上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Face Alignment by Deep Convolutional Network with Adaptive Learning Rate

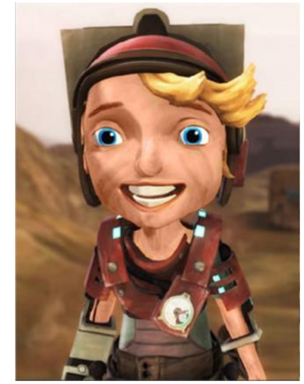
Zhiwen Shao, Shouhong Ding, and Lizhuang Ma

Shanghai Jiao Tong University



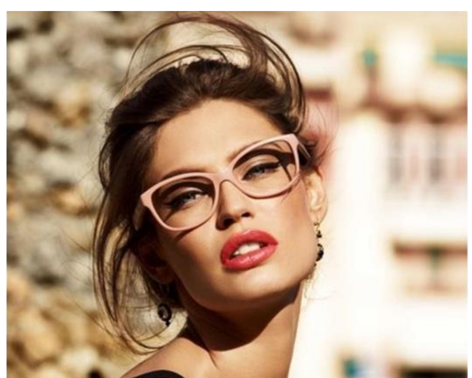

Applications

Face animation [1]



Face beautification

Face preprocessing



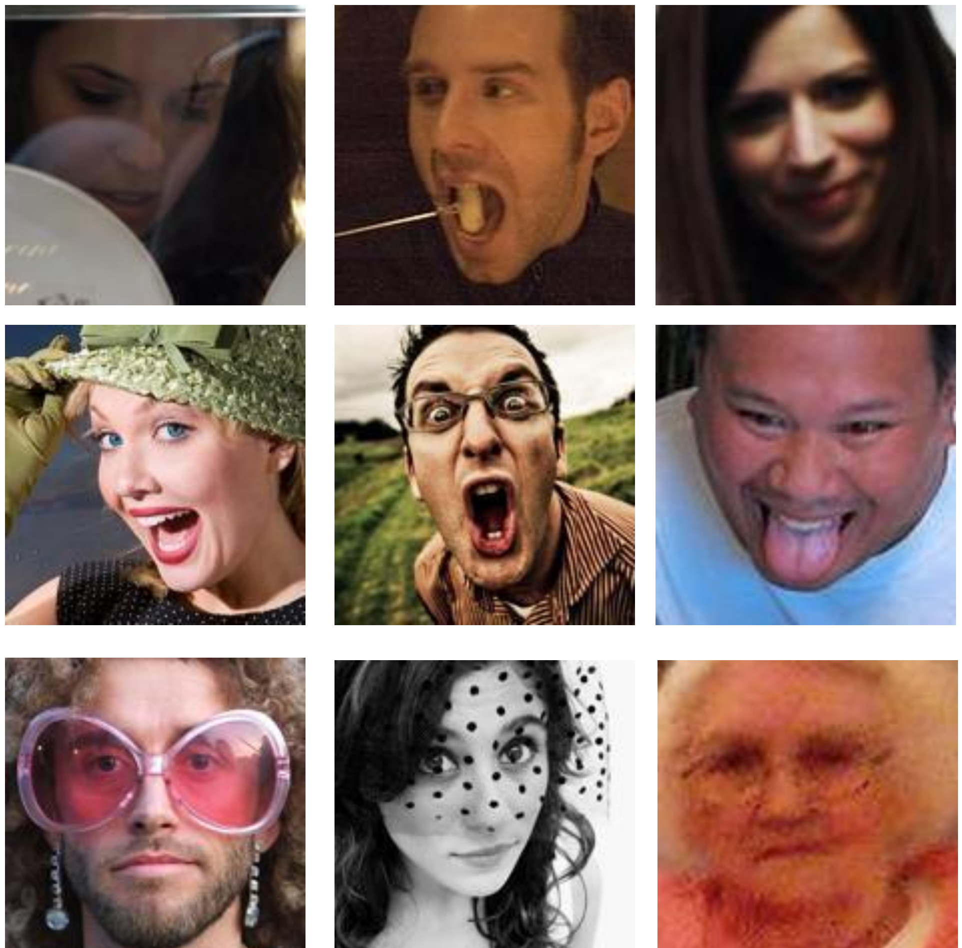
[1] The two images were downloaded from https://www.mixamo.com/faceplus

