Delta Distillation for Efficient Video Processing Supplementary material

Amirhossein Habibian[1], Haitam Ben Yahia[1], Davide Abati[1], Efstratios Gavves[2], and Fatih Porikli[1]

[1]Qualcomm AI Research** [2]University of Amsterdam
{ahabibia,hyahia,dabati,fporikli}@qti.qualcomm.com
egavves@uva.nl

Algorithm 1: Delta distillation (training $\mathcal{L}_{dd}$)

```
// η is learning rate, globally given
function train_dd($f_\theta$, $g_\phi$, x_list)
    // compute groundtruth deltas
    z_list ← $f_\theta$(x_list);
    z_cur ← z_list[1:];
    z_past ← z_list[:-1];
    $\Delta\mathbf{z}_t$ = (z_cur - z_past).detach();
    // estimate deltas with student
    x_cur ← x_list[1:];
    x_past ← x_list[:-1];
    $\Delta\tilde{\mathbf{z}}_t = g_\phi$(x_cur, x_past);
    // optimize distillation loss
    $\mathcal{L}_{dd} = \|\Delta\mathbf{z}_t - \Delta\tilde{\mathbf{z}}_t\|_2$;
    $\phi \leftarrow \phi - \eta\frac{\nabla\mathcal{L}_{dd}}{\nabla\phi}$;
```

Algorithm 2: Delta distillation (inference)

```
// T is globally given
function inference($f_\theta$, $g_\phi$, x_list)
    z_list ← [];
    $\mathbf{x}^{t-1}, \mathbf{z}^{t-1}$ ← none, none;
    for $i \leftarrow 0$ to len(x_list) do
        $\mathbf{x}^t$ ← x_list[i];
        if $i$ % $T$ == 0 then
            // predict with teacher
            $\mathbf{z}^t \leftarrow f_\theta(\mathbf{x}^t)$;
        else
            // update with student
            $\mathbf{z}^t \leftarrow \mathbf{z}^{t-1} + g_\phi(\mathbf{x}^t, \mathbf{x}^{t-1})$;
        z_list.append($\mathbf{z}_t$);
        $\mathbf{x}^{t-1}, \mathbf{z}^{t-1} \leftarrow \mathbf{x}^t, \mathbf{z}^t$;
    return z_list
```

A Training and inference algorithms

For clarity, we report the main pseudo-code algorithms characterizing delta distillation. More specifically, Alg. 1 illustrates the optimization of delta distillation objective $\mathcal{L}_{dd}$. Furthermore, Alg. 2 describes the inference schedule alternating between teacher (providing representations) and student (updating them). We however recall that the full optimization of a delta distillation model, described in Sec. 3.3, also entails the optimization of the task objective $\mathcal{L}_t$ and of the sparsity objective $\mathcal{L}_s$, that we omit to avoid cluttering Alg. 1.

B Further segmentation experiments

B.1 Latency measurements

While the amount of GFLOPs is agnostic to hardware, software implementation, and measurement noise, it might not reflect latency gains exactly. In particular, it does not capture the memory usage overhead, which can become a limiting factor, especially for low-power devices. To this end, we hereby report the latency of several image models for semantic segmentation, as well as the one from the

** Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

Table 1: The efficiency gains by Delta Distillation translate to latency improvement in resource constrained devices.

Model	**FLOPs**		**Latency**	
	(G)	↑	**(ms)**	↑
DDRNet39	282.0	-	41.0	-
+ **Delta Distillation**	131.6	(2.1×)	16.5	(2.5×)
DDRNet23	143.7	-	22.6	-
+ **Delta Distillation**	71.3	(2.0×)	12.4	(1.8×)
HRNet-w18-small	77.9	-	39.4	-
+ **Delta Distillation**	35.7	(2.2×)	33.6	(1.2×)
DDRNet23-slim	36.6	-	6.6	-
+ **Delta Distillation**	13.7	(2.7×)	5.1	(1.3×)

