Pruning the Seg-Edge Bilateral Constraint Fully Convolutional Network for Iris Segmentation

Hui Zhang[1], Junxing Hu[2], Jing Liu[1], Zhaofeng He[3(✉)], Xingguang Li[1], and Lihu Xiao[1]

[1] Beijing IrisKing Co., Ltd., Beijing, China
[2] Institute of Automation Chinese Academy of Sciences, Beijing, China
[3] Beijing University of Posts and Telecommunications, Beijing, China
zhaofenghe@bupt.edu.cn

Abstract. Iris semantic segmentation in less-constrained scenarios is the basis of new generation of iris recognition technology. In this paper, we reexamined our approach iris segmentation framework, named Seg-Edge bilateral constraint network (SEN), which contains an edge map generating network which passes detailed edge information from low level convolutional layers to iris semantic segmentation analysis layers and segmentation-edge bilateral constraint structure for focusing on interesting objects. To reduce the number of network parameters, we propose pruning filters and corresponding feature maps that are identified as useless by l_1-norm and l_2-norm, which results in a lightweight iris segmentation network while keeping the performance almost intact or even better. A novel l_1-norm or [l_1-norm, l_2-norm] clustering based pruning method is proposed to improve pruning effect and avoid the time consuming manual design. Experimental results suggest that the proposed SEN structure outperforms the state-of-the-art iris segmentation methods, and the clustering based pruning methods outperform manual design in both compression ratio and accuracy.

Keywords: Iris segmentation · Bilateral constraint domain transform · Model pruning

1 Introduction

Iris recognition is a reliable identification technique due to the stability and uniqueness of the iris in biometrics. In the last decades, iris recognition has been widely used in mines, prison, banks, police and entry and exit control. However, the traditional iris recognition only can be used under strictly limited conditions and asks for highly cooperated users. The user friendliness iris recognition system inevitably leads to a decline in iris image quality, which seriously affects the difficulty of iris image preprocessing. For example, it is more difficult to locate iris parts in defocus blur, motion blur, occlusion, or reflection iris images than

H. Zhang and J. Hu—These authors contribute equally to this article.

Y. Peng et al. (Eds.): ICIG 2021, LNCS 12889, pp. 641–653, 2021.
https://doi.org/10.1007/978-3-030-87358-5_52

in high quality iris images. In fact, the performance of iris segmentation [1–3] directly affects the performance of iris recognition. The higher the segmentation accuracy, the more precise the retained iris information.

Traditional iris segmentation methods usually includes preprocessing, denoising, boundary detection, and post-processing. There are several typical methods for iris segmentation. Daugman [1] proposes using an integro-differential operator to detect eyelids and locate iris boundaries. Wildes [2] presents exploiting a circular Hough transform to localize iris boundaries in iris images. He et al. [4] propose the method which is inspired by Hooke's law.

In the past few years, some algorithms based on neural network are proposed for the pixel level iris segmentation. Proença [5] exploits the neural network with one hidden layer. Tan et al. [6] use a typical 3-layer feed-forward neural network. Liu et al. [7] propose a multi-scale fully convolutional network. We proposed the Seg-Edge bilateral constraint network (SEN) for iris segmentation in [8], which improves the average segmentation errors over the state-of-the-arts by 2.22% and 22.03% on UBIRIS.v2 [9] and CASIA.v4-distance [10] dataset respectively.

To reduce the computational complexity and model size, we study the network pruning base on the l_1-norm and l_2-norm of feature maps. The pruning foundation includes: filters corresponding to feature maps with small l_1-norm or l_2-norm values extract little information and contribute little to the segmentation task; filters corresponding to feature maps with large different l_1-norm and l_2-norm values may contain valid information for segmentation. We prose a novel l_1-norm and l_2-norm clustering based pruning strategy, which not only save thousands of hours for manual design, but also achieve better accuracy with larger compression ratio.

2 Related Work

2.1 Semantic Segmentation

With the development of deep learning methods, the semantic segmentation has attracted more and more attention. The appearance of Fully Convolutional Networks (FCN) [11] leads to a rapid increase in the number of end-to-end semantic segmentation networks. The FCN model transforms all of the fully connected layers to convolutional layers which allows the input image of any size. SegNet [12] is an encoder-decoder structure, but it uses max-pooling indices to enhance location information. The DeepLab model proposed in [13] uses atrous convolution and fully connected Conditional Random Field to avoid the reduction of the spatial resolution of feature maps. The atrous convolution effectively enlarges the field-of-view of filters to incorporate a larger context without increasing the number of parameters and the amount of computation. DeepLab v2 [14] uses Atrous Spatial Pyramid Pooling (ASPP) with multiple sampling rates to robustly segment objects.

