Optical Flow Training under Limited Label Budget via Active Learning (Appendix)

Shuai Yuan, Xian Sun, Hannah Kim, Shuzhi Yu, and Carlo Tomasi

Duke University, Durham NC 27708, USA
{shuai,hannah,shuzhiyu,tomasi}@cs.duke.edu, xian.sun@duke.edu

Table of Contents

A Methodology Details

A.1 Semi-supervised Training

In this work, we explore the spectrum between totally supervised (100% labeled) and totally unsupervised (0% labeled) training. The question is: what is the intermediate state of semi-supervised learning if we have exactly a fraction r of training samples labeled ($0 < r < 1$)? Intuitively, the performance should be monotonically increasing when we increase the label ratio r.

Specifically, the most ideal setting requires us to find a semi-supervised learning scheme that

- trains using the information of all labeled and unlabeled samples at the same time, and
- is continuous at $r = 0$ (unsupervised) and $r = 1$ (supervised), *i.e.* , if we set $r = 0$, it should be equivalent as the current unsupervised learning pipeline, and if we set $r = 1$, it should be equivalent as the fully supervised setting.

We want to define our semi-supervised setting as a smooth transition between the supervised and unsupervised settings. The naive solution is to train a fixed neural network architecture with a semi-supervised loss that works differently for the labeled and unlabeled samples. For example,

$$\ell_{\text{semi}}(\boldsymbol{x}) = \begin{cases} \ell_{\text{unsup}}(\boldsymbol{x}), \text{ if } \boldsymbol{x} \text{ is unlabeled,} \\ \alpha \ell_{\text{sup}}(\boldsymbol{x}), \text{ otherwise,} \end{cases} \tag{1}$$

where $\alpha > 0$ is the coefficient to balance the two different losses. We then have the final loss term

$$\mathcal{L}_{\text{semi}} = \sum_{\boldsymbol{x} \in \mathcal{D}} \ell_{\text{semi}}(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in \mathcal{D}^u} \ell_{\text{unsup}}(\boldsymbol{x}) + \alpha \sum_{\boldsymbol{x} \in \mathcal{D}^l} \ell_{\text{sup}}(\boldsymbol{x}),$$

where $\mathcal{D}^u$ and $\mathcal{D}^l$ are the unlabeled and labeled sample set, and $\mathcal{D} = \mathcal{D}^u \cup \mathcal{D}^l$. Under this setting, the label ratio is $r = |\mathcal{D}^l|/(|\mathcal{D}^u| + |\mathcal{D}^l|)$, and changing r from 0 to 1 will change the setting smoothly from unsupervised to supervised.

However, one concern with this setting is that we need to fix the same dataset $\mathcal{D}$ for all experiments with varying $0 \leq r \leq 1$, but the datasets used in current state-of-the-art supervised and unsupervised methods are usually different. For example, to get the best results on the Sintel dataset [1], unsupervised methods first train on the Sintel raw movie dataset and then fine-tune on Sintel. However, the latest supervised methods usually first train on the FlyingChairs [2] and FlyingThings3D [7] datasets before training on the small Sintel set. This difference in dataset is important to notice because it is one of the advantages of unsupervised learning that it can use much more data (probably from the same data distribution as the test data) than supervised training.

In light of the problem mentioned above, we decide to use the unsupervised datasets as our data in the semi-supervised training. There are mainly two reasons. First of all, the label ratio can be very low in daily practice, so defining our setting closer to the unsupervised setting may be more practical. Second, the unsupervised training is harder to converge than the supervised training because of the lack of supervisory signals, so using a framework that is closer to the unsupervised training may be better for convergence in both scenarios.

Another concern is in the training schedule. In the setting defined above, we only have one stage of training that use all labeled and unlabeled samples in the same stage. Another option is to split to two stages, one unsupervised stage using all unlabeled samples (similar as a pre-training stage) and a supervised stage using the labeled samples. We discuss the pros and cons of both schedules in Sec. B.3.

A.2 Network Choice: Why ARFlow? Why Not RAFT?

We found that RAFT is not appropriate to be tested at this stage because it has been mostly proven to work in the supervised setting, but our semi-supervised flow is actually much closer to the unsupervised setting in the following two ways.

– Our label ratio is very low (5-10%), which is almost unsupervised. The supervision signal is extremely sparse.
– Our first training stage is unsupervised, so the model is initialized in an unsupervised way.

