CE-PeopleSeg: Real-time people segmentation with 10% CPU usage for video conference

Ziyu Jiang[1] Zhenhua He[1] Xueqin Huang[1] Zibin Yang[2] Pearl Tan[2]
[1]Texas A&M University [2]EEO

Abstract

*Nowadays, video conference solutions are widely adopted for companies, education, and government. People segmentation is crucial for supporting virtual background, an essential video conference function to protect users' privacy. This paper demonstrated a people segmentation framework called CE-PeopleSeg, which employed an efficient segmentation method, structural pruning, and dynamic frame skipping techniques, leading to a fast inference speed on CPU. Our extensive experiments show that the proposed CE-PeopleSeg can achieve a high prediction mIoU of **87.9%** on Supervised People Dataset while reaching a real-time inference speed of **32.40 fps** on CPU with very low usage of **10%**. Our code would be released at https://github.com/geekJZY/EfficientPeopleSeg.git.*

1. Introduction

The demand for video conference techniques significantly increases recently. Especially, the outbreak of Coronavirus Disease 2019 (COVID-19) [31] greatly facilitates the adaptation of video conferencing solutions for companies, government, and education. As many offline events like seminars, trade shows, regular discussions, and teaching are canceled for suppressing the infection of COVID-19, people have to turn to video conferencing solutions such as Zoom, Teams, and google meets to continue hosting these events. For instance, in 2020, the number of monthly active users grew by 2.22 million [2].

Virtual background serves as an essential privacy protection method for video conferences. While showing the actual home layout may release privacy information, the virtual background function allows you to use any picture as background, which greatly protects your privacy like your physical locations and personal belongings.

People segmentation is a necessary technique supporting virtual background since it can generate the people contour required for background replacement. As video conference is a real-time service and its client machine is usually a personal computer without powerful computation tools like GPU, people segmentation's efficiency becomes crucial.

The advance of computer vision in recent years largely benefits People Segmentation. Many solutions based on Fully Convolutional Neural Network [24] are proposed [35]. Recent solutions like [35, 21] can achieve a real-time speed on powerful GPU. However, while most users have no access to such devices, a real-time solution on CPU is more practical.

In this paper, we propose a CPU Efficient People Segmentation Framework (CE-PeopleSeg) for video conferences. CE-PeopleSeg leverages an efficient segmentation frame called Feature Pyramid Network (FPN) [18] with mobileNetV2 [30] as feature extractor. FPN merges the low-level details with high-level semantics without using dilated convolution as in Deeplab [4]. Meanwhile, mobileNetV2 [30] further accelerates the feature extraction. Moreover, we explore the influence of resolution and find that People Segmentation Task can be conducted under very low resolution with marginally performance sacrificing. Besides, we extend structural pruning [20] to depthwise separable convolution employed in mobileNetV2 [30]which further boost the efficiency by avoiding redundant computation. Also, considering people typically would keep the same pose in the video conference, we employ a dynamic frame skipping technique to save more computational costs.

The main contributions can be summarized as follows:

- We present CE-PeopleSeg, an efficient people segmentation backbone on CPU for video conference. It achieves fast segmentation speed by incorporating an efficient model, structural pruning, and dynamic frame skipping.
- We extend the structural pruning technique to depthwise convolution, which enabling us further improving the efficiency of mobileNetV2.
- We proposed a simple and effective dynamic frame skipping technique that can automatically adapt to the target video.
- Our extensive experiments verify the excellent efficiency of the proposed CE-PeopleSeg. It can gener-

ate high-quality segmentation mask with **87.9%** mIoU on Supervisely Person Dataset while achieving a **real-time** inference speed of **32.40 fps** on CPU (Intel(R) Core(TM) i5-8400 CPU@ 2.80GHz) with super low usage of **only 10%**. Such fast speed not only satisfies the virtual background function requirement, but it also saves CPU computation resources for other tasks like video compression and transmission.

The rest of this paper is structured as follows. In Section 2, we introduce the related works. In section 3, we focus on describing the framework of the proposed CE-PeopleSeg. Data introduction and experiment results are illustrated in Section 4. Finally, we will conclude in Section 5.

