1000 Fps Highly Accurate Eye Detection with Stacked Denoising Autoencoder

Wei Tang, Yongzhen Huang, and Liang Wang(✉)

Center for Research on Intelligent Perception and Computing,
National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, China
wangliang@nlpr.ia.ac.cn

Abstract. Eye detection is an important step for a range of applications such as iris and face recognition. For eye detection in practice, speed is as equally important as accuracy. In this paper, we propose a super-fast (1000 fps on a general PC) eye detection method based on the label map of the raw image without face detection. We firstly produce the label map of a raw image according to the coordinates of its bounding box . Then we train a stacked denoising autoencoder (SDAE) which is specifically designed to learn the mapping from the raw image to the label map. Finally, through an effective post-processing step, we obtain the bounding boxes of two eyes. Experimental results show that our method is about 2,500 times faster than the deformable part-based model (DPM) while maintaining a comparable accuracy. Also, our method is much better than the popular LBP+Cascade model in terms of both accuracy and speed.

Keywords: Eye detection · Autoencoder · Label map

1 Introduction

As a challenging problem in computer vision, eye detection has attracted increasing attention in recent years due to its importance in some real applications such as iris and face recognition. Eye detection aims to solve the problem of getting the accurate position of eyes in a given image. Great achievements have been made on the accuracy of object detection over the past years [2] [3] [7] and these methods could be directly utilized to eye detection.

However, when facing truly practical problems, we find that few methods can run at a fast speed and keep a high accuracy at the same time. On one hand, despite of the great accuracy achieved by many recently proposed methods such as DPM [2] and RCNN [3], they usually rely on tools of high performance computing (HPC) for the demand of real-time detection. Sometimes, even though the HPC technology is adopted, the speed still cannot meet the requirements in applications such as on the embedded devices. On the other hand, traditional methods like LBP+Cascade can run in real time, but their detection accuracy is usually not satisfactory.

H. Zha et al. (Eds.): CCCV 2015, Part II, CCIS 547, pp. 237–246, 2015.
DOI: 10.1007/978-3-662-48570-5_23

In this paper, we propose a novel method based on the label map to address fast and accurate eye detection. To obtain great acceleration, we adopt SDAE [14] to learn the mapping from raw image to label map image, which can be very fast in testing because SDAE needs only a few times of matrix multiplication.

Label map has been proposed in [8] for face parsing. Our method differs from that in two aspects. Firstly, the method in [8] deals with the face parsing problem, so the label map needs segmentation for each pixel. However, our task is specific object detection and the label map with the location of the bounding box is enough, which means traditional object detection datasets can be directly utilized to train our model. Secondly, face parsing in [8] needs the face detection results as the input. In our approach, to avoid the use of face detector, we adopt a different strategy called *patch based label map training*. This strategy ensures getting the whole label map with a high accuracy, at the same time at a super-fast speed.

The major contributions of this paper are summarized as follows:

1) We propose a novel method based on the label map for reliable eye detection, which avoids the time-consuming face detection. It is robust to illumination changes, non-rigid deformation, incomplete object and partial occlusion.
2) We specifically design a SDAE model to learn the mapping from a raw image to the label map, and propose a patch based label map training strategy to effectively train the SDAE model.
3) Our approach is about 2,500 times faster than DPM while maintaining a similar accuracy and is much better than the LBP+Cascade model in terms of both accuracy and speed, which gives the potential for our approach to be used in computing-limited scenarios such as the embedded mobile devices.

2 Related Work

The work proposed in this paper is related to object detection and deep learning. We simply introduce some related work as follows.

Object Detection. Object detection has long been studied and attracted increasing attention in recent years. As for dealing with real-time tasks such as face detection in videos, LBP+Cascade [7] might be one of the most mature methods and has been proved very effective in common use. Deformable part based model (DPM) proposed in [2] is another milestone because of its high detection accuracy. Both LBP+Cascade and DPM in essence are the sliding window based methods.

