Masked Loss Covolutional Neural Network For Facial Keypoint Detection

Junhong Xu, Shaoen Wu, *Senior Member, IEEE* Honggang Wang, *Senior Member, IEEE* and Chonggang Wang, *Fellow, IEEE*

***Abstract*—In this paper, we present an effective and easy to implement approach to detect facial keypoints on the dataset that has missing target labels using convolutional neural networks(CNN). The main challege of this task is the images that have missing target values can not be trained by traditional CNNs. The image dataset is from Kaggle Facial Keypoints Detection competition[1]. It consists of 7049 training images, but with only 2140 images that have full target keypoints labeled. To address this problem, we proposed a *masked objective function* that is able to ignore the error of predicting missing target keypoints in the output layer. In addition, we used data augmentation strategies to increase the number of training data in order to prevent overfitting. We compared our new approach that trained on full dataset to the traditional CNN facial keypoints recognition method proposed in [7] that trained on 2140 images. The experiement result shows that the new approach is robust to training CNNs on datasets that have missing target values.**

I. Introduction

Detecting facial keypoints is a critical task becuase of variations of faces, light exposures, and different viewpoints. It is essential for analyzing facial expressions, and tracking faces. Recent development of CNNs has shown great successes in computer vision field [9] [3] [4] [12] [10]. The deep structures extract raw data into a high level abstraction. This high level abstraction represents aspects of the input that are important for discrimination and discard irrlatvent information [10]. Previous works on facial keypoint detection task were commonly based on searching local image features [16] [11]. Specifically, Each keypoint is detected by a classifier called component detector based on local patches. Local minima may be caused by ambiguous or corrupted local patches. Recent approaches applied CNNs to facial keypoints detection task to address this problem by constructing cascading CNNs [14]. It uses several CNNs at different levels to predict and fine-tune facial keypoint positions. Another method that also uses CNN to predict keypoints in [7] with data augmentation has also shown improvment in Labeled Faces in the Wild(LFW) dataset [5]. However, these two methods have a common shortcoming when using CNNs as a predictor. They are not able to be trained on samples that have missing target values. Many face images do not have all facial keypoints represented because different viewpoints of faces may cause keypoints being invisible. Training a deep neural network requires large amount of data to prevent overfitting, discarding these samples may cause the network not be able to generalize to unseen datasets and become overfitting on the training dataset.

In this paper, we present an effective and easy to implement objective function to overcome the above issue. We added a mask matrix at the output layer to mask out the predicted value at the same index with the missing target value, so the error term between the target values and predicted values will not be affected by the missing values. Thus, the network will not be updated by backpropagation on missing target values. Besides the proposed objective function, we used deep residual convolutional neural network(ResNet), which won the first place in ILSVRC 2015 classification task [4], as our network structure to replace the conventional CNN. By introducing an identity mapping shorcut connection, the framework is able to prevent the problem of degradation and allow to be trained on deeper networks. We also adapted a recent advanced technique, batch normalization method to avoid internal covariate shift [6]. We used Facial Keypoints Detection dataset hosted at Kaggle which contains 7094 training images, but with only 2140 images that have all target keypoints labeled. We compared the performance of the traditional CNN and CNN with *masked loss* . In training phase, data augmentation method mentioned in [7] was used to improve the generalization of the two models. The results showed that the performance of our model surpassed the traditional CNN mentioned in [7] when trained on missing target keypoints dataset.

In the Kaggle dataset, each image sample is a grayscaled image with 96×96 pixels with 15 facial keypoints to predict. The total number of image samples is 7094, but with only 2140 of the target keypoints are fully labeled. Some samples are shown in Figure 1.

II. Background

Images are often stored in a 3 dimensional array, where dimensions represent height, width, and color channel of the image. Since the ordering of pixels are important, ignoring this ordering during image processing may cause unwanted affects. Convolutional layers are able to retain this spatial ordering by doing 2 dimensional convolution operation along h and

Junhong Xu and Shaoen Wu are with the Department of Computer Science, Ball State University, Muncie, IN 47306 USA. E-mails: {jxu7,swu}@bsu.edu.

