RAFT: Supplementary Material

Zachary Teed and Jia Deng

Princeton University
{zteed,jiadeng}@cs.princeton.edu

1 Network Architecture

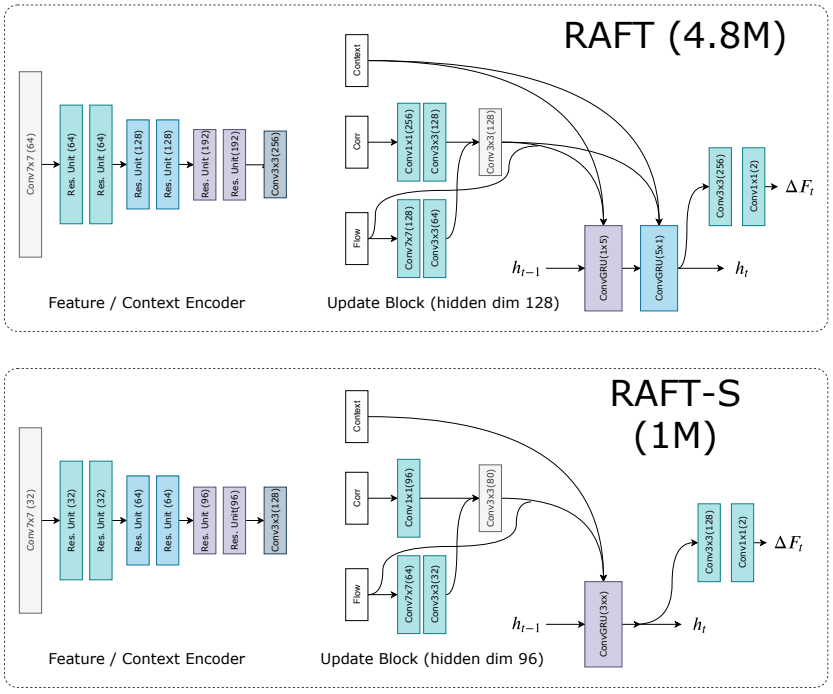

Fig. 1: Network architecture details for the full 4.8M parameter model (5.3M with upsampling module) and the small 1.0M parameter model. The context and feature encoders have the same architecture, the only difference is that the feature encoder uses instance normalization while the context encoder uses batch normalization. In RAFT-S, we replace the residual units with bottleneck residual units. The update block takes in context features, correlation features, and flow features to update the latent hidden state. The updated hidden status is used to predict the flow update. The full model uses two convolutional GRU update blocks with 1x5 filters and 5x1 filters respectively, while the small model uses a single GRU with 3x3 filters.

2 Upsampling Module

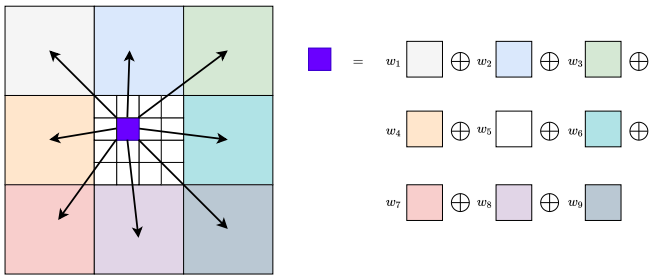

Fig. 2: Illistration of the upsampling module. Each pixel of the high resolution flow field (small boxes) is taken to be the convex combination of its 9 coarse resolution neighbors using weights predicted by the network.

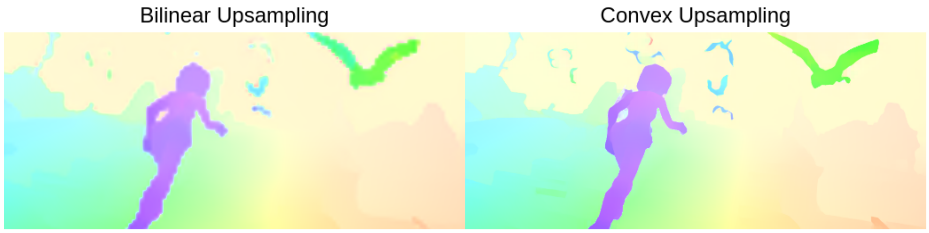

Fig. 3: Our upsampling module improves accuracy near motion boundaries, and also allows RAFT to recover the flow of small fast moving objects such as the birds shown in the figure.

