Optical Flow Models and Training Techniques in Data-Constrained Environment

Senem Isik
Stanford University
senemi@stanford.edu

Shawn Zixuan Kang
Stanford University
kangzx@stanford.edu

Zander Lack
Stanford University
zanderll@stanford.edu

Abstract

Due to the difficulties of collecting real-world optical flow data, the optical flow domain lacks any large, real-world datasets. And while large synthetic datasets have been created for proving model architectures, little work has been done to determine the best way to train and adapt these models for data-constrained scenarios. We analysed the effectiveness of data augmentation techniques and model architectures in creating models that can succeed in several of data-limited regimes. We then used those insights to create a variation of a current model architecture and test it in a data-constrained environment.

1. Introduction

Optical flow estimation aims at calculating a pixel-wise displacement field between two images. The flow field provides crucial correspondence information that has significant applications to tasks such as image interpolation, super-resolution reconstruction, object segmentation and tracking, action recognition, and autonomous navigation [14]. Specifically for the autonomous vehicle field, it can provide 3D object motion information used for scene understanding, obstacle avoidance or path planning. The main challenges facing the domain are fast-moving objects, occlusions, motion blur, and textureless surfaces [13].

Optical flow can be defined by the apparent motion of individual pixels on the image plane. It is a 2D velocity field, representing the apparent 2D image motion of pixels from the reference image to the target image. The task can be defined as follows:

Given two input images $I_1, I_2 \in R^{H \times W \times 3}$, the label, the ground truth flow field, $U \in R^{H \times W \times 2}$ describes the horizontal and vertical image motion between I_1 and I_2. Depending on the domain of the dataset and label-generating method used, the label flow can be sparse or dense. Dense flow labels give the flow for each pixel, while sparse flow labels contain an additional mask that specifics the relevant pixels. Sparse labels are often used in real-world data sets as sensors often cannot make readings over the entire image.



(a) Image 1

(b) Image 2

(c) Output

(d) Color Wheel

Figure 1. Inputs and Outputs

Optical flow estimation not only needs precise per-pixel localization but also the correspondences between two input images. This involves learning image feature representations as well as matching them at different locations in the two images. In this respect, optical flow estimation fundamentally differs from other deep learning tasks in computer vision [2].

The optical flow labels can be visualized where each color would indicate a vector in 2 dimension according to a color wheel. Above, we visualize a single example, taken from the Sintel dataset [1]. Image 1 and Image 2 are given as input to the model, which outputs a predicted flow. We can visualize the flow to capture the movement of the pixels between Image 1 and Image 2. Here, the blue hue of the flow label indicates that the frame has shifted to the left.

2. Related Work

Since the pioneering work by Horn and Schunck that formulated optical flow as a continuous optimization prob-

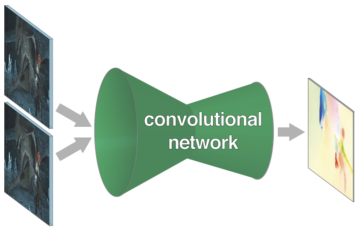


Figure 2. Deep Learning for Optical Flow

lem, various methods have dominated optical flow estimation [5]. For example, DeepFlow (2013) blended a matching algorithm that correlates multi-scale patches with a variational approach to handle large displacements [16]. Later, deep learning was shown to be able to side-step formulating an optimization problem and train a network to directly predict flow. FlowNet (2015) was the first CNN model that solved the optical flow estimation problem as a supervised learning task. It consisted of a generic CNN network with an additional layer that represents the correlation between feature vectors in different locations in the image [2]. FlowNet2 (2017) improved upon FlowNet by stacking multiple FlowNetC/S architectures and warping of the second image with intermediate optical flow. To optimally deal with small displacements, FlowNet2 introduced smaller strides into the FlowNet architecture [8]. However, FlowNet2 required over 160M parameters to achieve accurate flow estimation. Following studies attempted to create models that decrease the model size.

SpyNet (2016) combined a classical spatial-pyramid formulation with deep learning to estimate optical flow. This architecture was able to estimate large motions in a coarse-to-fine approach by warping one image of a pair at each pyramid level by the current flow estimate and updating to the flow. As a 96% smaller model than FlowNet, SpyNet introduced a new direction to the field [11].

