Recurrent Convolutional Shape Regression

Wei Wang, *Member, IEEE,* Sergey Tulyakov, *Member, IEEE,* and Nicu Sebe, *Senior Member, IEEE*

Abstract—The 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 based on features with respect to the current estimated shape. 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 independently 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 Shape Regression (RCSR) method that overcomes these limitations. We formulate the standard cascaded alignment problem as a recurrent process and learn all shape increments jointly, by using a recurrent neural network with a gated recurrent unit. Importantly, by combining a convolutional neural network with a recurrent one we avoid hand-crafted features, widely adopted in the literature and thus we allow the model to learn task-specific features. Besides, we employ the convolutional gated recurrent unit which takes as input the feature tensors instead of flattened feature vectors. Therefore, the spatial structure of the features can be better preserved in the memory of the recurrent neural network. 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 supports the importance of learning a single end-to-end model for face alignment.

Index Terms—Recurrent Neural Network, Gated Recurrent Unit, Shape Regression, Facial Landmarks

1 Introduction

Face alignment methods trace their lineage from Active Shape Models (ASM) [1] and Active Appearance Models (AAM) [2], developed a couple of decades ago. These works first build a statistical shape and appearance model 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. Today's mainstream face alignment methods belong to Cascaded Regression Method (CRM) group [3] [4] [5] [6]. These methods operate in a cascaded fashion, *i.e.* starting from an initial shape (*e.g.*, mean shape) and producing several shape increments that move the initial shape closer to the desired location with the help of a sequence of regressors which are trained independently. As shown in Fig. 1, 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 features extracted using the previous shape estimate $\hat{\mathbf{S}}_t$, $\Delta \mathbf{S}_{t+1}$ is a shape increment 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 the CRM framework. The final output of the CRM writes as:

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

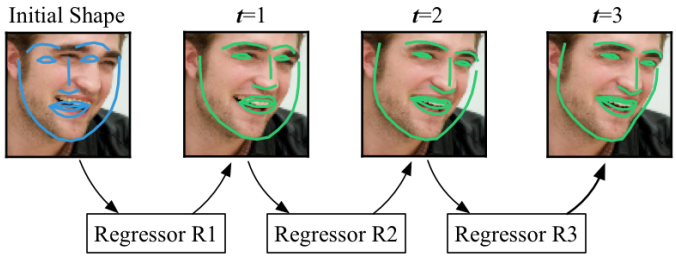


Fig. 1: The Cascaded Regression Methods (CRMs) train the regressors independently. We convert CRM into an end-to-end recurrent process, and the shape regressors are learned jointly. The regressor at each step is generated dynamically based on the new input features and the memory of the recurrent regressor.

where T is the total number of layers in the cascade. In order to frame the task 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 (*e.g.*, HOG [6], SIFT [3] [4], pixel differences [7], [8], [9], local binary features [10]), and different regression functions have been tried (*e.g.*, linear regression [3], random ferns [11], regression trees [7], [8], [9]). This brings to light two major limitations of the CRM that we are going to fix 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 [12] and SIFT features for object recognition [13], 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* the best features out of an overcomplete set [7], [8], [9], [10]. However, this feature selection is suboptimal, since it is still performed on a

- *Wei Wang, and Nicu Sebe are with the Department of Information Engineering and Computer Science (DISI), University of Trento, Trento 38123, Italy. E-mail: wei.wang@unitn.it, sebe@disi.unitn.it.*
- *Sergey Tulyakov is with Snap Research, US. Email: stulyakov@snap.com.*

Manuscript received June 12, 2017; revised June 12, 2017 and October 2,2017; accepted February 9, 2018.

generic set. Recently, it has been shown for object detection [14], tracking [15], image labeling [16] and other fields that features learned for a specific problem using deep convolutional neural networks (CNNs) show much better performance. Moreover, features learned for image classification often generalize well for different tasks, showing the ability of CNNs (*e.g.*, AlexNet [14]), to learn a generic image representation.

The second limitation of the CRM is the independence of the regressors at each layer 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 [3] 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 Networks (RNNs) are extremely powerful in modeling the sequential inputs and outputs [17]. Given the advantage of dealing with image sequences, RNN has also been applied for face alignment in videos. For example, a recurrent encoder-decoder network has been proposed in [18] for video-based face alignment. In order to model long time-varying sequences, various RNN units have been proposed. In particular, Long-Short Term Memory (LSTM) cells and later Gated Recurrent Units (GRU) have been proved to be efficient in modeling time-varying processes and sequence-to-sequence learning [19]. In this paper, we employ the convolutional GRU as our recurrent unit since it has a simpler and lighter structure than LSTM while has better performance. Besides, different from the traditional GRU employed in [20], convolutional GRU replaces all the product operations with convolutional layers. Therefore, the input of the convolutional GRU are feature tensors instead of flattened feature vectors. Thus, the spatial structure of the input can be better preserved in the memory of the unit, and better performance can be obtained compared with [20]. Recently, vanilla recurrent unit has also been applied for face alignment [21]. However, the vanilla RNN suffers the the vanishing/exploding gradient problem. Therefore, the vanilla RNN can only be trained with limited recurrences while our convolutional GRU unit has no such restriction and can be trained with more recurrences. Moreover, it has been shown that using a CNN for feature extraction and an RNN for classification bring extra advantages [22], [23]. 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. Our framework consists of two modules, which are the recurrent module and the convolutional module.

The benefit of using RNN: As shown in Fig. 1, the CRMs learn the sequential regressors $\{\mathbf{R}_1(\cdot), \mathbf{R}_2(\cdot), ...\}$ independently. Once the regressors are obtained, they will be fixed rigidly. On the contrary, instead of learning a sequence of fixed regressors, our RCSR framework learns a model to generate the regressors dynamically. In other words, our framework can generate specific regressors $\{\mathbf{R}_1^i(\cdot), \mathbf{R}_2^i(\cdot), ...\}$ adaptively with respect to the specific input image i and the memory of the recurrent module. Besides, the traditional CRMs have the pre-mature problem, *i.e.*, in the testing phase, the predicted shape might have already been accurate after the first several regressions, and the following redundant regressors may harm the accuracy. However, the predicted landmarks have to go through all the regressors learned in the training phase. Different from [20], we propose a new stopping criterion based on the shape increment. Thus, our model is able to automatically decide when to terminate the recurrence and it is also capable of generalizing beyond the fixed number of recurrent steps in the testing phase.

The benefit of end-to-end learning: Most traditional regression methods use hand-crafted features directly, such as HOG and SIFT [3], [5], and few of them try to learn the discriminative features jointly with the regressor [10]. Instead of splitting the feature and the regressor learning processes, we design an end-to-end unified framework by replacing the manually hand-crafted feature descriptor $\mathbf{F}_t(\cdot, \cdot)$ by learning a patch-based CNN and plug it into the RNN module. One advantage of using CNN is that it can learn task-specific features automatically. In addition, it has been proved that the features extracted by CNN have the state-of-the-art performance in various tasks. Usually, the CNN model learned from one task can be transferred to other tasks by fine tuning the model with the new domain data. In this paper, we take the super resolution CNN (SRCNN) and plug it into our framework. The reason why we use SRCNN is because of its light structure (it contains only 3 convolutional layers) and its good performance in dealing with image patches. Thus, we can obtain an end-to-end learning framework. The parameters of both the CNN module and the RNN module are learned jointly. Thus, the end-to-end learning strategy can encourage the system to learn the task specific features which are dedicated for face alignment. The experimental evaluation we detailed 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 the works in face alignment as well as the recent advances in the convolutional and recurrent neural networks which are the main bricks to formulate our recurrent convolutional shape regression (RCSR) framework.

2.1 Face alignment

Face alignment and registration are very important for face analysis, such as face aging [24], [25] and smile video generation [26]. According to the widely accepted classification, face alignment methods can be grouped into three broad categories [27]: Active Appearance Models (AAMs), Constrained Local Models (CLMs) [28], and Cascaded Regression Methods (CRMs). Initial works on face alignment such as ASMs [1] and AAMs [2], 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 any unseen subject. Although such methods still attract the attention of researchers, the more recent CRMs have shown higher accuracy at impressive frame rates [9], [10]. In the following, we focus on this latter group of works.