2.2 Model Pruning

For network model compression, a lot of methods have been proposed. LeCun et al. [15] propose Optimal Brain Damage to remove unimportant weights from a network. They use second-derivative information to make a tradeoff between network complexity and training set error. Han et al. [16] remove the connection whose weight is lower than a threshold after an initial training phase and converts a dense, fully-connected layer to a sparse layer. Li et al. [17] present removing filters together with their connecting feature maps to reduce the computation cost. Under the guidance of l_1-norm, this approach does not produce sparse connectivity patterns and not need the additional regularization.

3 Proposed Methods

3.1 Seg-Edge Bilateral Constraint Network

The structure of Seg-Edge bilateral constraint network (SEN) is shown in Fig. 1. It is composed of three components: backbone; iris edge map using rich convolutional features (RCF); bilateral constraint domain transform (BCT).

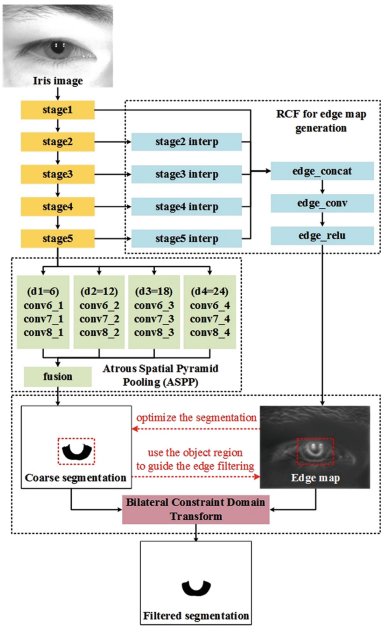


Fig. 1. The architecture of the proposed Seg-Edge bilateral constraint network (SEN).

Backbone Structure. We exploit the FCN model DeepLab v2 [14] as the backbone structure. The 'stage1' to 'stage5' in Fig. 1 stands for a bunch of convolutional layers of VGG-16. See [8] for more details.

Rich Convolutional Features for Edge Map Generation. Low-level convolutional layers contains richer edge features at different scales and sharpness degrees. We exploit all of them to predict the edge map following [18]. For medial convolutional layers, the method rescales their outputs to the original size, denoted as 'stage2 interp' to 'stage5 interp' in Fig. 1. A concat layer concatenates them in the channel dimension to one output. A convolutional layer with 1×1 kernel is used to produce the iris edge prediction.

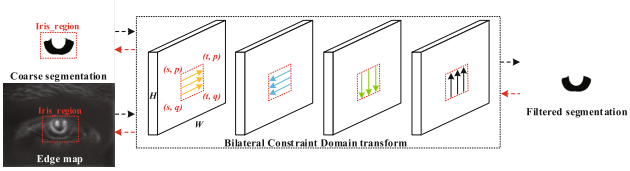


Fig. 2. Illustration of the bilateral constraint domain transform. The black dashed arrow indicates the forward propagation. The red dashed arrow indicates the back propagation. The red dashed box indicates the iris region, in which the edge map and the coarse segmentation are recursively filtered across rows and columns. (Color figure online)

Bilateral Constraint Domain Transform. We propose a novel bilateral constraint domain transform (BDT) structure in [8]. The iris edge map generated from intermediate convolutional layers can pass more detailed edge information to segmentation results. We use the iris region produced by coarse segmentation to constrain the edge filtering scope, which allows the edge filtering to focus on the interesting object parts. As illustrated in Fig. 2, the BDT recursively filters inputs across rows and columns through K iterations.

During the forward propagation, the filtering is performed along with four directions (left to right, right to left, top to bottom, and bottom to top) in sequence. For 2-D inputs of height, H and width W, the output $y_{i,j}$ is computed as: $y_{i,j} = (1-w_{i,j})x_{i,j}+w_{i,j}y_{i,j-1} i = p, ..., q \quad (1 \le p \le q \le H), j = s, ..., t \quad (2 \le s \le t \le W)$, where $x_{i,j}$ is the pixel value at $(i, j)$of the coarse iris segmentation, p and q are the lower and upper bound of the iris region in the vertical direction, s and t are the bound in the horizontal direction. The weight $w_{i,j} \in [0, 1]$ is a feedback coefficient which is related to the iris edge map.