These methods first turn an image into a large amount of image windows by slidingly sampling windows in the image pyramid. Then features (LBP in LBP+Cascade and HOG in DPM) are extracted from each window and classified by a category specific classifier (Cascade in LBP+Cascade and Latent SVM in DPM). Finally through a post-processing method named non-maximum suppression (NMS) [2], a bounding box surrounding the specific object can be obtained. Sliding window based methods turn out to be effective in some object

detection tasks such as face detection [7], pedestrian detection [2]. A main problem for sliding window based methods is the large amount of image windows. Especially, when objects in an image have a large range of size variance, to guarantee a high recall, the number of layers in the image pyramid should be larger and the stride between two windows should be smaller, which will produce a huge number of image windows. Classifying these image windows will be quite time-consuming. Our method adopts a label map based measure which avoids constructing the image pyramid, and thus significantly reduces the number of image windows (Section 3).

Deep Learning. The deep learning technology, with its strong representation learning capacity, has been utilized to deal with various computer vision problems and great success has been achieved in many areas such as image classification [6] [11] and object detection [3]. There are different deep learning models that have been proposed, such as DBN [5], Autoencoder [10], Convolutional Neural Network (CNN) [6]. Among all these models, CNN may be the most widely used, but its high computing cost is the biggest obstacle in real-time scenarios. Autoencoder (AE) is trained in an unsupervised manner by setting the output equal to the input. Lots of variants of AE have been developed in recent years such as denoising AE(DAE)[12], dropout AE [10], sparse AE [9] and stacked AE (SAE) [13]. AE has its own advantage that all it needs is just a few times of matrix multiplication, which leads to a super-fast forward propagating speed when testing.

3 Our Method

In this section, we detail our method. The whole framework includes three parts: data preparation, SDAE training and bounding boxes acquisition. We explain each part one by one below. More implementation details will be further introduced in Section 4.

3.1 Data Preparation

Two key problems we should solve before training a SDAE are 1) how to get the label map from traditional object detection datasets and 2) how to ensure we have enough data to effectively train the SDAE.

Getting Label Map. Unlike the method in [8], we do not need accurate label map with the segmentation for each pixel during training. Traditional object detection datasets can be directly used in our framework. In particular, we just utilize the size of the input image and the labeled bounding box to create a binary image with the same size. As shown in Fig. 1, we set the value of pixels in the bounding box to one and other pixels to zero, and thus get the label map used in our method.

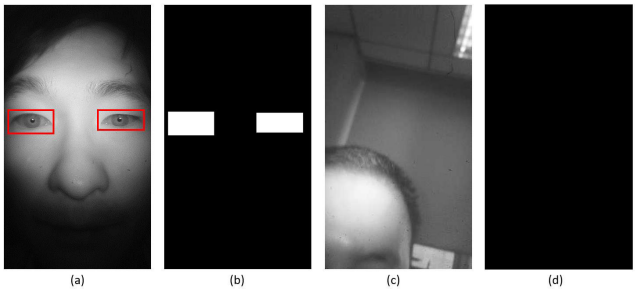

Fig. 1. (a) is an original image with bounding boxes and (b) is the corresponding label map. (c) is an image without the target object and (d) is the corresponding label map.

Patch Based Label Map Training. In our method, instead of first performing face detection as some other methods do, we directly perform eye detection on the whole input image. But training a SDAE with the whole image as input is proved to be unreasonable. For example, although the input image is 480x268, we experimentally find that the size of 80x44 is almost the limit to guarantee good performance. A simple AE with such a size will have ten millions of parameters. Optimizing so many parameters needs huge amount of training data and the optimized model will be very large so that it is hard to be loaded to a normal PC memory.

To solve this problem, we propose a training strategy called *patch based label map training*. In this strategy, instead of using the whole image as the input, we randomly crop image patches from the whole image with a reasonable size as the training data.