Challenges

Large pose, illumination and expression variations

Partial occlusion

Low quality

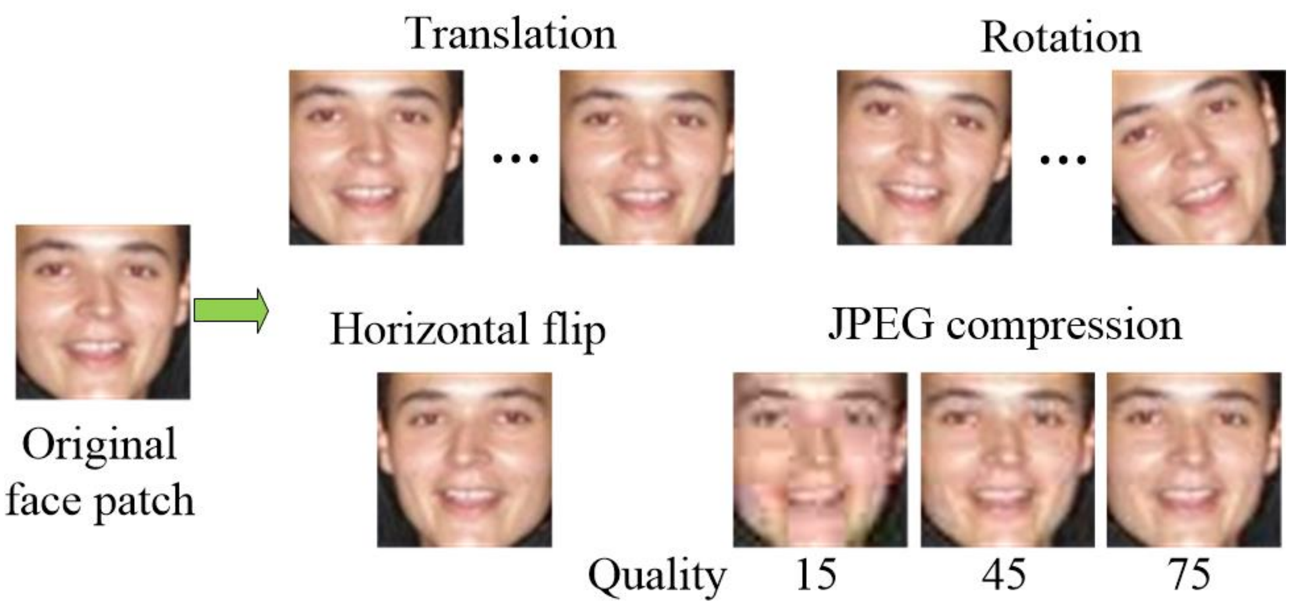


How to solve these problems

Deep convolutional neural network

a small number of training images annotated with landmarks

We propose an effective data augmentation strategy



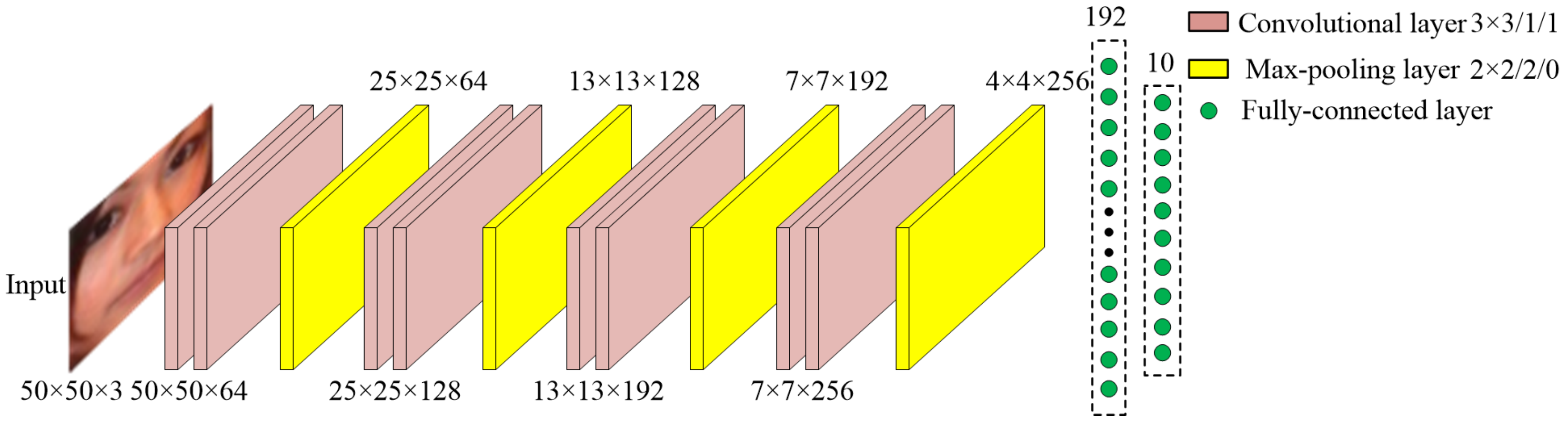
Data augmentation

Training data are enlarged dozens of times

Also

Translation	improve the robustness of landmark detection in the condition of tiny face shift
Rotation	adapt pose variation of in-plane rotation
JPEG compression	be robust to poor-quality images

