Appendix: Efficient Semantic Segmentation by Altering Resolutions for Compressed Videos

Yubin Hu[1] Yuze He[1] Yanghao Li[1] Jisheng Li[1] Yuxing Han[2] Jiangtao Wen[3] Yong-Jin Liu[1]*
[1]Department of Computer Science and Technology, Tsinghua University
[2]Shenzhen International Graduate School, Tsinghua University
[3]Eastern Institute for Advanced Study
{huyb20, hyz22, liyangha18}@mails.tsinghua.edu.cn, jas0n1ee@icloud.com,
yuxinghan@sz.tsinghua.edu.cn, jtwen@eias.ac.cn, liuyongjin@tsinghua.edu.cn

A1. Local Attention in CReFF

To show how the features are aggregated in CReFF, we visualize the segmentation results generated from different semantic features, including the up-sampled LR features of P frame ($\bar{F}_P$), the warped HR features from I frame ($\hat{F}_I$), and the aggregated HR features of P frame ($\tilde{F}_P$). As shown in Figure A1, the segmentation results generated from $\tilde{F}_P$ are much better than those from $\bar{F}_P$ and $\hat{F}_I$, which demonstrates the effectiveness of feature aggregation in CReFF. Benefiting from the local attention mechanism, CReFF selectively fuses $\hat{F}_I$ into $\bar{F}_P$ and produces better semantic features. According to Figure A1, we explain the function of local attention from three aspects.

- **It corrects the wrong features in $\bar{F}_P$.** In Figure A1(a), $\bar{F}_P$ leads to wrong segmentation at the edge of the large-scale background. The local attention mechanism rectifies such mistakes by referring to the correct information from $\hat{F}_I$.
- **It complements the missing features in $\bar{F}_P$.** In Figure A1(b), segmentation results of $\bar{F}_P$ lack relatively small segments due to low input resolution, e.g., sign symbols and bicyclists. With higher input resolution, $\hat{F}_I$ contains semantic features of these segments in detail, which can be utilized to enhance $\bar{F}_P$ by local attention.
- **It resists the misleading features from $\hat{F}_I$.** In Figure A1(c), noisy motion vectors result in $\hat{F}_I$ of low quality, which leads to incorrect segmentation results as a misleading reference. On the contrary, $\bar{F}_P$ provides accurate segmentation for these large-scale regions. Our proposed local attention allocates small weights to the misleading areas, thus resisting the noise in motion vectors.

*Corresponding author.

In each row of Figure A1, the query pixel of $\bar{F}_P$ is denoted by white crosses. In the fourth column, we visualize its normalized correlation scores inside the local neighborhood of $\hat{F}_I$. Higher brightness of the area indicates that local attention allocates larger weights while aggregating $\hat{F}_I$ into $\bar{F}_P$ at the query pixel. With the help of local-attention-based feature fusion, the query pixel obtains better semantic features in $\tilde{F}_P$, which leads to correct segmentation.

A2. Additional Experimental Results

A2.1. Performance Comparison to LR Baselines

Here we report the full comparison to image-based methods, including the 1.0x baselines and the LR baselines ranging from 0.5x-0.7x. For a fair comparison, we retrain the baseline models on LR images and evaluate them. The evaluation results of LR baselines and AR-Seg models are reported in Table A1. We also present the relative performance and computational cost changes compared to the 1.0x baseline in the $\widetilde{\Delta}$mIoU and $\widetilde{\Delta}$GFLOPs columns ($\widetilde{\Delta}x = \frac{\Delta x}{|x|}$), respectively. As shown in Table A1, the LR baselines reduce the computational cost but bring severe performance degradation. For example, on the Cityscapes dataset, BiseNet-0.5x saves 75.0% computational cost compared to BiseNet-1.0x, but the segmentation mIoU is reduced by -4.2%. As a comparison, the proposed AR$^{0.5}$-Bise18 models preserve the segmentation performance of 1.0x baselines with the help of the CReFF module and the FST strategy, while saving more than 65% computational cost.

A2.2. Performance of Video-based Methods and Corresponding Backbone Models

Since current state-of-the-art approaches conduct experiments with different single-frame backbone models, it isn't easy to provide a completely fair comparison. To provide

Figure A1. Visualization of the local attention mechanism in CReFF on the CamVid test set with $d = 11$. From *left* to *right*: P frame, warped I frame, segmentation generated from upsampled LR features $\bar{F}_P$, segmentation generated from warped I frame features $\hat{F}_I$, segmentation generated from aggregated features $\tilde{F}_P$ and the segmentation ground-truth. For the pixel denoted as the white cross in the P frame, its local attention weights are visualized over the segmentation results of $\hat{F}_I$ in the fourth column. The brighter color represents that more weight is placed on the neighborhood location. Better viewed in color.