Another key point in this strategy is the cleaning of the generated label map patches before training. Let us define the response rate of a label map patch as:

$$\mathrm{r} = \frac{\sum_{i=1,j=1}^{i=N,j=N} I(i,j)}{N^2} \tag{1}$$

where I denotes the label map patch, N denotes the size of I, $I(i,j) \in \{0,1\}$ denotes the pixel value at (i,j) in I. We find that, if the training data contains label map patches with a small r, the output label map will have many small noisy response regions, which will be a severe problem in the post-processing procedure. Therefore if r of a patch is smaller than a predefined threshold, we set the label map of this patch to 0.

Because of the above strategy we propose, we can easily sample thousands of image patches from one original image. Meanwhile, the model can be smaller so that fewer parameters need to be optimized. Fig. 2 shows the procedure of our strategy. To increase the contrast, the red border is added for sampled patches. The r value of the upper patch in Fig. 2 is above the threshold, so no further processing is needed. But the r value of the lower patch in Fig. 2 is smaller than the threshold, so all the values in this patch are set to 0.

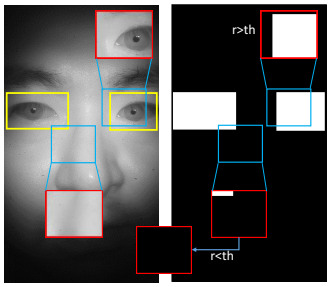

Fig. 2. The procedure of the training strategy described in Section 3.1. (Best viewed in color)

3.2 SDAE Training

The whole training procedure includes two parts: pre-training and fine tuning. The whole model architecture is shown in Fig. 3. (a) and (b) are two DAEs used for pre-training. (c) is the SDAE used in our framework.

Pre-training. Pre-training is an unsupervised layer-wise initialization procedure for deep network to avoid getting stuck in local minima or plateaus [1]. In our framework, we choose a variant of the conventional AE called denoising AE (DAE) as the building block of SDAE. DAE learns to recover a data sample from its corrupted version, which means it can learn more robust features than conventional AE. The architecture of DAE is shown in Fig. 3(a).

Suppose there are N training samples. Let i_k denote the kth image patch and $\tilde{i}_k$ denote the corrupted i_k, where corruption can be Gaussian or salt-and-pepper noise. Let W^1 and W^2 denote the weights (including the bias) for the encoder and decoder respectively. A DAE is learned by solving the following optimization problem:

$$\min_{W^1,W^2} \sum_{k=1}^{N} ||i_k - \hat{i}_k||_2^2 + \lambda(||W^1||_F^2 + ||W^2||_F^2) \tag{2}$$

where

$$h_k = f(W^1 \tilde{i}_k) \tag{3}$$

$$\hat{i}_k = f(W^2 h_k) \tag{4}$$

Here λ is a parameter that balances the reconstruction loss and weight penalty terms, $||\cdot||_F^2$ denotes Frobenius norm, and $f(\cdot)$ is a nonlinear activation function which is typically a sigmoid function or hyperbolic tangent function. As shown in Fig. 3, two DAEs are trained in our framework, thus leading to three hidden layers in SDAE.

Specifically Designed SDAE and Fine-Tuning. In the fine-tuning procedure, we utilize the two pre-training DAEs to build a SDAE. The weights of SDAE are all initialized with the weights from the pre-training stage as shown in Fig. 3(c) except that the weights between the last hidden layer and the output