2. Related works

2.1. People segmentation

People segmentation has been studied since a few decades ago. Shio and Sklansky [34] presented a method of segmenting monocular images of people in motion from a cinematic sequence of frames based on image intensities, motion, and an object model of the image of a person in motion. Zhao and Nevatia [46] used an efficient Markov chain Monte Carlo (MCMC) method with domain knowledge as proposal probabilities to segment individual humans in crowded situations. Fernández-Caballero et al. [7] introduced a new approach to process infrared-camera images for real-time human segmentation. A human candidate blob will be divided if it contains more than one person. Song et al. [35] proposed a fast and accurate people segmentation method with deep convolutional neural networks. This method was more than 10,000 times faster than the champion algorithm on the database of Baidu people segmentation competition. Due to the increasing demand for real-life applications, research about "humans" in the computer vision community, for example, portrait segmentation [32, 33, 34], has become a hot research area. Standard instance segmentation approaches combine object detection and segmentation of the object from bounding boxes. Zhang et al. [44] proposed a pose-based people segmentation framework based on the human pose rather than the proposal region. Lin et al [21] proposed a real-time, high-resolution background replacement method,which runs at 30fps in 4K resolution, and 60fps for HD on a modern GPU. Two neural networks were employed with a base network for computing low-resolution results and a high-resolution refinement network on selective patches. The method in Gruosso et al. [8] can automatically recognize and segment humans in videos without constraints on camera and human activity in the scene. Despite the fast development, current works still rely on powerful GPU to guarantee real-time speed, while the proposed CE-PeopleSeg can achieve real-time speed with low CPU usage.

2.2. Semantic segmentation

As a highly related topic, the developments in semantic segmentation would often benefit People Segmentation. For instance, the fully convolutional network (FCN), which achieves efficient pixel-wise prediction via replacing the fully connection layer with convolutional layer, is also widely adopted in the people segmentation works [35, 32, 21]. To improve the efficiency of FCN, ENet [28] employs an early down-sampling network. ICNet [45] leverages additional supervision on multi-resolution branches. BiSeNet [43] combined the spatial and contextual path for efficiently preserving both high-resolution and sufficient receptive field. Marin et al. [26] designs a content-adaptive down-sampling technique that favors sampling more for near boundary region. FasterSeg [5] automatically search an efficient framework. Some works also consider the memory efficiency [6]. While these works focus on improving GPU efficiency, we aim to build an efficient people segmentation framework on CPU.

2.3. Pruning

Pruning is an essential class of network compression techniques. It can be divided into unstructural and structural pruning methods.

Optimal Brain Damage [19] and Optimal Brain Surgeon [11] are early unstructural pruning works. They use the Hessian of the loss function as an indicator. Following works like Han et al. [10] proposed to prune the small weights in the neural networks. Some methods can also prune neural networks in a data-free manner [36]. Variatonal Dropout [17] can also be used to prune redundant weights [27]. Louizos et al. [25] learn sparse networks through L0-norm regularization. [14, 15] employ gating function to dynamically prune weights according to the input. However, since the learned weight is always sparse, these methods cannot achieve real speed-up with widely-used libraries/hardware [9].

On the other hand, structural pruning methods often prune the channels or even layers. The pruning structure is more organized and thus can be beneficial on widely-used libraries/hardware. [38, 42] also shows the structural pruning can be leverage for saving training cost. While a few works aim at pruning layers [37], most works focus on pruning channels. Li et al. [20] prune channels according to the weight norm of filters while Hu et al. [13] use the percentage of zeros in the output of the corresponding layer. Group sparsity is utilized in [39, 3]. [23, 41] learn which layer to be pruned via embedding sparse regularization to channel-wise scaling parameters. [40] groups convolutional layer weights to clusters to compress the model. We adopt the training method proposed in [23] given its good performance and conceptually simple implementation.