Deep convolutional network

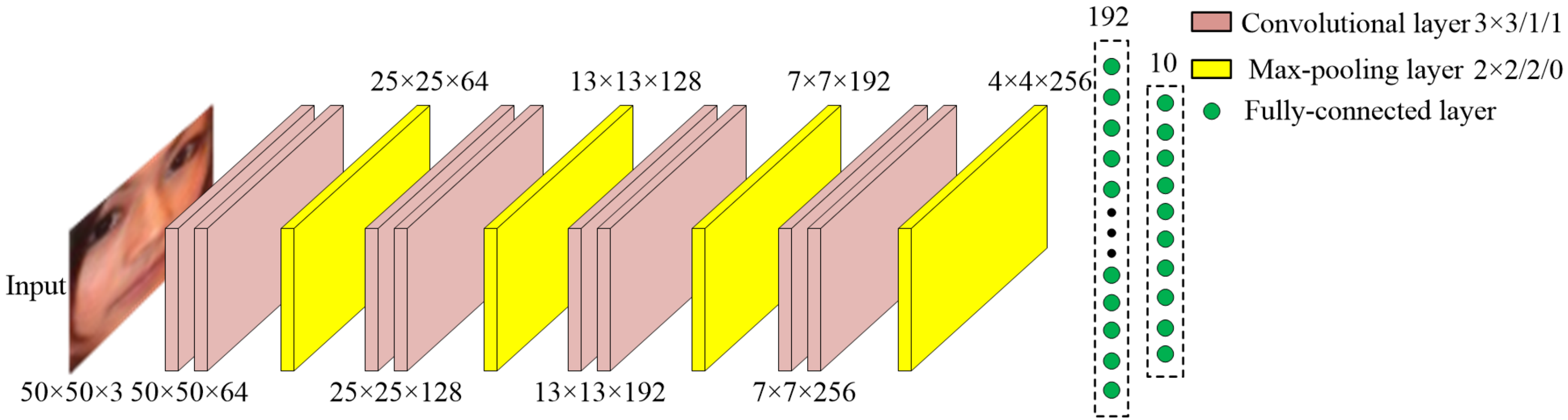


- ➢ Eight convolutional layers followed by two fully-connected layers

Convolution operation:

$$y^j = \sum_i k^{ij} * x^i + b^j$$

Deep convolutional network

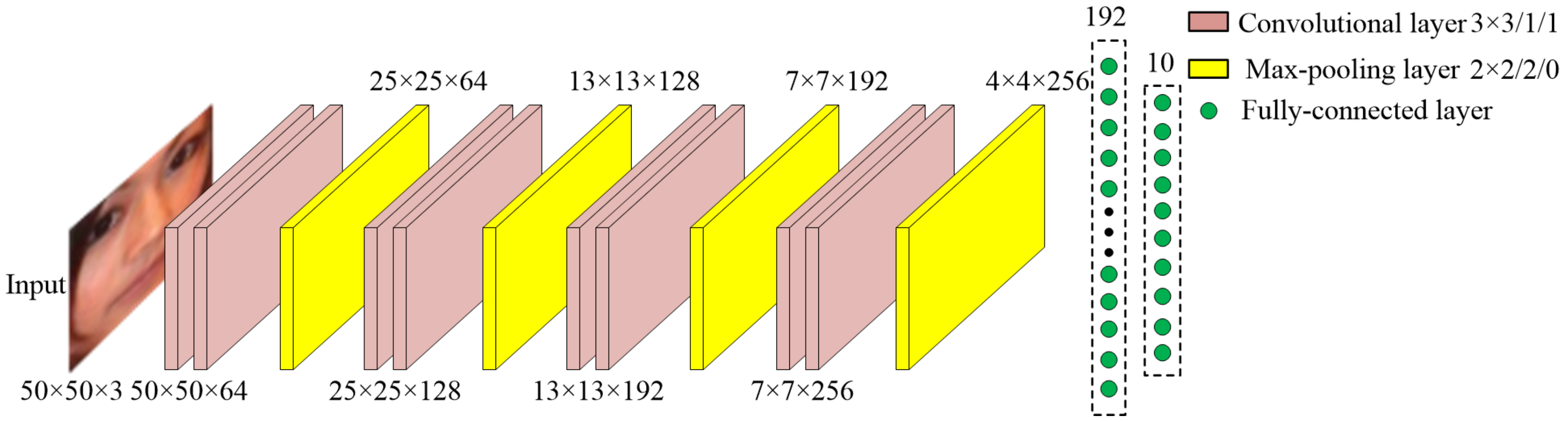


- Every two continuous convolutional layers connect with a max-pooling layer

Max-pooling operation:

