Masked Loss Residual Convolutional Neural Network For Facial Keypoint Detection

Junhong Xu, Shaoen Wu,
Shangyue Zhu, Hanqing Guo
Computer Science
Ball State University
Muncie, IN 47306
jxu7,swu,szhu,hguo@bsu.edu

Honggang Wang
Electrical and Computer Engineering
University of Massachusetts
Dartmouth
Dartmouth, MA 02747
hwang1@umassd.edu

Qing Yang
Computer Science
Montana State University
Bozeman, MT 59717
qing.yang@montana.edu

ABSTRACT

This paper presents an effective and novel CNN-based deep learning solution, named *Masked Loss Residual Convolutional Neural Network* (ML-ResNet), to facial keypoint detection on the datasets that have missing target labels. The core of the ML-ResNet solution is a *masked loss objective function* that ignores the error in predicting the *missing* target keypoints in the output layer of a CNN. To compensate for the loss induced by the masked loss objective function to prevent overfitting, we design a data augmentation strategy in ML-ResNet to increase the number of training data. The performance of ML-ResNet has been evaluated on the image dataset from Kaggle Facial Keypoints Detection competition, which consists of 7,049 training images, but with only 2,140 images that have full target keypoints labeled. In the experiments, ML-ResNet is compared to a pioneer literature CNN facial keypoint detection work. The experiment results clearly show that the proposed ML-ResNet is robust and advantageous in training CNNs on datasets that have missing target values. ML-ResNet can improve the learning time by 30% during the training and the detection accuracy by eight times in facial keypoint detection.

ACM Reference format:
Junhong Xu, Shaoen Wu, Shangyue Zhu, Hanqing Guo, Honggang Wang, and Qing Yang. 2017. Masked Loss Residual Convolutional Neural Network For Facial Keypoint Detection. In *Proceedings of 10th EAI/ACM International Conference on Mobile Multimedia Communications, Chongqing, China, July 2017 (MOBIMEDIA'17)*, 6 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Detection of facial keypoints is challenging because of variations of faces, light exposures, different viewpoints, etc. In face keypoint detection, it is essential to analyze facial expressions and track faces. Recent development of CNNs has shown great success in computer vision [1–5]. The deep structures of CNNs extract raw data into a high level abstraction, which consists of various aspects of an information [5]. Traditional image processing based works on facial keypoint detection are commonly based on searching local image features [6, 7]. Specifically, each keypoint is detected by a classifier called *component detector* based on local patches. As a result, local minima may be incurred by ambiguous or corrupted local patches. Recent approaches address the problem of local minima by employing cascading CNNs to facial keypoints detection. Sun, Wang and Tang propose to use several CNNs at different levels to predict and finely tune facial keypoint positions [8]. Another CNN based solution predicts keypoints with data augmentation [9] and it has shown significant improvements in Labeled Faces in the Wild(LFW) dataset [10]. Most CNN solutions in literature, however, have *a common issue in using CNNs as the predictor that they are not able to be trained on samples that have missing target values*. Many facial images unfortunately do not have all facial keypoints available because some angles of viewing faces can result in undisclosed keypoints. Excluding these images will significantly reduce the number of sample images available for the training of deep neural networks that requires a large amount of data to prevent overfitting. Therefore arises **a research challenge that how to fit into deep neural networks the facial images of missing keypoints if they are retained in the training dataset to prevent the overfitting.**

In this work, we design and evaluate an effective solution, named Masked Loss Residual ML-ResNetConvolutional Neural Network (ML-ResNet), that proposes a novel objective function to address the above research challenge. ML-ResNet adds a mask matrix at the output layer of the CNN to mask out the predicted value at the same index that expects to be a missing target value. As a result, the error between the target value and predicted value will not be affected by the missing value. The neural network will not be updated by the back-propagation on missing target values. In addition to the proposed objective function, we also design a deep residual convolutional neural network (ResNet) to replace the conventional CNN. Introducing an identity mapping shortcut connection [3], the ResNet can prevent the problem degradation and allow to be trained on deeper networks. We furthermore adapt a recent technique, *batch normalization* [11], to avoid internal covariate shift. We evaluated the performance of our solution ML-ResNet and a traditional CNN solution [9] with the Facial Keypoints Detection dataset hosted at Kaggle that contains 7094 training images, but with only 2140 images that have all target keypoints labeled. In the Kaggle dataset, each image sample is a grayscaled image with 96×96 pixels with 15 facial keypoints to predict. In the training phase, a data augmentation method [9] is employed to improve the generalization of the two models. The results show that the performance of our ML-ResNet largely surpasses the traditional CNN when trained on missing target keypoints dataset.