Initially CRMs were introduced in the medical image processing community for anatomic structure prediction [29]. 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 [11], a method for cascaded pose regression was introduced. The authors used *pose-indexed features* and learned a sequence of weak-regressors (random ferns

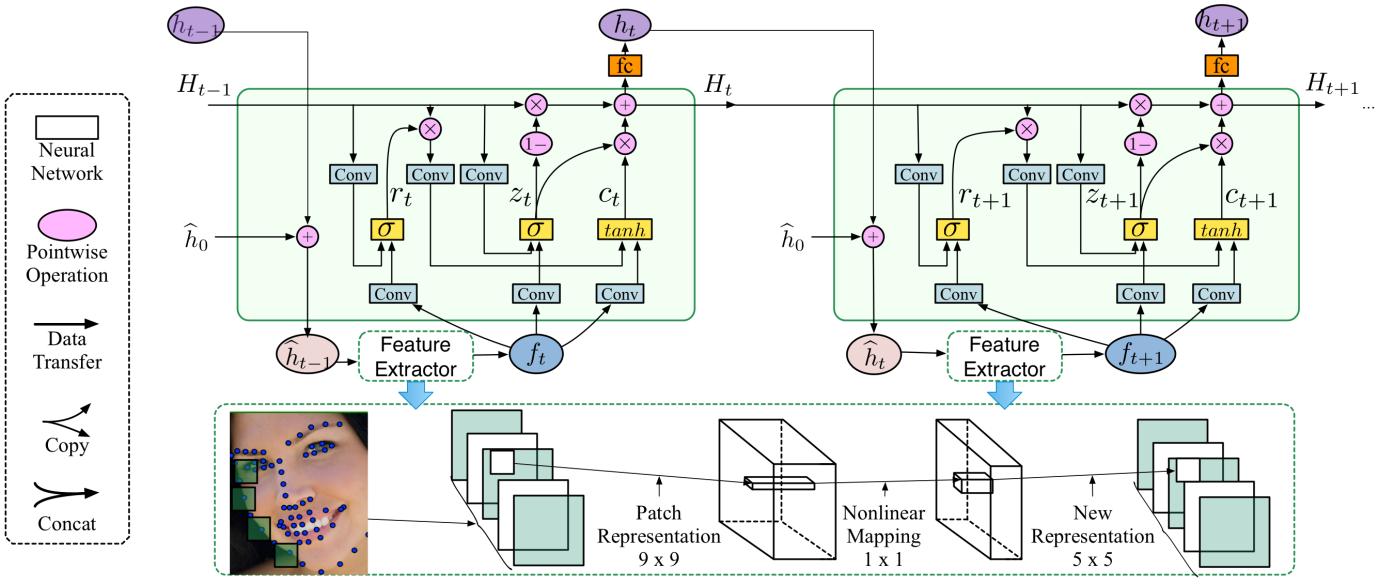


Fig. 2: The overview of the proposed approach. Top: the RNN with gated recurrent units unrolled in time. Bottom: the CNN architecture used for feature extraction. Note that feature extraction is performed at every recurrent step.

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 [30]. Importantly, it was shown that regressing the whole shape imposes the result to lie in the space constructed by all the training images. Recently, multiple CRM-based 3D methods have been proposed [7], [8], [31].

The supervised descent method (SDM) [3] 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 are applied sequentially to solve the optimization problem. The authors replace the feature extraction method with SIFT [32] 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 [6] by introducing a global SDM to automatically learn several descent maps at every layer of the cascade to handle complex cost-functions. These ideas were extended in [4], where the authors learn both the Jacobian and the Hessian matrices, in a manner inspired by the Gauss-Newton optimization method. Similar to the original SDM, the authors use hand-crafted SIFT features extracted around the keypoints. A different strategy for feature extraction is used in [7], [8], [9], [10]. Instead of employing manually hand-crafted features, they perform feature selection using a framework of regression trees. Alleviating the need to compute hand-crafted features, these works reach impressive processing speed.

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 [3], [4], HOG [6]. Some form of feature learning is employed in [7], [8], [9], [10], while each cascade layer in the second step remains 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, RCSR does not require manual supervision in defining feature extraction functions. Additionally, our method replaces a cascade of independent regressors by a single recurrent model, where all recurrent regressors are leaned jointly. This formulation merges the two steps of the typical CRM pipeline into a single unified framework and they are trained simultaneously. Different from our previous work [20], we apply a convolutional GRU instead of the traditional GRU as the recurrent unit. The convolutional GRU combines the strengths of both the CNN and RNN. The input to the convolutional GRU are feature tensors, instead of flattened feature vectors. Therefore, the spatial information is preserved in the memory of recurrent unit. Besides, we have also explored the appropriate depth of the convolutional layers in the feature extractor. Moreover, a new stopping criterion has been designed by analyzing the derivative of the shape increment. The framework could iterate beyond the fixed recurrence steps in the training phase, and the extra recurrences can improve the performance further. We also study the effect of the number of recurrent iterations during training on the performance of the final model. We then, evaluate different feature extraction models and show how they affect the performance. Finally, we visualize the learned CNN filters and feature maps.

2.2 Recurrent and convolutional neural networks

Recurrent Neural Networks (RNNs) have become increasingly popular to learn complex dynamic systems, because of their impressive capability to recurrently operate with sequential inputs. During the last decade, we have seen a lot of successes in applying RNNs to various application domains, such as machine translation [17], face aging [24], [25] and speech recognition [33]. Usually, the input and the output of the traditional convolutional neural networks have a fixed size. For example, AlexNet [14] takes a batch of images with the same size as the input, and its output is a probability matrix whose entry denotes the class probability of the input image. Most of the RNN architectures (*e.g.*, LSTM [34], GRU [19]) do not have such constraint. Their inputs and outputs

can either be sequences with different lengths or the fixed-size vectors or matrices.

The most significant feature of RNN is its capability to capture the dependencies. The underlying reason why RNN is capable of processing sequential signals is attributed to its hidden state. During each recurrence of the traditional RNN [35], an input signal is mapped to the hidden state, which is passed forward to the next recurrence. In this way, the information of the previous states is memorized and persists during the whole process, and the hidden state is always considered as the memory of the network. The output of the network is calculated based on the new input and the memory. Therefore, RNNs have been proved to have an advantage in modeling sequences with long-term dependencies. LSTM and GRU are better at capturing long-term dependencies than the traditional vanilla RNNs as vanilla RNNs have the vanishing/exploding gradient problem [36].

In the traditional deep neural networks, each layer has its unique parameters. On the contrary, the recurrent module in a RNN shares the same parameters across all the recurrent steps. Thus, this setting reduces the amount of parameters that need to be learned. Given the success of RNNs, a lot of RNN variants have been explored, such as the LSTM [34], GRU [19], and Clockwork RNN [37]. 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 the necessary information for future recurrences. Although the combination of gates/states varies depending on the selected architectures, each subsequent recurrence is performed in the same way, by processing a new input using the information of the hidden state. These architectures show varying performances for different tasks. In [38] it was shown that, in general, GRU-based models feature superior performance compared with other architectures.

Convolutional Neural Networks (CNNs) have recently demonstrated notable success in multiple tasks, such as image classification [14] and image super-resolution [39]. 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 with the commonly adopted hard-crafted features, in numerous application domains. Thus, it is very promising to combine the RNN together with the CNN into a hybrid architecture. This hybrid architecture has been successfully applied to many tasks, such as scene labeling [22] and object recognition [23].

Our framework is different from these tasks [22], [23]. First, the main recurrent module in our model is GRU which is free of the vanishing/exploding gradient problem. Second, the input for each recurrence is dynamic. The locations of the new input patches are determined by the predicted shape from the previous recurrence. In addition, the supervised descent method can be formulated as a special case of GRU.

3 METHOD

The overview of the proposed Recurrent Convolutional Shape Regression (RCSR) framework is given in Fig. 2. The framework mainly consists of two parts, the *recurrent module* and the *convolutional module*. During the t-th recurrence in the RNN, the current shape estimate $\mathbf{h}_t$ is imposed onto the image and the CNN is applied to the patches centered at the landmarks. The output of the last layer of the CNN is passed to the RNN as an input. During the first recurrence, the average shape of all the images in the training set is employed as the initial shape estimate: $\widehat{\mathbf{h}}_0 = \bar{\mathbf{S}}$.

3.1 Recurrent module

In the current study we use an RNN with a convolutional GRU module for its simplicity and superior performance as compared with other RNN types [38]. The convolutional GRU replaces all the product operations in the original GRU with convolutional layers. The convolutional GRU takes as input the feature tensors and process these tensors with convolutional layers while the traditional GRU takes as input the flattened feature vectors. Therefore, the convolutional GRU can preserve the spatial structure of the features in the memory compared with the traditional GRU. The structure of the recurrent module is given in Fig. 2. A GRU contains two gates (*i.e.*, the *reset* gate and the *update* gate) and one state (*i.e.*, the *hidden* state). The *hidden* state $\mathbf{H}_t$ is the memory of the RNN and $\mathbf{h}_t$ represents the overall shape increment after the adjustment in the t-th recurrence. Then the predicted shape after t recurrences is $\widehat{\mathbf{h}}_t{=}\widehat{\mathbf{h}}_0{+}\mathbf{h}_t$.

As shown in Fig. 2, the *reset* gate $\mathbf{r}_t$ controls whether the memory of the features from the previous recurrence should be ignored, *i.e.* if $\mathbf{r}_t$ is close to 0, the information of the previous landmark features will be forced to be discarded. Then the unit will focus on its current features without referring to the previous features. To sum up, the reset gate allows the unit to remember or drop the landmark features 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 memory which is implemented by the term $\mathbf{z}_t$, and the second one is to control the acceptance of the new input features which is implemented by the term $1 - \mathbf{z}_t$. As shown in Eq. 4, $\mathbf{z}_t$ is the output of a $sigmoid$ and takes values from 0 to 1. If $\mathbf{z}_t$ is close to 0, the memory from the previous recurrence will be kept and the new input features will be discarded. Thus, the new shape increment will be exactly the same as the previous one. However, if $\mathbf{z}_t$ is close to 1, $1 - \mathbf{z}_t$ will be close to 0, and the next shape increment will be mainly based on the new input features without referring to the memory.