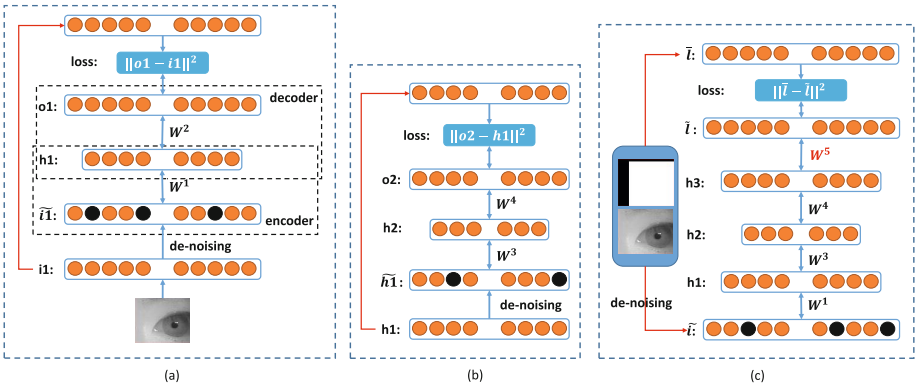

Fig. 3. (a) and (b) are two DAEs used for pre-training.(c) is the SDAE used in our framework. The weights in (c) are initialized with weights from (a) and (b) except that the weights between $h3$ and $\tilde{l}$ are randomly initialized.

layer are randomly initialized. These weights in conventional SDAE should be set to W^2, because the output is just a re-construction of the input. But in our method, we expect the output to produce the corresponding label map of the input image patch, so we initialize these weights with a random matrix denoted W^5 aiming to let the optimizing algorithm search for the optimal solution in a larger scope.

Other Strategies Adopted in Training. Overcomplete filters are used in the hidden layer of DAE. In conventional AE, the hidden layer is always in a "bottleneck" style. Overcomplete filters demand that the number of units in the hidden layer is larger than that of the input layer because it has been found that an overcomplete basis can usually capture better image structure [16].

To further learn more meaningful features, we also adopt sparsity constraints [4] imposed on the hidden units. If a sigmoid activation function is used, the output of each neural unit can be regarded as the probability of being active. Let ρ_i denote the target sparsity level of the ith unit and $\hat{\rho}_i$ denote its average empirical activation rate. The cross-entropy of ρ_i and $\hat{\rho}_i$ can then be introduced as an additional penalty term to Eqn.(2):

$$\sum_{j=1}^{s_2} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{5}$$

where s_2 is the number of hidden units.

3.3 Bounding Boxes Acquisition

Once we obtain many label map patches of the input image, we can merge these patches to form the whole label map according to their original positions. Example merging result is shown in Fig. 4(a) and we usually binarize the label

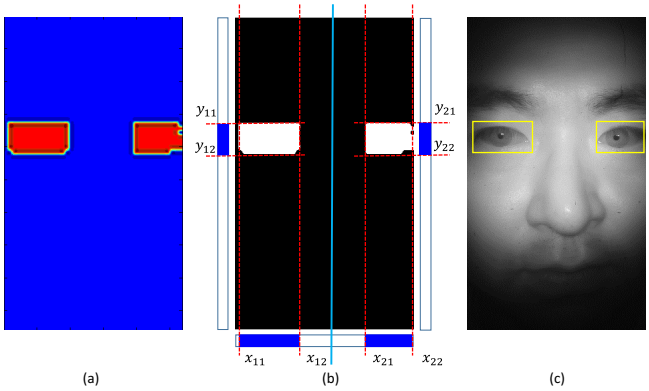

Fig. 4. (a) is the label map image obtained by merging all the label map patches from SDAE. (b) is the binary label map image, showing how to produce bounding boxes from label map. Please see the text for details. (c) is the result we get finally.

map as shown in Fig. 4(b). To get the bounding box from such a label map, we introduce a simple but very effective method in our eye detection task.

This method is first to add the matrix of the binary image along y axis to get one or two longest continuous non-zero sequence, which corresponds to the x coordinates of the eyes. Then we can separate the binary image into two parts according to the x coordinates. Adding the matrix of each part along x axis will get the corresponding y coordinates. We show this procedure in Fig. 4(b).

When response areas in the label map cannot be separated by x or y axis, we can also adopt some methods such as finding contours to get the bounding box from the label map.

