Real-time communication with Virtual Background

Keyuan Zhang

Abstract

I demonstrate replace background in real-time video communication by tackling the problem of portrait matting on mobile devices. The proposed model achieves real-time inference speed while maintain high visual performance. The proposed feature is able to attain over 25 FPS on 720P on iPhone11 in real-time video communication.

1 Introduction

There is an increasing need of using virtual background in real-time video communication, the key of this technology is to apply portrait matting first. However, the current method based on portrait segmentation or matting cannot meet the requirement in industry application due to their poor inference efficiency and robustness. Therefore, I have done extensive literature review and design a compact portrait matting model with high visual performance.

Replacing background in a video is a widely used technique that enable movie director or video creators to create more flexible content. Traditional method, which is shooting video on green screen, and replace the background in later editing, takes a lot of time, and shooting environment is constrained, therefore is impossible to apply this in real-time video communication.

The key of applying virtual background in real-time video is to develop a method to separate foreground and background, which we called image segmentation or image matting. With the success of convolution neural networks (CNN) in computer vision task, there has been an increasing number of works on this area [5, 16].There are some approaches to matting specific object as foreground [3, 10, 15, 9], for example, portrait matting. Portrait matting is useful especially in real-time video communication due to its specific situation.

In this work, I propose a compact network for portrait matting, which can run on mobile devices in real-time. The proposed network is U-Net [7]like architecture with skip connection between encoder and decoder. U-Net is a well-known network used in biomedical segmentation, the skip connections restore low-level features during up-sampling, which can reduce the loss in down-sampling. Inspired by [12, 8], I replaced traditional convolution with deepthwise convolution, and applied linear bottleneck to reduce model size for real-time efficiency while maintaining the performance.

The proposed virtual background feature is included video customer service of in several Chinese banks. Due to the confidentiality agreement with the company, I cannot upload the related code or model file in public, but the overall performance is shown in the web demo

2 Related Work

2.1 Encoder-Decoder

Encoder-Decoder architecture is commonly applied in image segmentation task [7, 6, 2], the encoder is used to extract image feature in low level through sequential convolution and down-sampling. The decoder, in other hand, up-sample the feature map into original image size while keep the high level information.

2.2 Depthwise Convolution

Depthwise convolution is a powerful operation used to reduce the computation cost while maintaining similar performance. This operation has been widely adopted in several works [4, 14, 13]. In this work, traditional convolution is been replaced by depthwise convolution in order to reduce computation cost.

2.3 Linear Bottleneck

Linear bottleneck [8] combines with depthwise convolution reduce the model size while maintaining high performance. According to [8], this module is particular suitable for mobile design due to it reduces the memory footprint needed during inference. This module is used extensively in encoder module to achieve real-time high performance goal.

2.4 Skip Connection

skip connection [7] restore low-level features during up-sampling, which enables low-level features restore in feature map therefore enhance the accuracy. This connection is been used between encoder and decoder in this work to ensure the low-level information been stored during up-sampling.

3 Methods

3.1 Model Architecture

The overall model architecture follows a standard encoder-decoder structure that is widely used in semantic segmentation [7, 6, 2]. Encoder reduces the size of inputs by summarizing the low-level information into high-level semantic information which store in feature map. Decoder, in the other hand, recovers the spatial information and restores the original image size through upsampling. The model architecture is shown in Figure 1

3.1.1 Encoder

Encoder block follows a multi-branch dilated depthwise convolution with a linear bottleneck. The dilation rates are different for all branches to capture multi spatial information. The outputs of different branches are concatenated to a tensor, which contains multi-level information. After concatenation, linear bottleneck is imposed

where the output of intermediate layers is thinner than the input. The linear bottleneck is a decomposition of traditional convolution, which connect two encoder blocks into cheaper one by reducing channels. The encoder block is shown in Figure 2

3.1.2 Decoder

The decoder block performs upsampling to restore the original resolution. In order to restore the low-level information from the encoder, skip connection is employed here