Table A1. Comparison of AR-Seg and the image-based methods on CamVid *test* set and Cityscapes *valid* set.

	Method	PSPNet18 [23]				BiseNet18 [22]			
		mIoU(%)↑	$\widetilde{\Delta}$mIoU↑	GFLOPs ↓	$\widetilde{\Delta}$GFLOPs ↓	mIoU(%)↑	$\widetilde{\Delta}$mIoU↑	GFLOPs ↓	$\widetilde{\Delta}$GFLOPs ↓
CamVid	1.0x	69.43	-	309.02	-	71.57	-	58.83	-
	0.7x	69.03	-0.6%	151.43	-51.0%	69.92	-2.3%	29.08	-51.6%
	$AR^{0.7}$-	71.23	+2.6%	169.86	-45.0%	71.78	+0.3%	31.89	-45.8%
	0.6x	68.16	-1.8%	111.26	-64.0%	69.64	-2.7%	21.21	-64.0%
	$AR^{0.6}$-	70.82	+2.0%	133.09	-56.9%	71.60	+0.0%	24.68	-58.0%
	0.5x	66.87	-3.7%	77.27	-75.0%	68.01	-5.0%	14.97	-74.6%
	$AR^{0.5}$-	70.48	+1.5%	101.98	-67.0%	70.38	-1.7%	18.96	-67.8%
Cityscapes	1.0x	69.00	-	560.97	-	70.09	-	178.96	-
	0.7x	67.87	-1.6%	277.23	-50.6%	68.52	-2.2%	88.53	-50.5%
	$AR^{0.7}$-	70.23	+1.8%	302.95	-46.00%	70.86	+1.1%	97.10	-45.7%
	0.6x	67.03	-2.9%	202.97	-63.8%	67.66	-3.4%	66.56	-63.4%
	$AR^{0.6}$-	69.45	+0.7%	234.91	-58.1%	70.72	+0.9%	76.06	-57.5%
	0.5x	66.75	-3.3%	140.24	-75.0%	67.17	-4.2%	44.74	-75.0%
	$AR^{0.5}$-	69.03	+0.0%	177.44	-68.4%	70.57	+0.7%	57.00	-68.2%

Table A2. Comparison of AR-Seg and the video-based methods on CamVid *test* set and Cityscapes *valid* set.

		Single-frame baseline			Video approach			
	Method	Backbone	mIoU(%)↑	GFLOPs ↓	mIoU(%)↑	GFLOPs ↓	$\widetilde{\Delta}$mIoU↑	$\widetilde{\Delta}$GFLOPs ↓
CamVid	Accel-DL18 [10]	DeepLab18 [3]	58.13	245.65	66.15	397.70	**+13.8%**	+61.9%
	TD^4-PSP18 [8]	PSPNet18 [23]	69.43	309.02	70.13	363.70	+1.0%	+17.7%
	BlockCopy [18]	SwiftNet-RN50 [13]	70.41	215.90	66.75	107.52	-5.2%	-45.7%
	TapLab-BL2 [6]	MobileNetV2 [16]	69.93	236.40	67.57	117.73	-3.1%	-50.2%
	Jain et al. [9]	DeepLab50 [3]	70.65	318.12	67.61	146.97	-4.3%	-53.8%
	$AR^{0.6}$-PSP18	PSPNet18 [23]	69.43	309.02	70.82	101.98	+2.0%	-57.0%
	$AR^{0.6}$-Bise18	BiseNet18 [22]	**71.57**	**58.83**	**71.60**	**24.68**	+0.0%	**-58.0%**
Cityscapes	Accel-DL18 [10]	DeepLab18 [3]	57.64	516.20	68.25	1011.75	**+18.4%**	+96.0%
	TD^4-PSP18 [8]	PSPNet18 [23]	69.00	560.97	70.11	673.06	+1.6%	+20.0%
	BlockCopy [18]	SwiftNet-RN50 [13]	**72.47**	500.35	67.69	294.20	-6.7%	-41.2%
	TapLab-BL2 [6]	MobileNetV2 [16]	71.85	480.34	68.90	237.29	-4.1%	-50.6%
	Jain et al. [9]	DeepLab50 [3]	72.26	721.41	68.57	342.67	-5.1%	-52.5%
	$AR^{0.6}$-PSP18	PSPNet18 [23]	69.00	560.97	69.45	234.91	+0.7%	**-58.1%**
	$AR^{0.6}$-Bise18	BiseNet18 [22]	70.09	**178.96**	**70.72**	**76.06**	+0.9%	-57.5%