$$y^i_{j,k} = \max_{0 \le m,n < h} \{x^i_{j \cdot h+m,\, k \cdot h+n}\}$$

Deep convolutional network



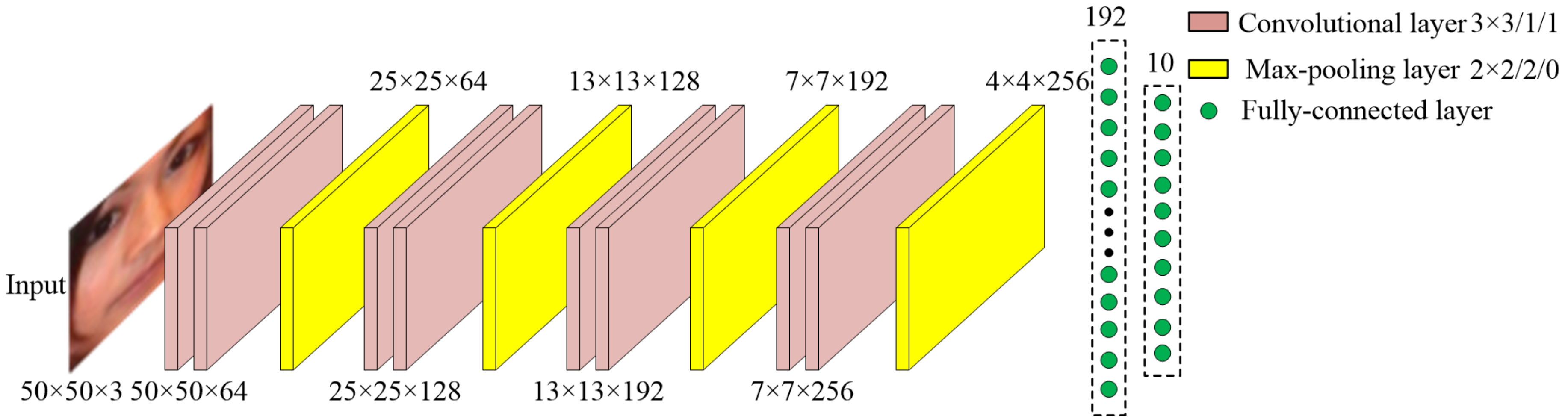
- VGG net: continuous convolutional layers, jointly extract complex features

Batch normalization $y = \gamma \dfrac{x - E(x)}{\sqrt{Var(x)}} + \beta$

ReLU nonlinearity $y = \max(0, x)$

⇒ accelerate training

Deep convolutional network



- The network input is $50 \times 50 \times 3$ for color face patches
- The output is predicted coordinates of five landmarks

Shrink the coordinates ⇨ guarantee numerical stability
reduce computational cost

Euclidean loss: $L = \frac{1}{2}(f - \hat{f})^2$

Deep convolutional network

Initially the value of Euclidean loss may be several hundred or even several thousand

⇩

Initial learning rate must be very small

⇩

converge slowly

search the solution in a small range

How?

Adaptive learning rate

Adaptive learning rate

Algorithm 1 The training algorithm with adaptive learning rate.

Input: Network N with trainable initialized parameters Θ_0, training set Ω, validation set Φ, control parameters α, t, k.

Output: Trainable parameters Θ.

1: Testing N and calculating the loss L_0 on Φ;
2: Setting the learning rate $\eta = \alpha / L_0$ and calculating the loss L on Ω;
3: **while** $L > t$ **do**
4:     Training N with back propagation (BP) [20] algorithm and calculating L;
5:     **if** l hasn't been reduced for k iterations **then**
6:         $\eta = \eta \cdot 0.1$;
7:     **end if**
8: **end while**
9: Setting $\eta = \alpha / L$;
10: **while** not convergence **do**
11:     Executing step 4 to 7;
12: **end while**

We firstly assign learning rate depended on initial testing loss, to avoid the network link weights changed sharply

Adaptive learning rate

Algorithm 1 The training algorithm with adaptive learning rate.

Input: Network N with trainable initialized parameters Θ_0, training set Ω, validation set Φ, control parameters α, t, k.

Output: Trainable parameters Θ.

1: Testing N and calculating the loss L_0 on Φ;
2: Setting the learning rate $\eta = \alpha / L_0$ and calculating the loss L on Ω;
3: **while** $L > t$ **do**
4:     Training N with back propagation (BP) [20] algorithm and calculating L;
5:     **if** l hasn't been reduced for k iterations **then**
6:         $\eta = \eta \cdot 0.1$;
7:     **end if**
8: **end while**
9: Setting $\eta = \alpha / L$;
10: **while** not convergence **do**
11:     Executing step 4 to 7;
12: **end while**

When network loss was reduced significantly, changing learning rate to be a larger value

Adaptive learning rate

Algorithm 1 The training algorithm with adaptive learning rate.

Input: Network N with trainable initialized parameters Θ_0, training set Ω, validation set Φ, control parameters α, t, k.

Output: Trainable parameters Θ.

```
1:  Testing N and calculating the loss L_0 on Φ;
2:  Setting the learning rate η = α/L_0 and calculating the loss L on Ω;
3:  while L > t do
4:      Training N with back propagation (BP) [20] algorithm and calculating L;
5:      if l hasn't been reduced for k iterations then
6:          η = η · 0.1;
7:      end if
8:  end while
9:  Setting η = α/L;
10: while not convergence do
11:     Executing step 4 to 7;
12: end while
```

The algorithm consists of two iterative procedures for adaptive learning rate decrease