The described process is schematically presented in Fig. 2. The feature extraction function $f_t{=}\mathbf{F}(\mathbf{I}, \widehat{\mathbf{h}}_t)$ is performed using a super resolution convolutional neural network, described in Section 3.2. Given f_t, a single recurrence is governed by the following equations:

$$\mathbf{z}_t = \sigma(\mathbf{W}_{zH} * \mathbf{H}_{t-1} + \mathbf{W}_{zf} * f_t + \mathbf{b}_z) \quad (4)$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_{rH} * \mathbf{H}_{t-1} + \mathbf{W}_{rf} * f_t + \mathbf{b}_r) \quad (5)$$

$$\mathbf{c}_t = tanh(\mathbf{W}_{cH} * \mathbf{r}_t \odot \mathbf{H}_{t-1} + \mathbf{W}_{cf} * f_t + \mathbf{b}_c) \quad (6)$$

$$\mathbf{H}_t = (1 - \mathbf{z}_t) \odot \mathbf{H}_{t-1} + \mathbf{z}_t \odot \mathbf{c}_t \quad (7)$$

where $\odot$ represents element-wise multiplication, and $*$ represents convolution operation. $\mathbf{c}_t$ denotes the new feature candidate created by the $tanh$ layer that could be merged into the current memory using Eq. (7). If the reset gate is always activated, the system will only have the short-term memory. If the update gate is always deactivated, the system can have the long-term memory and the previous features will be memorized. After fusing the current features with the memory of the previous features, an updated memory $\mathbf{H}_t$ can be obtained. By feeding the memory to a fully connected layer (fc in Fig. 2), we can calculate the shape increment $\mathbf{h}_t$ in t-th recurrence.

3.1.1 Progressive Recurrent Regression

Within this framework, the RNN acts as a refinement process which tries to find the optimal shape increment by changing the shape progressively. We use T recurrent steps to train the RCSR.

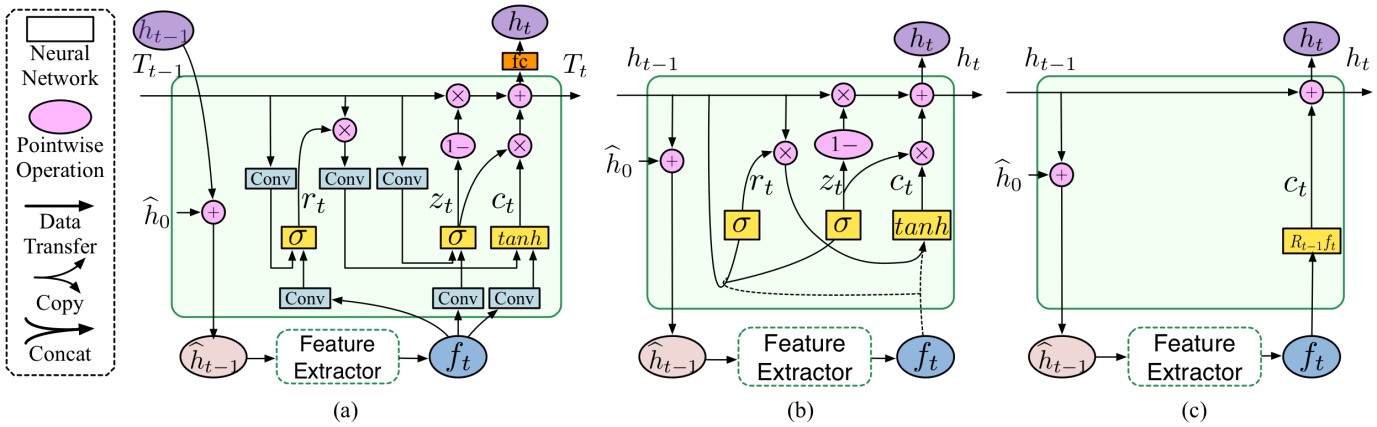


Fig. 3: The architectures of different regressors: (a) convolutional recurrent GRU Regressor (b) traditional recurrent GRU regressor (c) the traditional linear regressor. The linear regressor is a degenerate version of the traditional GRU regressor.

As a progressive refinement process, we expect the system to focus more on the latter recurrences. To achieve this target, we define a series of weights $\mathbf{w}=[w_1, w_2, ..., w_t, ..., w_T]$, each one of which corresponds to a single recurrence. These weights increase monotonically, therefore forcing the recurrent network to adjust the shape progressively and penalizing the model more for the error during the later recurrent steps. For simplicity, the weights can be designed as $w_t = 10^{\frac{t}{T}}$. Formally, the loss writes as:

$$J = \sum_{i=1}^{n}\sum_{t=1}^{T} w_t \|(\widehat{\mathbf{h}}_0 + \mathbf{h}_t^i) - \mathbf{h}_*^i\|_F^2, \tag{8}$$

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 recurrent steps, $\mathbf{h}_*^i$ is the target shape, the superscript i defines the i-th 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 [39]. We apply the SRCNN to the pixel values around the landmarks position as shown in Fig. 2. We denote the patch around a landmark as $\mathbf{Y}$, and use it as an input for the SRCNN. The SRCNN consists of three convolution layers, formulated as the following operations:

$$\begin{aligned} F_1(\mathbf{Y}) &= \max(0, \mathbf{W}_1 * \mathbf{Y} + \mathbf{B}_1) \\ F_2(\mathbf{Y}) &= \max(0, \mathbf{W}_2 * F_1(\mathbf{Y}) + \mathbf{B}_2) \\ F_3(\mathbf{Y}) &= \mathbf{W}_3 * F_2(\mathbf{Y}) + \mathbf{B}_3 \end{aligned} \tag{9}$$

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 $n_2 \times f_3 \times f_3 \times c = [32 \times 5 \times 5 \times 1]$. 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 a n_1 channel feature map to a n_2 channel feature map. Originally, the last layer in the SRCNN works as an averaging filter which projects a n_2 channel feature map to a high-resolution patch and take the average of the overlapping high-resolution patches. However, instead of projecting a n_2 channel feature map to a high-resolution patch, the last layer in our network will project a n_2 channel feature map to a feature space which can be then passed to the recurrent module directly.

3.3 Supervised descent method as GRU

In this section we show that the traditional GRU [20] as shown in Fig. 3 (b) is a generalization of the widely adopted Supervised Descent Method [3] as shown in Fig. 3 (c). Given a set of images $[\mathbf{I}^1, \mathbf{I}^2, ...\mathbf{I}^i, ..., \mathbf{I}^n]$, $\mathbf{h}^i$ denotes the shape of 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 shape $\mathbf{h}_*^i$. Then we have the following objective function for face alignment with respect to image $\mathbf{I}^i$,

$$\min \|\mathbf{F}(\mathbf{h}^i) - \mathbf{y}_*^i\|_2^2. \tag{10}$$

SDM applies the gradient descent rule to Eq. 10, and yields the following discrete update equation:

$$\begin{aligned} \mathbf{h}_t^i &= \mathbf{h}_{t-1}^i - \mathbf{R}_{t-1}(\mathbf{F}(\mathbf{h}_{t-1}^i) - \mathbf{y}_*^i) \\ &= \mathbf{h}_{t-1}^i - \mathbf{R}_{t-1}\mathbf{F}(\mathbf{h}_{t-1}^i) + \mathbf{R}_{t-1}\mathbf{y}_*^i, \end{aligned} \tag{11}$$

where $\mathbf{R}_{t-1} = \alpha\mathbf{F}'(\mathbf{x}_{t-1}^i)$, 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. 11 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. 11 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:

$$\mathbf{h}_t^i = \mathbf{h}_{t-1}^i - \mathbf{R}_{t-1}\mathbf{F}(\mathbf{h}_{t-1}^i) + \mathbf{b}_{t-1}, \tag{12}$$

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

$$\sum_i \|\mathbf{h}_*^i - \mathbf{h}_{t-1}^i + \mathbf{R}_{t-1}\mathbf{F}(\mathbf{h}_{t-1}^i) - \mathbf{b}_{t-1}\|^2 \tag{13}$$

where $\mathbf{h}_0^i$ is obtained using Monte Carlo integration. Thus, Eq. 13 can be considered as a special case of Eq. 8, making the SDM a special case of our GRU network.

Actually, the traditional linear regressor is equivalent to traditional 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:

$$\mathbf{h}_t^i = \mathbf{h}_{t-1}^i - \mathbf{R}_{t-1}\mathbf{F}(\mathbf{h}_{t-1}^i) \tag{14}$$

Eq. 14 is a recurrent version of Eq. 12 except for the term $\mathbf{b}_{t-1}$ which can be implemented by expanding the feature space by several columns set to 1.

