Efficient Heterogeneous Video Segmentation at the Edge

Jamie Menjay Lin Siargey Pisarchyk Juhyun Lee David Tian Tingbo Hou
Karthik Raveendran Raman Sarokin George Sung Trent Tolley Matthias Grundmann
Google, Mountain View, CA, USA
{jmlin,siargey,impjdi,dctian,krav,tingbo,sorokin,gsung,trenttolley,grundman}@google.com

Abstract

We introduce an efficient video segmentation system for resource-limited edge devices leveraging heterogeneous compute. Specifically, we design network models by searching across multiple dimensions of specifications for the neural architectures and operations on top of already light-weight backbones, targeting commercially available edge inference engines. We further analyze and optimize the heterogeneous data flows in our systems across the CPU, the GPU and the NPU. Our approach has empirically factored well into our real-time AR system, enabling remarkably higher accuracy with quadrupled effective resolutions, yet at much shorter end-to-end latency, much higher frame rate, and even lower power consumption on edge platforms.

1. Introduction

Video segmentation is a foundational technology powering various computer vision tasks in business and entertainment use cases, such as video editing in augmented reality (AR) and background blur and replacement in video conferencing. Machine learning (ML) inference for high-quality real-time video segmentation has been a challenging problem particularly on resource-limited edge devices, *e.g.* smartphones, with model accuracy, real-time latency, and power consumption being the key areas for improvement.

We present practical techniques for real-time high-quality video segmentation at the edge (Figure 1). We describe i) our advances made to common light-weight network architectures, *e.g.* MobileNetV3 [19] and EfficientNetLite [9], and ii) our optimizations to the inference pipelines at the edge, such as WebGL-based [4] browsers and the NPU inference pipelines on mobile devices.

2. Related Work

In the course of neural network-based model development for semantic segmentation, fully convolutional networks [31] were introduced as an earlier progress. Numerous follow-up works continued to improve on accuracy and efficiency, proposing various types of model architectures for segmentation, such as DeepLab [16], Vortex Pooling [18], PSPNet [37], and HRNet [32, 36]. Another recent line of work targets the problem of image matting for foreground-background separation, which typically requires either the availability of a trimap [14, 24, 33] or a pre-captured background image [14, 24, 30]. Dynamic routing [23] was introduced on a generalized view over various architectures for segmentation. Neural architecture search techniques have been another line of work in searching for optimized segmentation model architectures [15, 25, 28], as well as particular types of model improvements such as transformer-based segmentation models [17, 27].

Figure 1. Real-time video segmentation comparison on Google Pixel 6 smartphone. Left column: Baseline ML inference at 89.5% mIoU [2] (segmentation quality metric) and 2.03 Watts. Right column: Our ML inference at 95.1% mIoU and 1.84 Watts.

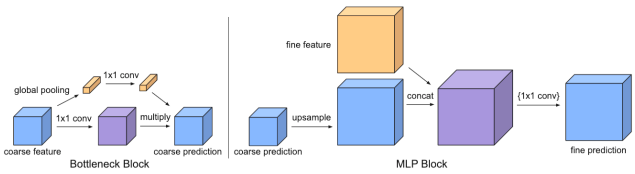

Figure 2. Left: The bottleneck block paralleled with a squeeze-and-excitation operation, which increases GPU latency due in part to the global pooling operations. Right: Our decoder with the MLP operation, which does not involve global pooling operations.

3. Edge Segmentation Model Design

Neural networks running on edge devices are often tailored to the available computational resources. As a baseline for the segmentation task, we consider a commonly used fully convolutional U-net architecture based on a MobileNetV3 backbone. Running CPU model inference with an input resolution of 256×256 takes over 50ms on an Intel Celeron N3060 as used in common EDU-targeted Chromebooks, suggesting that even a low-quality segmentation model is too expensive to run at real-time on older hardware.

To allow for larger models while still supporting real-time video, we targeted GPU and NPU that can deliver the required FLOPS [1] or TOPS [3] for the task of high resolution video segmentation. We analyze the impact of our design choices including the structure of the encoder and decoder blocks, convolution types, input resolutions, and width and depth multipliers. Table 1 summarizes the full ablation study discussed in this section.

While the squeeze-and-excitation blocks [20] in MobileNetV3 are efficient on the CPU, they increase the latency on GPUs due to the global average pooling of large tensors. We re-base our architecture on EfficientNetLite where each encoder block comes with an inverted residual bottleneck and produces an output tensor that is $1/32$ of the input. The decoder then begins with a bottleneck block that predicts a coarse segmentation mask as shown in Fig. 2. In this instance, the global average pooling only needs to operate on a much smaller tensor and the prediction accuracy benefits from such channel attention mechanism.