Honggang Wang is with Department of Electrical and Computer Engineering, University of Massachusetts Dartmouth, Dartmouth, MA 02747 USA. E-mail: hwang1@umassd.edu.

Chonggang Wang is with InterDigital Communications, Wilmington, DE 19809 USA.

[1]This competition is hosted at: https://www.kaggle.com/c/facial-keypoints-detection

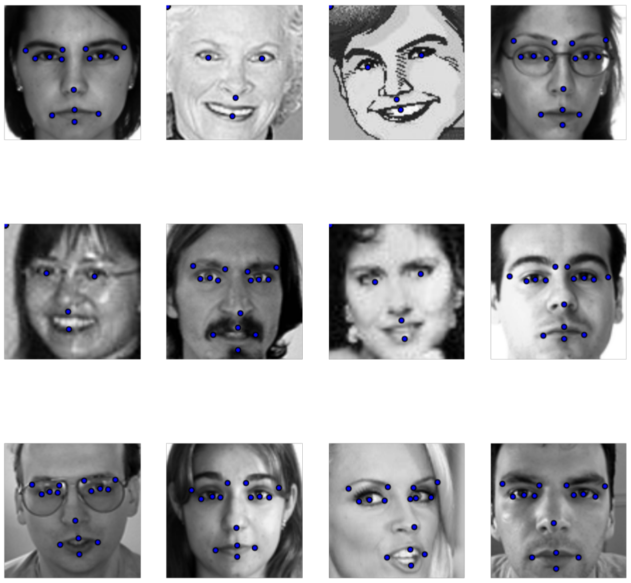

Fig. 1: Some of the faces have keypoints fully labeled, but some have missing labels

w axes. The input and output of each convolutional layer is called *feature map*, which is a 3 dimensional array with size $h \times w \times c$, where h and w are spatial dimensions representing height and width and c is the depth. The input to the first convolutional layer is an image with $h \times w$ pixels with c color channels(for gray-scale image, c is 1 and RGB image, c is 3). Each convolutional layer is consisted of n learnable $kernels$ of size $h^{'} \times w^{'} \times c$ representing height, width, and depth. The height and width are usually small to learn local features. The depth is the same as the input feature map. A demonstration is represented in Figure 2. In Figure 2, the dimensions of input *feature map* is a $6 \times 6 \times 2$. Three $kernels$ with dimension of $3 \times 3 \times 2$ convolve through the *feature map*. The dimension of the output *feature map* is decided by two parameters *stride* s and *padding* p. *Stride* s determines the distance between two consectutive positions to do matrix multiplication between each $kernel$ and local features in *feature map*. *Padding* p determines how many 0 valued pixels to pad along h and w axes of the input *feature map* to maintain the dimension of output *feature map*. The output dimension is shown in equation 1, where l indicates the layer number, h, w, c represent the height, width, and depth of the feature map, $h^{'}$, $w^{'}$, and k are the height, width, and the number of the kernels in a specific layer.

$$\begin{aligned} h_{l+1} &= (h_l - h_l^{'} + p)/s + 1 \\ w_{l+1} &= (w_l - w_l^{'} + p)/s + 1 \\ c_{l+1} &= k_l \end{aligned} \tag{1}$$

Convolution operation over a 2D image is defined in equation 2,

$$(I * k)(x, y) = \sum_{u,v} I(x, y)k(x - u, y - v) \tag{2}$$

where $I(x, y)$ is a function of image, $k(x, y)$ is a function of each kernel. Each element of the output *feature map* is the summation of the multiplied values between the kernel and the local *feature map*. Figure 3 shows how a single element is computed through one convolution operation.