After replacing the product operations in the traditional GRU with convolutional layers, we obtain the convolutional GRU shown in Fig. 3 (a). The inputs of the convolutional GRU are feature tensors while the inputs of the traditional GRU are the flattened feature vectors. Therefore, the spatial structure of the features can be well preserved in the memory of the convolutional GRU, while the traditional GRU totally discards it. 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 the memory regarding previous states, as it is shown in Eq. 13, every step has a separate loss function. In contrast, for our model the overall loss over all the recurrent steps is defined and the regressors are learned jointly by summing up the losses from all the steps (Eq. 8). 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: In this section, we first evaluate the performance of our algorithm using the 300-W dataset [40] with 68 landmark setting. We also evaluate our RCSR framework on the HELEN [41] and the LFPW [42] datasets with 49 landmark setting.

300-W dataset [40]: This dataset is a combination of several in-the-wild datasets, including AFW [43], LFPW [42], HELEN [41] and XM2VTS [44], that are annotated with 68-point marks in a consistent manner. Similar to previous works [5] [10], 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 which includes the testing images from the LPFW and HELEN, (ii) the *challenging* set which includes recently released 135 images also known as the IBUG set, and (iii) the *full* set which is a combination of the first two.

HELEN [41]: This dataset contains 2,330 high resolution face images which are downloaded from Flickr. 2,000 images are used for training and 330 images are used for testing. We conduct detection for both 68 and 49 landmark settings provided by [40].

LFPW [42]: The Labeled Face Parts in the Wild (LFPW) dataset originally contains 1000 training images and 300 test images. As only URLs are provided and some of them are invalid, only 811 training and 244 test images are available in [40]. We perform both 68 and 49 landmark detection on this dataset. These images are collected from internet search sites, and they are unconstrained: The images have different resolutions, various illumination conditions, and various facial expressions and poses. Some images have occlusions (*e.g.* eyes being occluded by glasses and hairs, face parts being occluded by hand or cellphone). Some images have artistic rendering effects.

Evaluation Metrics: To evaluate the performance, we follow the widely adopted evaluation metric [5], [7], [8], [10], 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

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. We also rely on these replicates to design our stopping criterion.

4.2 Deep CNN module for feature extraction

This module extracts features for all the image patches. It is suggested in [45] that CNN could benefit from deeper structures by adding more non-linear layers. Here we also explored deeper CNN modules. In [39], Dong *et al.* tried a 3-layer architecture for super resolution, but they did not observe superior performance even after weeks of training. Kin *et al.* in [46] proposed a much deeper CNN model (*i.e.*, 20 layers) than the one in [39] (*i.e.*, 3 layers), and conclude that the deeper CNN model is beneficial for super resolution task.

To see if the deeper CNN model is beneficial for our task, we exploit architectures detailed in [39], [46]. First, the SRCNN framework from [39] is adopted as our basic CNN module. SRCNN has three convolutional layers $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{W}_3$. 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. $\mathbf{W}_2$ is of the size $[64{\times}1{\times}1{\times}32]$ and $\mathbf{W}_3$ has the size of $[32{\times}5{\times}5{\times}1]$. It is suggested in [39] that larger filter size is better as it could grasp richer structural information. Following [39], we fixed the bottom and top layers ($\mathbf{W}_1, \mathbf{W}_3$) as they have large filter sizes. In front of the middle layer $\mathbf{W}_2$, we added compound convolutional layers $[\mathbf{W}_{2,1}, .., \mathbf{W}_{2,i}, ..., \mathbf{W}_{2,n}]$ proposed in [46] with the depth of n. The dimension of $\mathbf{W}_{2,i}$, $(i{=}1, 2, ..., n)$ is $[64{\times}3{\times}3{\times}64]$. We have tested with different depths $n{=}0, 5, 10, 15, 20$. After obtaining the feature maps from the CNN module for each image patch, we stack the feature maps of the 68 landmarks and feed the feature tensor to the RNN module. For the RNN module, we set the total number of recurrent steps T to be 5. We have also tested a different number of fully connected layers on top of the RNN module. The number of fully connected layer is set in the range of $[1, 2, 3]$.

Fig. 5 shows the performance and running time of different combinations of CNN module and fully connected layers. From Fig. 5, we can observe that the smallest error is obtained when the depth is set to 10 with one fully connected layer (*i.e.* depth-10-fc-1). Compared with the case where the depth is set to 0 with one fully connected layer (*i.e.*, depth-0-fc-1), the error stays almost the same, however, the computation time increases dramatically from 0.44 to 5.22 seconds. Therefore, depth-0-fc-1 should be a good

Fig. 4: Landmark localization for 5 recurrent steps. On the left, we show some examples, for which 5 recurrences are sufficient, while the examples on the right require additional recurrences.

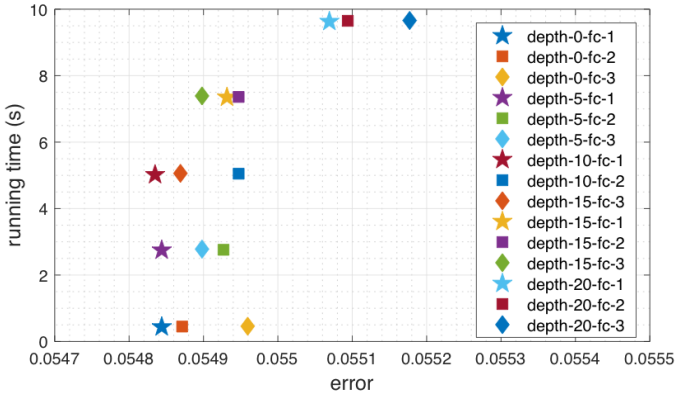


Fig. 5: Comparison between different combinations of the depth of the augmented CNN layers and fully connected layers: running time vs. error for the 300W dataset.

choice as it has a good trade-off between accuracy and speed. In addition, we also observe that adding more CNN layers and fully connected layers does not always bring better performance. Similar behavior was reported in [39].

4.3 Understanding when to stop iterating

One of the further advantages of our RCSR is that the model can be easily extended beyond the learned number of recurrent steps without the need of retraining the whole pipeline. More importantly, there is no upper bound on the number of recurrent steps one can perform. This, however, requires a stopping strategy which knows when to stop for a testing image. Fig. 4 shows several qualitative examples of different number of recurrent steps required for different testing examples. The examples on the left show that 5 steps are sufficient to localize the landmarks, as one can easily see that the landmarks on the jawline in the first row fit perfectly after the fifth recurrence. A similar observation can be made for the subject shown in the second row. The examples on the right show cases when 5 recurrences are insufficient, due to difficult illumination conditions and extreme head poses. Therefore, for the images shown on the left of Fig. 4, the iteration should be stopped as the predicted landmarks have already converged while the images shown on the right of Fig. 4 require more iterations. To explore the appropriate number of iterations, we have studied the relation between the error and the number of iterations. The results are plotted in the first row of Fig. 6.

Fig. 6 shows the convergence curve of the algorithm. The first row shows the average error against the recurrent steps. We can observe that for challenging cases the error continues to decrease when the recurrence number is greater than 13, while for the easy ones it remains stable between 5-th and 9-th recurrences and then goes up. This observation demonstrates that for easy images from the common set, the redundant recurrences actually harm the performance, while the hard cases from the challenging set benefit from more recurrences (Fig. 6, top middle). For the full set, the average error curve is similar to the common set (Fig. 6, top right). Therefore, the model should require less recurrences for an image with frontal face which has no expressions, while difficult examples which have extreme poses or challenging expressions may need more recurrences. We can observe that more iterations do not always bring better performance. This is especially true for the easy cases with frontal poses and good lighting conditions. During the testing phase, the regressor behaves normally and the error decreases if the number of iterations is the same as during the training phase. However, if the number of iterations is larger, the behavior of the regressor can be unpredictable as it goes far beyond the learned number of iterations.

During training we set the number of recurrent iterations to 5. As shown in Fig. 4, this is sufficient to regress the shape from an initial coarse shape to a shape which is closer to the ground truth, but the regressor has limited ability to further fine-grain the shape to make it more accurate. The regressor does not know what to do when the predicted shape becomes closer to the ground truth. Therefore, the error raises with more recurrences. One solution to this problem is to design a stopping criterion for the recurrence. Another solution to prevent the error from raising is to train the regressor with more recurrences. With more recurrences, the regressor will have the chance to be closer to the ground truth and as such will be able to learn what to do when the predicted shape is closer to the ground truth. The experiments of training the regressor with more recurrences are available in Section 4.4.