In the rest of this paper, Section 2 introduces the model of an image in a CNN network and the related solutions in literature that employ deep neural networks to address facial keypoints detection. Then, Section 3 discusses the detail design of the proposed solution. The performance evaluation is next presented in Section 4. Finally the paper is concluded by Section 5.

2 BACKGROUND

In this section, we first introduce how an image is modeled and computed by a CNN, followed by the a summary of related work proposed for facial keypoint detection.

2.1 Image Model in CNN

Images is often stored in a 3-dimensional array, where dimensions (h, w, c) represent the height, width, and color channel of an image. Because the spatial positions of pixels are important, convolutional layers in CNNs retain this spatial position information by performing a 2-dimensional convolution operation along h and w axes. The input and output of each convolutional layer constitute a *feature map*, which is a 3-dimensional array with the size of $h \times w \times c$, where h and w are spatial dimensions representing height and width and c is the color depth. The input to the first convolutional layer is an image of $h \times w$ pixels with c color channels (for gray-scale image, c is 1 and RGB image, c is 3). Each convolutional layer consists of n learnable *kernels* of size $h' \times w' \times c$ representing height, width, and depth. The height h' and width w' are usually small to learn local features. The depth is the same as the input feature map. An illustration is plotted in Figure 1. On the figure, the size of the input *feature map* is $6 \times 6 \times 2$. Three *kernels* with a dimension of $3 \times 3 \times 2$ convolve through the *feature map*. The size of the output *feature map* is decided by two parameters *stride s* and *padding p*. *Stride s* indicates the distance between two consecutive positions in the matrix multiplication between each *kernel* and local features in the *feature map*. *Padding p* represents how many 0 valued pixels to pad along h and w axes of the input *feature map* to maintain the size of the output *feature map*. The output size is calculated as 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 particular 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, 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 2 shows how a single element is computed through one convolution operation.

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

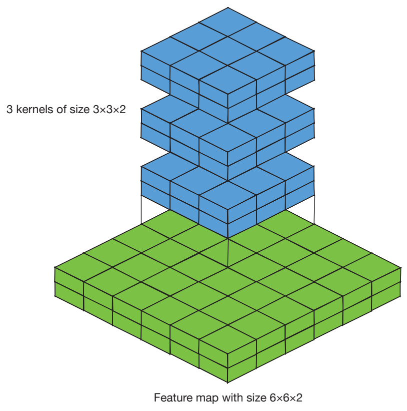

Figure 1: Demonstration of kernels and feature map in a convolutional layer. Three kernels with the size of $3 \times 3 \times 2$ **computing on a feature map of the size of** $6 \times 6 \times 2$

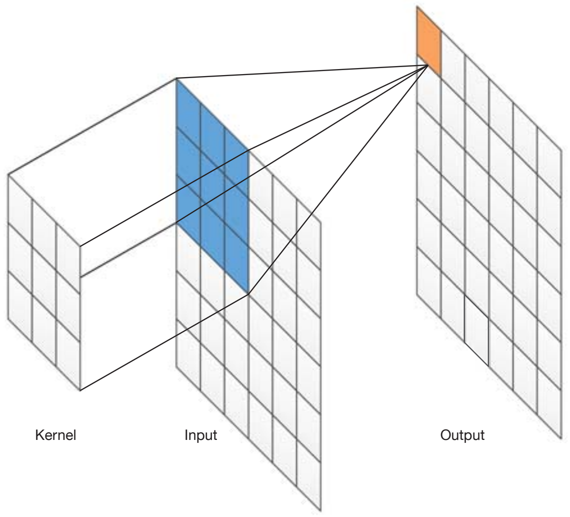

Figure 2: An example of how the output is computed through convolutional layer.

