# Real-Time, Accurate, and Consistent Video Semantic Segmentation via Unsupervised Adaptation and Cross-Unit Deployment on Mobile Device

Hyojin Park* Alan Yessenbayev Tushar Singhal Navin Kumar Adhikari
Yizhe Zhang† Shubhankar Mangesh Borse Hong Cai Nilesh Prasad Pandey Fei Yin
Frank Mayer Balaji Calidas Fatih Porikli

*Qualcomm AI Research*‡

## Abstract

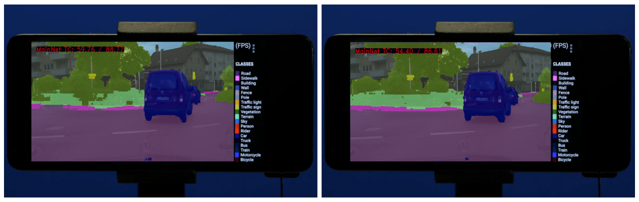
*This demonstration showcases our innovations on efficient, accurate, and temporally consistent video semantic segmentation on mobile device. We employ our test-time unsupervised scheme, AuxAdapt, to enable the segmentation model to adapt to a given video in an online manner. More specifically, we leverage a small auxiliary network to perform weight updates and keep the large, main segmentation network frozen. This significantly reduces the computational cost of adaptation when compared to previous methods (e.g., Tent, DVP), and at the same time, prevents catastrophic forgetting. By running AuxAdapt, we can considerably improve the temporal consistency of video segmentation while maintaining the accuracy.*

*We demonstrate how to efficiently deploy our adaptive video segmentation algorithm on a smartphone powered by a Snapdragon® Mobile Platform*[1]*. Rather than simply running the entire algorithm on the GPU, we adopt a cross-unit deployment strategy. The main network, which will be frozen during test time, will perform inferences on a highly optimized AI accelerator unit, while the small auxiliary network, which will be updated on the fly, will run forward passes and back-propagations on the GPU. Such a deployment scheme best utilizes the available processing power on the smartphone and enables real-time operation of our adaptive video segmentation algorithm. We provide example videos in supplementary material.*

## 1. Introduction

Semantic segmentation is a core functionality for visual scene understanding and is of great importance to various applications, such as AR/VR, self-driving/ADAS, robotics, and mobile image/video processing. Given the rapid growth of video data in recent years, the ability to segment video content has become increasingly important. However, directly applying an image-based segmentation model to video data often results in temporally inconsistent (e.g., flickering) outputs, an example of which is shown in Fig. 1 (a). These artifacts can significantly impact downstream applications like background editing and degrade user experience.

In the literature, researchers have looked into various ways to improve temporal consistency. A major line of work utilizes optical flow during training and/or testing, as it captures pixel correspondence across frames [1–5]. However, their performance is constrained by the quality of the


(a) Sample video segmentation *without* AuxAdapt

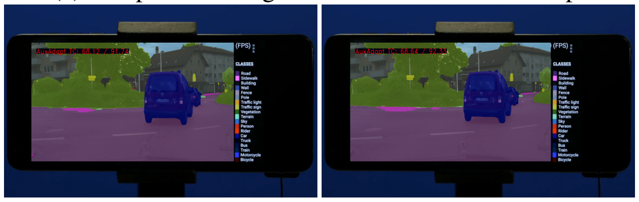

(b) Sample video segmentation *with* AuxAdapt

Figure 1. Snapshots of our video segmentation demonstration. (a) Sample video segmentation without using AuxAdapt. (b) Sample video segmentation with AuxAdapt. When using AuxAdapt, we see higher temporal consistency and an overall better quality of video segmentation. On the other hand, without AuxAdapt, the segmentation suffers from flickering artifacts, e.g., in the regions that transition between road/sidewalk and vegetation.

[1]Snapdragon is a product of Qualcomm Technologies, Inc. and/or its subsidiaries.
*Corresponding author. Email: hyojinp@qti.qualcomm.com
†Contributed to this work while at Qualcomm AI Research
‡Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

estimated optical flow and they also require accurately annotated video training sets which can be costly to collect. Some other works apply test-time adaptation to enhance model performance, but they are too computationally expensive to run in real time [6–8].