Next, we explore three variants for the decoder design:

1. **Bilinear Upsampling**: Upsamples bilinearly the lower resolution tensor to match the resolution of the next higher encoder block. This applies a non-linearity and sum with the tensor from the encoder skip connection.

2. **Channel Attention**: Applies squeeze-and-excitation blocks after upsampling, followed by a set of 1×1 convolutions and a 3×3 depth-wise convolution.

3. **Multilayer Perceptron (MLP)**: Applies upsampling followed by a sequence of 1×1 convolutions (MLPs).

Ablation Study	Parameter	Model Inf Time (ms)	Model Size (MB)	OPs (10^9)	Segment IOU (%)	Pose IOU (%)
Convolution Type	**Depthwise Conv**	**4.7**	**1.7**	**1.5**	97.19	89.60
	Group Conv*	**4.7**	7.2	4.8	**97.41**	**90.20**
Width Multiplier	**1.0***	**5.7**	**9.4**	**5.3**	97.50	91.40
	1.5	8.2	21.0	12.1	97.89	92.37
	2.0	11.4	37.0	20.6	**97.96**	**92.63**
Decoder Type	**Bilinear Upsampling**	**6.5**	**8.7**	**7.4**	97.65	90.65
	Channel Attention	8.2	**8.7**	7.5	97.75	**91.81**
	Multilayer Perceptron*	6.9	**8.7**	7.6	**97.81**	91.80
Image Resolution	**256x256**	**1.6**	**8.7**	**1.9**	96.94	87.23
	384x384	3.1	**8.7**	4.3	97.51	90.10
	512x512*	6.9	**8.7**	7.6	97.79	91.80
	640x640	9.5	**8.7**	12.0	**97.98**	**92.94**

Table 1. Ablation study over edge segmentation model design parameters. "$*$" denotes our final choice in each ablation category.

We found that 1×1 convolutions make a significant difference to the accuracy of the model when it comes to finer details, and that the MLP block serves as a reasonable performance compromise compared to the more expensive attention block. We also considered the effect of various convolution types on the network, ranging from standard convolutions to other types of convolutions. We found while separable convolutions are a natural fit for modern GPUs, certain hardware such as NPUs can benefit from group convolutions due to increased chip utilization, enhancing the model expressivity without increasing the latency. Finally, we performed a hyperparameter search for the EfficientNetLite family and settled on a multiplier of 1.0.

Having settled on an architecture, we measured the trade-offs between input resolution, latency, and accuracy. In each of these cases, we trained the network using images down-scaled to the specified input resolution and measured the IoU by down-scaling the ground truth mask to the output resolution. We found that resolution has the largest impact on the quality of segmentation, but observed rapidly diminishing returns to increasing the resolution beyond 640×640.

4. E2E Heterogeneous Pipeline Optimization

One of the most important performance criteria of a video segmentation system for AR applications is the E2E frame rate, which is directly perceivable by the application users. In this section, we discuss our particular considerations for ML inference pipeline optimization on web browsers and mobile devices leveraging the GPU and the NPU, respectively.

GPU for the Web: In browsers, ML frameworks with WebGL support [6, 10, 11] were our natural choices for hardware-aided ML inference in the browser, as NPUs currently lack a web standard. Since these implementations ran significantly slower than native OpenGL [22], we wrote our own WebGL inference engine that can run at near-native OpenGL performance. In our WebGL implementation, performance-critical operations, *e.g.* convolutions, leverage a modern GPU feature called Multiple Render Tar-

	System Configurations				Pipeline Performance		ML Model Accuracy							
Model for Mobile	**Image Resolution**	**ML Engine on Pixel 6**	**Hardware Buffer**[1]	**Sync Fence**[2]	**E2E ↓ Latency**[3] (ms)	**Power ↓** (W)	**Video Meeting ↑**		**Video Blogging ↑**		**Upper-Body Poses ↑**		**Full-Body Poses ↑**	
							J Mean*	F Mean*	J Mean*	F Mean*	J Mean*	F Mean*	J Mean*	F Mean*
#M0 (Baseline)	256x256	GPU	built-in	built-in	19.1	2.03	95.7%	93.5%	92.8%	83.0%	81.2%	87.4%	88.2%	82.1%
#M1[4]	512x512	GPU	built-in	built-in	42.5	2.48	97.0%	95.7%	95.0%	88.8%	88.0%	93.2%	91.5%	87.2%
#M2		NPU			35.0	2.31	**97.9%**	**97.6%**	**96.7%**	**93.8%**	**91.9%**	**96.4%**	**94.0%**	**92.1%**
#M3			✓		26.2	1.89								
#M4 (Final)[5]			✓	✓	**23.4**	**1.84**								

