goes up. This suggests a simple and efficient stopping criteria that was used to generate the results for the RCFA adaptive in the Table 1. If the difference between the previous landmark positions and the current landmark positions is smaller than a threshold, we stop iterating. To set the threshold value, we take the average difference between the 4-th and 5-th recurrence of all the images in the training data. This simple stopping strategy allows our model to automatically decide whether any additional iterations are necessary.

Fig. 4 shows several qualitative examples of different number of recurrent iterations required for different testing examples. The first two rows show that 4 steps are not sufficient to localize the landmarks, as one can easily see that the landmarks on the jawline in the first row do not fit perfectly until the fifth recurrence. A similar observation can be made for the subject shown in the second row. The last two rows show cases, when even 5 iterations are not sufficient

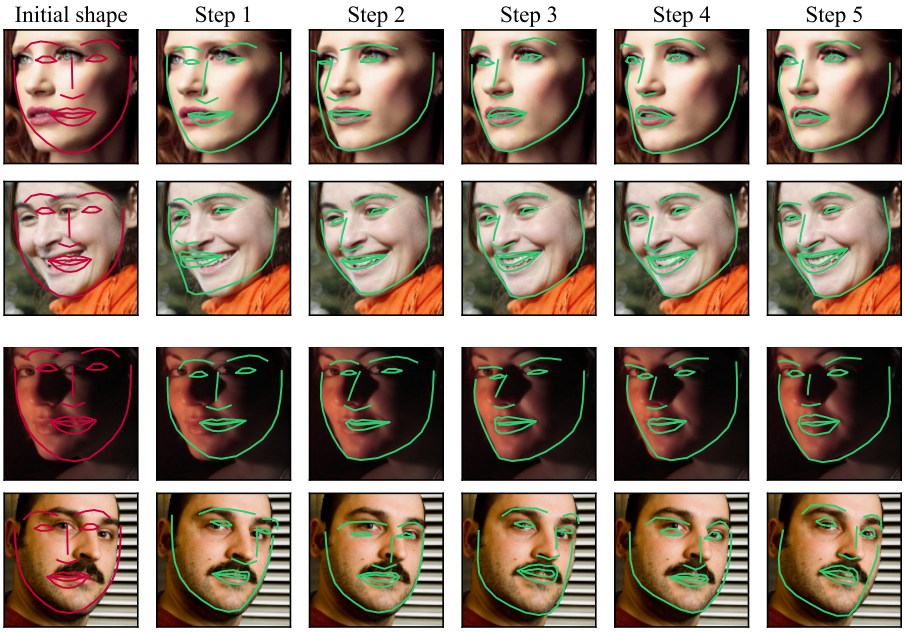

Fig. 4. Landmark localization for 5 recurrent steps. The top two rows show examples, for which 5 iterations is sufficient, while the examples in the last two rows require additional iterations.

for the method to converge, due to difficult illumination conditions and extreme head poses. This further supports the importance of the adopted stopping strategy.

4.3 Experimental Results

We report evaluation results on the three subsets of the 300-W dataset in Table 1. It compares three different result of the same RCFA model against best performing state-of-the-art methods. The reported RCFA results are obtained using 5, 10 recurrent iterations and the proposed stopping strategy. We would like to highlight, that due to the end-to-end structure, our model shows better performance than the up-to-date face alignment methods regardless of the number of iterations for the common set and the full set. Notably, when the proposed stopping strategy is used, the proposed method outperforms other works by a large margin for all three testing sets.

Interestingly, RCFA outperforms CFSS [8] by a margin of 16% on the common subset, while showing a little bit lower performance gain for the challenging set and the full set (10% and 9% correspondingly). There are mainly two reasons for this. Firstly, the commonly accepted evaluation metric is severely affected by a small portion of hard examples. Secondly, these hard examples are not evenly

Table 1. Experimental results obtained on the three subsets of the 300-W dataset.