2.2 Related Work

With the availability of strong computational power and big data, deep learning models like CNNs have shown great success in extracting low level features from high level input. CNNs have been successfully applied to various computer vision tasks such as image classification [1, 3, 4], object tracking [12], face verification [13], and image generation [14]. Recent advances on CNNs mainly focus on network structures and training pipeline. Some examples include: in [15], the authors analyze the performances of shallow neural networks and deep neural networks, rectified linear unit has be shown to stabilize training and has been mostly used in recent deep models [16], and dropout regularization has been applied to deep models to overcome overfitting problem by disabling randomly selected neurons at test time [17]. Deep residual networks have been shown to improve model performance by adding identity mappings between layers. Some works have extended deep residual networks to gain more performance [18], estimate depth information [19], and interpret residual networks [20].

Two CNN-based facial keypoints detection solutions [8, 9] are the most related to our work. Predicting the positions of keypoints on a facial image is essentially a regression task. In these two related solutions, their output layers are linear regressions that predict x and y coordinate values of each keypoints in the target values. In [9], the authors adopt a CNN of three convolutional layers and one fully connected layer containing 400 hidden units. This work also employs a data augmentation method that increases the data size during training and therefore effectively prevents overfitting. Although this solution reports better performance than conventional training methods, its CNN structure is not robust to the case that a number of face images have missing keypoint values. Ignoring these image samples, the CNN network likely results in overfitting due to the significant reduction on the number of samples.

Sun $etal$. proposes a framework that adopts different levels of CNNs to predict from a whole face region to local patches [8]. They used LFW dataset [10] that provides 13,466 face images to download from the web. Each face image is labeled with the positions of five facial keypoints without any missing labels. Although the framework achieves the state-of-art performance, it also lacks of the feature to address the problem of missing target values.

3 DESIGN OF MASKED LOSS CONVOLUTIONAL NEURAL NETWORK

In this section, we first present the network architecture of our proposed facial keypoints detection solution, ML-ResNet. We then introduce the data augmentation techniques designed to prevent overfitting in the training phase. Finally, we explain the *masked loss objective function* in details, which is the core of this solution.

3.1 ML-ResNet Architecture

The network of ML-ResNet is essentially a residual convolutional neural network, a.k.a ResNet. The basic unit of a ResNet [3] is called "residual block" that consists of two convolutional layers with a shortcut connection from the input to the output of the second batch normalization layer as shown in Figure 3. Although an ResNet can have arbitrary m residual blocks, a large m requires a large number of samples to train the ResNet. Otherwise, the model will become overfitting. To accommodate the datasets with moderate samples, we designed a special 6-layer ResNet to predict the positions of each facial keypoints. The input layer accepts facial images of a size of $96 \times 96 \times 1$ pixels.

The ResNet architecture of our ML-ResNet is shown in Figure 4. As described in [8], detecting facial keypoints from the whole face region to smaller 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.

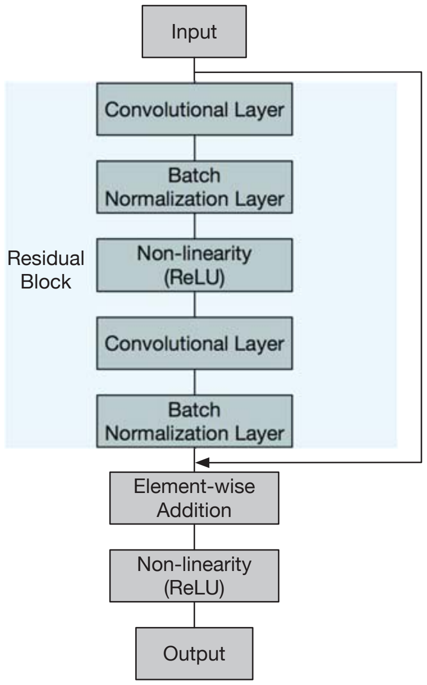

Figure 3: A residual black consists of two convolutional layers with a shortcut connection.