the most reasonable comparison, we report the results of the single-frame backbone models along with the video-based methods. For example, TD^4-PSP18 [8] makes use of PSPNet18 [23] as the single-frame backbone model. As shown in Table A2, PSPNet18 achieves 69.00 mIoU on the compressed Cityscapes dataset with 309.02 GFLOPs. Based on PSPNet18, the application of TDNet brings 17.7% additional computational cost and 1.0% performance improvement due to attention modules. We denote the relative changes using $\widetilde{\Delta}$GFLOPs=+17.7% and $\widetilde{\Delta}$mIoU=+1.0%. As for the Accel [10] framework, although it changes the network depth to achieve efficient VSS, it still raises the computational burden to a large extent due to the heavy optical flow network. Other video-based baselines, including BlockCopy [18], TapLab [6], and Jain et al. [9], reduce the computational cost by around 50%, but fail to maintain the performance of single-frame backbone model, resulting in accuracy degradation ranging from 3% to 7%.

As a comparison, the proposed AR-Seg successfully preserves the segmentation accuracy of single-frame backbone models while saving more than 55% computational cost. The computational cost reduction mainly comes from the altering resolution strategy, and the capability of preserving performance comes from the composition of CReFF and FST. Utilizing the same PSPNet18 backbone as TD^4-PSP18, $AR^{0.6}$-PSP18 achieves similar performance on the compressed CamVid and Cityscapes datasets at the cost of much less computation. Compared to these video-based methods, our proposed AR-Seg is a better solution to maintain performance and reduce computational costs. With

the lightweight backbone model BiseNet, AR$^{0.6}$-Bise18 achieves good accuracy and computational cost performance.

A2.3. Per-class Analysis

We report the IoUs of some representative classes of CamVid in Table A3. Our method performs better on categories like fence, pedestrian, and bicyclist. These classes benefit from our carefully designed CReFF module. Note that similar to other methods using feature propagation [6, 8], AR-Seg performs worse on small objects, e.g., the column pole category. As for the color coding, the bus and traffic light in CamVid share the same color because they belong to the ignored classes [8, 10].

Table A3. IoUs(%) of some classes in the CamVid dataset.

Class	Fence	Pedestrian	Bicyclist	Column Pole
PSPNet18(1.0x)	61.81	48.57	60.74	**33.10**
AR$^{0.5}$-PSP18	**66.47**	**50.97**	**65.15**	31.21

A2.4. Experiment with Strong Backbones

We further evaluate AR-Seg with stronger CNNs backbones and a transformer backbone on Cityscapes. The experimental results are summarized in Table A4, showing that all these backbones can benefit from our method. Note that the mIoUs of these backbones are lower than the original papers due to the compression artifacts in compressed videos.

Table A4. Results of more backbones on the Cityscapes dataset.

Backbone	1.0x baseline		AR$^{0.6}$-	
	mIoU↑	GFLOPs↓	mIoU↑	GFLOPs↓
PSPNet-50 [23]	72.91	1395.85	73.27	577.88
PSPNet-101 [23]	75.30	2181.71	75.53	918.50
SegFormer-MIT-B3 [21]	77.54	987.75	78.01	390.16

A3. Experiment Details

A3.1. Video Compression Configuration

The original CamVid and Cityscapes datasets contain 60Mbps videos that are almost losslessly encoded without motion vectors. However, since 60Mbps doesn't match the bit-rate for videos around 1080p in realistic streaming and storage applications, we pre-process the original video frames with an HEVC encode-decode process to simulate the compressed videos with reasonable bit-rates. First, with a pre-defined length of Group Of Picture (GOP) L, the frames in a GOP are compacted into yuv format by FFmpeg. Then we use x265[1] to encode the yuv frames into the HEVC video stream. The codec settings to produce our main experiment results are as follows:

- *CamVid*: L is set to 12, the frame rate of HEVC video stream is set to 30 , and the bitrate is set to 3Mbps.
- *Cityscapes*: L is set to 12, the frame rate of HEVC video stream is set to 17 , and the bitrate is set to 5Mbps.