To address these challenges, we leverage our recently proposed adaptation scheme, AuxAdapt [9], to efficiently perform online model updates to enhance temporal consistency. AuxAdapt is designed based on the key insight that inconsistency often arises from uncertainty in the network's decisions. As such, AuxAdapt enforces the segmentation model to learn from its own outputs and reinforce its prediction confidence. In addition, to significantly reduce the computation cost of online adaptation, AuxAdapt employs a small auxiliary network (AuxNet) to perform updates while keeping the main segmentation network (MainNet) unchanged. In other words, when processing each frame, MainNet is frozen and only AuxNet is updated, while the integrated model streams through the video. The final segmentation is determined by the aggregated outputs of the two networks. By doing this, we avoid costly model updates and catastrophic forgetting, and can enhance temporal consistency without requiring optical flow or other temporal features which can be unreliable.

In this demonstration, we deploy our adaptive video segmentation algorithm, AuxAdapt, on a Snapdragon-powered smartphone via a novel cross-unit deployment strategy. While a common approach would be to simply run the entire algorithm on the mobile GPU, this does not fully utilize the available processing power on the phone. Although AuxAdapt is significantly less computationally demanding as compared to previous adaptation methods, running all the forward and backward operations on the GPU may not lead to real-time video segmentation. Therefore, we additionally leverage the highly optimized AI accelerator on the smartphone and run a quantized version of the MainNet on it since its weights will not change. Meanwhile, the AuxNet runs on the GPU and adapts it weights as the model streams through the video. By doing this, we enable real-time operation of our adaptive video segmentation model, which provides accurate and temporally consistent segmentation results.

When evaluating the temporal consistency of video segmentation, it is common to utilize estimated optical flow. However, optical flow estimation may itself contain errors and thus cause unreliable evaluation results. Recently, we have proposed a new measure based on perceptual consistency [10]. As we have shown in the original paper, perceptual consistency can more accurately capture the temporal consistency of video segmentation. As such, in addition to the existing flow-based measure, we also perform temporal consistency evaluation using our perceptual consistency metric. Furthermore, in the demonstration, we will provide concrete examples to demonstrate the advantages of using perceptual consistency over estimated optical flow.

Via this demonstration, we hope to motivate further research in temporally consistent and adaptive video segmentation algorithms, as well as the efficient deployment of them on mobile devices. We summarize the main contributions of this demonstration as follows.

- We employ our latest online adaptation scheme, AuxAdapt, to obtain accurate and temporally consistent video semantic segmentation. As compared to previous adaptation frameworks (e.g., Tent [7], DVP [8]), AuxAdapt only requires a fraction of their computational costs for model update.
- We propose a novel cross-unit deployment strategy to efficiently run our adaptive video segmentation algorithm on a commercial smartphone powered by Qualcomm AI Engine.[2] Specifically, the larger, frozen part of the model runs on a highly efficient AI accelerator and a smaller, adaptive part runs on the mobile GPU. This allows us to better leverage the processing power on the phone and achieve real-time performance.
- When evaluating temporal consistency, in addition to the conventional approach based on estimated optical flow, we further utilize a new, perceptual-consistency-based method. This new evaluation method can more properly capture temporal consistency as compared to using an estimated optical flow, as we shall also show in this demonstration.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of our adaptive video segmentation algorithm, AuxAdapt and our novel way of evaluating temporal consistency, as well as sample evaluation results. In Section 3, we present our novel scheme that deploys the algorithm across different computing units (i.e., GPU and AI accelerator) on the smartphone to enable real-time performance. In Section 4, we describe details of our implementation and how the demonstration will be conducted. Conclusions will be given at the end.

## 2. AuxAdapt and Temporal Consistency Evaluation

In this section, we briefly explain the AuxAdapt [9] framework, which is applied as a online, unsupervised, test-time adaptation scheme to enhance the temporal consistency of a video semantic segmentation model. We then describe two evaluation schemes for measuring temporal consistency, based on estimated optical flow and perceptual consistency [10], respectively, and compare evaluation results without and with using AuxAdapt.

[2]Qualcomm AI Engine is a product of Qualcomm Technologies, Inc.