During the backward propagation, the segmentation errors at the output $y_{i,j}$ are back propagated through the BDT onto its inputs. To avoid the interference from the non-iris area and constrain the edge filtering inside the object region, we add the same limitation to both the coarse segmentation and the edge map. The derivative is calculated as:

$$\frac{\partial L}{\partial x_{i,j}} \leftarrow (1 - w_{i,j})\frac{\partial L}{\partial y_{i,j}}; \qquad \frac{\partial L}{\partial y_{i,j-1}} \leftarrow \frac{\partial L}{\partial y_{i,j-1}} + w_{i,j}\frac{\partial L}{\partial y_{i,j}}$$

$$\frac{\partial L}{\partial w_{i,j}} \leftarrow \frac{\partial L}{\partial w_{i,j}} + (y_{i,j-1} - x_{i,j})\frac{\partial L}{\partial y_{i,j}}; \qquad \frac{\partial L}{\partial g_{i,j}} \leftarrow -\frac{\sqrt{2}}{\sigma_k}\frac{\sigma_s}{\sigma_r}w_{i,j}\frac{\partial L}{\partial w_{i,j}}$$

$$\sigma_k = \sqrt{3}\sigma_s \frac{2^{K-k}}{\sqrt{4^K - 1}}; i = q, ..., p(1 \leq p \leq q \leq H), j = t, ..., s(2 \leq s \leq t \leq W)$$

3.2 Iris Segmentation Model Pruning

To reduce the parameters and floating point operations (FLOP) of model, the model pruning is an intuitive and efficient way. We adopt the l_1-norm based pruning strategy in [17] as the basic pruning method. The original pruning method needs one layer by one layer experiments and manual design, which is time consuming and difficult to operate. To overcome the drawback of manual design and improve pruning effectiveness, we propose clustering based methods to determine the pruning ratio with less human intervention.

Pruning Filters in a Single Layer. It is crucial to determine the pruning object and the corresponding importance measure. We first prune the single layer to observe its sensitivity to pruning. To avoid producing sparse connectivity patterns and the additional regularization, we use the filter of the convolutional layer as the pruning object following [17]. The procedure of pruning is as follow: Step 1: For the ith convolutional layer, let $f_{i,j}$ denotes jth filter, calculate its l_1-norm $l_{ij} = \sum |f_{i,j}|$; Step 2: Sort filters by l_{ij}; Step 3: Prune m filters with the smallest l_{ij} and connecting feature maps; Step 4: Create a new model with the remaining filter weights and retrain it. We compare the pruned models and retrained pruned models with the original model. Some layers are sensitive to pruning as we can not recover the accuracy after pruning them.

Pruning Filters Across Multiple Layers by Manual Setting. For layers which are sensitive to pruning, we prune fewer or no filters of them. We adopt the *one-shot* pruning method which prunes filters across multiple layers at once and retrains the model [17]. We decide which filters can be pruned based on the single layer pruning experimental results.

Pruning Filters Across Multiple Layers by l_1-norm Clustering. We propose directly calculating the pruning ratio of each layer in the original model by using the clustering method. The k-means clustering is adopted. The clustering based pruning avoids manually setting thresholds which brings a lot of conveniences for practical usage. The pruning ratio (p_i) for the i-th layer is calculated as:

- Step 1: Calculate the l_1-norm l_{ij} of the j-th filter.
- Step 2: Sort all filters of the layer by the l_1-norm.
- Step 3: Prune 50% filters with the smallest l_1-norm and test the pruned model without retraining. The layer whose pruned model has small accuracy loss will be pruned more, otherwise less pruning.
- Step 4: Cluster the filters into C_i categories by l_{ij}, and a_c is the c-th category. The $(C_i + 1)$-quantiles are used as initial centers for the repeatability.
- Step 5: Sort clusters by l_{ij}, and a_1 has the smallest value.