As for the structure of GOP, we consider I and P frames in our experiments for simplicity. We also turn on the rectangular and asymmetric motion partitions to fully utilize the HEVC/H.265 standard. For the last pre-processing step, we decode the HEVC video stream to get the motion vectors and compressed video frames for model training and evaluation. We follow the practice in previous work [20] to trace back the MV of P frames to the I frame. For video formats that contain B frames, motion vectors can also be obtained in the same way. We describe the treatment including B frames in Section A4.

Since the pioneering work CoViAR [20], the auxiliary information in compressed videos, like motion vectors and residuals, has been well exploited in computer vision tasks. However, existing literature mainly conducts experiments based on MPEG4 and H.264 codec [6, 9], which are far behind the state-of-the-art video compression standard: HEVC/H.265 and VVC/H.266. As far as we know, we are the first to extract and utilize the motion vectors of the HEVC/H.265 video stream. The code for pre-processing will be released together with the code for experiments.

A3.2. Training Details

Our models and baselines are initialized with ImageNet [5] pre-trained parameters. We use random flipping, random scaling, and random cropping as data augmentation.

We train AR-Seg in two stages. In the first stage, we train the HR branch on HR images using the segmentation loss. Note that if a pre-trained image segmentation model is available, the first stage can be skipped. In the second stage, we fix the HR branch and train the LR branch with FST. The neighborhood size of CReFF is set to 7×7 in our experiments. Given a GOP length L, we train the LR branch with image pairs (i, p), where p refers to the P frame with annotation and $i = p - (L - 1)$ refers to the I frame as a reference. We denote the distance between the annotated and the reference frames as d, then $d = L-1$ for the training pairs.

[1] x265 is a software codec for creating digital video streams in the HEVC/H.265 video compression format. The source code is available at: https://github.com/videolan/x265

For the CamVid [2] dataset, networks are trained by Adam optimizer [11] for 100 epochs with batch size 8. The learning rate is initialized as 0.001 and decays following the cosine annealing schedule.

For the Cityscapes [4] dataset, networks are trained by stochastic gradient descent [17] with momentum 0.9 and weight decay 5e-4 for 200 epochs with batch size 16. The learning rate is initialized as 0.01 and follows the same cosine annealing decay schedule.

We train our models with image frames from the high-bit-rate (HBR) original videos and motion vectors from the low bit-rate (LBR) compressed videos, and evaluate them on image frames and motion vectors extracted from the low bit-rate (LBR) compressed videos. All other video-based baseline methods [6, 8–10, 18, 24] are also trained on the HBR original video frames. We directly evaluate their pre-trained models on the compressed LBR videos for a fair comparison. As explained in Section A3.1, the HBR videos are encoded at 60Mbps, and the LBR videos are encoded at 5Mbps and 3Mbps for Cityscapes and CamVid, respectively.

We also directly trained our method and baselines on the compressed LBR video frames and motion vectors. However, the LBR models consistently perform worse than the HBR models. For example, we present the evaluation results on the CamVid validation set in Table A5. The compression artifacts in LBR videos could be the main obstacle to training good models.

Table A5. Performance of models trained on different bit-rate versions of CamVid. Models are evaluated on the compressed 3Mbps CamVid validation videos.

Method	HBR Model	LBR Model	GFLOPs
PSPNet18-1.0x	**69.43**	68.48	309.02
PSPNet18-0.6x	**68.16**	67.51	77.27
BiseNet18-1.0x	**71.57**	70.82	58.83
BiseNet18-0.6x	**69.64**	68.73	14.97
Accel-DL18 [10]	**66.15**	65.48	397.70
TD^4-PSP18 [8]	**70.13**	69.26	363.70
BlockCopy [18]	**66.75**	66.07	43.46
TapLab-BL2 [6]	**67.57**	66.83	167.53
Jain et al. [9]	**67.61**	66.39	146.97
AR$^{0.6}$-PSP18 (ours)	**70.82**	70.11	101.98
AR$^{0.6}$-Bise18 (ours)	**71.60**	70.87	18.96

A3.3. Evaluation Details

For evaluation, since only a part of video frames in the CamVid and Cityscapes datasets are annotated with semantic labels, we follow the previous works [10, 24] to simulate the mIoU evaluation on densely per-frame labeled video datasets. Specifically, for each frame p with annotation, we select its reference keyframe i by $d \in [0, L-1]$. For $d = 0$, we treat frame p as the keyframe and process it by the HR branch. Otherwise, we feed frame p into the LR branch. The average of $mIoU_d$ for each distance d is reported as the mIoU result.