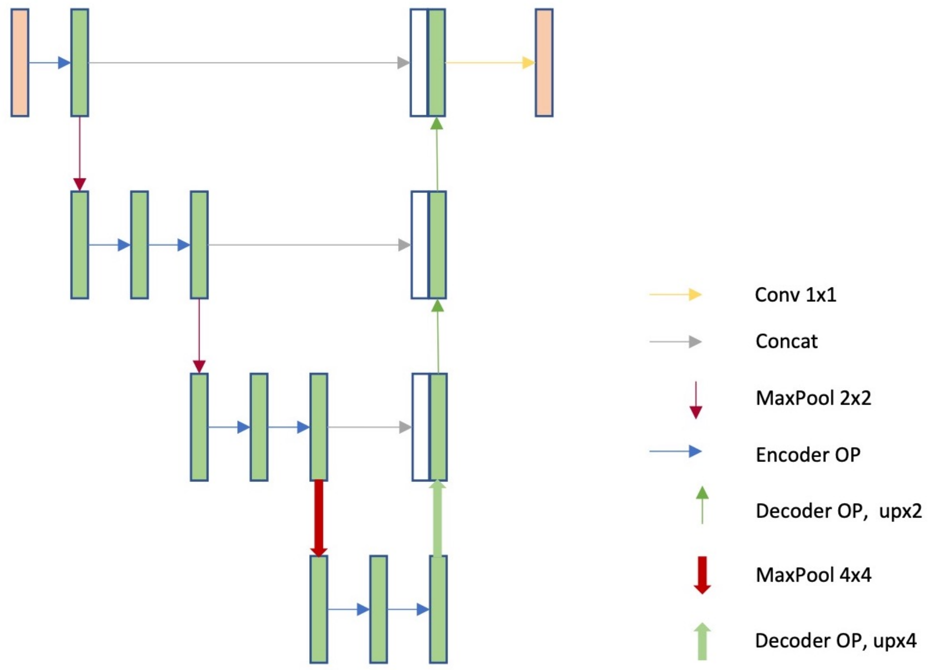
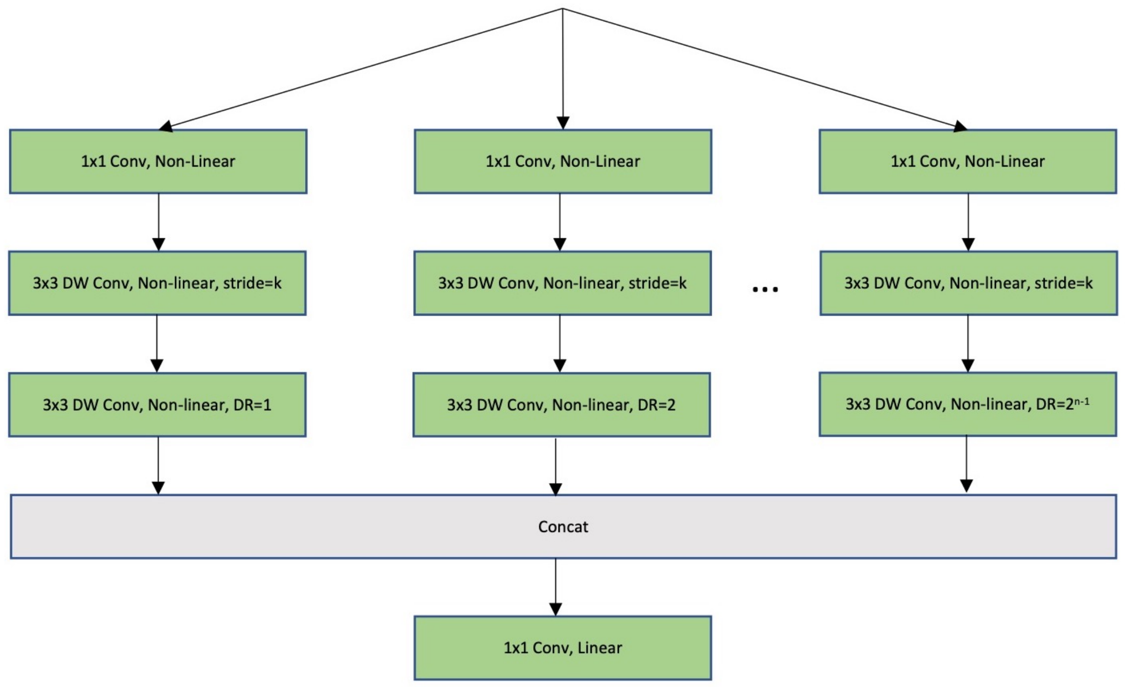