Method	Common	Challenging	Full
Zhu *et al.* [55]	8.22	18.33	10.20
DFMF [59]	6.65	19.79	9.22
ESR [60]	5.28	17.00	7.58
RCPR [61]	6.18	17.26	8.35
SDM [5]	5.57	15.40	7.50
Smith *et al.* [62]	-	13.30	-
Zhao *et al.* [63]	-	-	6.31
GN-DPM [64]	5.78	-	-
CFAN [65]	5.50	-	-
ERT [11]	-	-	6.40
LBF [13]	4.95	11.98	6.32
LBF fast [13]	5.38	15.50	7.37
CFSS [8]	4.73	9.98	5.76
CFSS Practical [8]	4.79	10.92	5.99
RCFA 5 iterations	4.08	12.81	5.81
RCFA 10 iterations	4.13	11.14	5.51
RCFA adaptive	**4.03**	**9.85**	**5.32**

presented in the 300-W training/testing sets. The dataset is rather biased towards having less extreme head poses, facial expressions and poor illumination conditions.

Fig. 5 shows the qualitative results for the images taken from the full set. Clearly, due to end-to-end learning our framework handles even challenging face images, such as facial expressions, extreme head poses, difficult lighting conditions. It is also very interesting to observe that RCFA can handle faces with severe occlusions, including sun-glasses, hands and hats. The reason why our framework can work well for these images is because our RNN network can not only learn the dependencies between each regressor, it also learns the location dependencies between the landmarks. Thus, even though parts of the face is occluded, our framework can still predict the location of the occluded landmarks based on other landmarks.

In the current implementation, a single forward pass through the pipeline takes around 10ms on average on Tesla K40, making it possible to apply the proposed model for real-time video processing at 100 frames per second. We would like to note, that no specific performance optimizations were used, therefore, we believe the running time can be decreased dramatically.

5 Conclusions

In this paper, we reformulate the classical cascaded regression face alignment problem as a recurrent process, alleviating the two major limitations of the CRMs. The proposed recurrent framework features end-to-end learning, starting from the raw pixel data, removing the previously used hand-crafted features.

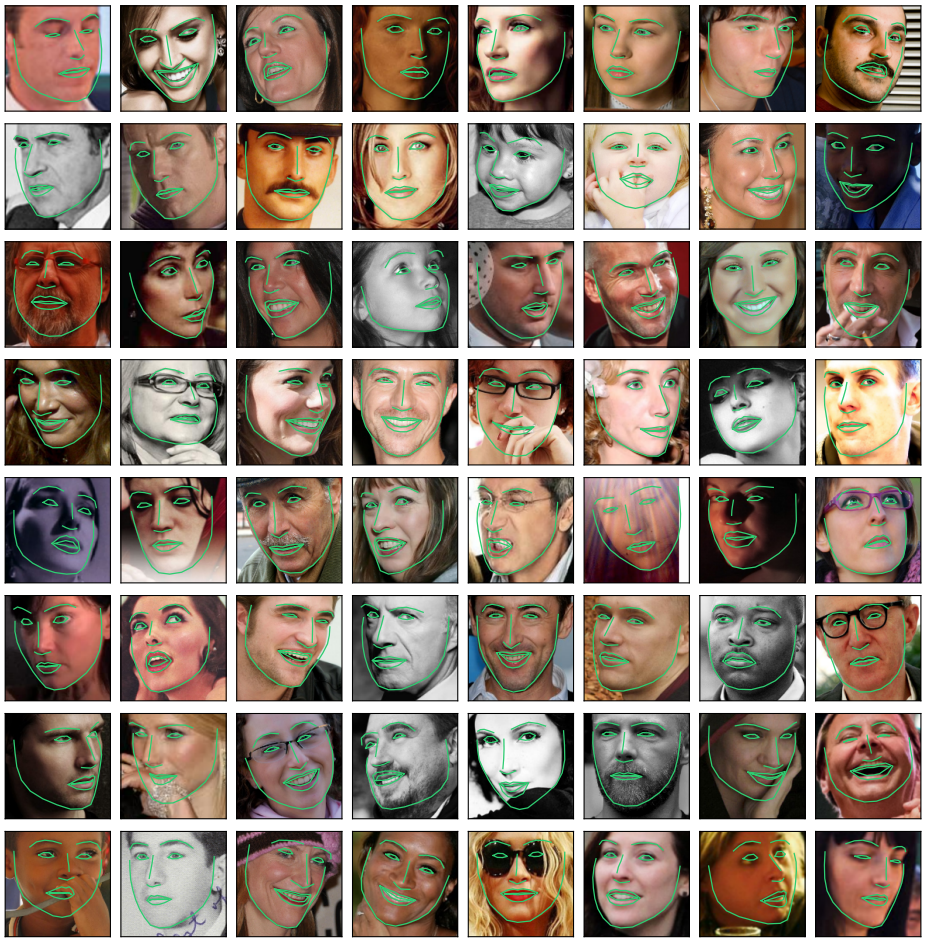

Fig. 5. Selected qualitative examples taken from the full set of the 300-W dataset.