Therefore, a reliable unsupervised base model is preferred in our setting. This is why we choose ARFlow [5] (unsupervised SOTA) instead of RAFT [11] (supervised SOTA).

Admittedly, there is recent work [10] on unsupervised versions of RAFT. However, this work is based on multi-frame inputs, and it also adds too much complexity (such as self-supervision) into the model, so we do not think it is the right time to move towards RAFT now. However, we do agree that it is worth trying in the future once a simple and reliable unsupervised appraoch for RAFT is available.

B Experiment Details

B.1 Summary of Available Datasets

Official Datasets We train and evaluate our method on two large synthetic datasets, FlyingChairs [2] and FlyingThings3D [7], as well as two more realistic datasets, Sintel [1] and KITTI [3,8].

FlyingChairs [2] and FlyingThings3D [7] consist of image pairs generated by moving chairs or everyday objects across the background images along randomized 3D trajectories. These two datasets are large but unrealistic, so they are usually only used to pre-train supervised networks.

Sintel [1] is a challenging benchmark dataset obtained from a computer-animated movie. This dataset is closer to real-life scenes as it contains fast motions, large occlusions, and many realistic artifacts like illumination change and fog or blur. It provides both clean and final passes with corresponding dense optical flow labels. Apart from that, the unlabeled raw movie frames have also been used in many recent unsupervised work [6,5].

KITTI dataset was first released in 2012 [3] and extended in 2015[8]. The dataset contains frame pairs of road scenes from a camera mounted on a car. Sparse optical flow labels are provided using 3D laser scanner and egomotion information. KITTI raw frames with no labels are also available and used in unsupervised training [9,5,12].

Tab. 1 summarizes the dataset information. We have excluded the labeled samples from the raw Sintel and KITTI dataset, so all splits in the table are disjoint.

Our Data Splits We cannot test our model using the official KITTI and Sintel test set for all the experiments since the website restricts the number of submissions. We test on the official test set only for the major final models in our paper. Thus, we need to split our own validation set from Sintel and KITTI.

Table 1. Available official datasets for optical flow estimation.

Dataset	Split	# of samples	Labeled?
FlyingChairs	train	22,232	✓
	val	640	✓
FlyingThings3D	train	19,621	✓
	val	3,823	✓
Sintel	raw	12,466	✗
	clean	1,041	✓
	final	1,041	✓
KITTI	raw	27,858	✗
	2012	194	✓
	2015	200	✓

Table 2. Our train/val split of Sintel and KITTI

Dataset	Our train split	# train samples	# val samples
Sintel clean+final	alley_1, ambush_4, ambush_6, ambush_7, bamboo_2, bandage_2, cave_2, market_2, market_5, shaman_2, sleeping_2, and temple_3	1082	1000
KITTI 2015+2012	first 150 samples for each	300	94

Our own train/val split is shown in Tab 2. For Sintel, as suggested in the official implementation of ARFlow [5], we split the following folders of both clean and final passes as our *train* split of Sintel: alley_1, ambush_4, ambush_6, ambush_7, bamboo_2, bandage_2, cave_2, market_2, market_5, shaman_2, sleeping_2, and temple_3. For KITTI, we take the first 150 samples in each of the 2015 set and 2012 set as our *train* split and the rest as our *val* split.

B.2 Data Augmentation Parameters

The data augmentation parameters in our experiments are summarized in Tab. 3. Our data augmentation implementations are borrowed from the official code base of ARFlow [5] and RAFT [11]. We use ColorJitter from the torchvision.transforms package to implement the appearance transformations. In addition, "gamma" means raising the normalized image color value (between 0 and 1) to a power sampled between 0.7 and 1.5 uniformly, and "gblur" means applying gaussian blur with radius 3 with probability 0.5.

B.3 Training Schedule Design

Three options of the training pipelines are listed below. We now explain and discuss them one by one.