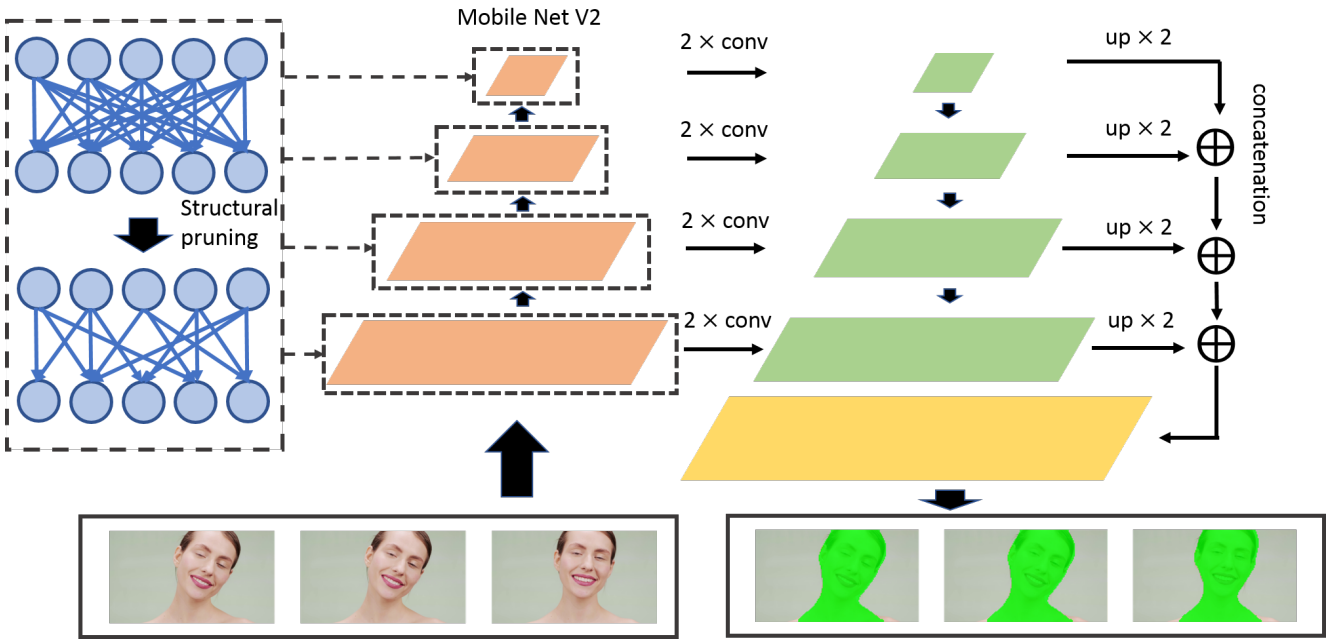

Figure 1. The overview of the proposed CE-PeopleSeg. It contains three modules: a Feature Pyramid Network segmentation network with MobileNetV2 as encoder, structural pruning applied on MobileNetV2, and the dynamic frames skipping for input frames.

3. Method

3.1. Framework overview

As demonstrated in Figure 1, the proposed CE-PeopleSeg is efficient on both model and data level. At the model level, we leverage FPN as the segmentation framework with MobileNetV2 as the backbone. The structural pruning technique is also utilized for further reducing the computation cost. As the neighboring frames are similar, we propose to dynamically skip the similar frames and directly adopt the previous frame's prediction, which further improves the overall efficiency.

3.2. Efficient Segmentation Backbone

We choose Feature Pyramid Network (FPN) as the segmentation framework. As shown in Figure 1, it is consists of a bottom-up and a top-down pathway. The bottom-up pathway is the feature encoding part, which computes the feature maps at multiple spatial scales. The spatial stripe is always set as 2. There are often many layers for processing the feature map at each scale. We define the multiple convolutional layers that process the feature map in the same scale as a *stage*. We choose mobileNetV2 here as the bottom-up pathway, given its efficiency. It contains four *stages* and has strides of $\{4, 8, 16, 32\}$. In the top-down pathway, inference starts from the top layer, where the semantical information is strong while being spatially coarser. This feature would be gradually upsampled and enhanced with the features from lower pyramid levels, and thus it can better localize the target object. For segmentation, We choose to upsample all the features at the top-down pathway to the same scale and then concatenate them together. The feature is then inputted with a segmentation head, which is a convolutional layer.

Feature Pyramid Network (FPN) design efficiently enhances the semantically strong high-level features with high-resolution low-level features. In contrast, another popular backbone, deeplab [4], uses the dilated convolution, which scales up every feature map, leading to very slow inference speed and large memory consumption.

3.3. Structural pruning

In this section, we would start by reviewing the structural pruning idea of [23]. Then we would describe how we extend the idea to depth-wise convolutional layers.

Structural pruning with scaling factor. The main idea of [23] is to introduce a scaling factor γ to each channel of the convolutional layers, which would be multiplied with the corresponding output channel. The γ and model weights are co-optimized in a data-driven manner. Finally, the channel with a small scaling factor can be pruned since it's not as important as the rest channels. Also, a sparsity regularization is imposed to make the scaling factors more sparse. Formally, the training loss can be expressed as

$$\mathcal{L} = \mathcal{L}_{CE}(f(x; \gamma, W), y) + \lambda \sum g(\gamma) \qquad (1)$$

where x and y denote the input and output of model f, respectively. W is the weight of the neural networks.