– Step 6: If the layer is compressed less, $p_i = num\,(a_1)/o_i$. If the layer is compressed more, $p_i = \sum_{c=1}^{C_i-1} num\,(a_c)/o_i, C_i \geq 3$. where o_i is the number of output channels for the i-th layer, $num(\cdot)$ is used to count filter numbers.

Pruning Filters Across Multiple Layers by l_1-norm and l_2-norm Clustering. As demonstrated in paper [17] and our previous experiments, it may have little effect on segmentation results to prune the filters whose outputs are relatively small l_1-norm or l_2-norm values. The l_1-norm and the l_2-norm of one feature map usually show similar distribution. However, there are still some feature maps have different l_1-norm and l_2-norm value distribution, as shown in Fig. 3. If l_1-norm ($/l_2$-norm) is large while l_2-norm ($/l_1$-norm) is small, corresponding filters of this map may include important information for segmentation. Pruning filters just based on l_1-norm may possibly delete these important filters. So, we consider both l_1-norm and l_2-norm for clustering. The l_1-norm and l_2-norm are contacted to form a vector [l_1-norm, l_2-norm]. Figure 4 shows some examples of clustering result.

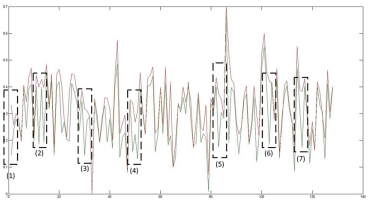
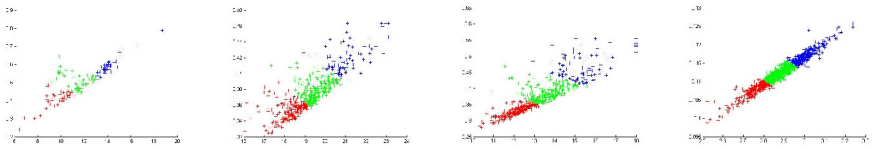


Fig. 3. An example l_1-norm and l_2-norm value distribution (conv2-2). Seven black dashed boxes illustrate some obviously inconsistent distributed of l_1-norm and l_2-norm.



(a) UBIRIS.v2 conv2_2 (b) UBIRIS.v2 conv5_1 (c) CASIA.v4-distance conv4_1 (d) CASIA.v4-distance conv7_2

Fig. 4. Examples of [l_1-norm, l_2-norm] clustering results.

The pruning ratio for the i-th convolutional layer is calculated as follows:

– Step 1: Calculate the l_1-norm l_{ij}^1 and l_2-norm l_{ij}^2 of the j-th filter respectively.
– Step 2: Sort all filters of the layer by the l_1-norm.
– Step 3: Same as step 3) in Sect. 3.2.

– Step 4: Calculate the mean of l_i^1 and l_i^2 for each layer respectively. α is a hyper-parameter and $\alpha = 4$ in our experiments, l_{ij}^1 and l_{ij}^2 are computed as:
  - if $l_{ij}^1 - mean(l_i^1) > \alpha * mean(l_i^1)$, $l_{ij}^1 = mean(l_i^1) + \alpha * mean(l_i^1)$;
  - if $mean(l_i^1) - l_{ij}^1 > \alpha * mean(l_i^1)$, $l_{ij}^1 = mean(l_i^1) - \alpha * mean(l_i^1)$;
  - if $l_{ij}^2 - mean(l_i^2) > \alpha * mean(l_i^2)$, $l_{ij}^2 = mean(l_i^2) + \alpha * mean(l_i^2)$;
  - if $mean(l_i^2) - l_{ij}^2 > \alpha * mean(l_i^2)$, $l_{ij}^2 = mean(l_i^2) - \alpha * mean(l_i^2)$.
– Step 5: Normalize l_{ij}^1 and l_{ij}^2 respectively by minimax normalization.
– Step 6: Cluster into C_i categories by $[l_{ij}^1, l_{ij}^2]$, and a_c is the c-th category.
– Step 7: Sort all clusters by l_{ij}^1 or $l_{ij}^1 + l_{ij}^2$, and a_1 has the smallest value.
– Step 8: Same as Step 6 in Sect. 3.2.

4 Experiments and Results

4.1 Datasets

We evaluate algorithms on two datasets: One is a subset of UBIRIS.v2 [9], which contains 500 images for training and 445 images for testing, short as UBIRIS; One is a subset of CASIA.v4-distance [10], which contains 300 images for training and 100 images for testing, short as CASIA.