A. train on *all* data (semi-sup)

Table 3. Data augmentation parameters

	FlyingChairs	FlyingThings3D	Sintel	KITTI
Cropping	384×448	384×768	384×768	320×960
Rescaling	✗	✗	scale $\in [2^{-0.2}, 2^{0.6}]$ with prob. 0.8	✗
Horizontal flip	with prob. 0.5	with prob. 0.5	with prob. 0.5	with prob. 0.5
Appearance	brightness = 0.5 contrast = 0.5 saturation = 0.5 hue = 0 gamma = `True` gblur = `True`	brightness = 0.5 contrast = 0.5 saturation = 0.5 hue = 0 gamma = `True` gblur = `True`	brightness = 0.4 contrast = 0.4 saturation = 0.4 hue = 0.16 gamma = `True` gblur = `True`	brightness = 0.3 contrast = 0.3 saturation = 0.3 hue = 0.1 gamma = `False` gblur = `True`

B. train on *all* data (unsup) → query partial labels from *all* data → train on *all* data (semi-sup)
C. train on *non-candidate* set (unsup) → query partial labels from *candidate* set → train on *candidate* set (semi-sup)

A one-stage training schedule (option A) can be used if our goal is only to visualize the change of performance when we gradually increase the label ratio from 0 to 1. We can simply train on the full dataset with partial labels using the semi-supervised loss in one stage. Thus, we use this setting in our first experiment to draw the label ratio-validation error curves. We randomly shuffle and mix labeled and unlabeled samples in mini-batches to stabilize our training. However, this assumes that the partial labels have to be assigned *before* training independent of the model and thus may be naive compared with the other two options, which use active learning.

Now, we want to explore a semi-supervised training pipeline where the labels are assigned *during* the training. This has to be a pipeline of at least two stages because we need to query labels at some point in the process. Specifically, as shown in option B, we first have a totally unlabeled dataset, so we train our first model using the unsupervised loss. Then, based on the current trained model, we pick a part of the dataset that can help the current model most to query labels. Subsequently, we continue training using the semi-supervised loss. This reflects a workflow that can be applied in real practice so that the researchers only need to pay for the partial labels that can help the most. Note that we can easily change the pipeline to query labels multiple times by stacking more stages in the end.

One problem for option B is that it assumes every sample can be labeled. However, in real life, it is possible that only a subset of the original dataset can be labeled. For instance, labeling the ground-truth flow of an autonomous driving dataset (like KITTI) requires lidar sensors deployed when the videos are collected. If a raw video does not have the corresponding lidar information, it cannot be labeled but can still be used in the unsupervised part of training. Therefore, a more general setting is to define a candidate set to indicate those

samples that can be labeled. Note that we can always split the full dataset manually to a candidate and a non-candidate splits even if every sample is eligible to get the label, which actually brings benefits in generalization as we will discuss next.

After splitting the dataset to a candidate and a non-candidate set, we can define the pipeline as in option C above. We first do unsupervised training on the non-candidate set and then use the current model to select samples out of the candidate set to get labels. This is beneficial because the model has not seen the candidate set in its first stage of unsupervised training. This can help add generalization ability because when we select samples to label, we are actually validating the current model on the new unseen candidate set. The selected samples are thus the ones that can help the current model generalize the most. This is why we stick to option C as our experiment settings in all the experiments on KITTI and Sintel.

Experiments on semi-supervised training (drawing the label ratio-validation error curves) Since our interest in this experiment is to see how the error changes when we assign different ratios of labels, we first use the simplest training schedule (option A) on two toy datasets, FlyingChairs and FlyingThings3D, to plot the whole figure. The experiment settings are as follows.

- FlyingChairs: train on the *train* split (semi-sup), evaluate on the *val* split
- FlyingThings3D: train on the *train* split (semi-sup), evaluate on the *val* split

Subsequently, we also would like to see the curve on two regular datasets, Sintel and KITTI, but since a large part of the data (raw dataset) is not labeled, we have to pre-train on those data in an unsupervised manner. This fits into the reason for specifying a candidate set, where only part of the data we have in hand are eligible to query labels. Moreover, to better fit the state-of-the-art unsupervised training schedule, we adopt option C as our training schedule. For Sintel and KITTI, we assign our *train* split as the candidate set, and the large unlabeled data (*raw* sets) as the non-candidate set[1], yielding the following training schedules.

- Sintel: train on *raw* Sintel videos (unsup) → *randomly* select and assign labels for our *train* split → train on our *train* split (semi-sup) → evaluate on our *val* split.
- KITTI: train on *raw* KITTI videos (unsup) → *randomly* select and assign labels from our *train* split → train on our *train* split (semi-sup) → evaluate on our *val* split.

Experiments on our active learning algorithms We consider many heuristics as the algorithms to select samples to label. We use Sintel and KITTI datasets and apply the same training schedule as in the previous experiments.

[1] We are not using the KITTI multi-view extension set for simplicity.

clean	final	label
c_1	f_1	l_1
c_2	f_2	l_2

(a) Sampling in pairs

clean	final	label
c_1	f_1	l_1
c_2	f_2	l_2

(b) Sampling separately

Fig. 1. Examples of two different sampling methods on Sintel

The only difference is that we now use our algorithms to select samples to label instead of random selection.