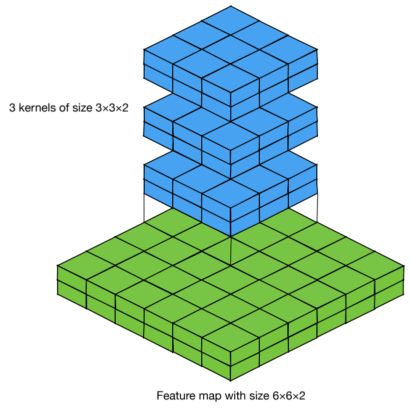

Fig. 2: Demonstration of kernels and feature map in a convolutional layer. 3 kernels with size $3 \times 3 \times 2$ computing on a feature map of size $6 \times 6 \times 2$

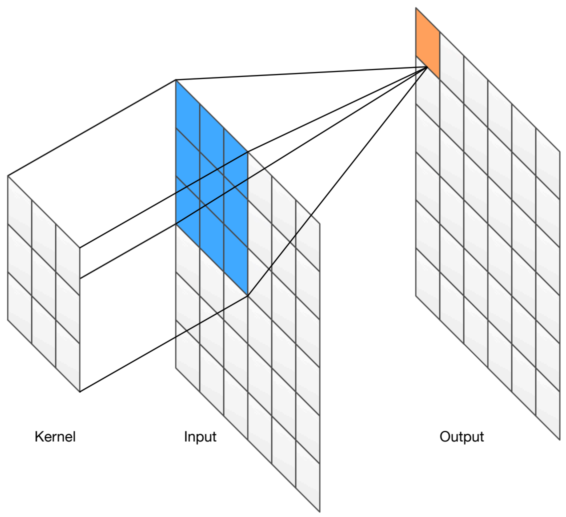

Fig. 3: An example of how the output is computed through convolutional layer.

III. Related Work

CNN-based facial keypoints detection methods [7] [14] are the most relative ones to our work. Predicting positions of keypoints is equivalent to a regression-based task. In the two works, output layers are linear regression layers that predict *x* and *y* coordinate values of each keypoints in the target values. In [7], they used a CNN with three convolutional layers and one fully connected layers with 400 hidden units. They used a training time data augmentation method that is able to increase data size during training and prevent overfitting. Although they achieved better performance than traditional training method, their CNN structure is not robust to many face images that have missing keypoint values. By ignoring these image sampels, the network may result in overfitting.

Sun et al proposed a framework that makes different levels of CNNs to predict from whole face regions to local patches [14] . They used LFW dataset [5] and face images downloaded from the web that result in 13,466 face images. Each face image is labeled with the positions of five facial keypoints without any missing labels. Although the framework achieved the state-of-art performance, it still did not address the problem of missing target values.

There are also many other regerssion based methods that do not use CNNs as regressors. For instance, Valstar et al. used Markov Random Fields to constrain the search space and support vector regressors to learn the typical appearance of areas surrounding a given keypoint [15]. Another example of regression based method was proposed by Cao et al [2]. They used the whole face region as input and random ferns as the regressor. Shapes to be predicted were expressed as linear combinations of training shapes.

IV. Methodology

In this section, we first describe our network archeticture for facial keypoints detection task. Then introduce the techniques to prevent overfitting in the training phase. Finally, we explain the mechanism of the *masked loss objective function* in detail.

A. Network Architecture

A basic building block of ResNet is consisted of two convolutional layers with a short cut connection from input to output of the last batch normalization layer as shown in Figure 4. If using original 18-layer ResNet, we saw the model becomes overfitting because the number of samples in the dataset are not enough to train an 18-layer ResNet. Instead, we used 5-layer ResNet to predict the positions of each facial keypoints. The input layer accepts face images of size of $96 \times 96 \times 1$ pixels.

The network architecture is shown in Figure 5. As described in [14], detecting facial keypoints from whole face region to local regions avoids local minima, we used 32 kernels with a large kernel size of 11×11 on the first convolutional layer and followed by a max pooling layer to both reduce the dimentionality and find out the most discriminative features. The following four convolutional layers are in two ResNet blocks. We used the kernal size of 3×3 in order to achieve predicting local small regions. The number of kernels are equal in the same residual block. The convolutional layers in the first residual block has 32 kernels and the second have 64 kernels. Instead of using pooling layer between residual blocks, we used a strided convolution at the last convolutional layer of each residual block to allow the network to learn its own spatial downsampling. The last fully connected layer has size of 1000 neurons. The output layer has 30 neurons representing facial keypoint positions in *(x, y)* paris. In addition, *strides* in all convolutional layers are one to prevent from losing information.