3.2 Data Augmentation

One of the most concerned problems in training deep neural network (DNN) models is the overfitting. Because DNNs are complex and expressive with many parameters, errors and noise in the training dataset resulted from insufficient training data will significantly hurt the accuracy and performance of the DNN models. In other words, without enough training data, DNNs only perform well on the training dataset, but not on the testing dataset. This problem is called overfitting. To address this problem, we design two techniques: one is to increase the size of the image dataset and the other is to improve the generalization of the DNN model. To increase the image dataset size, we develop a training time data augmentation technique called *horizontal rotation*, which horizontally rotates every image to create another new image. Figure 5 shows the horizontal transformation on six original images one the top with their rotated new images at the bottom. By applying the horizontal transformation on our Kaggle image dataset, this gives us $7,049 \times 2 = 14,096$ for total number of training samples with $2,140 \times 2 = 4,280$ training samples of fully labeled target keypoints. In addition to the data augmentation, we also design a dropout technique: during training, each neuron has a probability of $1 - p$ to output a 0 valued activation.

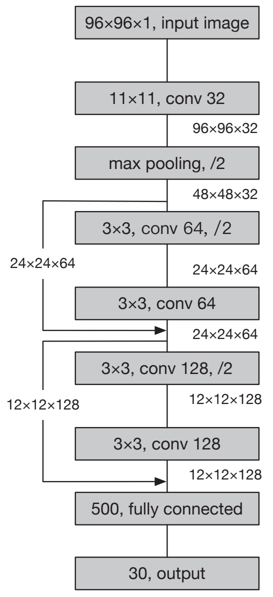

Figure 4: Network architecture of our facial keypoint predicting model. The number of parameters in each layer is shown before layer types. For convolutional layers, the number of kernels is shown after layer types. The number between two layers represents the dimension of the input to the next layer. /2 represents the downsampling factor. For clarity, non-linearity, batch normalization in each residual block shown in Figure 3 are ignored.

Figure 5: Top: the original images. Bottom: the images that vertically flipped.

3.3 Masked Loss Objective Function

Finding the *x, y* positions of a facial keypoint can be implemented as a linear regression task. A conventional linear regression objective function used in facial keypoint detection is *Mean Squared Error (MSE)* as defined in Equation 3, where 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 is applied to a mini-batch of samples, we will have:

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

M is the size of mini-batch samples. E is the average error over all sample.

In order to train on image samples that have missing target values, we propose a *masked loss objective function* to mask out the missing target values in Equation 4. The core of *masked loss objective function* is a $maskmatrix$, in which each row represents each sample and each column represents whether it is a missing target value. Therefore, *maskmatrix* is essentially a boolean matrix.

To construct a *maskmatrix*, we first add "0" to all missing values in target keypoints. Then we define a variable $mask(m, n)$ for each sample. The value of $mask(m, n)$ depends on the ground truth of target values and is determined as:

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

where y is the ground truth target value matrix which has M samples, and each sample has N values. Thus, $y(m, n)$ is the value at the sample index m and the value index n in the matrix y, where m satisfies $0 \le m \le M - 1$ and n satisfies $0 \le n \le N - 1$.

The DNN model parameters are updated through back propagation methods that feedback the gradient of the objective function *w.r.t* each model parameter. A typical back propagation is to use the gradient descent as defined in (6):

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

where E is the MSE in Equation(4), α controls the learning speed of the model, and $w_{ij}^{(l)}$ refers to the weight connecting from the neuron i at the layer l to the neuron j at the next layer.

The *masked loss objective function* works in this way: **after the** *maskedmatrix* **is decided and before back propagation, the** *masked matrix* **constructed as in Equation (5) multiplies the predicted target values of the model to mask out the information that would result in errors back to the network.** Figure 6 uses a fully connected network to illustrate the concept of the *masked objective function*. On the figure, a *masked matrix* is constructed according to the ground truth labels and multiplied 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 the back propagation. By multiplying *masked matrix* to the predicted target values, it is ensured that the predicted target values are the same as the missing target values, which are both 0. As a result, there is no difference between the ground truth value and the predicted value at a 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 the predicted values and the real values.