Figure 1: Model Architecture



1x1 Conv, Non-Linear
1x1 Conv, Non-Linear
1x1 Conv, Non-Linear
3x3 DW Conv, Non-linear, stride=k
3x3 DW Conv, Non-linear, stride=k
...
3x3 DW Conv, Non-linear, stride=k
3x3 DW Conv, Non-linear, DR=1
3x3 DW Conv, Non-linear, DR=2
3x3 DW Conv, Non-linear, DR=2^{n-1}
Concat
1x1 Conv, Linear

Figure 2: Encoder Block

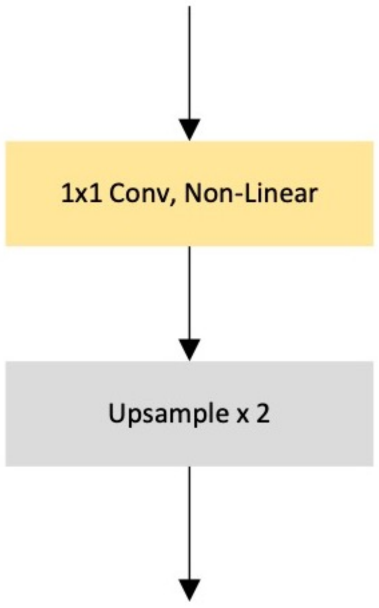


Figure 3: Decoder Block

to connect the encoder and decoder. The decoder block is illustrated in Figure 3

3.2 Loss Function

The loss function is the combination of three different loss functions, which monitor the mean absolute difference (MAD) between ground truth mask and the predicted mask, guide the model to diverge faster(not sure yet) and capture the fine-grained details in the edges.

The pixel-wise loss is frequently used in matting tasks. The pixel-wise loss L_1 measure the mean difference between ground truth mask and predicted one.

$$L_1 = \frac{1}{N}\sum_{i=1}^{N} |p_i - g_i^{gt}|$$

N denotes the total number of pixels, g is the ground truth, and p is vectorized output where each pixel value is indexed by subscript i.

The second loss is softmax cross entropy[find the ref] (don't know yet)

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N}(glogp + (1-g)log(1-p))$$

The last loss is the gradient loss [9]. Different from segmentation, matting requires finer edges, which needs additional loss function to monitor edge training. The gradient loss is shown as follows:

$$\begin{aligned} L_3 &= \frac{1}{N}\sum_{i=1}^{N} |G(A)_i - G(A^{gt})_i| \\ &= \frac{1}{2N}\sum_{i=1}^{N} \big(|(S * A - S * A^{gt})_i| + |(S^T * A - S^T * A^{gt})_i|\big) \end{aligned}$$

Where S is Sober-like filter S:[find some ref]

$$S = \begin{bmatrix} -\frac{1}{8} & 0 & \frac{1}{8} \\ -\frac{2}{8} & 0 & \frac{2}{8} \\ -\frac{1}{8} & 0 & \frac{1}{8} \end{bmatrix}$$

$G(A) = [S * A, S^T * A]$, where * is a convolution. $G(A)$ represents a two-channel output that contains gradient information along x-axis and y-axis of predicted mask, and $G(A^{gt})$ denotes the same information of ground truth mask.

The whole loss function used in this work is shown in below:

$$L = L_1 + L_2 + L_3$$

4 Evaluation

The performance metrics proposed in this work are mean absolute difference (MAD) and gradient error connectivity (Gradient) [17]. I used MAD to measure the pixel-wise accuracy of prediction, and Gradient to evaluate the edge fining performance. The MAD and Gradient error are defined as:

$$\frac{1}{N}\sum_{i} |p_i - g_i|$$

$$\frac{1}{N}\sum_{i} \|\nabla p_i - \nabla g_i\|$$

The training dataset comes from two different datasets, the first one is provided by Shen ty al.[11], which consists 2000 images of 600×800 resolution, and the second one obtains from [1], which consists 34427 images of the same resolution.