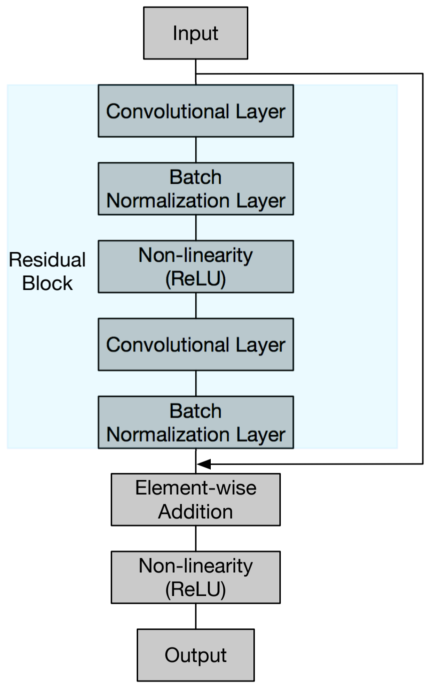

Fig. 4: A basic building block of ResNet is consisted of two convolutional layers with a short cut connection from input to output.

B. Prevent Overfitting

One of the most concerned problems while training deep models is overfitting. Deep neural networks are complex and expressive models with many parameters, but this deep architecture will learn errors and noise in the training dataset without enough training data. In other words, without enough training data, deep neural networks only perform well on training dataset, but perform poorly on testing dataset. We used two techniques to increase the size of the image dataset and improve generalization of the model. [7] shows a training time data augmentation technique with stochastic gradient descent (SGD) that performs better than traditional SGD. In our work, we just adpated one augmentation methods: *horizontal rotation*.This will give us $7049 \times 2 = 14096$ for total number of training samples and $2140 \times 2 = 4280$ for total number of fully labeled target keypoints training samples. The two

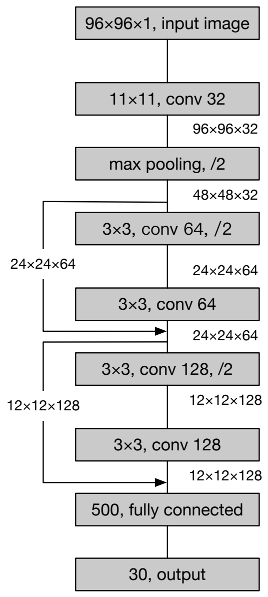

Fig. 5: Network architecture of our facial keypoint predicting model. The number of parameters in each layer is shown before layer types. For convolutional layers, the number of kernels is shown after layer types. The number between two layers represents the dimension of the input to the next layer. $/2$ represents the downsampling factor. For clarity, non-linearity, batch normalization in each residual block shown in Figure 4 are ignored.

figures shown in Figure 6 represent horizontal transformation. In addition to data augmentation, we also utilized dropout technique, where at training time, each neuron has a probability of $1-p$ to output a 0 valued activation, at testing time, each neuron if fully presented, in the fully connected layer in order to prevent overfitting [13].

C. Masked Objective Function

Finding *x, y* positions of facial keypoints can be categorized into linear regression task. The conventional objective function for linear regression task is *Mean Squared Error (MSE)* defined in equation 3. N is the number of predictions, y is the ground truth target keypoints positions, and $\bar{y}$ is the predicted keypoints positions.

$$e = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y_i})^2 \tag{3}$$

When equation 3 applies to a mini-batch of samples, we will have:

$$E = \frac{1}{M}\sum_{m=1}^{M} e_m \tag{4}$$

M is the number of mini-batch size. It averages over all error of each sample.