Experiments

Datasets and evaluation metric

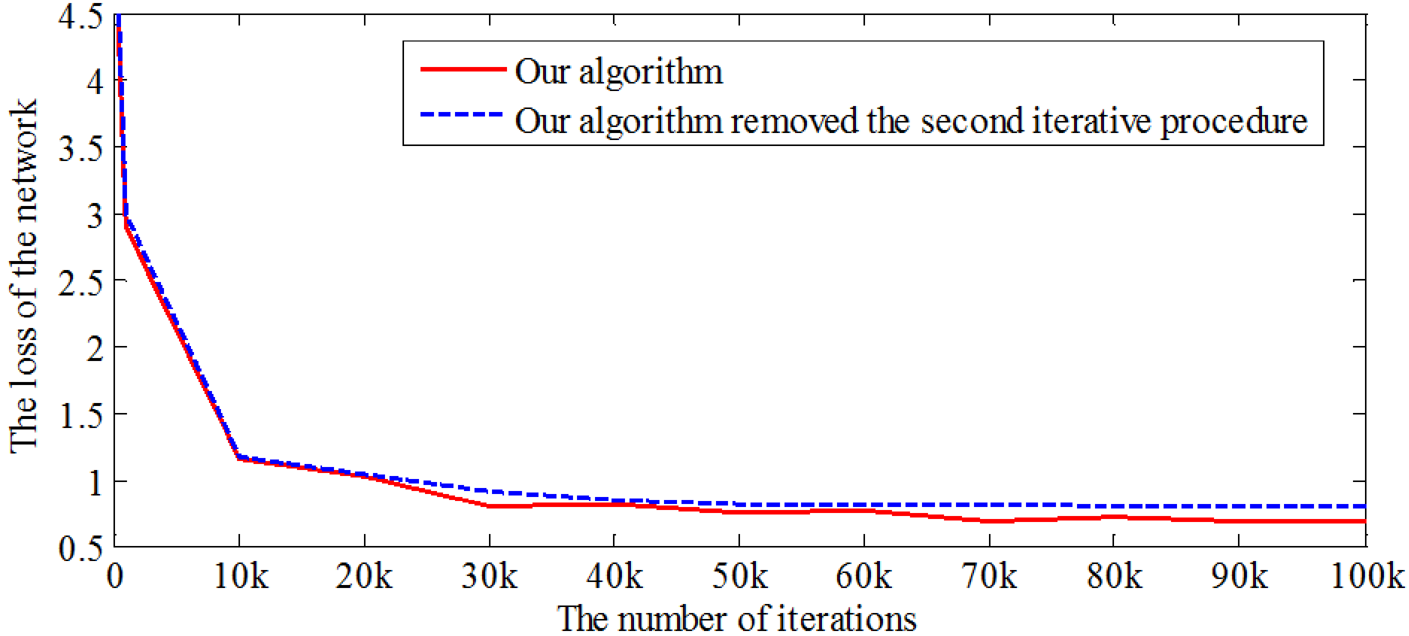
- Datasets

LFPW	1432 face Images, we use 1030 images as same as cascaded CNN [9]
AFLW	25993 face Images, we use 2995 images as same as TCDCN [17]

- Evaluation metric: mean error

the distance between estimated landmarks and the ground truths,
normalized with the inter-pupil distance,
averaged over all landmarks and images

Algorithm discussions

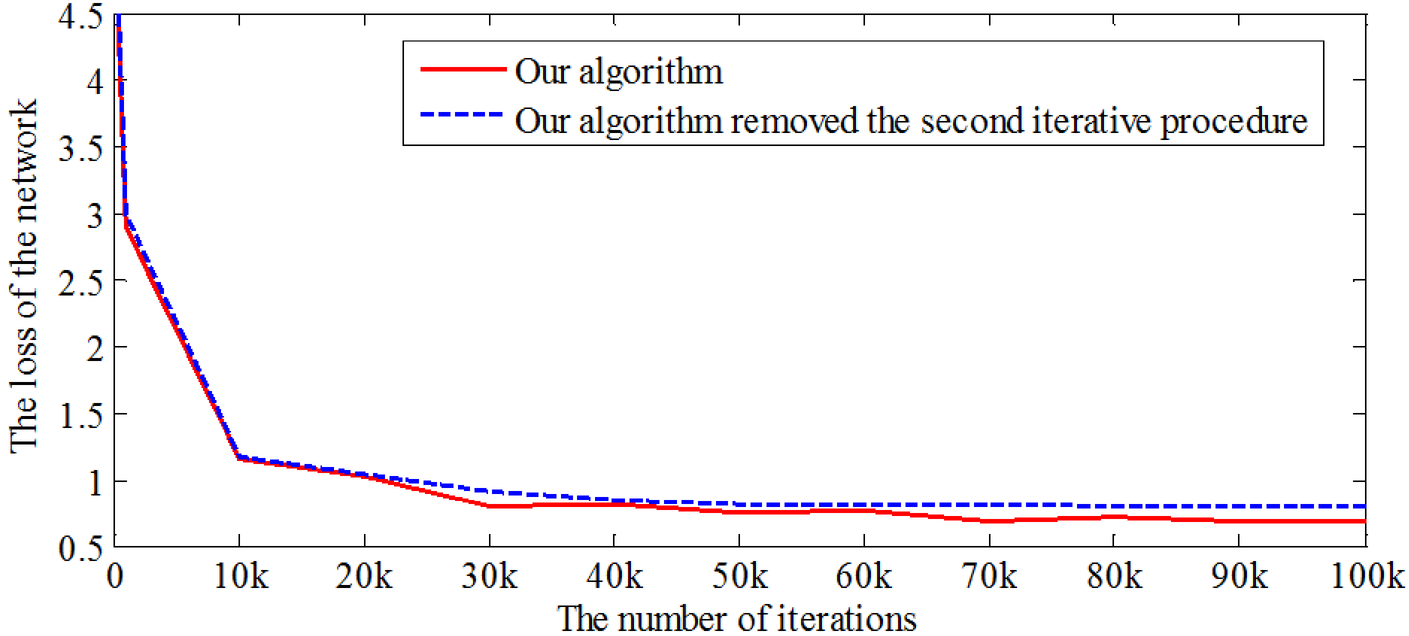


Train our network with adaptive learning rate algorithm

VS.