Replacing a standard cascade of independently learned shape regressors by a single recurrent regressor brings further advantage of iterating beyond the learned limit, making it possible to automatically decide when to stop.

The proposed RFCA method has room for further improvements. In our experiments an average shape is used to initialize the pipeline, while it has been shown that selecting a proper starting shape brings extra benefits [8]. Additionally, more rigorous data augmentation can alleviate the bias of the training set and can make the data more uniform. Furthermore, we believe similar recurrent-convolutional shape regression models can be employed to various other tasks such as action recognition [66] and human pose estimation.

References

1. Cootes, T.F., Taylor, C.J.: Active shape models - 'smart snakes'. In: BMVC. (1992)
2. Cootes, T.F., Taylor, C.J.: Active shape model search using local grey-level models: A quantitative evaluation. In: BMVC. (1993)
3. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. TPAMI (2001) 681–685
4. Cao, C., Weng, Y., Lin, S., Zhou, K.: 3d shape regression for real-time facial animation. In: SIGGRAPH. (2013)
5. Xiong, X., De La Torre, F.: Supervised descent method and its applications to face alignment. In: CVPR. (2013)
6. Yang, H., Patras, I.: Sieving regression forest votes for facial feature detection in the wild. In: ICCV. (2013) 1936–1943
7. Tzimiropoulos, G.: Project-out cascaded regression with an application to face alignment. In: CVPR. (2015)
8. Zhu, S., Li, C., Change, C., Tang, X.: Face alignment by coarse-to-fine shape searching. In: CVPR. (2015)
9. Xiong, X., Torre, F.D.: Global supervised descent method. In: CVPR. (2015)
10. Tulyakov, S., Sebe, N.: Regressing a 3d face shape from a single image. In: ICCV. (2015)
11. Kazemi, V., Josephine, S.: One millisecond face alignment with an ensemble of regression trees. In: CVPR. (2014)
12. Jeni, L.A., Cohn, J.F., Kanade, T.: Dense 3d face alignment from 2d videos in real-time. In: FG. (2015)
13. Ren, S., Cao, X., Wei, Y., Sun, J.: Face alignment at 3000 fps via regressing local binary features. In: CVPR. (2014)
14. Doll, P., Pietro, W., Perona, P.: Cascaded pose regression. In: CVPR. (2010)
15. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
16. Lowe, D.G.: Object recognition from local scale-invariant features. In: ICCV. (1999)
17. Wang, W., Yan, Y., Winkler, S., Sebe, N.: Category specific dictionary learning for attribute specific feature selection. TIP **25** (2016) 1465–1478
18. Wang, W., Yan, Y., Sebe, N.: Attribute guided dictionary learning. In: ICMR. (2015)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
20. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. In: NIPS. (2013)
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv (2014)
22. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. In: BMVC. (2014)
23. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. (2015)
24. Auli, M., Galley, M., Quirk, C., Zweig, G.: Joint language and translation modeling with recurrent neural networks. In: EMNLP. (2013)
25. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv (2014)