In many works (*e.g.*, [33], [47]) the early stopping criteria are heuristic and they are learned from the data. In other words, the stopping criterion is a data-driven approach. How to design a stop-

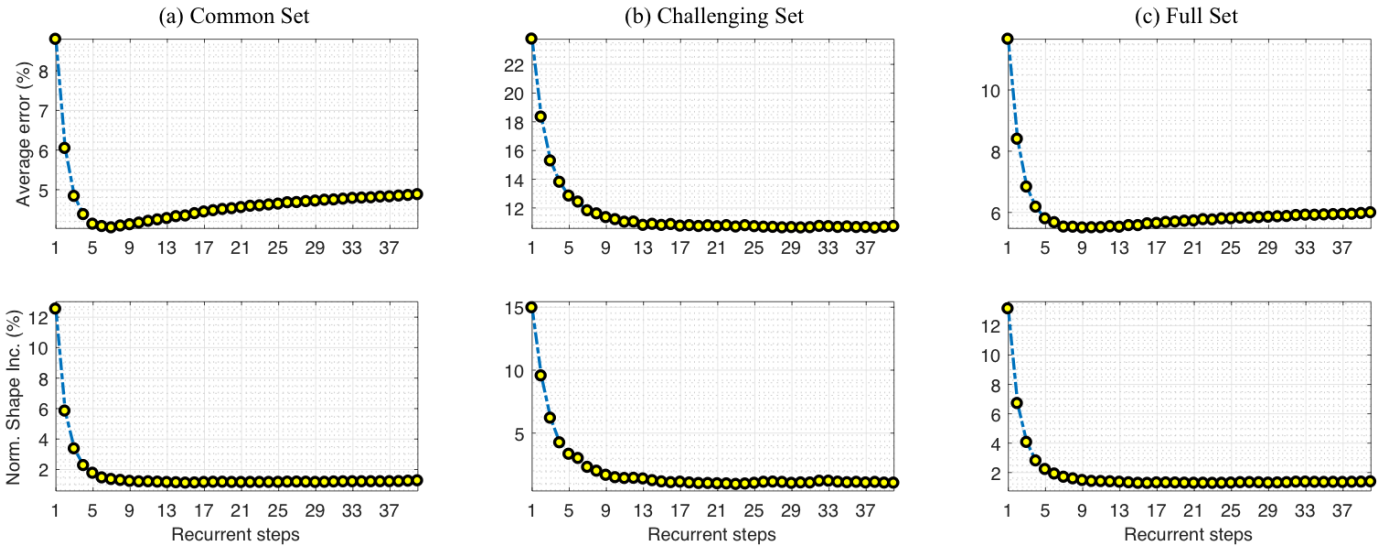


Fig. 6: Average error & normalized shape increment vs the number of recurrent steps.

ping criterion with a sound theory is still an open problem. Similar to [33], [47], we also rely on a heuristic method to design the stopping criterion. Even though there is no theoretical guarantee to be perfect, in practice, it is a satisfactory solution. From Fig. 4, we can observe that when the predicted shape $\mathbf{h}_t$ is far away from the ground truth, the predicted shape increment $\mathbf{h}_t - \mathbf{h}_{t-1}$ will be very large in the next recurrence. When the predicted shape is close to the ground truth, the shape increment will be very small and will oscillate around the ground truth. In this paper, we rely on the shape increment to design the stopping criterion. The second row in Fig. 6 shows the normalized shape increment against recurrent steps. After t-th recurrence, we can obtain the updated shape $\mathbf{h}_t$. The shape increment from the $(t-1)$-th to the t-th recurrence is defined as $\Delta\mathbf{h}_t = \mathbf{h}_t - \mathbf{h}_{t-1}$. We normalize the shape increment by dividing the l_2 norm of the shape increment with the distance between the outer corners of the eyes. In the first row in Fig. 6, we can observe that when the average error reaches its minimum, the normalized shape increment in the second row also becomes stable and remains as a small value. Therefore, if the normalized shape increment is smaller than a certain threshold and close to 0, we can say that the prediction difference is stable. Then we stop iterating. To set the threshold value, one simple option is to take the normalized shape increment when the minimum error is achieved. We observe that for the common, challenging and full test set, the value of the normalized shape increment is around 0.01 when the minimum error is reached. Therefore, we set the threshold γ to 0.01. In case the normalized shape increment is always greater than the threshold, we also set the maximum recurrence number. The maximum recurrence number is set to 13. This simple stopping strategy allows our model to automatically decide when to terminate the recurrence. Table 1 shows the performance of the proposed method and the baselines on the 300W dataset. We can observe that this simple stopping criterion can decrease the error by 3.10% from 4.10 to 4.01 on the common subset, by 2.39% from 8.79 to 8.58 on the challenging subset, and by 2.40% from 5.02 to 4.90 on the full subset.

4.4 Training with more recurrent iterations

Alternatively to designing a stopping criterion we train the RNN with varying number of recurrent iterations. We explored different recurrent steps n which are $[5, 10, 20, 30]$. The weight for the loss of i-th recurrence is set to 10^{w_i}. w_i is obtained by evenly sampling n values between -2 and 2. The weights of the loss in each recurrence are set following Section 3.1.1. The whole framework focuses more on the rear recurrences as the weights increase monotonically. We test different recurrent steps on the complete 300-W dataset [40].

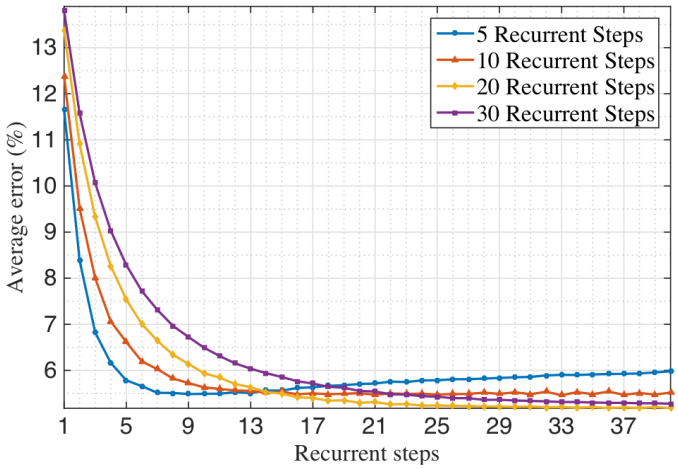


Fig. 7: Performance of RCSR trained with different recurrent steps (*i.e.*, 5, 10, 20, 30) on the full test set of the 300-W dataset.

Fig. 7 shows the results of the 4 models which are trained with 5, 10, 20, and 30 recurrent steps. For convenience, we name the 4 models as RCSR5, RCSR10, RCSR20 and RCSR30. We note that the models which are trained with more recurrent steps converge slower than those trained with less recurrent steps. However, the models which are trained with more recurrent steps are more stable. As shown in Fig. 7, the RCSR5 reaches its best performance around 9 steps and more recurrences harm its performance. On the other hand, RCSR10, RCSR20 and RCSR30 reach their best performance after more than 17 steps and then

their performances remain stable, and more recurrences do not harm their performances. We can also observe that RCSR20 has slightly better performance than RCSR30, and has marginal improvement compared with RCSR10. Therefore, more recurrent steps improve the performance and give a more stable behavior, but the performance gain is limited and is not monotonic with respect to the number of iterations. More importantly, the performance of the regressor trained with 10, 20 and 30 iterations are more stable, compared to 5 recurrent iterations.

4.5 Experimental results

We report evaluation results on the three subsets of the 300-W dataset in Table 1. It compares three different results of the same RCSR model against best performing state-of-the-art methods. The reported RCSR results are obtained using 5, 10 recurrent steps 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 recurrences 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.

TABLE 1: Landmark localization results on the 300-W dataset.

Method	Common	Challenging	Full
Zhu *et al.* [43]	8.22	18.33	10.20
DFMF [48]	6.65	19.79	9.22
ESR [49]	5.28	17.00	7.58
RCPR [50]	6.18	17.26	8.35
SDM [3]	5.57	15.40	7.50
Smith *et al.* [51]	-	13.30	-
Zhao *et al.* [52]	-	-	6.31
GN-DPM [53]	5.78	-	-
CFAN [54]	5.50	-	-
ERT [9]	-	-	6.40
LBF [10]	4.95	11.98	6.32
LBF fast [10]	5.38	15.50	7.37
CFSS [5]	4.73	9.98	5.76
CFSS Practical [5]	4.79	10.92	5.99
RCFA [20]	4.03	9.85	5.32
RCSR 5 recurrences	4.05	9.74	5.15
RCSR 10 recurrences	4.10	8.79	5.02
RCSR adaptive	**4.01**	**8.58**	**4.90**

We can observe from Table 1 that RCSR sharply reduces the error of CFSS [5] by of 15% on the full test set. Besides, we only use the mean-shape computed from the training set as the initial shape while CFSS [5] applies the multiple-shape-searching strategy. This observation validates the robustness of our RCSR method compared with other methods. We believe, the multiple-shape-search strategy may further improve the performance of our method. Fig. 8 shows the qualitative results for the images taken from the full set. Clearly, due to the 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 RCSR can handle faces with severe occlusions, including sun-glasses, hands and hats.

We also test our methods on HELEN and LFPW datasets. The results are shown in Table 2 and 3. For the 68 landmark setting, we reduce the error by 10% and 19% compared with CFSS [5]. For the 49 landmark setting, we reduce the error by 10% and 6% compared with CFSS [5].