4.2 Experimental Results

The accuracies of models are measured by the average segmentation error $ASE = (\sum_{i,j \in (H,W)} G(i,j) \oplus M(i,j))/(N \times H \times W)$, where N is the total number of the test images, H and W are height and width, G and M are the ground truth mask and the generated iris mask respectively. $\oplus$ represents an exclusive OR operation to compute the segmentation error. The comparational results of proposed SEN with other methods are listed in Table 1.

Table 1. Comparisons of the ASE with other methods.

Method	UBIRIS (%)	CASIA (%)
Proposed SEN	**0.88**	**0.46**
MFCNs [7]	0.90	0.59
RTV-L[1] [19]	1.21	0.68
Tan et al. [20]	1.31	–
Tan and Kumar [21]	1.72	0.81
Proença [5]	1.87	–
Tan and Kumar [6]	1.90	1.13

Table 2. Pruning ratios (%) of manual setting strategies. 'LY' indicates layer number. 1–21 indicate 21 layers from conv1_1 to conv7_4. 'PR' indicates pruning ratio.

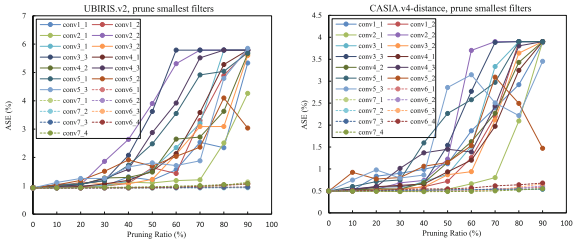
Prune_1	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
UBIRIS	PR	10	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	70	0	0	0	0
Prune_1	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
CASIA	PR	0	0	30	0	10	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	60
Prune_2	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
UBIRIS	PR	0	0	0	0	0	0	0	0	0	0	50	50	50	50	50	50	50	50	50	50	50
Prune_2	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
CASIA	PR	0	0	0	0	0	0	0	0	0	0	50	50	50	50	50	50	50	50	50	50	50
Prune_3	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
UBIRIS	PR	0	0	0	0	0	0	0	0	0	0	0	0	0	50	50	50	50	50	50	50	50
Prune_3	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
CASIA	PR	0	0	0	0	0	0	0	0	0	0	0	0	0	50	50	50	50	50	50	50	50

Table 3. The best *ASE* of retraining process are reported. 'Params' and 'FLOP' indicate the reduced percentage of parameters and FLOP for each pruning strategies.

Strategy	UBIRIS (%)			CASIA (%)		
	ASE	Params	FLOP	*ASE*	Params	FLOP
Prune_1	**0.926**	2.79	2.75	**0.489**	8.19	9.11
Prune_2	0.979	58.28	46.15	0.494	58.28	46.15
Prune_3	0.928	33.31	26.38	0.494	33.31	26.38

Pruning Results Based on Manual Setting. We design three different pruning strategies in total, and the design procedure is as follow:

a) After sorting filters by l_1-norm, we prune the smallest filters of each convolutional layer independently with different pruning ratios in the range of 10% to 90%, and evaluate the *ASE* of pruned model, seeing Fig. 5. Pruning some single layers may even improve the performance. We first prune these filters and retrain. The pruned layers and corresponding pruning ratios of models are denoted as Prune_1 as follows: (i) For UBIRIS, we prune 10%, 30%, 70% of conv1_1, conv7_1, conv7_2, respectively.



(a) UBIRIS (b) CASIA

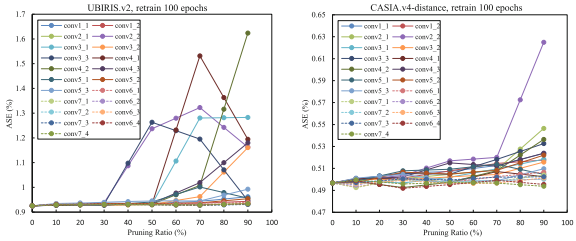
Fig. 5. The *ASE* after pruning filters with the smallest weights sum for each layer.