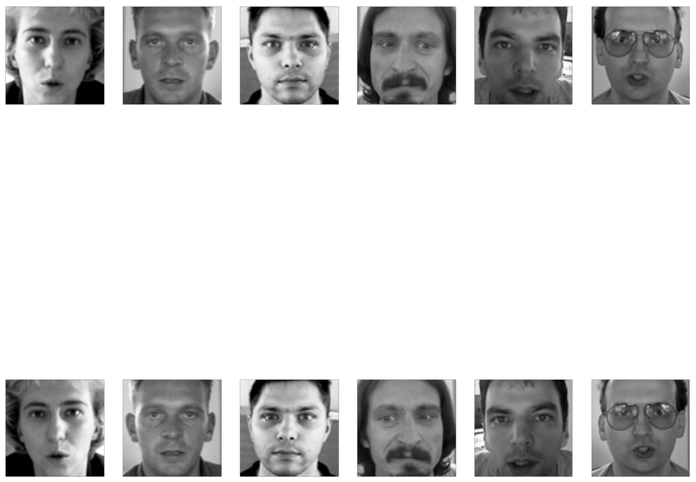

Fig. 6: Up: the original images. Down: the images that vertically flipped.

In order to train on image samples that have missing target values, we proposed a *masked objective function* to mask out the missing target values in equation 4

Concretely, we construct a $mask\ matrix$ based on the image labels. First, we added 0 values to all missing values in target keypoints. Then we define a *mask* variable based on the ground truth missing target values for each sample. Specifically, it is a boolean matrix, where each row represents each sample and each column represents whether it is a missing target value:

$$mask(m,n) = \begin{cases} 0, y(m,n) = 0 \\ 1, y(m,n) > 0 \end{cases} \tag{5}$$

Here, y is the ground truth target value matrix which has number of M samples and each sample has number of N values and $y(m,n)$ retrieves the value from y at sample index m and value index n. Therefore, m satisfies $0 \le m \le M$ and n satisfies $0 \le n \le N$. Parameters of models are updated through backpropagation method that back propagate the gradient of objective function *w.r.t* each model parameter. The simplest way to update parameters is gradient descent as described in (6):

$$w_{ij}^{(l)} = w_{ij} - \alpha\frac{dE}{dw_{ij}^{(l)}} \tag{6}$$

E is the MSE in equation(4), α controls the learning speed of this model, and $w_{ij}^{(l)}$ means the weight connects from neuron *i* at layer *l* to neuron *j* at next layer. Before backpropagating error terms to the network, the $masked\ matrix$ that constructs in (5) multiplies the predicted target values of the model to mask out the information that does not need to backpropagate. Figure 7 uses a fully connected network to illustrate how *masked matrix* is applied to the final predicted value. We constructed a *masked matrix* according to the ground truth labels and multiplied it with the predicted values. After this operation, the only element contributed to the final loss is y_2'. The gradients of y_1' and y_3' $w.r.t\ loss$ are 0. Therefore, the weights will not be updated by backpropagation. By multiplying *masked matrix* to the predicted target values, it ensures that the predicted target values are the same with missing target values, which are 0. In this way, there are no differences between the ground truth values and the predicted values at missing target value index. Thus, the gradients of the model parameters $w.r.t$ *masked MSE loss* are 0 on the missing target values and the weights are not updated by the errors between predicted values and real values. We ran several simple tests and observed that the gradient flowed in the model indeed did not change caused by the missing target values. This means our *masked objective function* sovles this problem successfully.