4 Experimental Results

In this section, we first introduce the implementation details of our experiments. Then we present our experimental results including the results of different strategies, comparison with other methods and the visualization of our detection results.

4.1 Experimental Setting

Dataset. We collect 2,732 near-infrared eye images as the dataset used in our experiments for the background of this task is the Asian iris recognition. We randomly choose 2,182 images for training and the remaining 550 images for testing. The image size is 480x268. In our experiments, we resize all the images to 80x44. The patch size is 36x36. In testing, we sample patches with a stride of 18, just the half of the patch size.

Implementation Details. We empirically set the threshold of r to 0.02. For DAE and SDAE, we set the denoising ratio to 0.5, the sparsity target to 0.05,

Table 1. Effectiveness comparison of different strategies.

F1	h1 = 1024, h2 = 512	h1 = 2048, h2 = 1024
$r = 0$	—	0.879
$r = 0.02$	0.892	0.906

the weight penalty to 0.0001, the batch size to 100 and the learning rate to 0.2 in DAE and 0.1 in SDAE respectively. The sizes of $h1$ and $h2$ are 2,048 and 1,024 respectively. For DPM, the settings suggested by the paper [15] are used.

Evaluation Metrics. For accuracy, we use the same criterion as in [2] that the predicted bounding box is valid if its overlap ratio with the ground truth is bigger than 50%. Because our method does not return a score for a predicted bounding box, we use $F1$ value as the performance metric instead of the ROC curve. Let P denote precision and R denote recall rate, $F1$ can be defined as follows:

$$F_1 = \frac{2PR}{P + R} \tag{6}$$

From the formula, we can see that $F1$ value is a balance between precision and recall rates.

For speed, we use frames per second (fps) as the performance metric.

4.2 Basic Results

Effectiveness of Cleaning Small Response Area. We set r to 0 and 0.02 separately. Results show that when r is equal to 0, the label map will have some noisy regions which results in the overlap between the predicted bounding box and ground truth being less than 50%. In our test, this will reduce the accuracy by about 2.7% compared with $r = 0.02$.

Effectiveness of Overcomplete Filters. Except for setting $h1$ to 2,048 and $h2$ to 1,024, we also use the conventional "bottleneck" hidden layer with $h1 =$ 1,024 and $h2 = 512$. It shows that by adopting overcomplete filters, the accuracy can be improved by about 1.4%. All these results are shown in Tab. 1.

4.3 Comparison

We compare our method with other methods from two aspects: speed and accuracy. As for the methods we choose to compare with, LBP+Cascade is the most widely used object detection method and DPM has achieved excellent results on the challenging PASCAL VOC dataset. The accuracy and speed comparisons are shown in Fig. 5. We can see that our method can get a surprising 1,000 fps on a general PC with CPU (i7-3770 in our experiments), which is 17 times faster than LBP+Cascade, 140 times faser than the fastest DPM [15] and 2,500 times faster than original DPM [2]. The accuracy of our method is slightly lower than DPM but obviously better than LBP+Cascade. Overall, our method achieves a super-fast speed while maintaining a high accuracy.

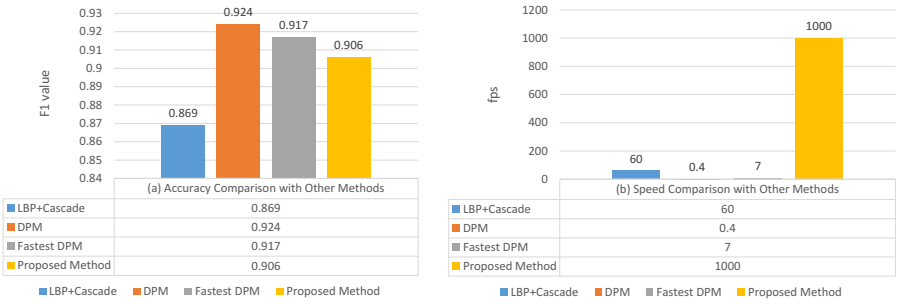

Fig. 5. Comparison with other methods on accuracy and speed.