- Sintel: train on *raw* Sintel videos (unsup) → apply our *active learning algorithms* to select and assign labels for our *train* split → train on our *train* split (semi-sup) → evaluate on our *val* split.
- KITTI: train on *raw* KITTI videos (unsup) → apply our *active learning algorithms* to select and assign labels from our *train* split → train on our *train* split (semi-sup) → evaluate on our *val* split.

B.4 A Special Note on Sintel Label Queries in Pairs

When we run experiments on Sintel, the same set of labels are provided for both clean and final pass input frames, since the final pass is simply another rendering of the same content with more realistic artifacts like motion blur. In other words, we always ensure that the corresponding clean and final samples are either both labeled or both unlabeled. The reasons are as follows.

In our project, we want to investigate the trade-off between model performance and annotation costs. We use label ratio r to represent annotation cost, so we need to make sure that the total fraction of labels needed in our experiment is consistent with the label ratio r. A simple example is shown in Fig. 1. Suppose we have a tiny training set of only two clean-final pairs (c_1, f_1) and (c_2, f_2), and we set the label ratio $r = 0.5$. We have a total of four samples, so we need to select two of them to be labeled. If we sample clean and final images in pairs (as it is done in our experiments), the results may be like in Fig. 1(a), where only l_1 (a half of the label set, consistent with $r = 0.5$) is needed in the experiment.

However, if we select clean and final samples separately, the selection may be like Fig. 1(b). In this case, c_1 and f_2 are selected to be labeled, so both l_1 and l_2 are needed. This is inconsistent with $r = 0.5$ because 100% of the label set is needed, meaning that the annotation cost here is 100%. Therefore, the label ratio r here does not represent the actual annotation cost needed in the experiment. Even though the labels are not used during 100% of the training time (*e.g.* , l_1 is not used when we train with f_1), we still need to pay full cost for annotating l_1 and l_2. Thus, this alternative sampling method does not support our investigation since r does not reflect the annotation cost accurately.

Another option may be to use only one split (clean or final) for the experiments, *i.e.* , training one semi-supervised model only on the clean split and

Table 4. Validation error for semi-supervised training on different datasets

Label ratio r	FlyingChairs	Sintel	
	EPE/px	clean EPE/px	final EPE/px
0	3.066(±0.044)	1.906(±0.013)	2.933(±0.010)
0.05	2.369(±0.033)	1.850(±0.014)	2.828(±0.020)
0.1	2.091(±0.046)	1.776(±0.018)	2.710(±0.023)
0.2	1.803(±0.018)	1.691(±0.011)	2.598(±0.022)
0.4	1.653(±0.037)	1.643(±0.014)	2.349(±0.031)
0.6	1.560(±0.039)	1.625(±0.008)	2.281(±0.022)
0.8	1.550(±0.043)	1.581(±0.015)	2.281(±0.015)
1	1.439(±0.052)	1.651(±0.018)	2.290(±0.013)
Label ratio r	FlyingThings3D	KITTI	
	EPE/px	2012 Fl/%	2015 Fl/%
0	12.037(±0.500)	5.827(±0.057)	12.742(±0.090)
0.05	10.588(±0.444)	5.525(±0.038)	11.462(±0.088)
0.1	10.205(±0.694)	5.325(±0.042)	11.030(±0.128)
0.2	9.584(±0.132)	5.137(±0.050)	10.357(±0.096)
0.4	8.395(±0.307)	4.899(±0.036)	10.109(±0.087)
0.6	8.296(±0.154)	4.973(±0.049)	9.947(±0.150)
0.8	7.833(±0.152)	4.709(±0.057)	9.784(±0.134)
1	7.876(±0.283)	4.562(±0.047)	9.448(±0.134)

another model only on the final split. This also solves the problem above that the label ratio r does not reflect the true annotation cost. Nevertheless, this setting is largely different from most of the previous work, where both clean and final images are used to train one model that works on both passes at the same time. In this case, we are not able to compare with previous results. Such comparisons are crucial because we want to show that our semi-supervised models are significantly better than the state-of-the-art unsupervised models and also close to the supervised results.

