BlockCopy: High-Resolution Video Processing with Block-Sparse Feature Propagation and Online Policies

Supplementary material

1. Introduction

The supplemental material contains

- a video visualizing the input/ouput, policy decisions, frame state and information gain, as shown in Figure 2 of the manuscript,
- the complete derivation of the reinforcement learning loss function given in Section 3.5 of the maniscript,
- the full numeric results of Section 4.3 of the manuscript.

2. Reinforcement learning

This section gives the full derivation of the reinforcement loss (Eq. 7 in the manuscript). The dependence on the time step t is omitted for simplicity of notation.

The policy predicts actions $\mathcal{A}$ with the goal of maximizing the expected reward, within the computational budget constraint, based on the input state $\mathcal{S}_t$. Actions $\mathcal{A}$ are given by the actions of the individual blocks b in the image:

$$\mathcal{A}_t = [a_1, \ldots, a_b, \ldots, a_B] \in \{0, 1\}^B \ .$$

With $a_b = 1$ resulting in execution of block b.

Actions should maximize the reward per image, with the objective given by

$$\max \mathcal{J}(\theta) = \max \mathbb{E}_{\mathcal{A} \sim \pi_\theta} \big[\mathcal{R}(\mathcal{A}) \big] \ . \tag{1}$$

We define the total reward as the sum of the block rewards in the image:

$$\mathcal{R}(\mathcal{A}) = \sum_{b=1}^{B} \big[\mathcal{R}_b(a_b) \big] \ . \tag{2}$$

Assuming that block actions are independent, the objective to be maximized is then given by

$$\max \mathcal{J}(\theta) = \max \sum_{b=1}^{B} \Big(\mathbb{E}_{a_b \sim \pi_{b,\theta}} \big[\mathcal{R}_b(a_b) \big] \Big) \tag{3}$$

with $\mathcal{R}_b$ the reward based on the Information Gain IG as described in Section 3.5 in the manuscript. The policy network's parameters θ can then be updated using gradient ascent with learning rate α:

$$\theta \leftarrow \theta + \alpha \nabla_\theta [\mathcal{J}(\theta)] \ . \tag{4}$$

The gradient of the objective is given by:

$$\nabla_\theta \mathcal{J}(\theta) = \nabla_\theta \sum_{b}^{B} \Big(\mathbb{E}_{a_b \sim \pi_{b,\theta}} \big[\mathcal{R}_b(a_b) \big] \Big). \tag{5}$$

The expected value over a_b is the sum of all possible values of a_b, weighted by the probability $\pi_{b,\theta}(a_b \mid \mathcal{S}_t)$:

$$\nabla_\theta \mathcal{J}(\theta) = \nabla_\theta \sum_b^B \sum_{a_b=\{0,1\}} \pi_{b,\theta}(a_b|S_t)\mathcal{R}_b(a_b). \tag{6}$$

Using the sum-rule

$$\nabla_\theta \mathcal{J}(\theta) = \sum_b^B \sum_{a_b=\{0,1\}} \nabla_\theta \pi_{b,\theta}(a_b|S_t)\mathcal{R}_b(a_b) \tag{7}$$

and the log-derivative trick

$$\nabla_\theta \mathcal{J}(\theta) = \sum_b^B \sum_{a_b=\{0,1\}} \pi_{b,\theta}(a_b|S_t) \frac{\nabla_\theta \pi_{b,\theta}(a_b|S_t)}{\pi_{b,\theta}(a_b|S_t)} \mathcal{R}_b(a_b) \tag{8}$$

results in

$$\nabla_\theta \mathcal{J}(\theta) = \sum_b^B \sum_{a_b=\{0,1\}} \pi_{b,\theta}(a_b|S_t) \nabla_\theta \log \pi_{b,\theta}(a_b|S_t)\mathcal{R}_b(a_b). \tag{9}$$

The sum over a_b can be replaced by the expectation:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_b^B \Big(\mathbb{E}_{a_b \sim \pi_{b,\theta}} \left[\nabla_\theta \log \pi_{b,\theta}(a_b|S_t)\mathcal{R}_b(a_b) \right] \Big). \tag{10}$$

In practice, the expectation is approximated using Monte-Carlo sampling. Maximizing the objective is equivalent to minimizing the loss function

$$\mathcal{L} = -\sum_{b=1}^B \left(\mathcal{R}_b(a_b) \log \pi_{b,\theta}(a_b \mid \mathcal{S}_t) \right) . \tag{11}$$

3. Semantic segmentation full results

The numeric results of Figure 7 in the manuscript can be found in Table 1. Inference times are as reported in the respective references, while also a normalized inference time is calculated based on the GPU power, in order to be equivalent of an Nvidia GTX 1080 Ti. The scaling factors to normalize inference time are given in Table 2.