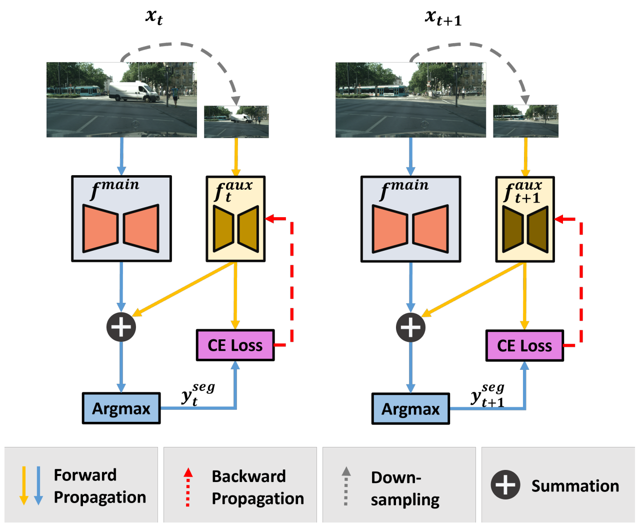


Figure 2. Illustration of the AuxAdapt framework. AuxAdapt uses a tiny auxiliary network AuxNet $f^{aux}$ on lower spatial resolution together with the main segmentation network $f^{main}$ to generate segmentation decisions. For each frame $x_t$, it only updates the AuxNet based on the aggregated segmentation output $y_t^{seg}$, which is used as the target for training the AuxNet with a cross-entropy loss.

## 2.1. AuxAdapt

Consider a pretrained semantic segmentation network $f^{\text{main}}$. It takes an input image, $x \in [0, 1]^{H\times W\times 3}$, and generates a 2D prediction map, $y^{\text{main}} \in \mathbb{R}^{H\times W\times K}$, for a $K$-class segmentation task. $H$ and $W$ are the height and width of the input image, respectively. $y^{\text{main}}(i, j, k)$ indicates how likely this pixel belongs to class $k$ for the pixel location $(i, j)$. For the segmentation decision, an argmax operation is applied pixel-wise to the last dimension of $y^{\text{main}}$ to assign the most probable class to each pixel. We denote this hard decision as $y^{\text{seg}} \in \{1, 2, \dots, K\}^{H\times W}$. To obtain semantic segmentation for a sequence of video frames $X = \{x_1, \dots, x_T\}$, $f^{\text{main}}$ can be applied to generate $Y = \{y_1^{\text{seg}}, \dots, y_T^{\text{seg}}\}$, where $T$ is the number of frames in the video.

Applying $f^{\text{main}}$ directly to $X$ usually results in temporally inconsistent segmentation outputs, as shown in Fig. 1 (a). This is mainly due to the network's uncertainty in its own predictions, which lead to different segmentation results on visually similar regions across frames. As such, AuxAdapt reinforces the network with its own predictions, i.e., training the network on its own decisions, in order to enhance the network's prediction confidence and consistency. While one can simply update the entire network based on its own outputs to achieve uncertainty reduction, performing back-propagation through the entire network is computationally expensive and for a long video, can make the network deviate too much from its stable pretrained weights, resulting in degraded accuracy.

and/or its subsidiaries.

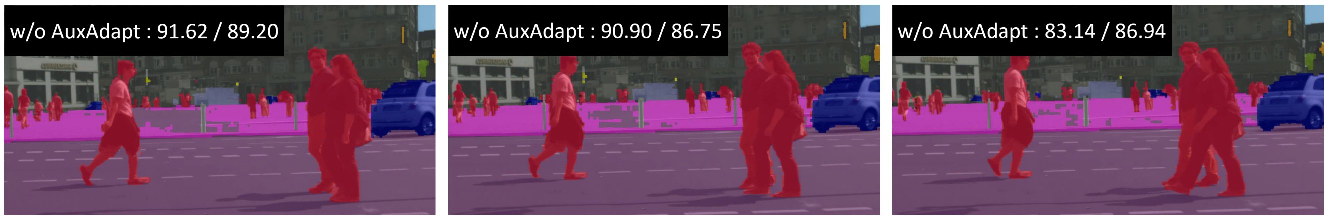
**Algorithm 1:** AuxAdapt

```
Input: $x_1, x_2, \dots, x_T$;
Output: $y_1^{\text{seg}}, y_2^{\text{seg}}, \dots, y_T^{\text{seg}}$;
Load trained MainNet $f^{\text{main}}$, which will be frozen;
Load trained AuxNet $f^{\text{aux}}$ as $f_1^{\text{aux}}$;
Initialize $t = 1$;
while $t \le T$ do
    $y_t^{\text{main}} = f^{\text{main}}(x_t)$,;
    $y_t^{\text{aux}} = f_t^{\text{aux}}(x_t)$;
    $y_t^{\text{seg}}(i, j) = \underset{k}{\text{argmax}}\, y_t^{\text{main}}(i, j, k) + y_t^{\text{aux}}(i, j, k), \forall (i, j)$;
    Compute cross-entropy loss: $\mathcal{L}(y_t^{\text{aux}}, y_t^{\text{seg}})$;
    Update $f_t^{\text{aux}}$ based on the above loss, which gives $f_{t+1}^{\text{aux}}$;
    $t \leftarrow t + 1$;
end
```