3 Training Details

Stage	Weights	Training Data	Learning Rate	Batch Size (per GPU)	Weight Decay	Crop Size
Chairs	-	C	4e-4	6	1e-4	[368, 496]
Things	Chairs	T	1.2e-4	3	1e-4	[400, 720]
Sintel	Things	S+T+K+H	1.2e-4	3	1e-5	[368, 768]
KITTI	Sintel	K	1e-4	3	1e-5	[288, 960]

Table 1: Details of the training schedule. Dataset abbreviations: C: FlyingChairs, T: FlyingThings, S: Sintel, K: KITTI-2015, H: HD1K. During the sintel Finetuning phase, the dataset distribution is S(.67), T(.12), K(.13), H(.08).

Photometric Augmentation: We perform photometric augmentation by randomly perturbing brightness, contrast, saturation, and hue. We use the Torchvision `ColorJitter` with brightness 0.4, contrast 0.4, saturation 0.4, and hue

$0.5/\pi$. On KITTI, we reduce the degree of augmentation to brightness 0.3, contrast 0.3, saturation 0.3, and hue $0.3/\pi$. With probablity 0.2, color augmentation is performed to each of the images independently.

Spatial Augmentation: We perform spatial augmentation by randomly rescaling and stretching the images. The degree of random scaling depends on the dataset. For FlyingChairs, we perform spatial augmentation in the range $2^{[-0.2,1.0]}$, FlyingThings $2^{[-0.4,0.8]}$, Sintel $2^{[-0.2,0.6]}$, and KITTI $2^{[-0.2,0.4]}$. Spatial augmentation is performed with probability 0.8.

Occlusion Augmentation: Following HSM-Net [5], we also randomly erase rectangular regions in I_2 with probability 0.5 to simulate occlusions.

4 Timing, Parameters, and Training Iterations

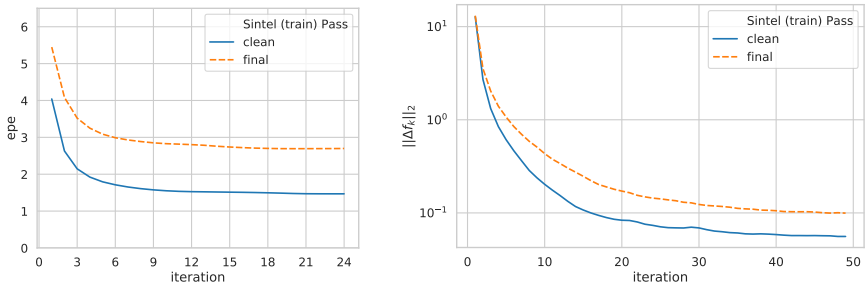

Fig. 4: (Left) EPE on the Sintel set as a function of the number of iterations at inference time. (Right) Magnitude of each update $||\Delta \mathbf{f}_k||_2$ averaged over all pixels indicating convergence to a fixed point $\mathbf{f}_k \rightarrow \mathbf{f}^*$.

Method	Parameters (M)	Time (Reported)	Time (1080Ti)	Training Iter. (#GPUs)	Accuracy
LiteFlowNetX[1]	0.9M	0.03s	-	2000k	4.79
LiteFlowNet[1]	5.4M	0.09s	0.09s	2000k	4.04
IRR-PWC[2]	6.4M	-	0.20s	850k	3.95
PWCNet+[4]	9.4M	0.03s	0.04s	1700k	3.93
VCN[6]	6.2M	0.18s	0.26s	220k(4)	3.63
FlowNet2[3]	162M	0.12s	0.11s	7000k	3.54
Ours (small)	1.0M	-	0.05s	160k(2)	3.37
Ours (mixed)	5.3M	-	0.10s	240k(1)	2.85
Ours	5.3M	-	0.10s	200k(2)	2.83

Table 2: Parameter counts, inference time, training iterations, and accuracy on the Sintel (train) final pass. We report the timing and accuracy of our method after 10 updates using a GTX 1080Ti GPU. If possible, we download the code from the other methods and re-time using our machine. If the model is trained using more than one GPU, we report the number of GPUs used to train in parenthesis. We can also train RAFT using mixed precision training Ours(mixed) and achieve similar results while training on only a single GPU. Overall, RAFT requires fewer training iterations and parameters when compared to prior work.

References

1. Hui, T.W., Tang, X., Change Loy, C.: Liteflownet: A lightweight convolutional neural network for optical flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8981–8989 (2018)
2. Hur, J., Roth, S.: Iterative residual refinement for joint optical flow and occlusion estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5754–5763 (2019)
3. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2462–2470 (2017)
4. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Models matter, so does training: An empirical study of cnns for optical flow estimation. arXiv preprint arXiv:1809.05571 (2018)
5. Yang, G., Manela, J., Happold, M., Ramanan, D.: Hierarchical deep stereo matching on high-resolution images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5515–5524 (2019)
6. Yang, G., Ramanan, D.: Volumetric correspondence networks for optical flow. In: Advances in Neural Information Processing Systems. pp. 793–803 (2019)