Table 2. Performance of *mobile* video segmentation systems between the baseline mobile GPU and our improved ML model plus optimized NPU pipeline with TensorFlow Lite [13] on Pixel 6 smartphone. *J mean [29] and F mean [29] metrics are for region similarity and contour accuracy, respectively. [1][2]More details are in Section 4. [3]The end-to-end latency includes image pre- and post-processing. [4]#M0 and #M1 use the same baseline ML model at different image resolutions. [5]The final model parameters are as indicated in Table 1.

gets (MRT) [35], which allows rendering multiple textures at once, substantially reducing the overhead of multiple draw calls. For efficient MRT, we separate logical tensors and physical GPU objects, which have a 1-to-1 correspondence in other frameworks, and allow tensors to take flexible layouts instead of one hard-coded layout.

NPU for Mobile Devices: Advanced mobile-optimized video processing frameworks, such as MediaPipe [26], offer streamlined pipelines primarily targeting image and video processing along with downstream co-processor interactions and GPU execution, typically for the tasks of image acquisition, pre-/post-processing, and rendering. For NPU-accelerated ML inference use cases, the system can benefit from a processing chain that is tightly coupled with the NPU. A naive replacement of GPU inference with its NPU counterpart can introduce major inefficiencies such as indirect inter-processor data flow and latency from inter-processor synchronization. We streamline the former with Native Hardware Buffers [5] supporting shared access by both the GPU and the NPU, and reduce the latter with sync fences in the Android Synchronization API [12].

5. Results

Our segmentation models are trained with a standard Jaccard [21, 34] loss on annotated datasets totaling 250k images that cover a wide range of video scenarios, from video conferencing to human activities. We evaluate our models on proprietary datasets, *e.g.* "video meeting" (800 samples) and "full-body poses" (947 samples), and report their mean Intersection over Union (mIoU) [2], J Mean [29] for region similarity, and F Mean [29] for contour accuracy.

GPU for the Web: For video segmentation in web apps designed for desktop and laptop web browsers, we run our EfficientNetLite- and MobileNetV3-based segmentation models with the baseline CPU (XNNPACK [7] with WebAssembly SIMD [8]) and with the GPU (our WebGL implementation). Table 3 presents the direction for ML inference in the web browser: (a) to employ the GPU for Web ML when possible, and (b) to prefer EfficientNetLite over MobileNetV3 for GPU-accelerated video segmentation on the web. In fact, the latter observation led to us confidently pursuing EfficientNetLite-based models for the NPU below.

Model for Web	**Model Base**	**Size** (MB)	**Resolution**	**CPU** (ms)	**GPU** (ms)
#W0 (Baseline)	MobileNetV3	1.62	256×256	25.6	3.3
#W1	**EfficientNetLite**	1.65		19.6	**2.6**
#W2	MobileNetV3	1.62	512×512	102.6	4.9
#W3 (Final)	**EfficientNetLite**	1.65		79.5	**3.9**

Table 3. Performance of *web* video segmentation models on the baseline CPU (XNNPACK WebAssembly SIMD) and the GPU (our WebGL implementation) on MacBook Pro 2019.

NPU for Mobile Devices: With our model improvement and pipeline optimization as discussed in Sections 3 and 4, we have achieved significantly higher accuracy and enabled much shorter inference latency, while reducing power consumption compared to the baseline. For end-to-end performance evaluation, we run video segmentation tasks with 512x512 input image resolution and 7.6B FLOPS in various architectures on a Google Pixel 6 smartphone. As shown in Tab. 2, the Model System #M4 (Final) with configurations of the NPU, shared data buffer, and asynchronous GPU-NPU execution along with other choices indicated in the ablation study of Sec. 3, outperforms *all* other variants in *both* latency and power consumption. Remarkably, our final model #M4 achieves higher accuracy with a speed up by 81% (at 42.7 inf/sec) while consuming only 74% power of #M1 with the baseline ML model as shown in Tab. 2.

6. Conclusion

In this paper, we showcase a development for edge-based video segmentation with our improvements and optimizations of ML model, inference acceleration and E2E heterogeneous compute pipeline. We visit our deliberations that cover the scope of model architecture design and system pipeline analysis, targeting specifically resource-limited edge ML. We present our design and optimization methodology in the ML model architecture and in the end-to-end ML pipeline with efficient use of resources and APIs to maximize the throughput for production-ready, heterogeneous edge ML solutions. We demonstrate that our methodology enables higher accuracy across various datasets, lower end-to-end inference latency, and lower power consumption. We expect our methodology to be beneficial and extensible to a wider range of edge CV/ML systems and real-time AR/VR applications.