To evaluate the computational complexity, we measure the FLOPs by PyTorch-OpCounter [15] following the previous methods [12, 14]. Note that within a GOP, the keyframe is processed by the HR branch, and the other non-keyframes are processed by the LR branch. We report the average FLOPs within a GOP computed with the following equation:

$$FLOPs = \frac{1}{L} \times FLOPs^{HR} + \frac{L-1}{L} \times FLOPs^{LR}. \quad (1)$$

All the comparisons are evaluated on the compressed videos.

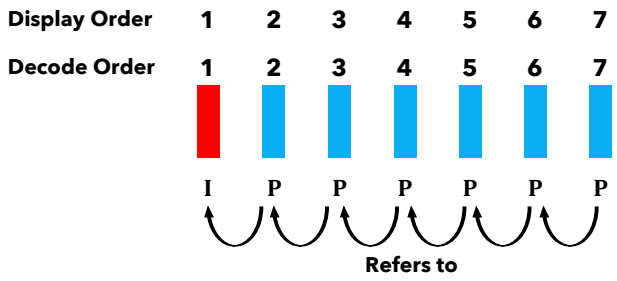
A4. Full Treatment Including B Frames

To simplify the description of our method, we mainly describe and experiment with our proposed AR-Seg framework based on compressed videos consisting of I frames and P frames in the main paper. In this section, we explain how to apply our method to compressed videos containing both P and B frames.

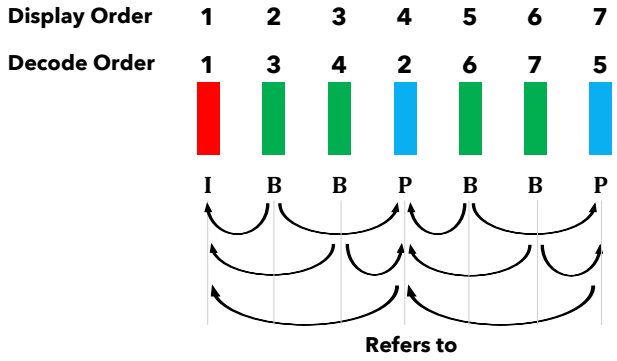
As shown in Figure A2(a), a P frame refers to its previous I frame or P frame. The motion vectors from the P frame to its reference frame (not always the I frame) have already been stored in the compressed video. While the video decoder reconstructs a P frame, we can obtain the motion vectors from this P frame to the I frame by accumulating along its reference trajectory, e.g., *3(P)-2(P)-1(I)* for frame *3(P)*. The tracing-back operation only accumulates existing data (per-frame motion vectors) and incurs little computational overhead.

For compressed videos containing B frames, as shown in Figure A2(b), the overall reference structure and decoding order become more complicated since a B frame bidirectionally refers to the I frame or P frame before and after it. However, despite the complicated decoding order in video decoders, the B frame's reference frames (the P frame and I frame) have already been decoded when the video decoder reconstructs it, which is the same as the P frame's condition. Then we can find a corresponding pixel in the I frame for each pixel in this B frame following its reference trajectory, e.g., two trajectories *2(B)-4(P)-1(I)* and *2(B)-1(I)* for frame *2(B)*. This way, we can obtain the motion vectors from the B-frame to the I-frame in the same trace-back manner as P frames with little computational overhead.

With the obtained motion vectors for P frames and B frames, our proposed AR-Seg can process the decoded video frames efficiently. Moreover, since AR-Seg identifies keyframes according to the GOP structure in com-



(a) GOP structure for compressed videos with I frames and P frames only. GOP=7.



(b) GOP structure for compressed videos with I frames, P frames and B frames. GOP=7.

Figure A2. Illustrations for different GOP structures in compressed videos.

pressed videos, our segmentation framework can process each frame as soon as the video decoder reconstructs it without incurring additional latency. Such an online processing property makes it possible to integrate AR-Seg with video decoders in actual low-latency real-time applications.