C More Data and Results

C.1 Raw Validation Data

Our raw data values are shown in Tabs. 4 and 5. All pseudo error bars are obtained by taking the standard deviations in the last 50 epochs or 50k iterations. Semi-supervised training validation errors (Fig. 2 in the paper) are shown in Tab. 4, and active learning validation errors (Fig. 3 in the paper) are shown in Tab. 5.

C.2 Benchmark Qualitative Results

Some qualitative results are shown in Fig. 2. We can see that our active learning method is especially effective at hard sequences like “ambush”, “cave”, “market”

Table 5. Active learning validation errors, mean and std

Label ratio r	Method	Sintel	
		clean EPE/px	final EPE/px
0	-	1.906 (±0.013)	2.933 (±0.010)
0.05	random	1.850 (±0.014)	2.828 (±0.020)
	photo loss	1.807 (±0.010)	2.731 (±0.015)
	occ ratio	**1.767 (±0.019)**	**2.693 (±0.017)**
	flow grad norm	1.797 (±0.017)	2.770 (±0.016)
0.1	random	1.776 (±0.018)	2.710 (±0.023)
	photo loss	1.706 (±0.006)	2.541 (±0.018)
	occ ratio	**1.686 (±0.013)**	**2.515 (±0.018)**
	flow grad norm	1.696 (±0.009)	2.545 (±0.017)
0.2	random	1.691 (±0.011)	2.598 (±0.022)
	photo loss	1.639 (±0.010)	2.383 (±0.025)
	occ ratio	1.643 (±0.013)	2.373 (±0.018)
	flow grad norm	**1.631 (±0.016)**	**2.299 (±0.019)**
1	-	1.651 (±0.018)	2.290 (±0.013)
Label ratio r	Method	KITTI	
		2012 Fl/%	2015 Fl/%
0	-	5.573 (±0.056)	12.062 (±0.153)
0.05	random	5.363 (±0.080)	11.456 (±0.158)
	photo loss	5.477 (±0.032)	11.705 (±0.112)
	occ ratio	**5.256 (±0.040)**	**10.689 (±0.101)**
	flow grad norm	5.353 (±0.047)	10.994 (±0.171)
0.1	random	5.273 (±0.034)	10.480 (±0.108)
	photo loss	5.175 (±0.040)	10.441 (±0.087)
	occ ratio	5.170 (±0.043)	**10.148 (±0.110)**
	flow grad norm	**5.159 (±0.039)**	10.880 (±0.135)
0.2	random	5.021 (±0.061)	9.962 (±0.096)
	photo loss	4.934 (±0.033)	9.759 (±0.140)
	occ ratio	4.929 (±0.043)	9.736 (±0.147)
	flow grad norm	**4.837 (±0.046)**	**9.731 (±0.122)**
1	-	4.446 (±0.034)	8.545 (±0.086)

and "temple", and less effective at easy sequences where errors are already very small even for the unsupervised model. KITTI qualitative results are also shown in Fig. 3, where the differences are less visible with the naked eye.

C.3 Analysis on More Uncertainty Scores

We have also tried more metrics with heuristics defined as below.

- *flow norm*: the 2-norm of the estimated flow vectors averaged across the frame, used to reflect large motions.
- *img grad norm*: the magnitude of gradients of the input images, used to reflect edges in the scene

- *texture score*: used to evaluate whether the input images have good textures (high scores for good textures); computed based on Good Features to track [4]. We select 16*16 windows with stride=8. For each window, we compute the Z matrix for each pixel from image gradients and add them up to get a summed 2-by-2 positive semi-definite symmetric matrix. We compute the smaller eigenvalues (must positive) of the matrix for each window and take the average.
- *color change*: used to indicate illumination change. We compute the color histogram for each RGB channel as well as its cumulative distribution. We compute the distances between the cumulative distributions of the first and second frames and take average across the RGB channels.
- *param grad norm*: the norm of the loss gradients with respect to the network parameters. Intuitively, if a sample contributes large gradients to the network, it is likely that this sample does not fit well with the current network, so it may need labels.
- *max corr vol*: the maximum value of the correlation volume at each pixel averaged across the whole frame. We use the correlation volume at the second-level decoder here. Intuitively, a large maximum correlation volume means that the pixel has a good match within the window, so the error may be small.

We plot similar correlation matrices (as the ones in the last part of the main paper) with all our metrics in Fig. 4. We can see that all metrics are more or less consistent with our intuitions. Note that the "img grad norm" and "texture score" are negatively correlated with the errors because larger values indicate better textures and thus smaller estimation errors. Also, "max corr vol" is negatively correlated because larger values indicate better matches found for the first image pixels. From Fig. 4, we can see that the metrics that are more correlated with the errors are "occ ratio", "flow grad norm", "photo loss", which are then used in our experiments.