We have validated this design with several tests and observed that the gradient flowing in the model indeed did not change along with the missing target values as shown on Figure 7. The validation test has two output values y_1' and y_2' and their corresponding ground

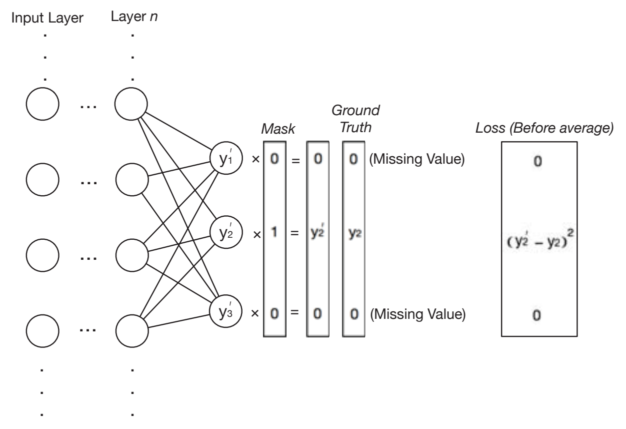

Figure 6: Illustration of how *masked matrix* is applied to predicted values

truth y_1 and y_2, where y_1 is the missing target value. We can observe from the figure that the gradient flowing in the model is not changed by the missing target values. This indicates our *masked loss objective function* solves this problem successfully.

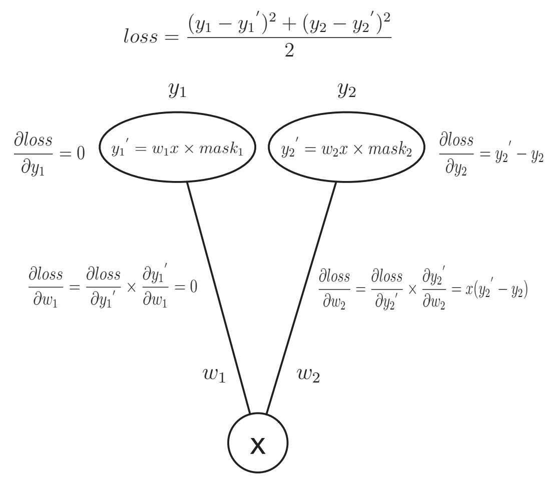
$loss = \frac{(y_1 - y_1')^2 + (y_2 - y_2')^2}{2}$
y_1
y_2
$\frac{\partial loss}{\partial y_1} = 0$
$y_1' = w_1 x \times mask_1$
$y_2' = w_2 x \times mask_2$
$\frac{\partial loss}{\partial y_2} = y_2' - y_2$
$\frac{\partial loss}{\partial w_1} = \frac{\partial loss}{\partial y_1'} \times \frac{\partial y_1'}{\partial w_1} = 0$
$\frac{\partial loss}{\partial w_2} = \frac{\partial loss}{\partial y_2'} \times \frac{\partial y_2'}{\partial w_2} = x(y_2' - y_2)$
w_1
w_2
x

Figure 7: A simple example shows that gradient flow for the missing target is 0. In this picture, y_1 is the missing target, so $mask_1$ should be 0 and $mask_2$ should be 1. w_1 and w_2 are the weights for this simple regression task and x is the input.

4 PERFORMANCE EVALUATION

To evaluate the performance of ML-ResNet, we have used the image dataset from Kaggle Facial Keypoints Detection competition, which consists of 7049 training images, but with only 2140 images that have full target keypoints labeled [?]. In the experiments, ML-ResNet is compared to a pioneer CNN facial keypoint detection work [9].

4.1 Training

In training stage, we have used the Adam training algorithm proposed by [21], which is an adaptive learning rate training method speeding up the learning process. Adam training computes individual learning rates for different parameters by calculating the estimates of first and second moments of the gradients. To have a fair comparison of our designed *masked MSE* loss and the original *MSE* loss in the related CNN work, we have tested three network models in our experiment:

- our designed ML-ResNet that is a ResNet with the proposed *masked loss objective function*
- a regular CNN with the same 6-layer architecture trained on fully *labeled* dataset (2140 samples)
- a regular CNN with the same 6-layer architecture, but with *masked loss objective function*, which allows it to be trained on the full dataset (7149 samples)

After random searching hyper-parameters for all three models [22], we select the parameters in the training as shown in Table 1: the learning rate of 0.0001, the weight decay of 0.0005, the batch size of 256, and the weight initialized from a zero-centered normal distribution with a standard deviation of 0.02. We have trained these three models on 1,000 epochs and recorded the loss on evaluation dataset for every epoch.