TABLE 2: Landmark localization results on the HELEN dataset.

Method	68 landmarks	49 landmarks
Zhu *et al.* [43]	8.16	7.43
DFMF [48]	6.70	-
RCPR [50]	5.93	4.64
SDM [3]	5.50	4.25
GN-DPM [53]	5.69	4.06
CFAN [54]	5.53	-
CFSS [5]	4.63	3.47
CFSS Practical [5]	4.72	3.50
RCFA [20]	4.65	3.45
RCSR adaptive	**4.16**	**3.13**

TABLE 3: Landmark localization results on the LFPW dataset.

Method	68 landmarks	49 landmarks
Zhu *et al.* [43]	8.29	7.78
DFMF [48]	6.57	-
RCPR [50]	6.56	5.48
SDM [3]	5.67	4.47
GN-DPM [53]	5.92	4.43
CFAN [54]	5.44	-
CFSS [5]	4.87	3.78
CFSS Practical [5]	4.90	3.80
RCFA [20]	4.74	3.69
RCSR adaptive	**3.93**	**3.57**

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. Our framework can work better than the other approaches for these images not only because our RNN network can learn the dependencies between each regressor, but also because it learns the location dependencies between the landmarks. Thus, even though parts of the face are occluded, our framework can still predict with reasonable accuracy the location of the occluded landmarks based on other landmarks. As shown in some examples in Fig. 8, even if some faces are partially occluded by the hands and glasses, the occluded landmarks can still be roughly estimated based on the features from the other non-occluded landmarks.

5 Selecting Recurrent-Convolutional Architecture

The proposed framework consists of two parts: the convolutional part, that learns a feature mapping centered at the predicted landmarks and the recurrent part, that performs localization and refinement of these landmarks. In the following we show that such combination is advantageous. First, we demonstrate the effect of the recurrent module, by learning it on conventional computer vision features. Second, we show that the convolutional module brings further advantages by improving the performance. Third, we report evaluation results of three different architectures of the convolutional part, supporting the selected architecture.

5.1 Effect of the recurrent module

As discussed in Section 3.3, SDM is a degenerate case of the traditional GRU. To show the advantages of the recurrent regressor over the classical SDM cascade, we employ the same feature descriptor for both of them. We remove the convolutional part of the

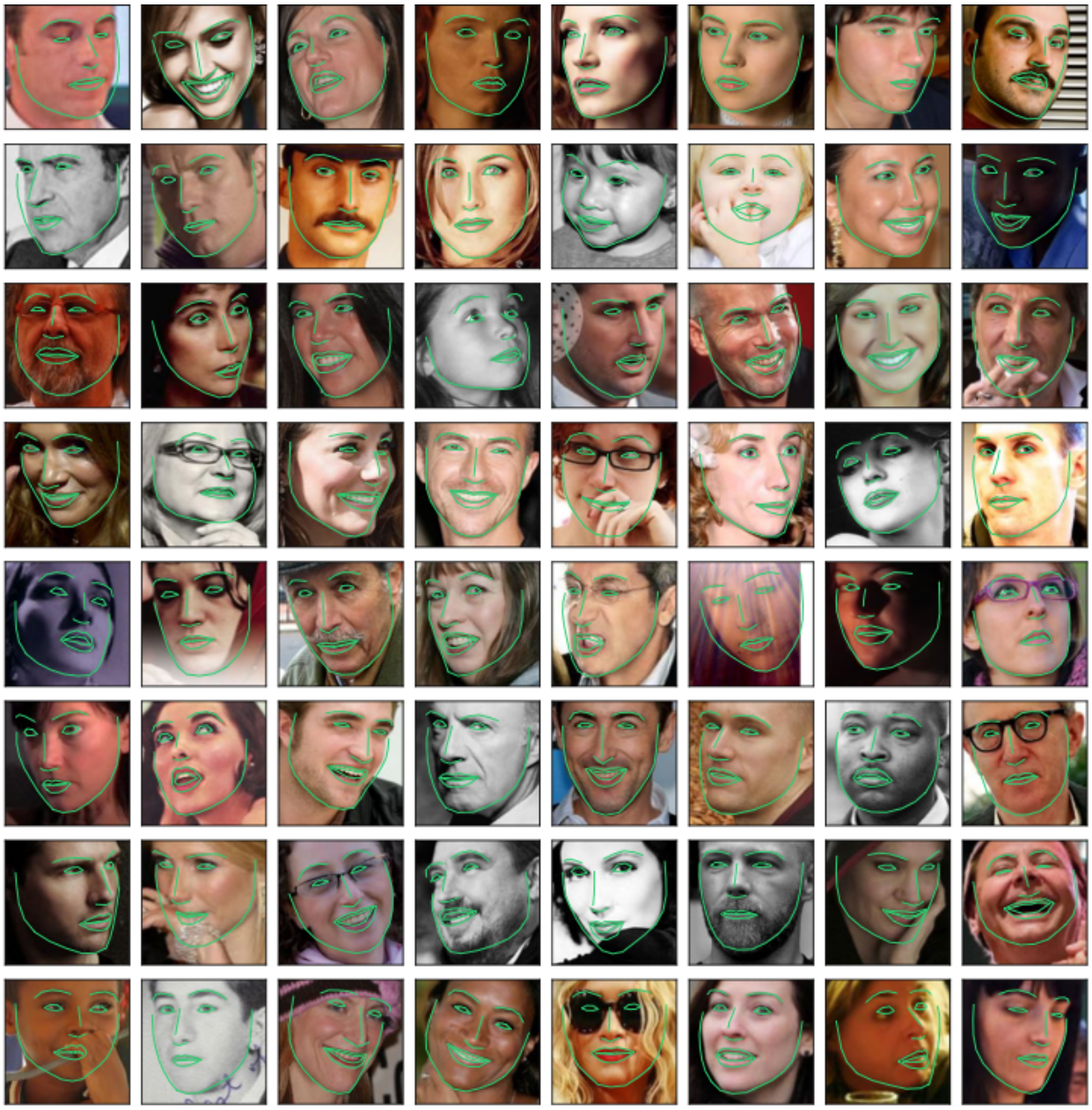


Fig. 8: Selected qualitative examples taken from the full set of the 300-W dataset.

TABLE 4: Evaluation of the effect of various recurrent-convolutional architectures (best results in bold). The last row shows the framework presented in the current paper. The third row shows the framework in the earlier work [20].

Method	Common	Challenging	Full
Linear Regressor + HOG (SDM [3])	5.70	15.77	7.68
Traditional GRU + HOG	4.61	9.94	5.63
Traditional GRU + SRCNN [20]	4.03	9.85	5.32
Convolutional GRU + LeNet	4.63	9.48	5.59
Convolutional GRU + FAUC	4.32	8.96	5.24
Convolutional GRU + SRCNN	**4.01**	**8.58**	**4.90**

proposed framework and replace it with HOG features extracted at patches centered at the landmarks' locations. Similarly, we also use HOG features in the SDM pipeline. Thus the only difference between the two pipelines is the recurrent module.

Comparison results are reported in the first two rows of Table 4. Clearly, the recurrent module offers significantly better performance, as compared to SDM. The classical SDM is known for performing poorly on hard examples, *i.e.*, those having non-frontal faces, facial expressions and with difficult illumination conditions. Therefore, the performance gain of the recurrent module is especially evident on the challenging set of the 300-W dataset, which contains these hard cases.

We further compare the performance of our newly introduced convolutional GRU and the traditional GRU presented in the earlier work [20] to show the effectiveness of combining convolutional neural network and the recurrent unit. We do not report the performance of convolutional GRU with HOG features, as the hand-crafted HOG features do not contain any spatial information, which is required when spatial convolutions are employed.