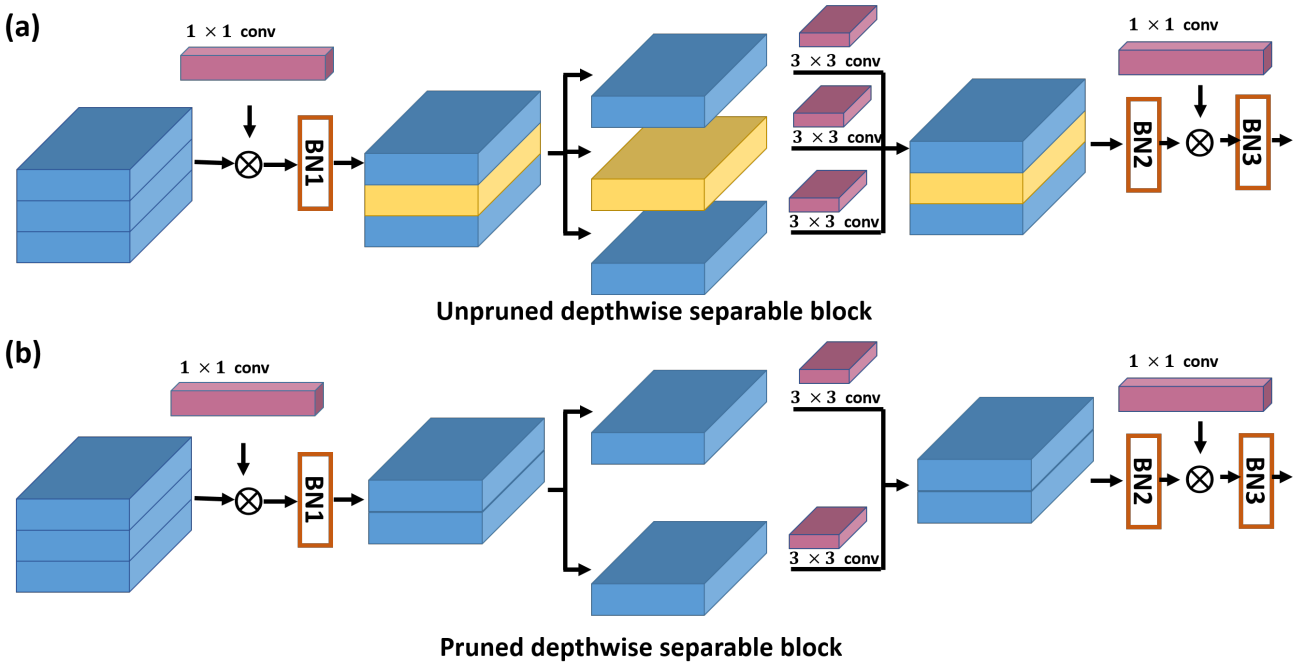

Figure 2. Illustration of depthwise separable block before (a) and after (b) pruning. The yellow (●) channel in (a) is the channel to be pruned.

$f(x;\gamma,W)$ indicates the prediction of model.

The first loss term measures the cross-entropy loss of the prediction, while the second one $g(\gamma)$ is the sparsity regularization term for scaling factors. In practice, we follow [23] adopting $g(\gamma)=|\gamma|$, which is also known as L1-norm. λ is a weight term employed to balance the two terms. We adopt $\lambda = 0.0001$ in our experiments.

Leverage the scaling factor in BN layers. Batch Normalization (BN) [16] is widely adopted in modern convolutional neural networks. It can accelerate convergence speed of networks and improve the generalizability. BN would first normalize the each input feature map as

$$\widehat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \tag{2}$$

where x_i denotes the ith tensor in the input batch $\mathcal{B} = \{x_{1...m}\}$. $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are the mean and variance of the features in the batch, respectively. The calculation of them can be formalized as

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i; \sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{3}$$

Afterwards, the normalized tensors would be scaled and shifted as

$$y_i = \gamma\widehat{x}_i + \beta \tag{4}$$

where γ and β are learnable scaling factor and bias, respectively.

As the BN layer is often employed after each convolutional layer, we follow [23] directly adopting scaling factor in batch normalization layer as scaling factor for pruning. This is theoretically more meaningful than employing an additional scaling factor that is directly multiplied on filters because the filter itself is a linear transform. It can manipulate the scale of itself to make the scaling factor smaller by enlarging its own magnitude. Thus, the scaling factor would fail to reflect the importance of each channel. In contrast, after normalization, the features' scale is fixed, the scale of the normalization layer can better reflect the importance of filters.

Extend to depthwise separable convolutional layers. Depth-wise separable convolution is a special convolutional structure proposed in MobileNet [12]. It decomposes one convolutional layer into a depth-wise and a point-wise convolutional layer. Single-layer filters are adopted for each channel of depth-wise convolution. While point-wise convolution, a 1×1 convolutional layer, would then combine the outputs of depthwise convolutional layers. In contrast, a traditional convolutional layer would do both filtering and combining in a single layer. This factorization can significantly save and reduce the model size. MobileNetV2 [30] further proposes to leverage another point-wise convolutional layer which serves as a bottleneck. A block of MobileNetV2 is illustrated in Figure 2(a) (Unpruned depthwise separable block).