A5. BD-FLOPs

In this section, we describe the idea and formulation of BD-FLOPs mentioned in **Resolution of LR branch** of Sec. 4.3. BD-FLOPs is designed to measure the average complexity reduction brought by the proposed Video Semantic Segmentation (VSS) algorithms. Following the widely applied metrics BD-Rate [1] in video compression, we derive BD-FLOPs by replacing the bitrate-PSNR of video encoders with the complexity-accuracy of VSS algorithms.

Specifically, given the computational cost X under different input resolutions and the corresponding segmentation accuracy Y, we fit a cubic polynomial function $\mathcal{F}(y)$ to the data points $(y_j, log(x_j))$, where $X = [x_0, x_1, \ldots, x_N]$, $x_j \in \mathcal{R}$, $j = 0, \ldots, N$, and $Y = [y_0, y_1, \ldots, y_N]$, $y_j \in \mathcal{R}$, $j = 0, \ldots, N$. Since X spans multiple orders of magnitude, similar to Bjontegaard's analysis of bitrate [1], we follow the practice in BD-Rate and calculate the relative complexity reduction in the logarithmic space. The fitted polynomial function $\mathcal{F}(y)$ represents the accuracy-complexity curve of a specific VSS algorithm, and can be used to calculate the average difference in computational cost between different methods.

In this paper, we measure the complexity reduction of our proposed altering resolution algorithm (AR) compared to the constant resolution baseline (CR) by BD-FLOPs. Our experiments are conducted on the CamVid [2] dataset. Both methods are equipped with the same image segmentation backbone, e.g., PSPNet18 [23], or BiseNet18 [22]. Based on the data points (X^{CR}, Y^{CR}) and (X^{AR}, Y^{AR}), we fit the polynomial curves $\mathcal{F}^{CR}(y)$ and $\mathcal{F}^{AR}(y)$, respectively. The average complexity reduction BD-FLOPs of AR-Seg is calculated by

$$\text{BD-FLOPs} = e^D - 1, \tag{2}$$

$$D = \frac{1}{y_{lim1} - y_{lim0}} \int_{y_{lim0}}^{y_{lim1}} [\mathcal{F}^{AR}(y) - \mathcal{F}^{CR}(y)]\, dy, \tag{3}$$

where $y_{lim0} = \max(y_0^{CR}, y_0^{AR})$ and $y_{lim1} = \min(y_N^{CR}, y_N^{AR})$ denote the integral limits. The calculation of BD-FLOPs with backbone BiseNet18 is visualized in Figure A3. Since $\mathcal{F}^{AR}(y)$ always lies under $\mathcal{F}^{CR}(y)$, the integral result D is less than zero for the illustrated example, leading to a negative value of BD-FLOPs.

We calculate BD-FLOPs with all data points. Each algorithm has eight points in total, with input resolution ranging from 0.3x to 1.0x. AR-Seg achieves -76.73% and -51.06% BD-FLOPs with PSPNet18 and BiseNet18, respectively. BD-FLOPs measures the average computational reduction at different accuracy levels. Such a curve-to-curve difference is a more compact and, in some sense, more accurate way to evaluate the efficient VSS algorithms, compared to the point-to-point difference used in [8, 19]. Furthermore, for other complexity-concerned algorithms, e.g., dynamic neural networks [7], our proposed BD-FLOPs can also be applied for better evaluation.

Similarly, the mentioned BD-mIoU in Sec. 4.3 can also be calculated following the above steps by simply exchanging xs and ys.

References

[1] G. Bjontegaard. *Calculation of average PSNR differences between RD-curves*. VCEG-M33, Austin, TX, USA, Apr. 2001. 6

[2] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognit. Lett.*, 30(2):88–97, 2009. 5, 6