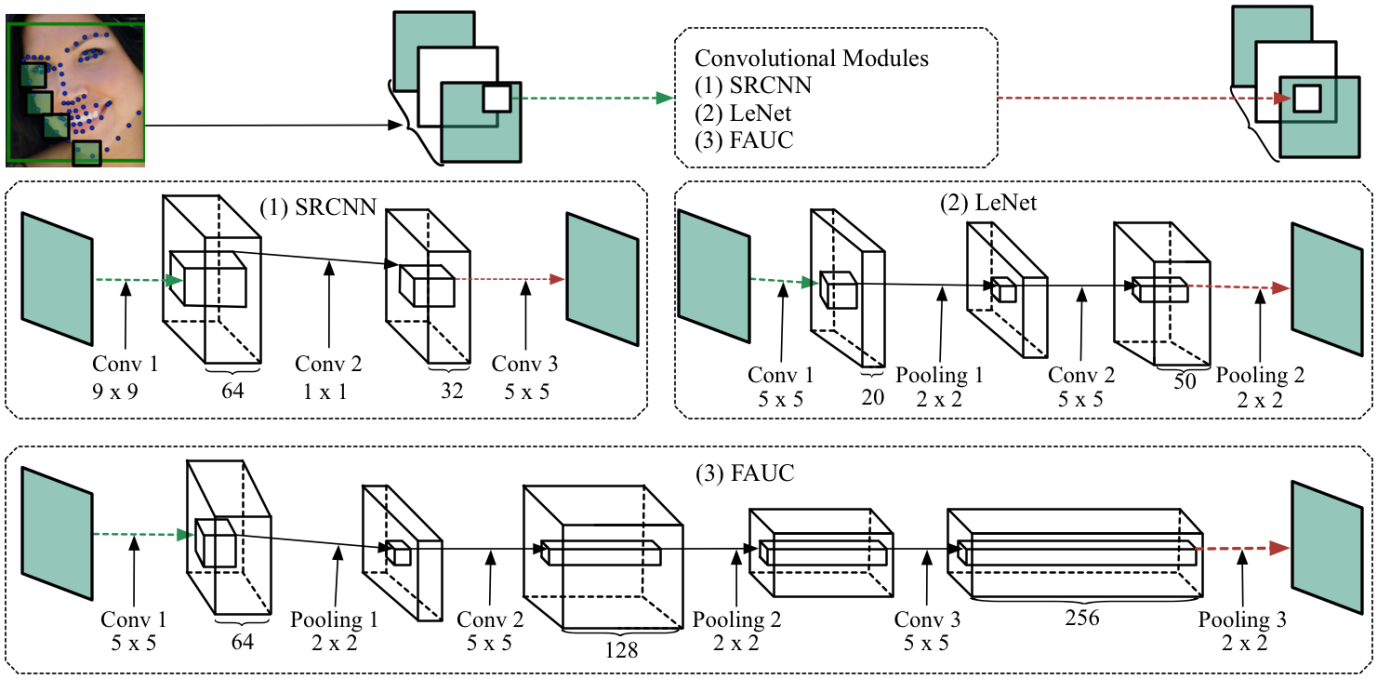


Fig. 9: Different convolutional modules for feature extraction which are (1) the super resolution convolutional neural network (SRCNN); (2) the LeNet (3) the facial action unit classification (FAUC) neural network.

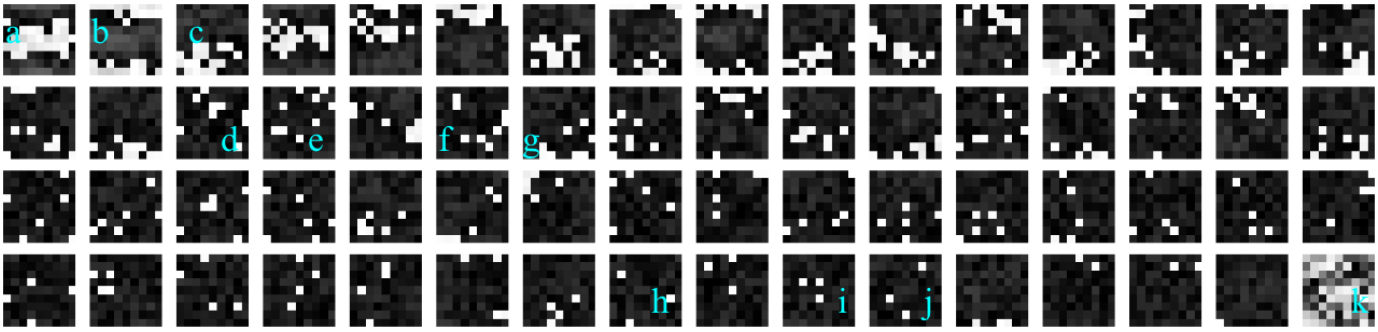


Fig. 10: The first-layer filters trained on the image batches. The filters are organized in the descending order of their respective variances. The size of each filter is 9×9.

Table 4 shows the error for each method evaluated on the 300W dataset. Since convolutional GRU combines the advantages of both convolutional and recurrent neural networks its performance is superior to the traditional GRU. Compared with the traditional GRU, the Covolutional GRU decreases the error by 0.50% from 4.03 to 4.01 on the common subset, 12.89% on the challenging subset from 9.85 to 8.58 and 7.89% on the full subset from 5.32 to 4.9. As the convolutional GRU is more computationally expensive, inference time increases from 0.44 seconds to 0.50 seconds. Depending on the application, increased inference time can be considered acceptible, as performance gain of the convolutional GRU is significant.

We visualized the feature maps of the output tensors of the convolutional GRU in Fig. 11. From Fig. 11, we can observe that the feature maps contain clear geometric meanings (*e.g.* wrinkles and shadings) demonstrating that the convolutional GRU can preserve the spatial structures of the input.

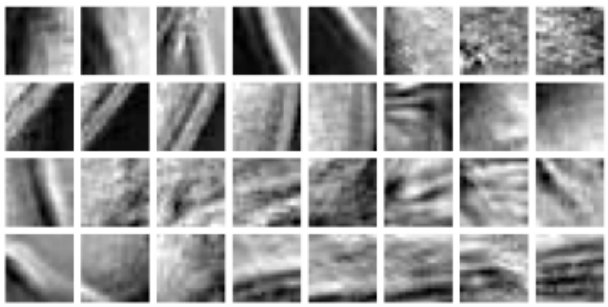


Fig. 11: Feature maps of the output of the convolutional GRU.

5.2 Effect of the convolutional module

If we use the convolutional module instead of HOG feature descriptor, performance improves even further (row 3), as compared to the version with HOG features. This supports that end-to-end learning is beneficial, since the features can be learned adaptively for the landmark localization task. To study the effect

of the selected SRCNN architecture, we explore several different convolutional architectures, widely adopted by the community. Since very deep neural networks require large amounts of training data, a key criterion for evaluating a particular architecture is its size. Faces do not have much variability as compared to more general computer vision tasks, such as object detection, therefore the capacity of a light neural network is sufficient. To this end, we select two neural networks as baselines, which are Facial Action Unit Classification (FAUC) Network [55] and LeNet [56]. The details of the network structures are shown in Fig. 9.

Facial Action Unit Classification (FAUC) network [55] has been originally designed for facial expression classification using as inputs 96×96 gray scale images. With the help of visualization technique introduced in [57], the image region which has the strongest activation towards the neurons can be reconstructed by deconvolutional neural network. It is observed that the learned convolutional filters in the learned FAUC network correspond to different facial action units. As shown in Fig. 9, FAUC network consists of 3 convolutional layers with $64, 128, 256$ filters, and each filter has the size of 5×5. Each convolutional layer is followed by the ReLU activation function, and a pooling layer. The first two pooling layers are max-pooling layers, while the last pooling layer is the quadrant pooling layer [58]. We follow the same network settings as in [55], where the bias terms of the convolutional filters are removed. Since we are not dealing with the facial expression classification problem, we removed the fully connected layer which is used for classification, leaving only feature extraction layers.

LeNet [56] has been originally designed for digit recognition using as inputs 28×28 gray scale images. It consists of two convolutional layers with $20, 50$ filters, and each of the filter has the size of 5×5. Each convolutional layer is followed by a max-pooling layer as shown in Fig. 9. The last four layers are the fully connected layer, ReLU, the fully connected layer and the loss layer. Similar to the FAUC network, the last four layers are removed as we are not interested in the digit classification problem. Thus, we can obtain the feature extraction layers.

The details of the SRCNN are illustrated in Section 3.2. Combining the convolutional GRU module with different convolutional modules (SRCNN, LeNet, and FAUC network), we can obtain three different structures. Table 4, shows that the SRCNN has the best performance. The reason why LeNet has inferior performance is because the max-pooling layer loses the information during feature extraction process. The architecture of LeNet is also the simplest among all the convolutional modules which is another reason for its poor performance, as it is not powerful enough to learn powerful feature representations. The FAUC has slightly better performance compared with the LeNet, but it is worse than SRCNN. The structure of FAUC is more complex than SRCNN, and it has more parameters which makes it much easier for FAUC to overfit the data.

6 Analysis of Learned Representations

In this section, we first visualize the filters of the convolutional module learned in the network. Then we analyze the connections between the convolutional module and the local binary feature descriptor which has very good performance among the hand-crafted feature descriptors. Finally, the benefit of stacking the features of all the landmarks is discussed.

6.1 Visualization of the convolutional module

Fig. 10 shows the learned first-layer filters of the SRCNN module which are quite different from the filters trained for the super-resolution task [39]. We can observe that most of the filters, such as a, b, and c in the first row focus more on regions while the rest of the filters focus more on some selected pixels. It is also very interesting to observe that many filters are symmetric with respect to the diagonals. For example, d, f, g and h are symmetric with respect to the principle diagonal while e, i, and j are nearly symmetric with respect to the counter-diagonal with a tilted angle. The reason for the tilted symmetry is probably because most of the edges in which the facial landmarks lie are tilted and the pixels on both sides of the edges are required to be activated in order to judge whether the pixel is a landmark or not. It is also very interesting to notice that the last filter k has all its pixels activated. This filter takes the sum of all the pixels, and this is equivalent to capturing the illumination information.