(ii) For CASIA, we prune 30%, 10%, 40%, 60% of conv2_1, conv3_1, conv6_3, conv7_4, respectively.

b) We retrain all pruned models and results are listed in Fig. 6. Some layers are sensitive to pruning, as we can not recover their accuracies even using small pruning ratios, such as conv2_2 layer. We empirically prune 50% filters of each layer from conv5_1 to the end of the model, denoted as Prune_2. Table 3 demonstrates that the *ASE* of the pruned model for the CASIA can be restored. But we can not restore the *ASE* for the UBIRIS.v2 because the layers of the fifth stage are sensitive to pruning for this dataset (seeing Fig. 6).

c) Based on the result of (2), we prune 50% filters of each layer from conv6_1 to conv7_4, denoted as Prune_3.

In conclusion, details of these three pruning strategy Prune_1, Prune_2, and Prune_3 are listed in Table 2, and results are shown in the Table 3. As a trade-off between the accuracy and the number of parameters, the proper pruning strategy for UBIRIS and CASIA is Prune_1 and Prune_2 respectively.



(a) UBIRIS (b) CASIA

Fig. 6. The *ASE* after retraining the pruned models.

Pruning Results Based on l_1-norm Clustering. As a trade-off between the accuracy and the number of parameters, we regard the pruning 50% filters as an appropriate pruning ratio. For results at 50% pruning ratio, we set the upper limit 10% for $loss_{ASE}$ (the loss of *ASE*). For the pruning method only using l_1-norm, the k-means is used to cluster sorted filters in each layer of the original models. Experimental results indicate 3 clusters is the best choice. Clustering results for models are shown in Table 4.

Following the clustering results, the pruning strategy PruneCl_1 ($loss_{ASE} \geq 10\%$: pruning less) is designed. We also find that some layers have large $loss_{ASE}$ so that not pruning them may lead to better results. Therefore, we add three strategies as PruneCl_2 ($loss_{ASE} \geq 10\%$: not pruning), PruneCl_3 ($10\% \leq loss_{ASE} < 50\%$: pruning less, $loss_{ASE} \geq 50\%$: not pruning), PruneCl_4 ($10\% \leq loss_{ASE} < 100\%$: pruning less, $loss_{ASE} \geq 100\%$: not pruning). Table 5 lists the detailed pruning ratios. We prune these filters and retrain, results are in Table 6.

Table 4. Clustering results of l_1-norm: the number of samples in each category

Layer	UBIRIS			CASIA			Layer	UBIRIS			CASIA		
	Cat 1	Cat 2	Cat 3	Cat 1	Cat 2	Cat 3		Cat 1	Cat 2	Cat 3	Cat 1	Cat 2	Cat 3
conv1_1	32	19	13	31	20	13	conv5_2	152	259	101	182	261	69
conv1_2	15	18	31	16	17	31	conv5_3	131	253	128	138	250	124
conv2_1	49	41	38	46	37	45	conv6_1	436	433	155	209	510	305
conv2_2	41	44	43	41	44	43	conv7_1	388	455	181	2	519	503
conv3_1	77	135	44	79	137	40	conv6_2	257	516	251	254	500	270
conv3_2	86	113	57	84	110	62	conv7_2	2	574	448	137	537	350
conv3_3	80	120	56	86	117	53	conv6_3	241	516	267	234	512	278
conv4_1	211	229	72	219	217	76	conv7_3	2	511	511	2	523	499
conv4_2	193	236	83	213	218	81	conv6_4	209	489	326	200	494	330
conv4_3	194	256	62	213	248	51	conv7_4	2	502	520	2	513	509
conv5_1	195	245	72	201	245	66							

Table 5. Pruning ratios (%) obtained by l_1-norm clustering. 'LY' indicates layer No.

PruneCl_1	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
UBIRIS	PR	50	20	40	30	30	30	30	40	40	40	40	30	30	80	80	80	60	70	50	70	50
PruneCl_2	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
UBIRIS	PR	0	0	0	0	0	0	0	0	0	0	0	0	0	80	80	80	60	70	50	70	50
PruneCl_3	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
UBIRIS	PR	50	0	40	0	0	30	0	0	0	0	0	0	0	80	80	80	60	70	50	70	50
PruneCl_4	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
UBIRIS	PR	50	20	40	0	30	30	0	40	40	0	0	30	30	80	80	80	60	70	50	70	50
PruneCl_1	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
CASIA	PR	50	30	40	30	30	30	30	40	40	40	40	40	30	70	50	70	70	70	50	70	50
PruneCl_2	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
CASIA	PR	0	0	0	0	0	0	0	0	0	0	0	0	0	70	50	70	70	70	50	70	50
PruneCl_3	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
CASIA	PR	0	30	40	0	0	0	0	0	0	0	0	0	0	70	50	70	70	70	50	70	50
PruneCl_4	LY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
CASIA	PR	0	30	40	0	30	30	0	40	0	0	0	0	0	70	50	70	70	70	50	70	50

Table 6. Accuracy result of pruning by l_1-norm clustering.