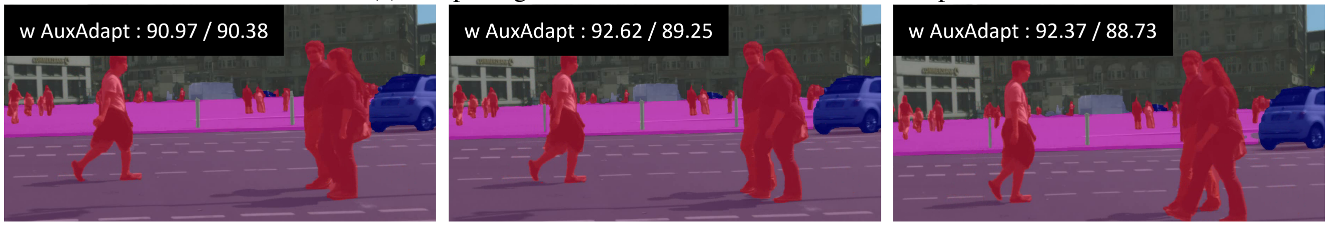
In order to perform adaptation and avoid the disadvantages of naively updating the entire model, AuxAdapt utilizes an auxiliary network (AuxNet), $f^{\text{aux}}$, to work with the main segmentation network (MainNet), $f^{\text{main}}$. AuxNet is a separately-trained small-sized segmentation network. For a video frame at time $t$, MainNet and AuxNet produce their respective prediction maps, $y_t^{\text{main}}$ and $y_t^{\text{aux}}$, and then an argmax operation is applied to their summation to obtain the final segmentation decision, $y_t^{\text{seg}}$. When streaming a video, the MainNet is frozen and the AuxNet is updated based on $y_t^{\text{seg}}$. Our AuxAdapt framework is illustrated in Fig. 2 and summarized in Algorithm 1.

In this process, while the MainNet does not change, the AuxNet adapts itself by learning from the final combined decision to reduce the overall prediction uncertainty. By only updating the small AuxNet, the forward and backward operations will be much cheaper than those of updating the MainNet. Furthermore, AuxAdapt can maintain the accuracy on both a given, specific test video and other general test images. Specifically, since MainNet is unchanged, we fundamentally prevent catastrophic forgetting and the variation of accuracy is limited due to the MainNet's contribution to the overall output. The AuxNet also distills knowledge from the MainNet through the combined output and learns to match the MainNet's performance.

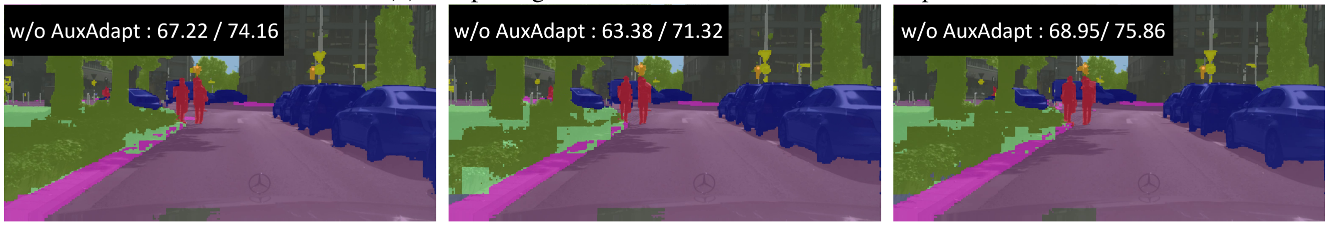
**Networks:** In this demonstration, we adopt the state-of-the-art segmentation model, HRNet-w18 [11], and apply its variants for both the MainNet and the AuxNet. In particular, for the AuxNet, we analyze the runtime for each layer of HRNet-w18 on GPU and reduce the computation (e.g., number of channels) of observed bottlenecks. We also use a lower input resolution for the AuxNet to further reduce computation.


w/o AuxAdapt : 91.62 / 89.20
w/o AuxAdapt : 90.90 / 86.75
w/o AuxAdapt : 83.14 / 86.94