References

[1] FLOPS. https://en.wikipedia.org/wiki/FLOPS. 2

[2] Intersection over Union. https://en.wikipedia.org/wiki/Jaccard_index. 1, 3

[3] TOPS. https://semiengineering.com/tops-memory-throughput-and-inference-efficiency/. 2

[4] WebGL. https://khronos.org/webgl, 2011. 1

[5] Native Hardware Buffer. https://developer.android.com/ndk/reference/group/a-hardware-buffer, 2017. 3

[6] TensorFlow JS. https://tensorflow.org/js, 2018. 2

[7] XNNPACK. https://github.com/google/XNNPACK, 2018. 3

[8] WebAssembly Core Specification. https://www.w3.org/TR/wasm-core-1/, 2019. 3

[9] EfficientNet-lite. https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet/lite, 2020. 1

[10] Paddle.js. https://github.com/PaddlePaddle/Paddle.js, 2020. 2

[11] ONNX Runtime Web. https://github.com/microsoft/onnxruntime, 2021. 2

[12] Android Synchronization Framework. https://source.android.com/devices/graphics/sync, 2022. 3

[13] TensorFlow Lite. https://www.tensorflow.org/lite, 2022. 3

[14] Yagiz Aksoy, Tunç Ozan Aydin, and Marc Pollefeys. Designing Effective Inter-Pixel Information Flow for Natural Image Matting. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 228–236, 2017. 1

[15] Liang-Chieh Chen, Maxwell D. Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jonathon Shlens. Searching for Efficient Multi-Scale Architectures for Dense Image Prediction. In *NeurIPS*, pages 8713–8724, 2018. 1

[16] 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, 2018. 1

[17] Mingyu Ding, Xiaochen Lian, Linjie Yang, Peng Wang, Xiaojie Jin, Zhiwu Lu, and Ping Luo. HR-NAS: Searching Efficient High-Resolution Neural Architectures with Lightweight Transformers. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2981–2991, 2021. 1

[18] Jiansheng Dong, Jingling Yuan, Lin Li, Xian Zhong, and Weiru Liu. An efficient semantic segmentation method using pyramid shufflenet V2 with vortex pooling. In *ICTAI*, pages 1214–1220. IEEE, 2019. 1

[19] Andrew Howard, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, Yukun Zhu, Ruoming Pang, Hartwig Adam, and Quoc Le. Searching for MobileNetV3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019. 1

[20] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018. 2

[21] Paul Jaccard. The Distribution of the Flora in the Alpine Zone. *New Phytologist*, 11(2):37–50, 1912. 3

[22] Juhyun Lee, Nikolay Chirkov, Ekaterina Ignasheva, Yury Pisarchyk, Mogan Shieh, Fabio Riccardi, Raman Sarokin, Andrei Kulik, and Matthias Grundmann. On-Device Neural Net Inference with Mobile GPUs. In *CVPR Workshop ECV*, 2019. 2

[23] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning Dynamic Routing for Semantic Segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8550–8559, 2020. 1

[24] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman. Real-Time High-Resolution Background Matting. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8758–8767, 2021. 1

[25] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 82–92, 2019. 1

[26] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. MediaPipe: A Framework for Building Perception Pipelines. In *CVPR Workshop on Computer Vision for AR/VR*, 2019. 3

[27] Eslam Mohamed and Ahmad El Sallab. Spatio-Temporal Multi-Task Learning Transformer for Joint Moving Object Detection and Segmentation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1470–1475, 2021. 1

[28] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast Neural Architecture Search of Compact Semantic Segmentation Models via Auxiliary Cells. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9118–9127, 2019. 1

[29] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[30] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. Background Matting: The World Is Your Green Screen. In

2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2288–2297, 2020. 1

[31] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017. 1

[32] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5696, 2019. 1

[33] Yanan Sun, Guanzhi Wang, Qiao Gu, Chi-Keung Tang, and Yu-Wing Tai. Deep Video Matting via Spatio-Temporal Alignment and Aggregation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6971–6980, 2021. 1

[34] Taffee T Tanimoto. Elementary mathematical theory of classification and prediction. 1958. 3

[35] Wikipedia contributors. Multiple render targets — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Multiple_Render_Targets&oldid=574813172, 2013. [Online; accessed 6-April-2022]. 3

[36] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-HRNet: A Lightweight High-Resolution Network. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10435–10445, 2021. 1

[37] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, 2017. 1