Method	Gradient	MAD
Proposed Model	1.99	4.78
MDv3	2.06	5.49

Table 1: Evaluation result of proposed model and Mobile DeepLabv3

I split the dataset into training and testing with the ratio of 17 : 3. To make the model more robust, I augment images by scaling, rotation and left-right flip. First, and image is descaled into input size 256×256 with random scaling factor selected from 1 to 1.15. Then the rotation by $[-30^{\circ}, 30^{\circ}]$ is applied with probability of 0.5, additional left-right flip is applied with the same probability. Finally, cropping is applied to ensure the size of image match the input size of model.

The training process consist two steps, I first trained the larger dataset around 500 thousands iterations to approach convergence, then fine tuning with the first dataset due to its high annotation quality. I trained model with a batch size of 32 and a fixed learning rate of 10^{-4}.

evaluation figure(todo)

5 Application

The final application platform is iOS for commercial purpose, with brief review of deep learning inference architecture, I found three mainstream mobile inference architectures are promising: TensorFlow Lite, CoreML, and Caffe2.

Comparing with other two architectures, CoreML is friendly for iOS device, it seamlessly takes advantage of CPU and GPU. For A11 and newer bionic chip, CoreML can take benefit from bionic silicon to speed up inference efficiency. Specifically, I implemented CoreML as the mobile inference architecture and quantize the model by CoreML to improve inference efficiency further.

During the demo test, there are some losses in visual performance. The first one is loss in rotation, because the training dataset did not contain images with camera rotation, when user rotate camera, the visual loss is heavy. This can be solved by random rotate image $[-90^{\circ}, 90^{\circ}]$ during training.

The second one is caused by image ratio of width and height. During the demo test, there are some losses in visual performance, because the ratio of input blob is 1 : 1, and main ratio of resolution in video communication is 16 : 9. To overcome