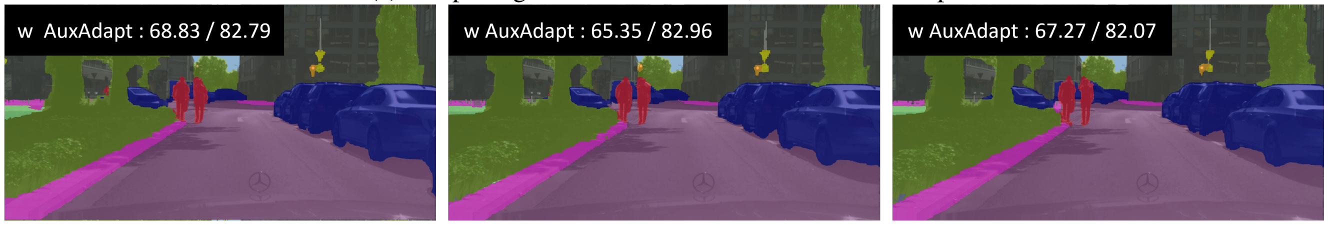
(a) Sample segmentation of video 1, *without* AuxAdapt


w AuxAdapt : 90.97 / 90.38
w AuxAdapt : 92.62 / 89.25
w AuxAdapt : 92.37 / 88.73

(b) Sample segmentation of video 1, *with* AuxAdapt


w/o AuxAdapt : 67.22 / 74.16
w/o AuxAdapt : 63.38 / 71.32
w/o AuxAdapt : 68.95/ 75.86

(c) Sample segmentation of video 2, *without* AuxAdapt


w AuxAdapt : 68.83 / 82.79
w AuxAdapt : 65.35 / 82.96
w AuxAdapt : 67.27 / 82.07

(d) Sample segmentation of video 2, *with* AuxAdapt

Figure 3. Segmentation results of two sequences of video frames, without and with using AuxAdapt. Rows 1 and 3 show the results *without* AuxAdapt, and rows 2 and 4 are obtained by applying *AuxAdapt*. For each frame, we also show the temporal consistency scores (between current and previous frames) based on optical flow (left) and perceptual consistency (right). As visible, AuxAdapt considerably enhances the temporal consistency of the video segmentation, which is also reflected by the scores.

## 2.2. Evaluation of Temporal Consistency

In this section, we first describe two ways of evaluating temporal consistency, and then evaluate segmentation results without and with using AuxAdapt on the Cityscapes dataset [12]. When measuring temporal consistency, the first method is based on estimated optical flow and has been commonly used. The second method is based on our newly proposed perceptual consistency, which leverages visual similarity and does not require optical flow. As we will see, our perceptual-consistency-based scheme offers a more reliable way to measure temporal consistency.

**Optical-flow-based measure:** Many existing papers use estimated optical flow to evaluate the temporal consistency of the video segmentation, e.g., [4, 5, 13, 14]. Following [5], we adopt FlowNet2 [15] to compute the optical flow between two adjacent frames and warp the segmentation at frame $t$ to frame $t-1$. We then compare the warped and estimated segmentation masks for each frame ($t < T$) using the standard mean Intersection-over-Union (mIoU) metric. The overall mIoU then serves as the flow-based temporal consistency metric, which we denote as *F-TC*. Note that this is the same metric used in [5] and more details can be found therein.

Although the flow-based measure has been widely used, there are several drawbacks. First, it is challenging to generate highly accurate and generalizable optical flow estimations. In addition, it is susceptible to occlusions and objects moving out of the frame.

**Perceptual-consistency-based measure:** Recently, we have proposed to utilize perceptual consistency to capture

| **Method** | **F-TC** | **PC** | **mIoU** | **GMAC** |
|---|---|---|---|---|
| w/o AuxAdapt | 73.06 | 81.02 | 72.47 | 17.0 |
| w AuxAdapt | 74.46 | 82.25 | 73.18 | 21.2 |

Table 1. Performance evaluation results on Cityscapes validation set. F-TC denotes the flow-based temporal consistency measure (using FlowNet2) and PC denotes the perceptual-consistency-based temporal consistency measure. GMAC is calculated based on an input size of $1024 \times 2048$.

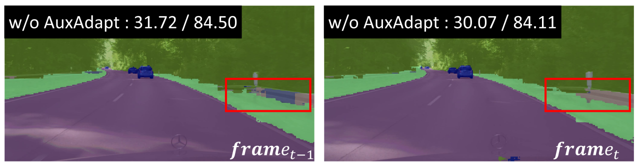
the temporal consistency of video segmentation [10]. Given two nearby video frames and the corresponding predicted segmentation maps, we assess how much the segmentation agrees with the cross-frame pixel correspondence established on the two frames' perceptual feature maps.