26. Pinheiro, P.H.O., Collobert, R.: Recurrent convolutional neural networks for scene parsing. arXiv (2013)
27. Liang, M., Hu, X.: Recurrent convolutional neural network for object recognition. In: CVPR. (2015)
28. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: AAAI. (2015)
29. Wang, N., Gao, X., Tao, D., Li, X.: Facial feature point detection: A comprehensive survey. arXiv (2014)
30. Saragih, J.M., Lucey, S., Cohn, J.F.: Deformable model fitting by regularized landmark mean-shift. IJCV **91** (2011) 200–215
31. Baltrusaitis, T., Robinson, P., Morency, L.P.: 3d constrained local model for rigid and non-rigid facial tracking. In: CVPR. (2012)
32. Yu, X., Huang, J., Zhang, S., Yan, W., Metaxas, D.N.: Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model. In: ICCV. (2013)
33. Cootes, T., Edwards, G., Taylor, C.: Active appearance models. PAMI (2001)
34. Gross, R., Gross, R., Matthews, I., Matthews, I., Baker, S., Baker, S.: Generic vs. person specific active appearance models. IVC (2005)
35. Tzimiropoulos, G., Pantic, M.: Optimization problems for fast aam fitting in-the-wild. In: ICCV. (2013)
36. Fanelli, G., Dantone, M., Van Gool, L.: Real time 3d face alignment with random forests-based active appearance models. In: FG. (2013)
37. Zhou, S.K., Comaniciu, D.: Shape regression machine. In: Medical Imaging. (2007)
38. Cao, X.: Face alignment by explicit shape regression. In: CVPR. (2012)
39. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60** (2004) 91–110
40. Tulyakov, S., Alameda-Pineda, X., Ricci, E., Yin, L., Cohn, J.F., Sebe, N.: Self-adaptive matrix completion for heart rate estimation from face videos under realistic conditions. In: CVPR. (2016)
41. Cao, C., Weng, Y., Zhou, S., Tong, Y., Zhou, K.: Facewarehouse: A 3d facial expression database for visual computing. In: TVCG. (2014)
42. Jourabloo, A., Liu, X.: Pose-invariant 3d face alignment. In: ICCV. (2015)
43. Tulyakov, S., Vieriu, R.L., Semeniuta, S., Sebe, N.: Robust Real-Time Extreme Head Pose Estimation. In: International Conference on Pattern Recognition. (2014)
44. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. TSP **45** (1997) 2673–2681
45. Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., Saenko, K.: Sequence to sequence-video to text. In: ICCV. (2015)
46. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: CVPR. (2015)
47. Wang, W., Cui, Z., Yan, Y., Feng, J., Yan, S., Shu, X., Sebe, N.: Recurrent face aging. (2016)
48. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: ICASSP. (2013)
49. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation (1997)
50. Koutnik, J., Greff, K., Gomez, F., Schmidhuber, J.: A clockwork rnn. arXiv (2014)
51. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: ICML. (2015)
52. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: ECCV. (2014)

53. Liang, X., Liu, S., Shen, X., Yang, J., Liu, L., Dong, J., Lin, L., Yan, S.: Deep human parsing with active template regression. TPAMI **37** (2015) 2402–2414
54. Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., Pantic, M.: 300 faces in-the-wild challenge: The first facial landmark localization challenge. In: ICCV Workshops. (2013)
55. Zhu, X., Ramanan, D.: Face detection, pose estimation, and landmark localization in the wild. In: CVPR. (2012)
56. Belhumeur, P.N., Jacobs, D.W., Kriegman, D.J., Kumar, N.: Localizing parts of faces using a consensus of exemplars. TPAMI **35** (2013) 2930–2940
57. Le, V., Brandt, J., Lin, Z., Bourdev, L., Huang, T.S.: Interactive facial feature localization. In: ECCV. (2012)
58. Messer, K., Matas, J., Kittler, J., Luettin, J., Maitre, G.: Xm2vtsdb: The extended m2vts database. In: Second international conference on audio and video-based biometric person authentication. (1999)
59. Asthana, A., Zafeiriou, S., Cheng, S., Pantic, M.: Robust discriminative response map fitting with constrained local models. In: CVPR. (2013)
60. Cao, X., Wei, Y., Wen, F., Sun, J.: Face alignment by explicit shape regression. IJCV **107** (2014) 177–190
61. Burgos-Artizzu, X., Perona, P., Dollár, P.: Robust face landmark estimation under occlusion. In: ICCV. (2013)
62. Smith, B., Brandt, J., Lin, Z., Zhang, L.: Nonparametric context modeling of local appearance for pose-and expression-robust facial landmark localization. In: CVPR. (2014)
63. Zhao, X., Kim, T.K., Luo, W.: Unified face analysis by iterative multi-output random forests. In: CVPR. (2014)
64. Tzimiropoulos, G., Pantic, M.: Gauss-newton deformable part models for face alignment in-the-wild. In: CVPR. (2014)
65. Zhang, J., Shan, S., Kan, M., Chen, X.: Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment. In: ECCV. (2014)
66. Wang, W., Yan, Y., Nie, L., Zhang, L., Winkler, S., Sebe, N.: Sparse code filtering for action pattern mining. In: ACCV. (2016)