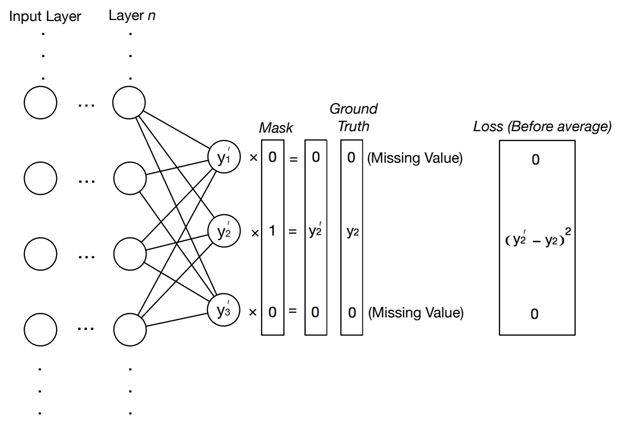

Fig. 7: Illustration of how *masked matrix* is applied to predicted values

V. Experients

A. Training

We used Adam proposed in [8] as a training algorithm instead of SGD. Adam is an adaptive learning rate training method that computes individual learning rates for different parameters from estimates of first and second moments of the gradients. It speeds up the learning process of our model. To have a fair comparison of *masked MSE* loss and original MSE loss used in the previous work proposed in [7], we trained three models in our experiment shown in table: *1)* the network described in IV-A with *masked objective function*, *2)* CNN with same 6-layer architecture trained on fully labeled dataset (2140 samples) without *masked objective function*, and *3)* the same 6-layer CNN as, but with *masked objective function*, which allows it to be trained on the full dataset (7149 samples). With random searching hyperparameters for all three models [1], we chose learning rate to be 0.0001, weight decay to be 0.0005, batch size to be 256, and weight initialized from a zero-centered Normal distribution with 0.02 standard deviation. We trained three models on 1000 epochs and recorded loss on evaluation dataset for every epoch.

TABLE I: Overview of three network architectures

Hyper-parameters	*Values*
Learning rate	0.0001
Weight decay	0.0005
Batch size	256
Weight initialization	Normal distribution with mean 0 and standard deviation 0.02
Total training epochs	1000

B. Results

We first evaluated the three models according to evaluation loss curve as shown in Figure 8. All the losses are normalized between 0 and 1 for clarity. The x axis represents number of epochs and the y axis is either the value of *masked MSE loss* or original *MSE*. The learning curve for CNN with MSE loss function converges at around 0.06 and the CNN with $masked\ loss$ is slightly better than it and converges at 0.03.The ResNet with $masked\ loss$ surpassed both the previous two models and converges at around 0.01. We also noticed that training ResNet is more stable than training CNN as the flucuation of evalution loss is much smaller.

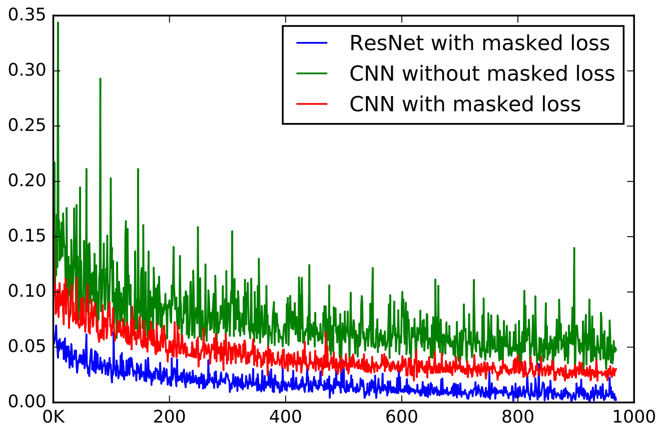

Fig. 8: Evaluation losses for the three models

In addition to evaluation loss, we adopted the evaluation method in [14]. We defined a *detection error* measured as Equation 7,

$$error = \sqrt{(x-\bar{x}) + (y-\bar{y})^2}/w \tag{7}$$

where x and y are the ground truth values, $\bar{x}$ and $\bar{y}$ are the predicted values, and the number w is the width of the input face pictures (in our case, it is 96). This error detection measurement represents how much the predicted values are deviated from the ground truth values. If the $error$ is larger than 10%, we count it as a failure. The average error rate is calculated as f/N, where f is the total number of failures and N is the total number of evaluation samples. The result is shown in Table II

Then, we visualized the result of the three models by predicting facial keypoint positions on the evaluation dataset as shown in Figure 9, 10, and 11. Red dots are ground truth labels and blue dots are predicted lables. All the models were able to learn to predict faces that are in the center and without turning. The CNN without *masked loss function* failed to predict faces with small turning, but both models with *masked loss function* are able to learn turning faces, especially for ResNet with masked loss(see Figure 11 row three, column 2).