References

[1] Mehwish Awan and Jitae Shin. Online keyframe selection scheme for semantic video segmentation. In *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–5. IEEE, 2020. 3

[2] Mehwish Awan and Jitae Shin. Semantic video segmentation with dense-and-dual warping spatial features. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 129–132. IEEE, 2020. 3

[3] Raghudeep Gadde, Varun Jampani, and Peter V Gehler. Semantic video cnns through representation warping. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4453–4462, 2017. 3

[4] Samvit Jain and Joseph E Gonzalez. Fast semantic segmentation on video using block motion-based feature interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 3

[5] Samvit Jain, Xin Wang, and Joseph E Gonzalez. Accel: A corrective fusion network for efficient semantic segmentation on video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8866–8875, 2019. 3

[6] Yule Li, Jianping Shi, and Dahua Lin. Low-latency video semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5997–6005, 2018. 3

[7] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6819–6828, 2018. 3

[8] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *European Conference on Computer Vision*, pages 852–868. Springer, 2016. 3

[9] Yu-Syuan Xu, Tsu-Jui Fu, Hsuan-Kung Yang, and Chun-Yi Lee. Dynamic video segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6556–6565, 2018. 3

[10] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2349–2358, 2017. 3

Table 1: Results on CityScapes validation set and comparison to other methods for video semantic segmentation. Normalized inference time is scaled by GPU power, to be equivalent to an Nvidia GTX 1080 Ti GPU.

			baseline				accelerated	
method	**GPU**	**architecture**	**mIoU**	**inference time**	**inference time (norm)**	**mIoU**	**inference time**	**inference time (norm)**
BlockCopy+DeepLab (t=0.3)	1080 Ti	DeepLabV3+ RN101	76.2%	204 ms	204ms	69.8 %	132 ms	132 ms
BlockCopy+DeepLab (t=0.5)						74.9 %	156 ms	156 ms
BlockCopy+DeepLab (t=0.7)						75.8 %	175 ms	175 ms
BlockCopy+SwiftNet (t=0.3)	1080 Ti	SwiftNet RN50	77.7%	90 ms	90ms	70.8 %	50 ms	50 ms
BlockCopy+SwiftNet (t=0.5)						76.4 %	63 ms	63 ms
BlockCopy+SwiftNet (t=0.7)						77.4 %	77 ms	77 ms
Accel-18	Tesla K80	DeepLab-RN101	75.2%	740 ms		72.1 %	440 ms	159 ms
Accel-34						72.4 %	530 ms	192 ms
Accel-50						74.2 %	670 ms	243 ms
Accel-101						75.5 %	870 ms	315 ms
LLVSS	unknown	N.A.	-	-	-	76.9 %	171 ms	-
LLVSS-LL						75.9 %	119 ms	-
DVSNet-DeepLabV2	1080 Ti	DeepLabV2	74.8 %	555 ms	555 ms	70.3 %	120 ms	120 ms
DVSNet-DeepLab-Fast		DeepLab-Fast	73.5 %	165 ms	165 ms	71.9 %	83 ms	83 ms
DVSNet-DeepLab-Fast		DeepLab-Fast	73.5 %	165 ms	165 ms	70.4 %	54 ms	54 ms
OKSS	Titan XP	-	-	-	-	74.2 %	250 ms	233 ms
Awan and Shin	Titan XP	DeepLabV3 RN50	74.9 %	670 ms	623 ms	74.1 %	340 ms	316 ms
DFF	Tesla K40		71.1 %	658 ms	292 ms	69.2 %	178 ms	79 ms
BMV-prop-mv	unknown	DeepLab	75.2%	770 ms		67.5 %	178 ms	-
BMV-prop-mv						71.2 %	290 ms	-
ClockWork	unknown	FCN	-	-	-	64.4 %	83 ms	-
NetWarp	Titan X	PSPNet-SSc	79.4 %	3000 ms	1764 ms	80.6 %	3040 ms	1788 ms
GRFP	Titan X	-	-	-		69.4 %	1700 ms	1000 ms

[5] (CVPR2019)
[6] (CVPR2018)
[9] (CVPR2018)
[1] (ICCE-Asia2020)
[2] (ICAIIC2020)
[10] (CVPR2017)
[4] (ECCVW2018)
[8] (ECCV2016)
[3] (ICCV2017)
[7] (CVPR2018)

Table 2: GPU performance and scaling factors

GPU	TFLOPS	factor
Nvidia GTX 1080 Ti	11.34	x1.0
Nvidia Titan XP	12.15	x1.07
Nvidia Titan X	6.691	x0.60
Nvidia Tesla K40	5.046	x0.44
Nvidia Tesla K80	4.113	x0.36