Train our network with only one iterative procedure (the learning rate is gradually reduced from a small value)

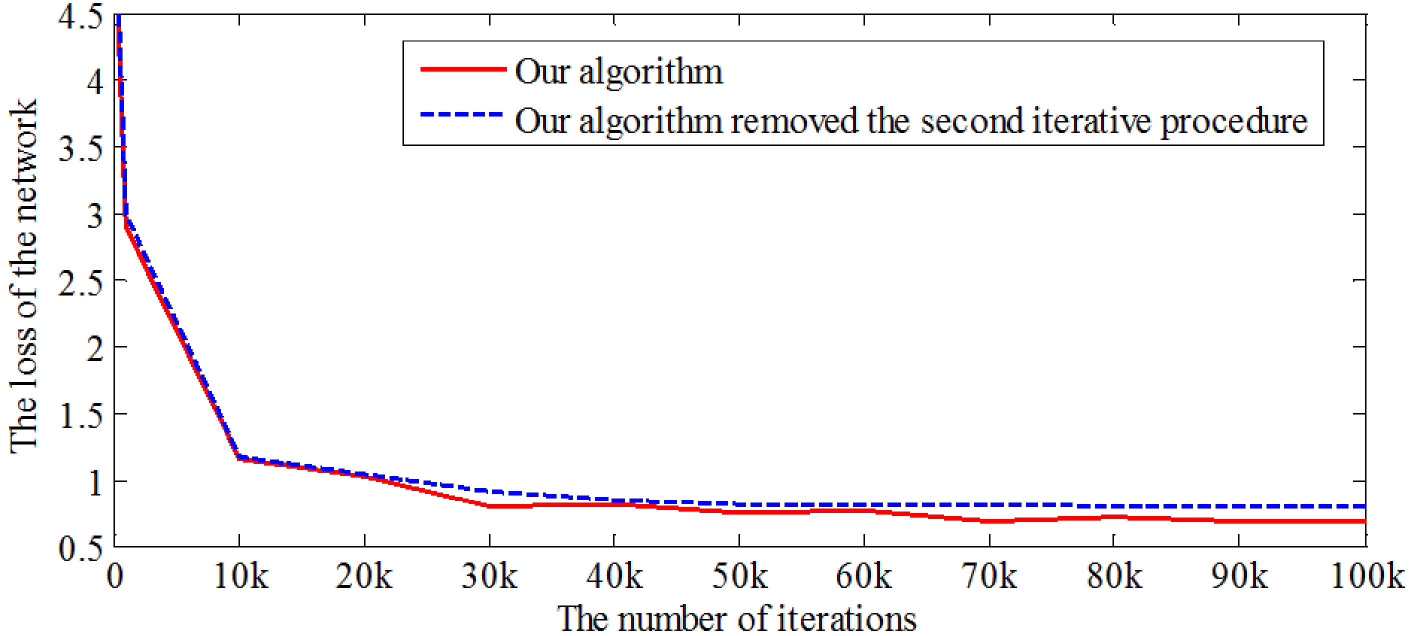
Algorithm discussions



- The minimal loss are 0.6823 and 0.7938 respectively
- The difference of average distance is approximately