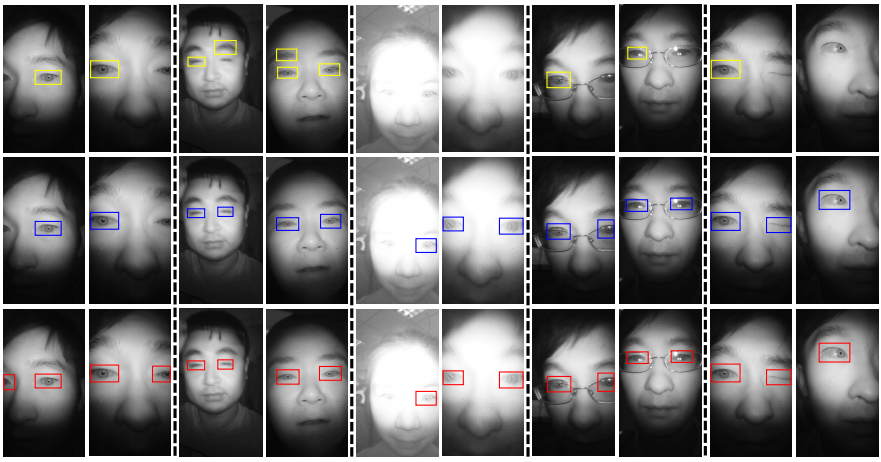

Fig. 6. Example results produced by LBP+Cascade(top), DPM (middle) and our approach (bottom). From left to right, we can see that our method is robust to the incomplete object, background disturbance, changes of illumination, occlusion and non-rigid deformation (best viewed in color).

4.4 Visualization

We show some detection results in Fig. 6. From left to right, we can see that our method can effectively deal with the incomplete object (such as the incomplete eyes), background disturbance (such as the disturbance of eyebrows), changes of illumination, occlusion (such as wearing glasses) and non-rigid deformation (such as the non-rigid deformation of eyes).

5 Conclusion

In this paper, we have presented a label map based eye detection method. By training a specifically designed SDAE, the label map can be accurately pro-

duced. The resulting method is 2,500 times faster than DPM while maintaining a comparable accuracy, which shows the great potential of our method to be used for real-time applications and in computing-limited scenarios.

Acknowledgments. This work is jointly supported by National Basic Research Program of China (2012CB316300), National Natural Science Foundation of China (61135002, 61420106015, 61203252), SAMSUNG GRO Program, CCF-Tencent Open Fund and 360 OpenLab Program.

References

1. Bengio, Y.: Learning deep architectures for AI. Foundations and trends in Machine Learning **2**(1), 1–127 (2009)
2. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. TPAMI **32**(9), 1627–1645 (2010)
3. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
4. Hinton, G.: A practical guide to training restricted boltzmann machines. Momentum **9**(1), 926 (2010)
5. Hinton, G.E.: Deep belief networks. Scholarpedia **4**(5), 5947 (2009)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
7. Liao, S.C., Zhu, X.X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: Lee, S.-W., Li, S.Z. (eds.) ICB 2007. LNCS, vol. 4642, pp. 828–837. Springer, Heidelberg (2007)
8. Luo, P., Wang, X., Tang, X.: Hierarchical face parsing via deep learning. In: CVPR. IEEE (2012)
9. Ng, A.: Sparse autoencoder. CS294A Lecture notes 72 (2011)
10. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
11. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. arXiv preprint arXiv:1409.4842 (2014)
12. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, pp. 1096–1103. ACM (2008)
13. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research 11 (2010)
14. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. In: NIPS, pp. 809–817 (2013)
15. Yan, J., Lei, Z., Wen, L., Li, S.Z.: The fastest deformable part model for object detection. In: CVPR. IEEE (2014)