While the channel pruning method is devised for the con-

volutional layer. Directly adopting it for depthwise convolution 2 of MobileNetV2 would lead to inconsistency in the number of channels. For instance, as shown in Figure 2(a), if we prune using the scaling factor in both BN1 and BN2, the resultant number of pruning channels would likely be different. However, this cannot fit the depth-wise convolutional layer requirement that the channel number of input and output tensor has to be the same. Fortunately, we find the scaling factor of BN2 can be used for pruning the convolutional layer before both BN1 and BN2. Because the depthwise convolutional layer processes each filter independently. As demonstrated in Figure 2, if a layer is removed in the output (yellow channel in Figure 2(a)), this channel can be removed in both input and output (yellow channel in Figure 2(b)). Therefore, we do not consider the scaling factor of BN1 when applying to each depthwise separable block.

3.4. Dynamic frame skipping

During video conferences, people would not perform large motion most of the time, which means the difference between neighboring frames would be too small to change the body's contour. We thus propose to save the computation resource by directly adopting the previous prediction in such a scenario.

To this end, we first need a metric to measure the frame difference, which we adopt the L1 norm between the gray version of the current frame and that of the frame predicted last time. Secondly, we need a threshold to determine which difference is large enough for a prediction. One straightforward solution is to adopt a fixed threshold. However, we find a problem for such a solution: when the scale of the body is different in frames, the threshold corresponding to "large motion" would largely differ, which means it is hard to find a universal threshold. To address this problem, we propose to employ a self-adaptive threshold picking method. The main idea of the method is to pick a relatively large threshold. Specifically, we would build a list that saves the value of the previous frame differences. When a new frame comes, we will check if the frame is larger than $\alpha\%$ values in the list. If true, we would predict the new frame. Otherwise, we adopt the last prediction. This method could avoid the threshold difference among videos while ensuring $\alpha\%$ frames are skipped on average.

In practice, α is set as 80%, which would bring a speed-up of five times as the skipping calculation has little computational overhead. We also set a maximal length of 3000 to the list for fast adapting to a new pose or person. When the list reaches the maximal length, the early value would be discarded.

4. Experiment

4.1. Settings

Dataset We choose dataset – Supervisely Person [1] for training our network. Supervisely Person contains 5711 images with 6884 high-quality annotated people. 5110 images are included in the training split, while the rest is employed as the testing dataset.

Data processing During training, the input image would first be resized to make the longest size equal to the *base size* to unify the resolution of input images. Then, it would be resized using a random ratio among [0.5, 0.75, 1.0, 1.25, 1.5]. Such augmentation can make the model most robust against scale difference. We then crop a patch in (H, W) for training. Other augmentations like flipping and color jittering are also employed. During testing, the input image would only be resized to the *base size*, and other augmentations would not be applied. In the following parts, unless specifically specified, *base size* is set as 160 and (H, W) is (160, 128).

Pruning ratio We do not employ structure pruning for Section 4.2 and 4.3. After the pruning ratio is ablationed in 4.4, we by default employ structure pruning with best searched pruning ratio of 30%.

Training strategy The backbone is initialized with ImageNet pre-trained weight, while the other parts are initialized with random weights. The standard cross-entropy loss is employed for training networks by default. An additional sparsity regularization term is used while searching pruning structure. We adopt SGD with a momentum of 0.9 as the optimizer and train for 50 epochs. The starting learning rate is set as 1e-4 and would decay following the "poly" learning rate [4].

Metric We adopt the standard metric called mean Intersection over Union (mIoU) for measuring the precision of the segmentation prediction.

Computation infrastructure Our codes are based on Pytorch [29], and all models are trained with a single GeForce GTX 1080 Ti.

Speed measuring We measure the speed by running on CPU (Model name: Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz) while restricting the resource to **10% usage**. The CPU usage is limited using the "cpulimit" library on Ubuntu.

4.2. Segmentation method ablation

We start by verifying the employed segmentation method's performance by comparing it with one of the state-of-the-art segmentation methods called DeeplabV2 [4] on Supervisely Person dataset using the same resolution. For a fair comparison, we also employ MobileNetV2 as its backbone.

As shown in Figure 2, DeeplabV2 is slower in terms of

Table 1. Ablation study on FPN with different pruning ratio on Supervised People dataset concerning four metrics: number of parameters, FLOPs, speed (fps) and mIoU (%). The first three metrics contain two attributes: value and ratio. While value denotes the measured value, the ratio is {value after prune}/{value before prune}. ↑ means the metric the higher the better and ↓ means the metric is the lower the better. "-" indicates Not Applicable.

Pruning ratio	Parameters		FLOPs		Speed		mIoU↑
	value↓	ratio↓	value↓	ratio↓	value↑	ratio↑	
0%	2.25M	-	3.60×10^8	-	6.24	-	88.8
30%	1.25M	0.56	3.18×10^8	0.88	6.53	1.05	87.9
50%	0.88M	0.39	2.89×10^8	0.61	8.80	1.41	83.9
70%	0.59M	0.26	1.54×10^8	0.43	9.23	1.48	76.7

Table 2. Comparison between two segmentation methods: DeeplabV2 and FPN on Supervisely People Dataset on four attributes: number of parameters, FLOPs, speed (fps), and mIoU (%). The MobileNetV2 is adopted as the backbone for both frameworks. ↑ means the metric the higher the better and ↓ means the metric is the lower the better.