Strategy	UBIRIS (%)			CASIA (%)		
	ASE	Params	FLOP	*ASE*	Params	FLOP
PruneCl_1	0.961	74.18	70.15	0.520	72.91	69.51
PruneCl_2	0.930	47.41	37.54	0.500	44.58	35.31
PruneCl_3	0.939	48.62	45.88	0.512	44.89	40.05
PruneCl_4	0.959	65.79	61.54	0.512	50.13	47.71

Table 7. Clustering results of [l_1-norm, l_2-norm]: the number of samples in each category for all layers of UBIRIS and CASIA model respectively

Layer	UBIRIS			CASIA			Layer	UBIRIS			CASIA		
	Cat 1	Cat 2	Cat 3	Cat 1	Cat 2	Cat 3		Cat 1	Cat 2	Cat 3	Cat 1	Cat 2	Cat 3
conv1_1	32	20	12	33	19	12	conv5_2	204	245	63	205	242	65
conv1_2	15	18	31	15	18	31	conv5_3	141	250	121	143	251	118
conv2_1	56	60	12	27	41	60	conv6_1	487	393	144	227	507	290
conv2_2	36	51	41	26	59	43	conv7_1	443	419	162	208	499	317
conv3_1	91	114	51	89	117	50	conv6_2	270	502	252	264	491	269
conv3_2	101	97	58	94	101	61	conv7_2	292	500	232	255	473	296
conv3_3	90	115	51	81	111	64	conv6_3	265	519	240	258	509	257
conv4_1	201	224	87	211	211	90	conv7_3	227	483	314	196	446	382
conv4_2	208	239	65	206	234	72	conv6_4	230	494	300	250	498	276
conv4_3	192	245	75	201	248	63	conv7_4	242	505	277	227	500	297
conv5_1	206	219	87	214	219	79							

Table 8. Pruning ratios (%) obtained by [l_1-norm, l_2-norm] clustering.

UBIRIS	LY	1	2	3	4	5	6	7	8	9	10	11
	PR	50.00	23.44	40.63	28.13	35.55	41.02	31.64	39.26	38.48	37.50	40.23
	LY	12	13	14	15	16	17	18	19	20	21	
	PR	39.84	27.54	85.94	84.18	75.39	77.44	76.56	69.34	70.70	72.95	
CASIA	LY	1	2	3	4	5	6	7	8	9	10	11
	PR	46.88	23.44	42.19	23.44	31.64	34.38	31.64	43.36	40.23	39.26	41.80
	LY	12	13	14	15	16	17	18	19	20	21	
	PR	40.04	27.93	71.68	69.04	73.73	71.09	73.93	62.70	73.05	71.00	
CASIA without conv1_1 pruning	LY	1	2	3	4	5	6	7	8	9	10	11
	PR	0	23.44	42.19	23.44	31.64	34.38	31.64	43.36	40.23	39.26	41.80
	LY	12	13	14	15	16	17	18	19	20	21	
	PR	40.04	27.93	71.68	69.04	73.73	71.09	73.93	62.70	73.05	71.00	

Pruning Results Based on l_1-norm and l_2-norm Clustering. For the pruning method using l_1-norm and l_2-norm, we also use k-means to cluster $[l_{ij}^1, l_{ij}^2]$ of filters into $K = 3$ clusters each layer. Clustering results for models are shown in Table 7. Following the clustering results, prune strategies for UBIRIS and CASIA are designed are illustrate in Table 8. We prune these filters and retrain, results are listed in Table 9. The pruning ratio of Params and FLOP of this strategy is the largest. But the *ASE* for UBIRIS.v2 are better than the manual designed strategies Prune_2 which have much smaller pruning ratios (see Table 3) and l_1-norm clustering based strategies PruneCl_1, PruneCl_4 which have smaller pruning ratios (see Table 6).