$$\sqrt{0.1115/(\lambda^2)\times 2/5} \approx 1.0559,\ \lambda = 0.2$$

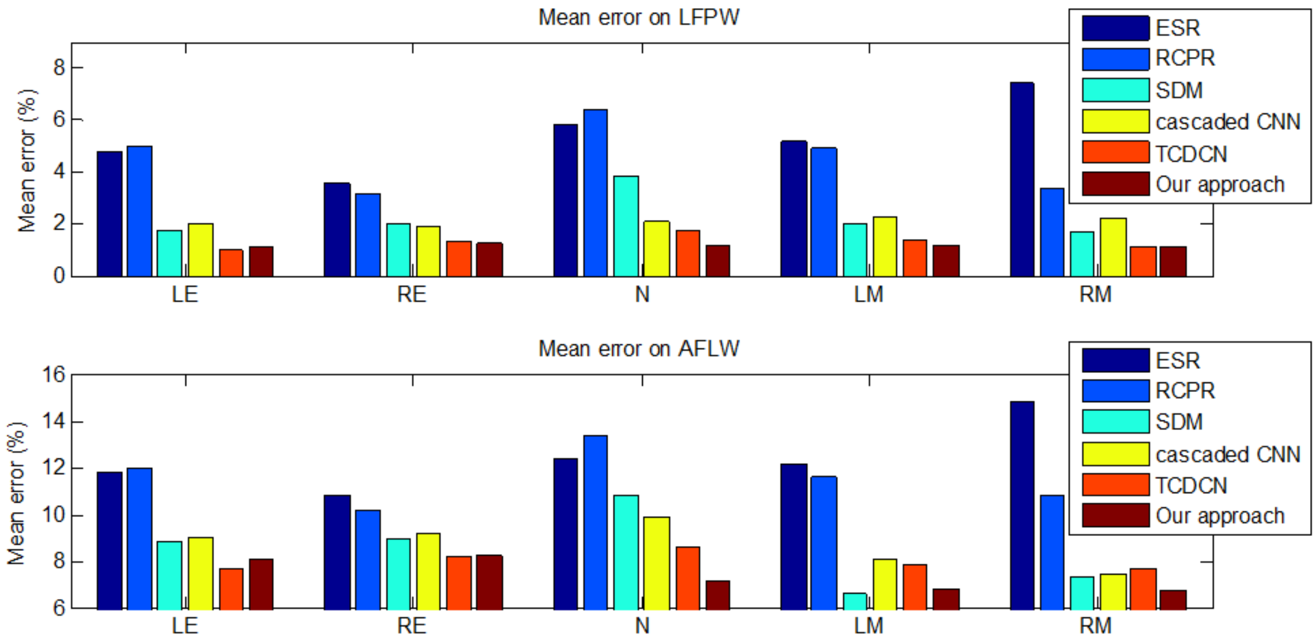
Algorithm discussions



Mean error (%)

Method	LFPW	AFLW
Our algorithm (one iterative procedure)	1.50	8.15
Our algorithm	1.17	7.42

Comparison with other methods



Method	LFPW	AFLW
ESR [7]	5.36	12.4
RCPR [8]	4.58	11.6
SDM [11]	2.26	8.5
cascaded CNN [9]	2.10	8.72
TCDCN [17]	1.33	8.0
Our approach	**1.17**	**7.42**

Comparison with other methods

Method	LFPW	AFLW
ESR [7]	5.36	12.4
RCPR [8]	4.58	11.6
SDM [11]	2.26	8.5
cascaded CNN [9]	2.10	8.72
TCDCN [17]	1.33	8.0
Our approach (quick)	**1.28**	**7.86**
Our approach	**1.17**	**7.42**

- Our approach (quick)

  Changing convolutional layers:

  2, 2, 2, 2 ⇨ 1, 1, 2, 2

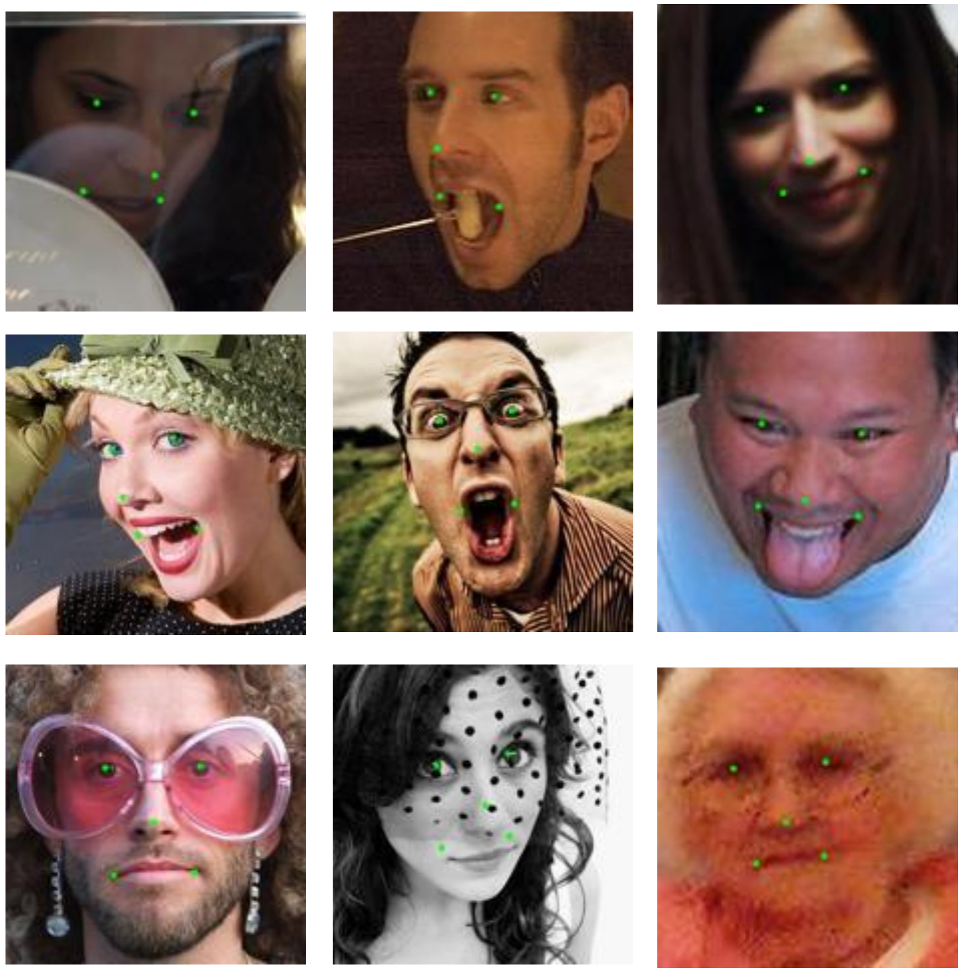
Comparison with other methods

Deep model	Time (Intel Core i5 CPU)
cascaded CNN [9]	120 ms
TCDCN [17]	17 ms
Our approach (quick)	39 ms
Our approach	67 ms