Method	DeeplabV2	FPN
Parameters↓	4.87M	2.25M
FLOPs↓	4.98×10^8	3.60×10^8
Speed↑	3.15	6.24
mIoU↑	87.4	88.8

Table 3. Ablation study on FPN with inputs in different resolution on Supervised People dataset concerning three metrics: FLOPs, speed (fps), and mIoU (%). The resolution is controlled via adjusting the *base size*. The corresponding cropping size for *base size* of [320, 160, 64, 32] is [(320, 256), (160, 128), (64, 64), (32, 32)], respectively. ↑ means the metric the higher the better and ↓ means the metric is the lower the better.

base size	FLOPs↓	Speed↑	mIoU↑
320	12.51×10^8	1.37	92.2
160	3.60×10^8	6.24	88.8
64	0.80×10^8	12.27	79.6
32	0.25×10^8	11.07	72.1

both FLOPs and Speed while shown lower mIoU by 1.4%. Moreover, the number of parameters in DeeplabV2 is more than twice larger compared to that of FPN, demonstrating the efficiency and effectiveness of the employed FPN.

4.3. Resolution ablation study

Resolution is an important factor affecting the speed of segmentation. The choosing of resolution is always a trade-off between performance and speed. We ablation study four different scaling via adjusting the *base size*. As demonstrated in Table 3, while *base size* is set as 320, highest mIoU of 92.2% can be achieved. However, its speed is very slow. For instance, when compared with the *base size* of 160, *base size* of 320 is 4.55 times slower in terms of speed while having 3.48 times more computational cost in terms of FLOPs. Moreover, the mIoU only marginally drops by 3.4%. But, when the *base size* continues decreasing to 64 and 32, performance would significantly drop by more than 9% while the speed improvement is less than two times. Therefore, we adopt *base size* of 160 in practice.

4.4. Structural pruning

We further study the speed improvement coming from structural pruning. As shown in Table 1, pruning can significantly reduce the parameters. The model size would be reduced by 60% when the pruning ratio is 50%. The Speed would also consistently improve. At the pruning ratio of 30%, the pruned model only leads to 0.9% performance drop. However, a large pruning ratio (e.g. 50%, 70%) would lead to a significant mIoU drop larger than 4.9%. In practice, we choose a pruning ratio of 30% since it captures most of the parameter saving while maintaining a similar mIoU compared to the baseline.

4.5. Dynamic frame skipping

We then study the performance of dynamic frame skipping on videos. As shown in Figure 3, the pose between frames is small. Moreover, our method can correctly detect large motion. For the first two rows of Figure 3, the head hardly changes, and our sampling is very sparse. When the significant motion is performed at the last row, the sampling ratio becomes far denser.

4.6. Overall Performance

Finally, we test the overall performance of CE-PeopleSeg on 20 videos. The average speed can achieve 32.40 fps, which is real-time. Meanwhile, the quality of prediction is also good, as shown in Figure 4.

5. Conclusion

This paper presents the efficient people segmentation framework called CE-PeopleSeg. It is supported by an effi-