Comparing the Sintel correlation matrix (Fig. 4(a)) from that of KITTI (Fig. 4(b)), we can see that the Sintel metrics are generally more correlated with the errors, whereas KITTI metrics are generally less effective in detecting the samples of large errors. Especially for the texture related scores like "img grad norm" and "texture score", Sintel errors have correlations around 0.5, but KITTI errors are almost independent of the sample errors. We guess it may be because KITTI can already achieve pretty decent results by merely learning the flow distribution patterns (the looming motion), so it does not have to track every patch closely.

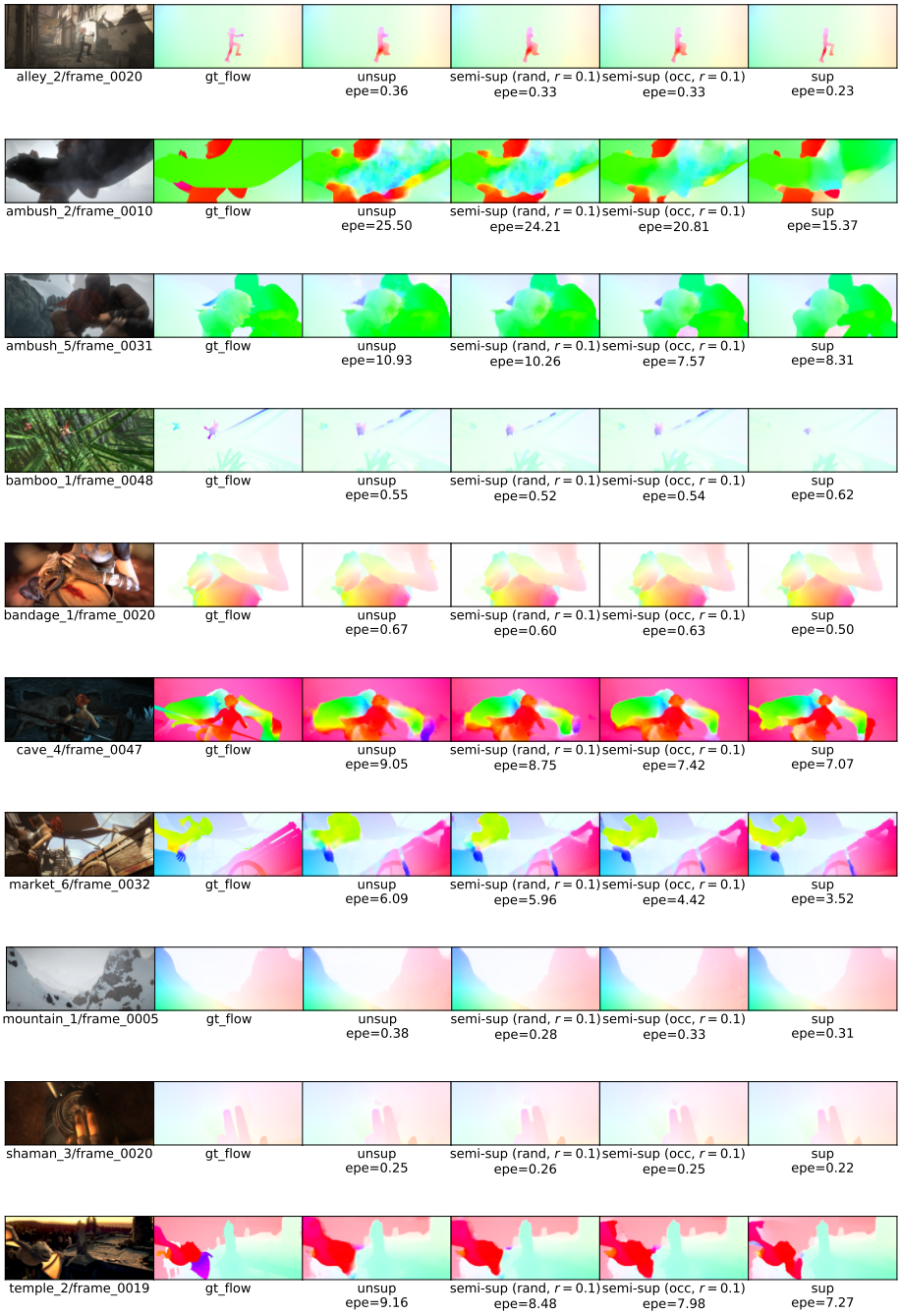


Fig. 2. Qualitative results on Sintel. Examples selected form our final pass validation split. Columns from left to right: the first frame image, the ground-truth flow, the unsupervised model prediction, the (random) semi-supervised model prediction (label ratio $r = 0.1$), our active learning model prediction (label ratio $r = 0.1$), the supervised model prediction. EPEs are shown in the subtitles.