corresponding delta distilled models. More specifically, we report runtime measurements on simulated hardware of a mobile SoC, as this aligns well with the practical use cases of delta distillation. Indeed, resource constrained devices represent a real-world setting where research on efficient inference methods might contribute to significantly. We illustrate such measurements in Tab. 1. In this experiment we opt, as a student architecture, for the non-linear channel reduction compression (see Sec. 3.1 in the main paper for details). Specifically, for every architecture, we compress a sequence of two residual blocks in a row into a cheaper student, with a compression ratio $\gamma = 4$. Although delta distillation increases memory usage, due to transferring inputs and outputs of every block, we appreciate how the GFLOPS gains translate to latency improvements. This behavior is more evident for heavier architectures (*i.e.* DDRNet39 and DDRNet23), where the latency speedups are in line with (or even better than) the theoretical ones. Improvements can still be appreciated when distilling cheaper architectures (*i.e.* HRNet-w18-small and DDRNet23-slim), although they become less evident.

B.2 Per class IoU results

We report in Fig. 1 the per-class IoU metric on the Cityscapes validation set, for three different variants of a DDRNet model, namely DDRNet23 slim (top), DDRNet23 (middle) and DDRNet39 (bottom). The figure also reports performances of the corresponding delta distilled variant, with students instantiated at every linear block with a compression factor $\gamma = 4$. Interestingly, the figure testifies how the original and the compressed model behave similarily on all classes. It is noteworthy how, for highly dynamic semantic classes such as cars, trains or bycicles, the student model can successfully update the representations from the teacher without any hurt in performance, showcasing that retained performances of the cheaper student model do not simply come from static classes. We remark