To quantify this, for each pixel in one frame, we first find the most correlated pixel from the other frame by matching perceptual features. These two pixels are expected to belong to the same class. Next, we find the most correlated pixel that is also agreed by the segmentation maps. If the segmentation agrees with the perceptual correspondence, the correlations found via unconstrained feature matching and segmentation-agreed feature matching will be equal. Otherwise, the constrained one will be smaller. As such, we can use the ratio between these two correlations to quantify the pixel-wise agreement between the segmentation and the perceptual correspondence. This can then be aggregated over the pixels to measure the perceptual consistency between the segmentation maps on the two frames, which naturally captures their temporal consistency.

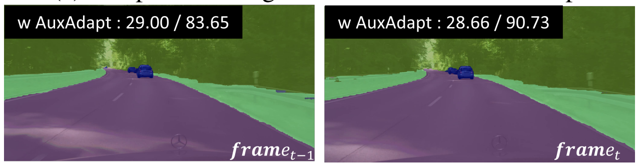
Unlike optical flow, perceptual consistency does not look for exact pixel correspondence across two images. Instead, it finds maximally correlated pairs of pixels. This makes perceptual consistency immune to cases like occlusions and objects moving out of frame where exact correspondence no longer exists.

**Evaluation results on Cityscapes:** We use the Cityscapes dataset to showcase the efficacy of our video segmentation by using AuxAdapt. As summarized in Table 1, our approach achieves higher temporal consistency in terms of both metrics.[3] Specifically, AuxAdapt achieves a temporal consistency score of 74.46 based on the flow-based measure (F-TC) and 82.25 based on the perceptual-consistency-based measure (PC). On the other hand, when not using AuxAdapt, i.e., the MainNet alone is applied to each video frame without any adaptation, the respective scores are lower: 73.06 and 81.02. In terms of segmentation accuracy (in mIoU), we achieve a similar (slightly higher) accuracy as compared to case of not using AuxAdapt. We also report the computational costs measured by GMAC and it can be seen that the additional computation due to the adaptation of AuxAdapt is merely 4.2G. By using the proposed AuxAdapt, we efficiently improve temporal consistency while preserving original accuracy.

[3]Note that higher scores indicate higher temporal consistency for both metrics.



(a) Sample video segmentation *without* AuxAdapt



(b) Sample video segmentation *with* AuxAdapt

Figure 4. Comparison of two temporal consistency metrics for a sample pair of consecutive video frames. On each frame, the temporal consistency scores based on the flow-based metric and the perceptual-consistency-based metric on shown on the left and right, respectively. It can be seen that perceptual consistency correctly assigns higher scores to the clearly more consistent results in the bottom row, whereas the flow-based metric incorrectly indicates that the video segmentation without AuxAdapt (top) is more consistent,

Sample video segmentation results without and with using AuxAdapt are shown in Fig. 3. It can be seen that AuxAdapt significantly improves the temporal consistency of the video segmentation. Furthermore, we show the temporal consistency scores on each frame (computed between the current and previous frames) based on optical flow (left) and perceptual consistency (right). The scores also indicate that the video segmentation based on AuxAdapt achieves better temporal consistency. In the demonstration, we will show consistent segmentation on full videos running live on a smartphone.

**Comparing two temporal consistency metrics:** Fig. 4 shows the flow-based and perceptual-consistency-based temporal consistency scores sample video segmentation results. It can be seen that in Fig. 4 (a), when not using AuxAdapt, the segmentation results has flickering artifacts (highlighted by the red boxes). The video segmentation with AuxAdapt, on the other hand, shows significantly more consistent results across the frames, as can be seen in Fig. 4 (b). We show the respective scores based on two temporal consistency measures for each frame (computed between current and previous frames), with the optical-flow-based score on the left and perceptual-consistency-based score on the right. We see that the measure based on our new perceptual consistency assigns higher scores to the more consistent segmentation (bottom row), whereas flow-based measure incorrectly gives higher scores to the less consistent results (top row). During the demonstration, we will provide more examples and analysis to better illustrate