Table 1: Overview of three network architectures

Hyper-parameters	*Values*
Learning rate	0.0001
Weight decay	0.0005
Batch size	256
Weight initialization	$\sigma(0, 0.02)$
Total training epochs	1000

4.2 Results

4.2.1 Training Loss. We first evaluate the learning loss performance of these three models in the training. Learning "loss" refers to the MSE defined in Equation 4, which significantly impacts the training speed and efficiency. The result is shown in Figure 8. All the losses on the figure have been normalized to the values between 0 and 1 for clarity. The horizontal axis represents the number of epochs and the vertical axis is either the value of *masked MSE* or original *MSE*. As we can observe from the figure, the learning curve of the CNN with *MSE* loss function converges around 0.06 and the CNN with *masked loss* is slightly better than it converging at 0.03.The ML-ResNet, however, largely surpasses both CNN models and converges at around 0.01. Another observation from the figure is that the training of ML-ResNet is far steadier than the training of both CNN models that fluctuates widely.

4.2.2 Prediction Accuracy. In addition to the training loss, we have also evaluated the prediction accuracy in the testing stage. The prediction accuracy indicates how accurate a predicted facial keypoint is to its ground truth point of the facial images in the testing pool. Two metrics are tested for the accuracy.

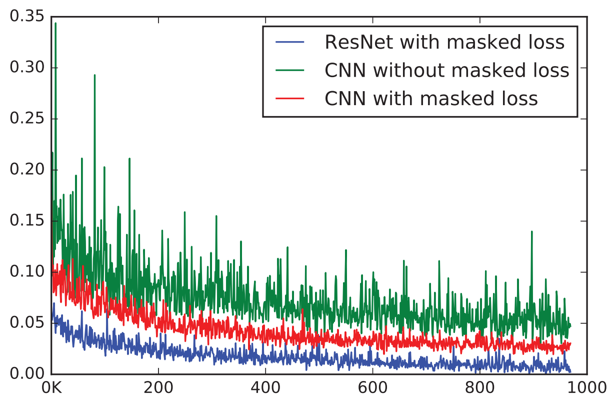

Figure 8: Evaluation losses for the three models

We first define a metric called *detection error* as in 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 measures the degree at which a predicted value deviates from its ground truth value. If the deviation, or error, is larger than 10%, the prediction is counted 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 in the test. The result is summarized in Table 2. The result shows that the *masked loss objective function* improves the test accuracy by eight times over the traditional CNN without considering missing target values.

Table 2: Detection Accuracy

Model	*Average error rate*
CNN without *masked loss*	75%
CNN with *masked loss* (ML-CNN)	**9%**
ML-ResNet	**9%**

5 CONCLUSION

In this paper, we design a new deep learning facial keypoint detection solution, Masked Loss Residual Convolutional Neural Network (ML-ResNet), which is a residual neural network with a masked loss objective function. ML-ResNet can effectively work on facial keypoint detection datasets that have missing target values. The *masked loss function* can mask out the errors between ground truth values and predicted values with a *masked matrix*. As a result, the detection performance will not be hurt by the errors introduced by the images that have missing target values, so that they can be attained in the datasets to prevent overfitting in the training stage. With extensive evaluation and comparison, ML-ResNet outperforms other CNN solutions in training efficiency and stability as well as the facial keypoint detection accuracy. In addition, *masked loss objective function* can be also used in other CNN solutions to improve their performance in both training and detection.

6 ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. #1408165.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
[6] Xiangxin Zhu and Deva 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.
[7] Lin Liang, Rong Xiao, Fang Wen, and Jian Sun. Face alignment via component-based discriminative search. In *European Conference on Computer Vision*, pages 72–85. Springer, 2008.
[8] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.
[9] Masatoshi Kimura, Takayoshi Yamashita, Yuji Yamauchi, and Hironobu 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.
[10] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik 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.
[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
[12] Hanxi Li, Yi Li, and Fatih Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Transactions on Image Processing*, 25(4):1834–1848, 2016.
[13] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.
[14] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
[15] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.
[16] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
[17] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
[18] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.
[19] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.
[20] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, pages 550–558, 2016.
[21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[22] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.