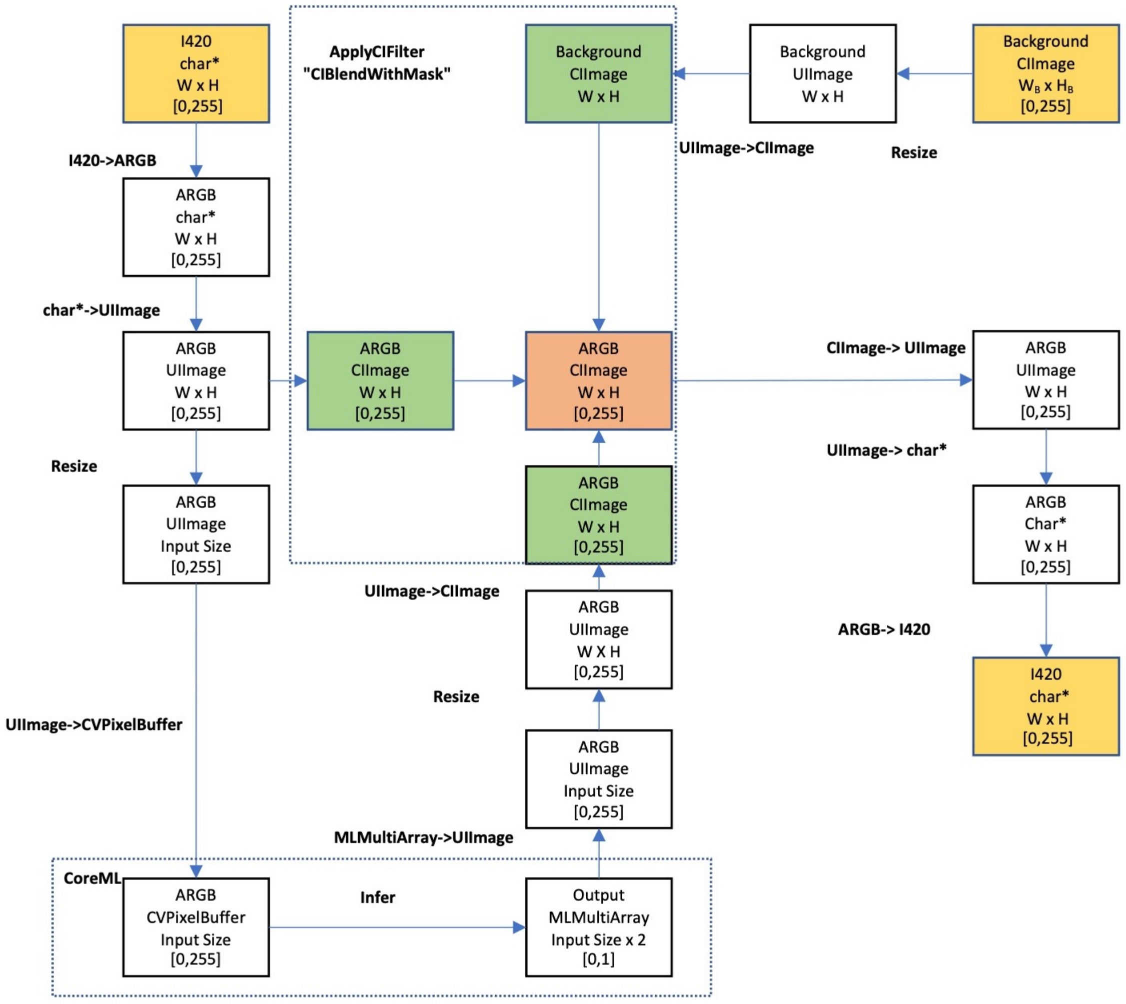


Figure 4: Application Flow Chart

Mobile Device	360P	720P	1080P
iPhone7	19FPS	15FPS	8FPS
iPhoneX	20FPS	16FPS	10FPS
iPhone11	29FPS	25FPS	19FPS

Table 2: Running Speed in Real-time video communication

this, I modified the ratio of input blob into 16 : 9. For example, with the input image size of 640×360, the input has to be resize to 256×144 to feed into network.

The speed performance, which include resizing, inference, replacing background and codec is shown in Table 5

The web demo is available at [demo link]

6 Conclusion

In this work, I have proposed an efficient model to perform real-time virtual background in video communication on mobile device. The proposed model can achieve 30 FPS on iPhone 11. I have also showed the pipeline of implementing model to real-time application, which includes literature review of related works, how to take advantage of open source dataset, design a compact model architecture, and necessary modification on model deployment.

References

[1] Aisgement. https://github.com/aisegmentcn/matting_human_datasets. Accessed: 2019-09-16. 4

[2] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 2.1, 3.1

[3] Q. Chen, T. Ge, Y. Xu, Z. Zhang, X. Yang, and K. Gai. Semantic human matting. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 618–626, 2018. 1

[4] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 2.2

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 1

[6] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2.1, 3.1

[7] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1, 2.1, 2.4, 3.1

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1, 2.3

[9] S. Seo, S. Choi, M. Kersner, B. Shin, H. Yoon, H. Byun, and S. Ha. Towards real-time automatic portrait matting on mobile devices. *arXiv preprint arXiv:1904.03816*, 2019. 1, 3.2

[10] X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia. Deep automatic portrait matting. In *European conference on computer vision*, pages 92–107. Springer, 2016. 1

[11] X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia. Deep automatic portrait matting. In *European conference on computer vision*, pages 92–107. Springer, 2016. 4

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceed-*

ings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015. 2.2

[14] M. Wang, B. Liu, and H. Foroosh. Design of efficient convolutional layers using single intra-channel convolution, topological subdivisioning and spatial" bottleneck" structure. *arXiv preprint arXiv:1608.04337*, 2016. 2.2

[15] N. Xu, B. Price, S. Cohen, and T. Huang. Deep image matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2970–2979, 2017. 1

[16] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 1

[17] B. Zhu, Y. Chen, J. Wang, S. Liu, B. Zhang, and M. Tang. Fast deep matting for portrait animation on mobile phone. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 297–305, 2017. 4