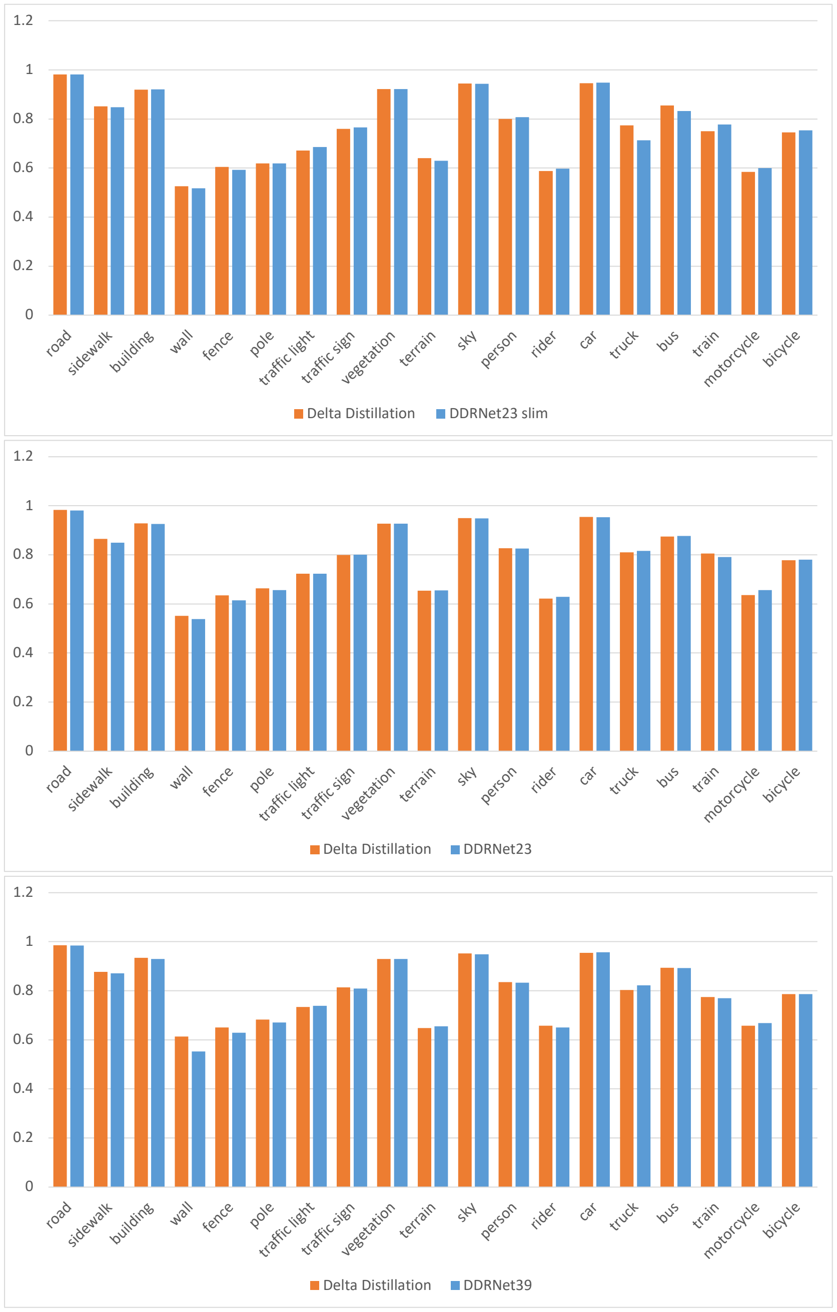

Fig. 1: Per class IoU on Cityscapes for different DDRNet models. Note that Delta distillation does not exploit any single class, but rather improves upon several or retains overall IoU at a lower cost.

how this result is achieved without relying on any explicit motion compensation during training or inference.

B.3 Temporal Consistency Results

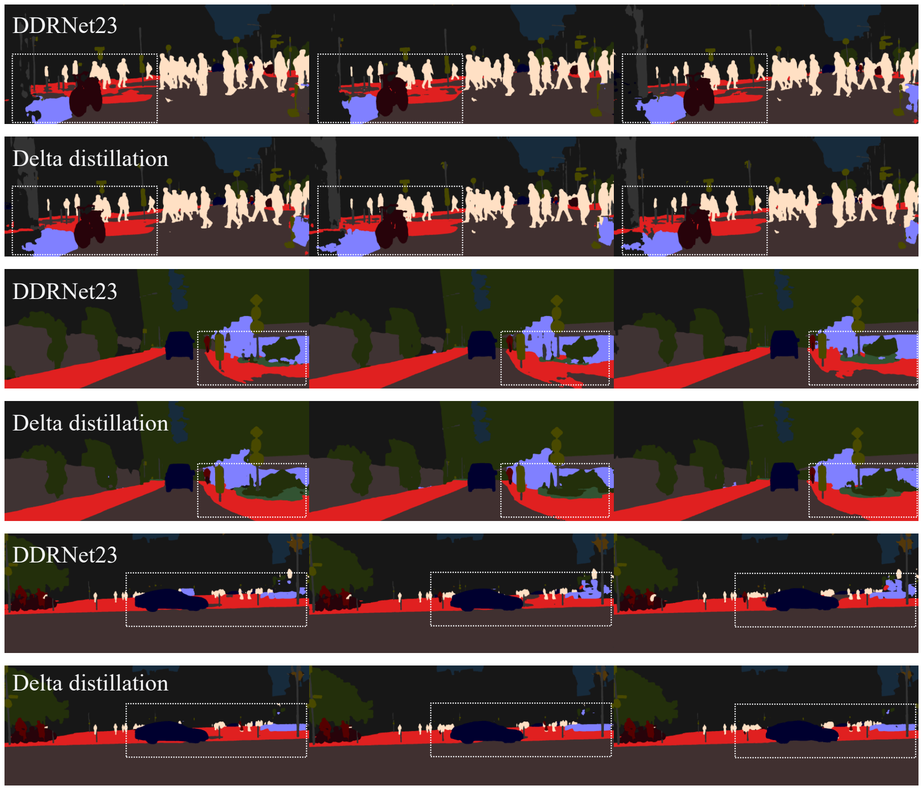

Fig. 2: Additional results of DDRNet23 and Delta Distillation in alternating rows respectively. We highlight large differences between methods in the boxes with white dashed lines. We observe that delta distillation tries to maintain large areas assigned to particular classes over time more so than the baseline model.