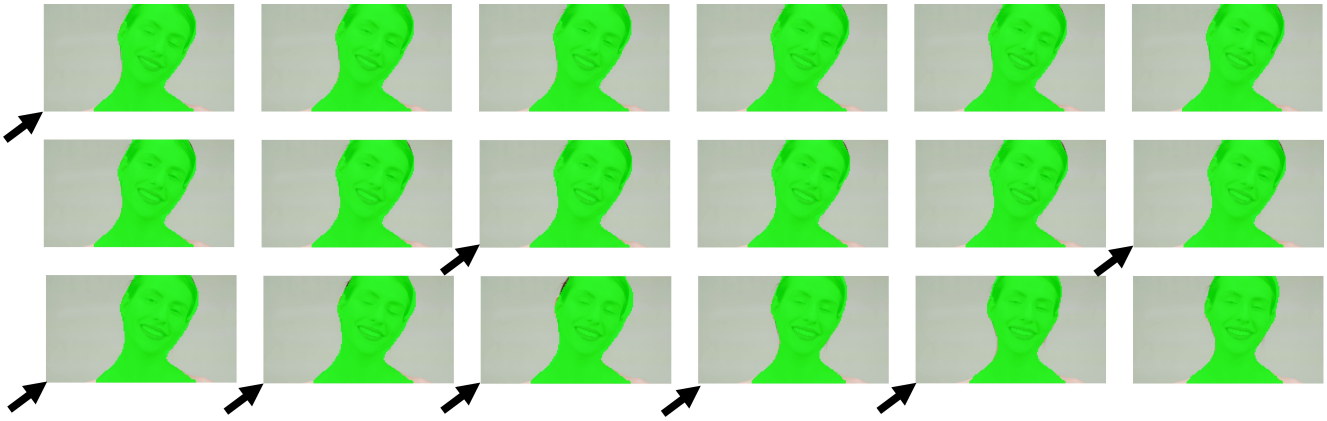

Figure 3. A sampled video clip with segmentation. For every two frames, we **only show the first one** to analyze a longer sequence. The video plays from left to right and would switch to a new line once it fills a row. We use an arrow to spotlight the frames where prediction is performed. For the rest frames, the closest previous prediction is adopted.

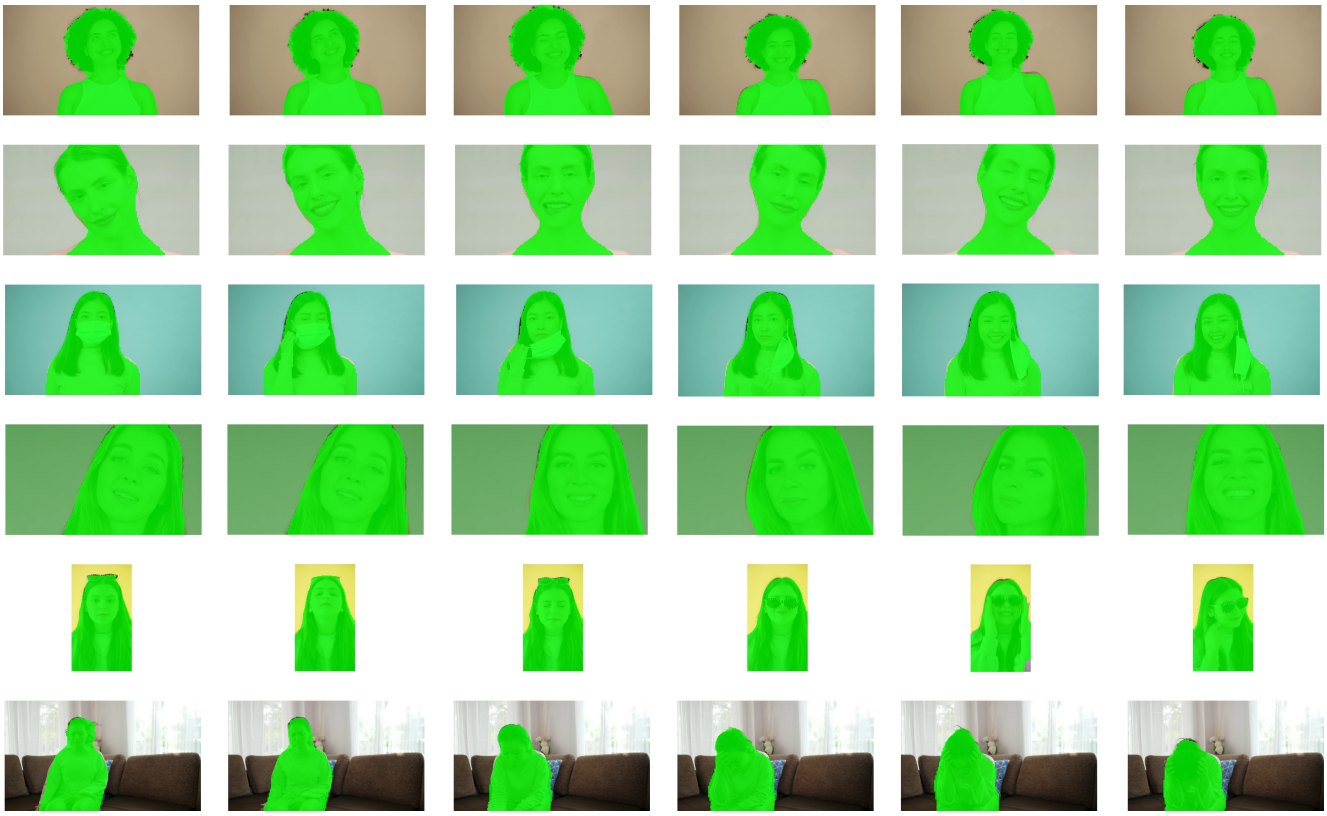

Figure 4. Prediction visualization for CE-PeopleSeg.

cient segmentation framework called FPN [22] and encoding network MobileNetV2 [30]. The structural pruning is also leveraged to compress the size of the model further. A dynamic frame skipping method is also proposed for skipping similar frames. Our extensive experiments demonstrated that the proposed CE-PeopleSeg could ensure the prediction mIoU of 87.9% while achieving a real-time inference speed on a single GPU with only 10% usage.