Among the hand-crafted features, the local binary feature descriptor has very good performance for landmark detection. Even though the convolutional module (SRCNN) and the local binary feature descriptor are quite different, they share a lot in common. First, both extract features based on the local patches instead of the whole image. Second, they utilize the features of all the landmarks to learn a global feature mapping function.

6.2 Why using local patches?

The reason why the local features are preferable is well illustrated in [10] where the local binary features are designed as the input to the regressor. The task is to predict the offset Δs between the current landmarks and the target landmarks. The intuition is that the size of the local patches should depend on the distribution of the Δs. Let σ denote the standard deviation of Δs, it is revealed in [10] that the optimal size of local patches has a linear relationship with σ, which is roughly twice as large as σ. Either too large or too small patch size will lead to inferior performance, and the optimal patch size should gradually shrink from the early stage to the later stage as the predicted landmarks becoming closer to the ground truth [10]. Thus, different local binary feature descriptors need to be trained at different stages. However, in our convolutional module, the image patch size is fixed and remains the same at different stages, but we can achieve similar shrinkage effect with the help of convolutional layers. As shown in Fig. 10, many filters have very sparse values so that these filters will focus on very small regions of the patches, and very few pixels will be employed to generate features. For example, only 3 pixels are activated for filters h, i and j. This is very similar to [9] [10] where they take the pixel pair differences as features. With the help of the filters, the local feature learning is self-adaptive in different stages.

6.3 Why using global feature mapping?

In our pipeline, the outputs of the convolutional module are the features of the patches which are centered at the predicted landmarks. Instead of considering features of each patch individually, we stack all feature maps to form a global feature representation and feed it into the recurrent module. Then there is only one single recurrent regressor trained to predict the offsets for all the landmarks. One main reason to use the global features is that the global information can enforce the global shape constraint and

make the predictions robust to the local errors caused by occlusions. As shown in Fig. 8, even though some faces are occluded by the hands and glasses, the shape can still be maintained well and the positions of the occluded landmarks can be roughly estimated based on the features from the other non-occluded landmarks.

7 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. 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 presented RCSR method employs a convolutional GRU as the recurrent unit instead of the traditional GRU. This allows the spatial structure of the features to be better preserved in the memory and thus leads to better performance.

The proposed RCSR method still 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 [5]. 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 human pose estimation.

References

[1] T. F. Cootes and C. J. Taylor, "Active shape models - 'smart snakes'," in *British Machine Vision Conference*, 1992.

[2] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, no. 6, pp. 681–685, 2001.

[3] X. Xiong and F. De La Torre, "Supervised descent method and its applications to face alignment," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2013.

[4] G. Tzimiropoulos, "Project-out cascaded regression with an application to face alignment," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2015.

[5] S. Zhu, C. Li, C. Change, and X. Tang, "Face alignment by coarse-to-fine shape searching," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2015.

[6] X. Xiong and F. D. Torre, "Global supervised descent method," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2015.

[7] S. Tulyakov and N. Sebe, "Regressing a 3d face shape from a single image," in *IEEE Int. Conf. on Computer Vision*, 2015.

[8] S. Tulyakov, S. L. Jeni, J. Cohn, and N. Sebe, "Viewpoint-consistent 3d face alignment," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2018.

[9] V. Kazemi and S. Josephine, "One millisecond face alignment with an ensemble of regression trees," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.

[10] S. Ren, X. Cao, Y. Wei, and J. Sun, "Face alignment at 3000 fps via regressing local binary features," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.

[11] P. Dollar, P. Welinder, and P. Perona, "Cascaded pose regression," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2010.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2005.

[13] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE Int. Conf. on Computer Vision*, 1999.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Int. Conf. on Neural Information Processing Systems*, 2012.

[15] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Int. Conf. on Neural Information Processing Systems*, 2013.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.

[17] M. Auli, M. Galley, C. Quirk, and G. Zweig, "Joint language and translation modeling with recurrent neural networks." in *Conference on Empirical Methods in Natural Language Processing*, 2013.

[18] X. Peng, R. S. Feris, X. Wang, and D. N. Metaxas, "A recurrent encoder-decoder network for sequential face alignment," in *Eur. Conf. on Computer Vision*, 2016.

[19] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv*, 2014.

[20] W. Wang, S. Tulyakov, and N. Sebe, "Recurrent convolutional face alignment," in *Asian Conference on Computer Vision*, 2016.

[21] G. Trigeorgis, P. Snape, M. A. Nicolaou, E. Antonakos, and S. Zafeiriou, "Mnemonic descent method: A recurrent process applied for end-to-end face alignment," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2016.

[22] P. H. O. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene parsing," *arXiv*, 2013.

[23] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2015.

[24] W. Wang, Z. Cui, Y. Yan, J. Feng, S. Yan, X. Shu, and N. Sebe, "Recurrent face aging," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2016.

[25] W. Wang, Y. Yan, Z. Cui, J. Feng, S. Yan, and N. Sebe, "Recurrent face aging with hierarchical autoregressive memory," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2018.

[26] W. Wang, X. , Alameda-Pineda, X. Dan, E. Ricci, and N. Sebe, "Every smile is unique: Landmark-guided diverse smile generation," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2018.

[27] N. Wang, X. Gao, D. Tao, and X. Li, "Facial feature point detection: A comprehensive survey," *arXiv*, 2014.

[28] X. Yu, J. Huang, S. Zhang, W. Yan, and D. N. Metaxas, "Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model," in *IEEE Int. Conf. on Computer Vision*, 2013.

[29] S. K. Zhou and D. Comaniciu, "Shape regression machine," in *Int. Conf. on Information Processing in Medical Imaging*, 2007.

[30] X. Cao, "Face alignment by explicit shape regression," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2012.

[31] C. Cao, Q. Hou, and K. Zhou, "Displaced dynamic expression regression for real-time facial tracking and animation," in *SIGGRAPH*, 2014.

[32] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[33] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[35] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[36] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE trans. on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[37] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork rnn," in *Int. Conf. on Machine Learning*, 2014.

[38] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Int. Conf. on Machine Learning*, 2015.

[39] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

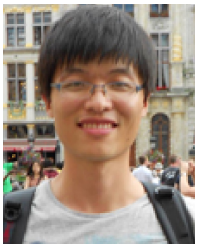
[40] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *IEEE Int. Conf. on Computer Vision Workshops*, 2013.

[41] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive facial feature localization," in *Eur. Conf. on Computer Vision*, 2012.

[42] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2930–2940, 2013.

[43] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2012.

[44] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre, "Xm2vtsdb: The extended m2vts database," in *Second international conference on audio and video-based biometric person authentication*, 1999.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2016.
[46] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2016.
[47] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv*, 2014.
[48] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, "Robust discriminative response map fitting with constrained local models," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2013.
[49] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, 2014.
[50] X. Burgos-Artizzu, P. Perona, and P. Dollár, "Robust face landmark estimation under occlusion," in *IEEE Int. Conf. on Computer Vision*, 2013.
[51] B. Smith, J. Brandt, Z. Lin, and L. Zhang, "Nonparametric context modeling of local appearance for pose-and expression-robust facial landmark localization," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.
[52] X. Zhao, T.-K. Kim, and W. Luo, "Unified face analysis by iterative multi-output random forests," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.
[53] G. Tzimiropoulos and M. Pantic, "Gauss-newton deformable part models for face alignment in-the-wild," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.
[54] J. Zhang, S. Shan, M. Kan, and X. Chen, "Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment," in *Eur. Conf. on Computer Vision*, 2014.
[55] P. Khorrami, T. Paine, and T. Huang, "Do deep neural networks learn facial action units when doing expression recognition?" in *IEEE Int. Conf. on Computer Vision Workshops*, 2015.
[56] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[57] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Eur. Conf. on Computer Vision*, 2014.
[58] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Int. Conf. on artificial intelligence and statistics*, 2011.



Wei Wang received the master degree from the University of Southern Denmark. He is currently a PhD student in Multimedia and Human Understanding Group with the Department of Information Engineering and Computer Science (DISI) in the University of Trento, Italy. His research interests include computer vision and deep learning. In particular, he is interested in face and human pose analysis.

Sergey Tulyakov, PhD is a Senior Research Scientist at Shap Research. His research focuses on computer vision, machine learning and face analysis, including 2D and 3D detection, tracking, pose estimation, heart rate estimation from videos, 3D face alignment techniques, with particular emphasis on realistic capturing conditions. He has co-organized the 3D Face Alignment in the Wild workshop held in conjunction with ECCV in 2016. He received his PhD degree from the University of Trento, Italy.

Nicu Sebe is with the Department of Information Engineering and Computer Science, University of Trento, Italy, where he is leading the research in the areas of multimedia information retrieval and human behavior understanding. He was GC of FG 2008 and ACM Multimedia 2013, and PC of ACM Multimedia 2007 and 2011, ECCV 2016 and ICCV 2017. He is a Senior Member of ACM and IEEE and a fellow of IAPR.