In Fig. 2, we illustrate further examples of the improved temporal consistency empowered by Delta Distillation. The figure, which serves as an extension of Fig. 7 in the main paper, showcases some segmentation results on the Cityscapes dataset, and it compares predictions from the original image-based architecture (DDRNet23) with the corresponding delta-distilled model. As it can be appreciated, especially by comparing predictions within the marked dashed line boxes, Delta Distillation enjoys more temporal consistency in outputs, in

Table 2: Specifications of the architectures used in Sec. 4 the main paper. We hereby report the Flops and the number of parameters separately for the teacher and student models.

Task	**Model**		**Flops (G)**	**Params (M)**	**Ref (paper)**
Obj. Detection	EfficientDet-D0	Teacher	2.5	3.84	Tab. 1
		Student	0.5	1.12	
		Amortized (T=10)	0.7	-	
Obj. Detection	Faster-RCNN	Teacher	149.1	81.22	Tab. 1
		Student	15.9	29.97	
		Amortized (T=10)	29.2	-	
Segmentation	DDRNet-39	Teacher	282.0	34.14	Tab. 2
		Student	68.9	8.09	
		Amortized (T=3)	140.0	-	
Segmentation	DDRNet-23	Teacher	143.7	20.79	Tab. 2
		Student	35.8	5.12	
		Amortized (T=3)	71.8	-	
Segmentation	DDRNet-23 slim	Teacher	36.6	5.91	Tab. 2
		Student	8.5	1.47	
		Amortized (T=3)	17.9	-	

regions where per-frame predictions result flickery. We remark that temporal consistency is not an explicit objective of Delta Distillation, but rather a beneficial side effect enabled by its inference scheme, that relies on the update of past representations, in a procedure that is inherently more temporally smooth with respect to independent processing of frames.

C Teacher and Student Parameters

In all experiments in Sec. 4 in the main paper we report amortized GMACs as computational cost measure. Therefore, the reported numbers illustrate the *average* cost required to perform a prediction and, due to the schedule we apply at inference time assigning frames unevenly to different models, they are affected by *i)* the teacher cost, *ii)* the student cost and *iii)* the schedule period T. For clarity and completeness, we report in Tab. 2 the unamortized costs of the teacher and student for the main models used in the paper. Additionally, the table reports the corresponding number of learnable parameters.

D Future work

We recognize two limitations of our work, that we plan to tackle in future work. First, delta distillation incurs some memory overhead: similarly to recurrent models, feature maps need to be stored in order to be propagated to the following time-steps. The corresponding impact on memory thus depends on the granularity at which our model operates: it might be negligible when a few (bigger) blocks are optimized, whereas it could become meaningful when the distillation involves every layer of the original model. Although we did not observe noticeable effects on latency (see Tab. 1) in our experiments, such an overhead might be impactful for large models with many layers.

Moreover, our proposal is capable of reducing the amortized runtime of the original network, yet it features unbalanced latencies between frames. Indeed, the inference schedule assigns frames either to the teacher or student model: frames processed by the former do not enjoy any latency reduction. We envision this could be addressed by deploying the two models in separate threads, and by implementing an asynchronous inference scheme similar to [1].

In conclusion, we acknowledge our proposal introduces a few additional hyperparameters into the network training schedule (*i.e.* α, β), whose determination required some trial-and-error based tinkering.

References

1. Liu, M., Zhu, M., White, M., Li, Y., Kalenichenko, D.: Looking fast and slow: Memory-guided mobile video object detection. arXiv preprint arXiv:1903.10172 (2019) 6