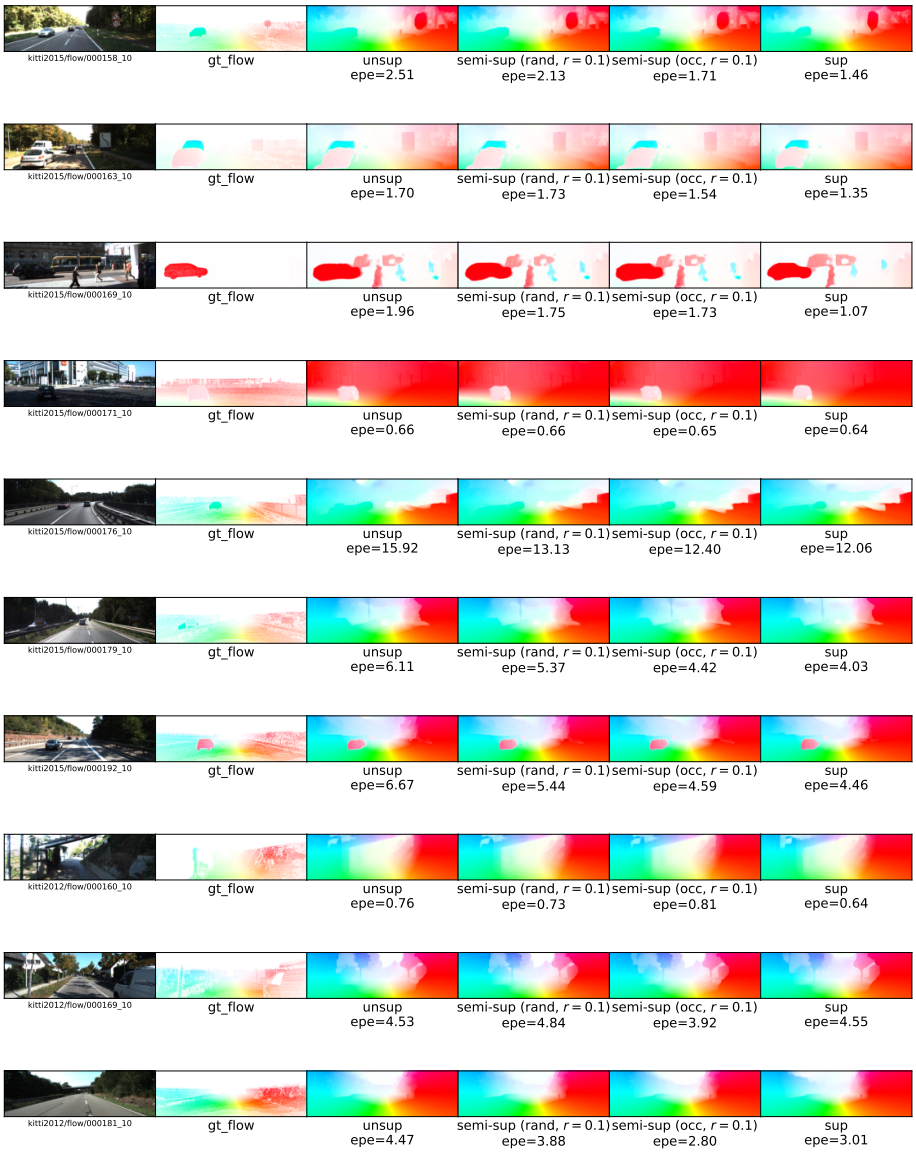
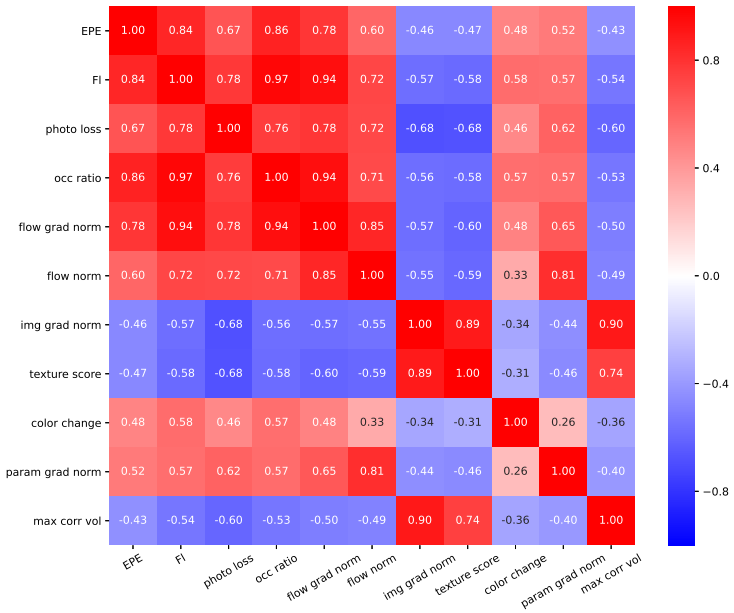
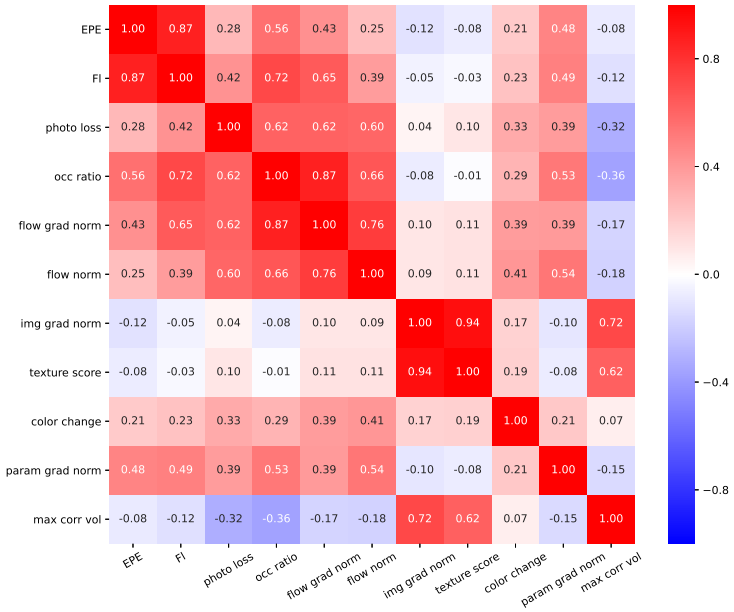