Results of two datasets show different changing situations with strategy changes. We infer that the different is due to the different of the visible and the near infrared images. The near infrared images include more detailed texture. So the conv1_1 layer may contain more detailed or finer low-level perceptual information that cannot be pruned. To verify the conjecture, we conduct an experiment without pruning the conv1_1, as the 3rd row of Table 8. The result is listed in Table 9. It is a little better than using the PruneCl12 pruning strategy.

Table 9. Result of pruning by [l_1-norm, l_2-norm] clustering.

Strategy	$ASE(\%)$	Params (%)	FLOP (%)
UBIRIS	0.947	75.87	72.34
CASIA	0.530	74.58	70.62
CASIA without pruning conv1_1	0.520	74.55	68.79

5 Conclusions

In this paper, we reviewed our proposed Seg-Edge bilateral constraint network for iris segmentation. The iris edge map generated from convolutional layers optimizes the iris segmentation by aligning it with the iris boundary. The iris region produced by the segmentation limits the scope which makes the edge filtering pay more attention to the interesting target. The proposed clustering based pruning method is not only a substitute for time and computing resources consuming manual design, but also a better pruning method achieving much better segmentation accuracy and larger pruning ratio.

Acknowledgement. This work was supported by National Key Research and Development Program of China No. 2020AAA0140002.

References

1. Daugman, J.: High confidence visual recognition of persons by a test of statistical independence. TPAMI **15**(11), 1148–1161 (1993)
2. Wildes, R.: Iris recognition: an emerging biometric technology. Proc. IEEE **85**(9), 1348–1363 (1997)
3. Ma, L., Tan, T., Wang, Y., Zhang, D.: Personal identification based on iris texture analysis. TPAMI **25**(12), 1519–1533 (2003)
4. He, Z., Tan, T., Sun, Z., Qiu, X.: Toward accurate and fast iris segmentation for iris biometrics. TPAMI **31**(9), 1670–1684 (2009)
5. Proenca, H.: Iris recognition: on the segmentation of degraded images acquired in the visible wavelength. TPAMI **32**(8), 1502–1516 (2010)
6. Tan, C., Kumar, A.: Unified framework for automated iris segmentation using distantly acquired face images. TIP **21**(9), 4068–4079 (2012)
7. Liu, N., Li, H., Zhang, M., Liu, J., Sun, Z., Tan, T.: Accurate iris segmentation in non-cooperative environments using fully convolutional networks. In: ICB (2016)
8. Hu, J., Zhang, H., Xiao, L., Liu, J., Li, X., Li, L.: Seg-edge bilateral constraint network for iris segmentation. In: CVPR, pp. 5872–5881 (2017)
9. Proenca, H., Filipe, S., Santos, R., Oliveira, J., Alexandre, L.A.: The ubiris. v2: a database of visible wavelength iris images captured on-the-move and at-a-distance. TPAMI **32**(8), 1529–1535 (2010)
10. Casia.v4 database. http://www.cbsr.ia.ac.cn/china/Iris%20Databases%20CH.asp
11. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR, pp. 3431–3440 (2015)

12. Badrinarayanan, V., Handa, A., Cipolla, R.: Segnet: a deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling, arXiv preprint arXiv:1505.07293 (2015)
13. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected CRFs, arXiv preprint arXiv:1412.7062 (2014)
14. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. TPAMI **40**(4), 834–848 (2018)
15. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in Neural Information Processing Systems, pp. 598–605 (1990)
16. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems, pp. 1135–1143 (2015)
17. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets, arXiv preprint arXiv:1608.08710 (2016)
18. Chen, L.C., Barron, J.T., Papandreou, G., Murphy, K., Yuille, A.L.: Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform. In: CVPR, pp. 4545–4554 (2016)
19. Zhao, Z., Ajay, K.: An accurate iris segmentation framework under relaxed imaging constraints using total variation model. In: ICCV, pp. 3828–3836 (2015)
20. Tan, T., He, Z., Sun, Z.: Efficient and robust segmentation of noisy iris images for non-cooperative iris recognition. Image Vis. Comput. **28**(2), 223–230 (2010)
21. Tan, C., Kumar, A.: Towards online iris and periocular recognition under relaxed imaging constraints. TIP **22**(10), 3751–3765 (2013)