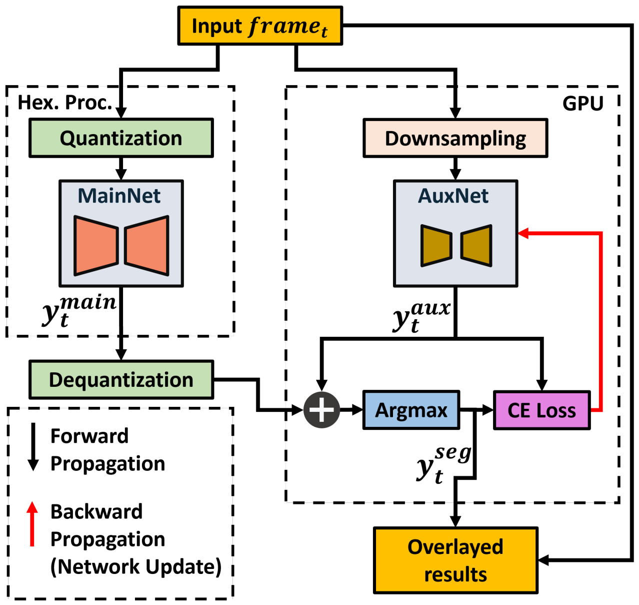


Figure 5. Cross-unit deployment of AuxAdapt for real-time application on mobile device. A quantized version of the MainNet is deployed on the Hexagon Processor (Hex. Proc.) and the AuxNet is deployed on the mobile GPU. MainNet's output is converted back to floating point before being sent to the GPU. On the GPU, the MainNet's and the AuxNet's outputs are aggregated to provide the final segmentation of a given frame. After that, the AuxNet is updated based on this combined segmentation decision via back-propagation.

the advantages of using perceptual consistency as a temporal consistency measure.

## 3. Cross-Unit Deployment on Mobile Device

We employ a novel cross-unit deployment strategy to best utilize the processing capabilities on a smartphone powered by the Qualcomm AI Engine. This platform is comprised of several hardware and software components to accelerate on-device AI-enabled user experiences. The hardware architectures supported within the AI Engine include the Qualcomm® Hexagon™ Processor,the Qualcomm® Adreno™ GPU, and the Qualcomm® Kryo™ CPU – all engineered to run AI applications quickly and efficiently on-device. The Hexagon Processor has a fused AI accelerator architecture, fusing together the scalar, vector, and tensor accelerators. This enables to provide fast fixed-point integer operations, while having a low power consumption.[4]

Since the MainNet is heavier and requires more computation operations as compared to the AuxNet, we take advantage of the Hexagon Processor and run a quantized version of the MainNet on it. This enables very fast inferences of the MainNet. Meanwhile, the smaller and less computation demanding AuxNet runs on the GPU, where it carries out the forward inferences and the back-propagations for model adaptation. By adopting such a cross-unit deployment approach, we properly utilize the available AI computation resources provided by the Qualcomm AI Engine on the smartphone. On the contrary, a simple approach of running both the MainNet and the AuxNet on the GPU would leave the powerful Hexagon Processor un-utilized and result in much slower performance. As we shall show in the demonstration, our cross-unit approach enables real-time, adaptive, accurate, and consistent video semantic segmentation.

Fig. 5 describes our cross-unit deployment strategy. More specifically, given a video frame, the quantized MainNet runs on the Hexagon Processor and generates a segmentation prediction $y^{main}$. $y^{main}$ is then converted to floating point and accessed by the GPU, which we denote as $\tilde{y}^{main}$. On the GPU, the input frame is first down-sampled before being fed to the AuxNet, which reduces the network's computation. The AuxNet then generates a segmentation prediction $y^{aux}$. Next, $\tilde{y}^{main}$ and $y^{aux}$ are summed up, and an argmax operation is applied on the sum to obtain the final segmentation decision $y^{seg}$ for the given input frame. To adapt the AuxNet, a cross-entropy loss is computed between $y^{seg}$ and $y^{aux}$, and then back-propagated through the AuxNet to update its weights.

## 4. Implementation and Demonstration Details

The demonstration will run as an Android application on a mobile phone powered by Qualcomm AI Engine. The application will take as input a video sequence and run our deployed adaptive video segmentation model to generate the corresponding segmentation maps in real time.