Cascaded CNN

Our approach

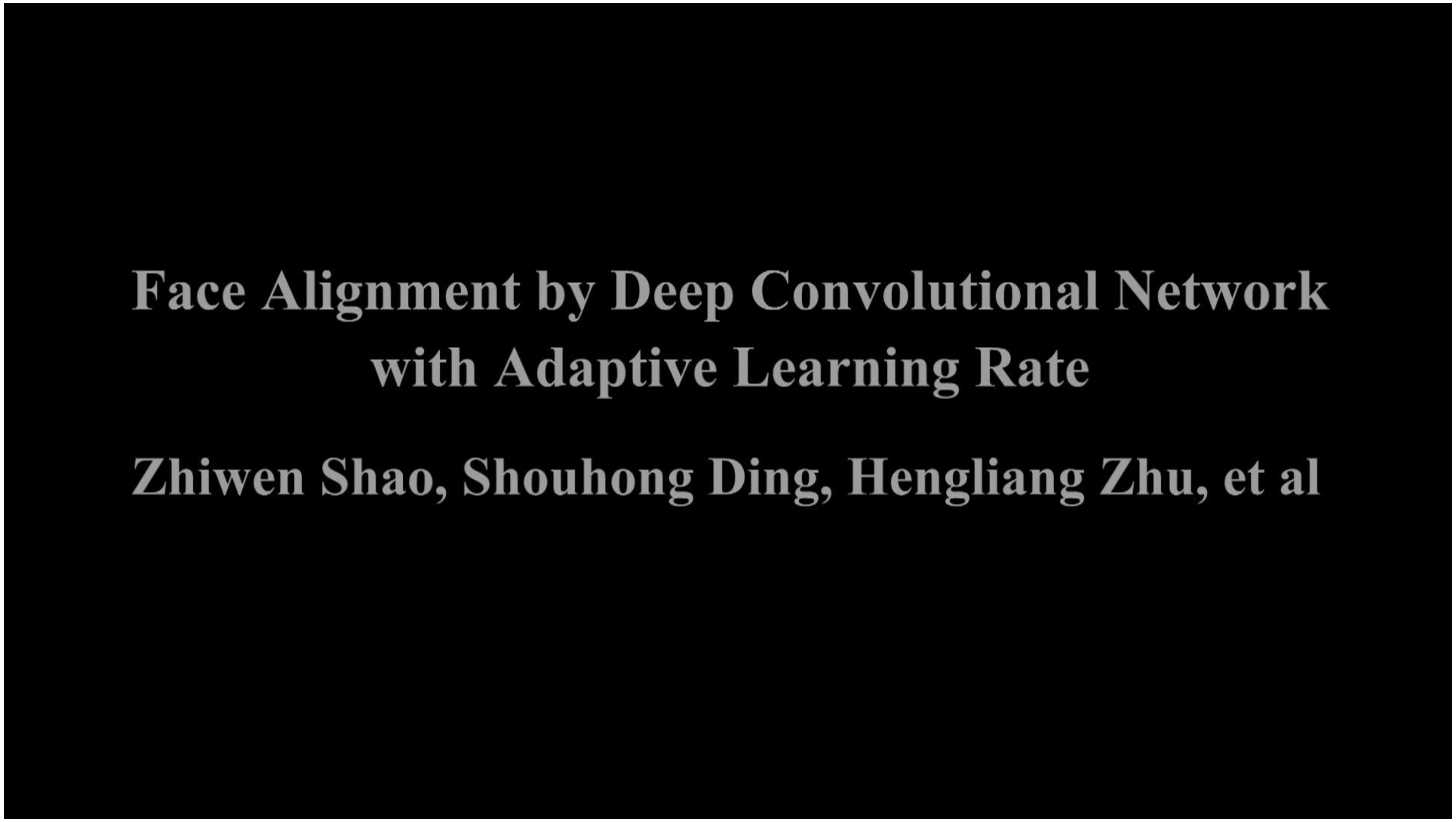
More examples



Conclusions

- We propose an effective deep convolutional network based on data augmentation and adaptive learning rate
- We achieve state-of-the-art performance
- The data augmentation and adaptive learning rate can also be applied to other problems like face recognition

Demo


Face Alignment by Deep Convolutional Network
with Adaptive Learning Rate
Zhiwen Shao, Shouhong Ding, Hengliang Zhu, et al

Thank you!