At last, we evaluated our models on the test set. The test set is evaluated by Kaggle server using *Root Mean Squared Error(RMSE)* is defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y_i})^2} \tag{8}$$

The final result is shown in Table II

TABLE II: RSME of the three models on the test set

Model	*RMSE*	*Average error rate*
CNN without *masked loss*	3.1	75%
CNN with *masked loss*	2.36	**9%**
ResNet with *masked loss*	**2.23**	**9%**

VI. Conclusion

In this work, we proposed a new loss function called *masked loss* to train neural networks on facial keypoint detection dataset that has missing target values. *Masked loss function* is able to ignore errors between ground truth values and predicted values by multiplying a *masked matrix*. We gave evidence that with *masked loss function*, the model is able to learn from the whole dataset which has missing target values. In addition to facial keypoints detection, *masked loss function* is also able to perfom on other dataset that has missing target values and we leave this exploration for the future work.

References

[1] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[2] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. *International Journal of Computer Vision*, 107(2):177–190, 2014.

[3] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[5] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[7] M. Kimura, T. Yamashita, Y. Yamauchi, and H. Fujiyoshi. Facial point detection based on a convolutional neural network with optimal mini-batch procedure. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2860–2864. IEEE, 2015.

[8] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[11] L. Liang, R. Xiao, F. Wen, and J. Sun. Face alignment via component-based discriminative search. In *European Conference on Computer Vision*, pages 72–85. Springer, 2008.

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[14] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[15] M. Valstar, B. Martinez, X. Binefa, and M. Pantic. Facial point detection using boosted regression and graph models. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2729–2736. IEEE, 2010.

[16] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.

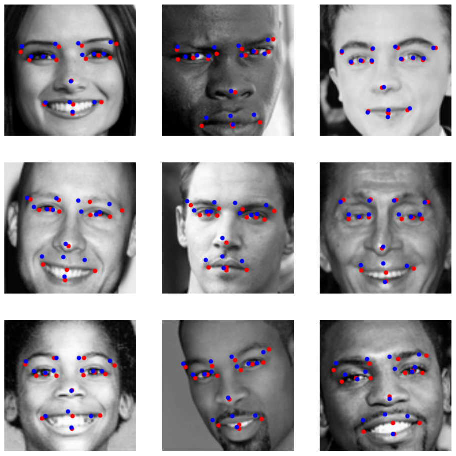

Fig. 9: Evaluation on CNN without masked loss

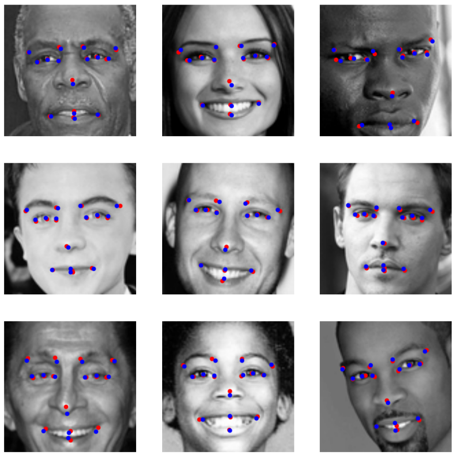

Fig. 10: Evaluation on CNN with masked loss

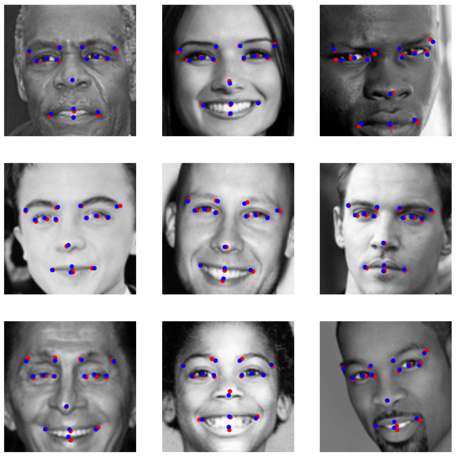

Fig. 11: Evaluation on ResNet with masked loss