References

[1] Supervisely person dataset. https://supervise.ly/explore/projects/supervisely-person-dataset-23304/datasets. Accessed: 2021-03-25. 5

[2] Video conference market report. https://markets.businessinsider.com/news/stocks/video-conferencing-market-to-reach-usd-10-92-billion-by-2027-penetration-of-internet-to-influence-growth-states-fortune-business-insights-1029669542. Accessed: 2021-03-23. 1

[3] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. *arXiv preprint arXiv:1611.06321*, 2016. 2

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 1, 3, 5

[5] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. *arXiv preprint arXiv:1912.10917*, 2019. 2

[6] Wuyang Chen, Ziyu Jiang, Zhangyang Wang, Kexin Cui, and Xiaoning Qian. Collaborative global-local networks for memory-efficient segmentation of ultra-high resolution images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8924–8933, 2019. 2

[7] Antonio Fernández-Caballero, José Carlos Castillo, Juan Serrano-Cuerda, and Saturnino Maldonado-Bascón. Real-time human segmentation in infrared videos. *Expert Systems with Applications*, 38(3):2577–2584, 2011. 2

[8] Monica Gruosso, Nicola Capece, and Ugo Erra. Human segmentation in surveillance video with deep learning. *Multimedia Tools and Applications*, 80(1):1175–1199, 2021. 2

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

[10] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015. 2

[11] Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993. 2

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 4

[13] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016. 2

[14] Weizhe Hua, Yuan Zhou, Christopher De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *arXiv preprint arXiv:1805.12549*, 2018. 2

[15] Xueqin Huang, Urmish Thakker, Dibakar Gope, and Jesse Beu. Pushing the envelope of dynamic spatial gating technologies. In *Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, pages 21–26, 2020. 2

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[17] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015. 2

[18] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 1

[19] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990. 2

[20] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1, 2

[21] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. *arXiv preprint arXiv:2012.07810*, 2020. 1, 2

[22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 7

[23] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017. 2, 3, 4

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[25] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017. 2

[26] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2131–2141, 2019. 2

[27] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017. 2

[28] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 2

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 5

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1, 4, 7

[31] Atul Sharma, Swapnil Tiwari, Manas Kanti Deb, and Jean Louis Marty. Severe acute respiratory syndrome coronavirus-2 (sars-cov-2): a global pandemic and treatment strategies. *International journal of antimicrobial agents*, 56(2):106054, 2020. 1

[32] Xiaoyong Shen, Hongyun Gao, Xin Tao, Chao Zhou, and Jiaya Jia. High-quality correspondence and segmentation estimation for dual-lens smart-phone portraits. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3257–3266, 2017. 2

[33] Xiaoyong Shen, Aaron Hertzmann, Jiaya Jia, Sylvain Paris, Brian Price, Eli Shechtman, and Ian Sachs. Automatic portrait segmentation for image stylization. In *Computer Graphics Forum*, volume 35, pages 93–102. Wiley Online Library, 2016. 2

[34] Akio Shio and Jack Sklansky. Segmentation of people in motion. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 325–332. IEEE, 1991. 2

[35] Chunfeng Song, Yongzhen Huang, Zhenyu Wang, and Liang Wang. 1000fps human segmentation with deep convolutional neural networks. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 474–478. IEEE, 2015. 1, 2

[36] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015. 2

[37] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018. 2

[38] Yue Wang, Ziyu Jiang, Xiaohan Chen, Pengfei Xu, Yang Zhao, Yingyan Lin, and Zhangyang Wang. E2-train: Training state-of-the-art cnns with over 80% energy savings. *arXiv preprint arXiv:1910.13349*, 2019. 2

[39] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665*, 2016. 2

[40] Junru Wu, Yue Wang, Zhenyu Wu, Zhangyang Wang, Ashok Veeraraghavan, and Yingyan Lin. Deep k-means: Retraining and parameter sharing with harder cluster assignments for compressing deep convolutions. In *International Conference on Machine Learning*, pages 5363–5372. PMLR, 2018. 2

[41] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018. 2

[42] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019. 2

[43] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018. 2

[44] Song-Hai Zhang, Ruilong Li, Xin Dong, Paul Rosin, Zixi Cai, Xi Han, Dingcheng Yang, Haozhi Huang, and Shi-Min Hu. Pose2seg: Detection free human instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 889–898, 2019. 2

[45] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 405–420, 2018. 2

[46] Tao Zhao and Ramakant Nevatia. Bayesian human segmentation in crowded situations. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–459. IEEE, 2003. 2