Fig. 3. Qualitative results on KITTI. Examples are selected from our validation split. Columns from left to right: the first frame image, the ground-truth flow, the unsupervised model prediction, the (random) semi-supervised model prediction (label ratio $r = 0.1$), our active learning model prediction (label ratio $r = 0.1$), the supervised model prediction. EPEs are shown in the subtitles.



(a) Sintel



(b) KITTI

Fig. 4. The correlation matrices of more active learning criteria with sample errors

References

1. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: Proceedings of the European Conference on Computer Vision. pp. 611–625. Part IV, LNCS 7577, Springer-Verlag (2012) 2, 3
2. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2758–2766 (2015) 2, 3
3. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013) 3
4. Jianbo, S., Tomasi, C.: Good features to track. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 593–600 (1994) 10
5. Liu, L., Zhang, J., He, R., Liu, Y., Wang, Y., Tai, Y., Luo, D., Wang, C., Li, J., Huang, F.: Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6489–6498 (2020) 3, 4
6. Liu, P., Lyu, M., King, I., Xu, J.: Selflow: Self-supervised learning of optical flow. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4571–4580 (2019) 3
7. Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proceedings of the IEEE International Conference on Computer Vision (2016), arXiv:1512.02134 2, 3
8. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015) 3
9. Ranjan, A., Jampani, V., Balles, L., Kim, K., Sun, D., Wulff, J., Black, M.J.: Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12240–12249 (2019) 3
10. Stone, A., Maurer, D., Ayvaci, A., Angelova, A., Jonschkowski, R.: Smurf: Self-teaching multi-frame unsupervised raft with full-image warping. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3887–3896 (2021) 3
11. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: Proceedings of the European Conference on Computer Vision. pp. 402–419. Springer (2020) 3, 4
12. Yin, Z., Shi, J.: Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1983–1992 (2018) 3