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous con-

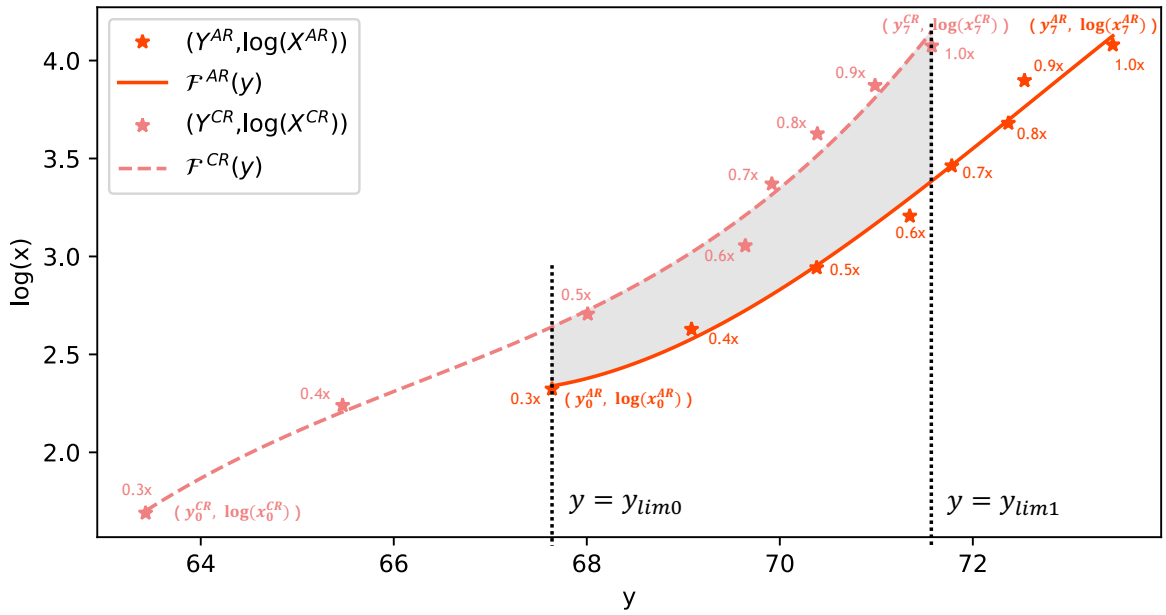


Figure A3. Illustration of the BD-FLOPs calculation with BiseNet18. $\mathcal{F}^{CR}(y)$ and $\mathcal{F}^{AR}(y)$ denote the fitted cubic polynomial function. $(Y, log(X))$ denotes the data points results from experiments with different resolutions. The integral interval of BD-FLOPs calculation is determined by y_{lim0} and y_{lim1}.

volution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018. 3

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3213–3223. IEEE Computer Society, 2016. 5

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009. 4

[6] Junyi Feng, Songyuan Li, Xi Li, Fei Wu, Qi Tian, Ming-Hsuan Yang, and Haibin Ling. Taplab: A fast framework for semantic video segmentation tapping into compressed-domain knowledge. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(3):1591–1603, 2022. 3, 4, 5

[7] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(11):7436–7456, 2022. 6

[8] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. Temporally distributed networks for fast video semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8815–8824. Computer Vision Foundation / IEEE, 2020. 3, 4, 5, 6

[9] Samvit Jain and Joseph E. Gonzalez. Fast semantic segmentation on video using block motion-based feature interpolation. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, volume 11132 of *Lecture Notes in Computer Science*, pages 3–6. Springer, 2018. 3, 4, 5

[10] Samvit Jain, Xin Wang, and Joseph E. Gonzalez. Accel: A corrective fusion network for efficient semantic segmentation on video. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8866–8875. Computer Vision Foundation / IEEE, 2019. 3, 4, 5

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5

[12] Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4061–4070. Computer Vision Foundation / IEEE, 2021. 5

[13] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *IEEE Conference on Computer Vision and Pattern Recognition,*

CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 12607–12616. Computer Vision Foundation / IEEE, 2019. 3

[14] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12607–12616. Computer Vision Foundation / IEEE, 2019. 5

[15] PyTorch-OpCounter Contributors. THOP: PyTorch-OpCounter, 11 2018. 5

[16] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018. 3

[17] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1139–1147. JMLR.org, 2013. 5

[18] Thomas Verelst and Tinne Tuytelaars. Blockcopy: High-resolution video processing with block-sparse feature propagation and online policies. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 5138–5147. IEEE, 2021. 3, 5

[19] Li Wang, Dong Li, Yousong Zhu, Lu Tian, and Yi Shan. Dual super-resolution learning for semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 3773–3782. Computer Vision Foundation / IEEE, 2020. 6

[20] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. Compressed video action recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6026–6035. Computer Vision Foundation / IEEE Computer Society, 2018. 4

[21] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12077–12090, 2021. 4

[22] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, volume 11217 of *Lecture Notes in Computer Science*, pages 334–349. Springer, 2018. 3, 6

[23] 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 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6230–6239. IEEE Computer Society, 2017. 3, 4, 6

[24] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4141–4150. IEEE Computer Society, 2017. 5