To deploy on the Hexagon Processor, the MainNet will be quantized using Qualcomm Innovation Center's open-source AI Model Efficiency Toolkit (AIMET) [16].[5] For online updating the AuxNet on the GPU, we use the OpenCL ML SDK [17, 18]. We further utilize several techniques to speed up the AuxNet's operation on the GPU. First, the forward passes are conducted using 16-bit floating-point numbers (FP16) and the backward passes are carried out with 32-bit floating-point numbers (FP32). Using FP16 reduces the forward inference time by half as compared to using FP32. In addition, we freeze early layers in the network to reduce the number of trainable layers, which accelerates the backward passes.

During the demonstration, we will first provide a brief overview of our technologies and innovations to be demonstrated. After that, we will show our online adaptive video

[4]Qualcomm Hexagon, Qualcomm Adreno, and Qualcomm Kryo are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

[5]AIMET is a product of Qualcomm Innovation Center, Inc.

segmentation model running real-time on the mobile phone. More specifically, we will highlight two aspects of our approach: 1) better temporal consistency and 2) a frame rate that meets real-time performance requirements. This will be done via an Android application that will display on the phone screen the input RGB video frame sequence and the consistent segmentation results generated at a high frame rate based on our deployed model, as well as another application which will display the baseline segmentation network on another phone. While running the application, we will pause the video at certain time instances to highlight example regions to showcase our improved segmentation. Additional information such as accuracy numbers, temporal consistency scores, and the real-time frame rate will also be shown. The screen recording will be provided to the audience and an interactive Q&A session will be held to answer questions.

## 5. Conclusion

In this work, we outlined our approach to demonstrate our innovations on efficient, accurate, and temporally consistent video semantic segmentation on a mobile device. We applied our test-time adaptation scheme, AuxAdapt, to enable an image segmentation model to adapt to a given video online. By running AuxAdapt, we can considerably improve the temporal consistency of video segmentation in a cost-efficient manner while maintaining the accuracy. To measure temporal consistency, we utilized our novel metric, Perceptual Consistency. We also showed the efficacy of this metric over the widely-used optical-flow-based metric. Finally, we exhibited a cross-unit deployment scheme which best utilizes the available AI processing power on a Snapdragon-powered smartphone and enables real-time operation of our adaptive video segmentation algorithm.

## References

[1] G. Floros and B. Leibe. Joint 2D-3D temporally consistent semantic segmentation of street scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 1

[2] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H.n Yang. Segflow: Joint learning for video object segmentation and optical flow. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 1

[3] M. Ding, Z. Wang, B. Zhou, J. Shi, Z. Lu, and P. Luo. Every frame counts: Joint learning of video segmentation and optical flow. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 1

[4] D. Nilsson and C. Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 4

[5] Y. Liu, C. Shen, C. Yu, and J. Wang. Efficient semantic video segmentation with per-frame inference. In *Proceedings of the European Conference on Computer Vision*, 2020. 1, 4

[6] W.-S. Lai, J.-B. Huang, O. Wang, E. Shechtman, E. Yumer, and M.-H. Yang. Learning blind video temporal consistency. In *Proceedings of the European Conference on Computer Vision*, 2018. 2

[7] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. Fully test-time adaptation by entropy minimization. In *Proceedings of the International Conference on Learning Representations*, 2021. 2

[8] C. Lei, Y. Xing, and Q. Chen. Blind video temporal consistency via deep video prior. In *Advances in Neural Information Processing Systems*, 2020. 2

[9] Yizhe Zhang, Shubhankar Borse, Hong Cai, and Fatih Porikli. Auxadapt: Stable and efficient test-time adaptation for temporally consistent video semantic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022. 2

[10] Yizhe Zhang, Shubhankar Borse, Hong Cai, Ying Wang, Ning Bi, Xiaoyun Jiang, and Fatih Porikli. Perceptual consistency in video segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022. 2, 5

[11] J. Wang, K. Sun, T. Cheng, B.i Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 3

[12] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 4

[13] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 4

[14] Serin Varghese, Yasin Bayzidi, Andreas Bar, Nikhil Kapoor, Sounak Lahiri, Jan David Schneider, Nico M Schmidt, Peter Schlicht, Fabian Huger, and Tim Fingscheidt. Unsupervised temporal consistency metric for video segmentation in highly-automated driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020. 4

[15] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 4

[16] AI Model Efficiency Toolkit (AIMET). https://quic.github.io/aimet-pages/index.html. 6

[17] OpenCL ML SDK. https://developer.qualcomm.com/blog/accelerate-your-models-our-opencl-ml-sdk. 6

[18] OpenCL ML SDK Training. https://developer.qualcomm.com/blog/ml-training-edge-training-mobile-devices. 6