PWC-net (2018), outperforming FlowNet, first downsampled the features in several layers and warped the features of the second image to the first as a layer with learnable weights. It then passed through a cost volume layer followed by a generic CNN. PWC-NET was 18 times smaller than FlowNet2 [12].

LiteFlownet (2018) was 30 times smaller in model size than FlowNet and also outperformed the previous models. This architecture owned a structure for pyramidal feature extraction with a more effective flow inference approach through a lightweight cascaded network and embraced feature warping rather than image warping. It also presented a novel flow regularization layer to overcome outliers and vague flow boundaries [6].

Taking inspiration from residual networks, Iterative Residual Refinement (IRR) proposed a scheme that takes optical flow output from a previous iteration and iteratively refines it by using a single network block of shared weights. Applying this scheme two networks FlowNet and PWC-Net yielded significant improvement in accuracy [7].

MaskFlowNet (2020) solved the ambiguity caused by occluded areas during warping with a matching module that learned a mask that filters useless (occluded) areas after feature warping without any explicit supervision [17]. Global Motion Aggregation (2021), on the other hand, proposed a transformer-based approach to find long-range dependencies between pixels in the first image, and perform global aggregation on the corresponding motion features to solve this ambiguity [9].

The state-of-art model Recurrent All-Pairs Field Transforms (RAFT) (2020) is a composition of CNN and RNN architectures. It maintains and updates a single fixed flow field at high resolution unlike all the previous work that uses coarse-to-fine design where flow is first estimated at low resolution and then upsampled to be refined at high resolution. Unlike the rest, it also ties the weights across iterations and, hence, the update operator of RAFT is recurrent and lightweight. It uses a novel design for its update operator as well. This model is advantageous because it does not have to recover errors at coarse resolutions, can detect small fast-moving objects, and requires less training iterations [13].

3. Problem Statement and Motivation

Given the depth of current research into model architectures, we decided to focus on a different aspect of optical flow in this paper. One current limitation of optical flow research is limited data. Getting optical flow ground truth for realistic video material is difficult [2]. There are only a few datasets and all of the larger datasets are generated with computer simulation techniques. These simulated datasets are useful for improving architectures and generally can improve a model's performance on a real-word dataset if the model is trained on sufficiently large training set. However, training the models only on a *small* portion of these *synthetic* datasets might not lead to models that generalizes well to other types of data. Thus, we will analyse the effectiveness of existing optical flow architectures specifically FlowNet and RAFT in a data-constrained environment, and use those insights to modify an existing architecture to hopefully improve its performance. Specifically, we will experiment with data augmentation and different dataset sizes to determine the effectiveness of each model.

4. Methods

4.1. Models

We chose to analyse two models: the first optical flow deep learning architecture, FlowNet-S, and current the state of the art, RAFT.

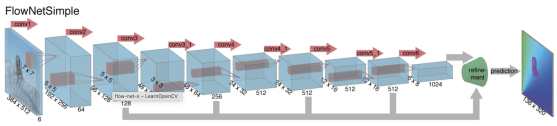


Figure 3. FlowNet-S Encoder

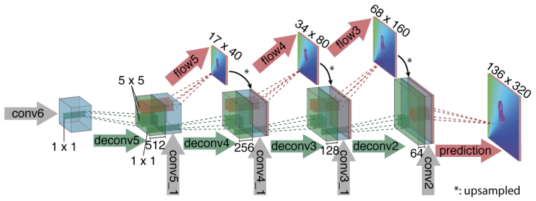


Figure 4. FlowNet-S Decoder

4.1.1 FlowNet-S

Encoder: The input two images are places into 6-channel tensor, where the first three channels belong to the first image, the three remaining channels belong to the second image. The encoder part consists of several convolution layers followed by activation functions. See Fig. 3 for the architecture [2].

Decoder: The trainable upconvolution layers upscale the encoder Optical Flow output. Each decoder stage concatenates upscaled results from the previous stage and the feature map from the corresponding layer of the encoder which helps with predicting the fine details. See Fig. 4 for the architecture [2].

Loss Function: The loss function for FlowNet is called Multi-scale training loss. The loss is calculated for every decoder stage with the impact-decreasing parameter to get the fine prediction on every decoder stage. As a result, the calculated loss for a small-sized prediction has more contribution to the total loss rather than the larger one [2]. The loss formula for every decoder stage is based on the endpoint error between prediction $V_{pred} = (\Delta x_{pred}, \Delta y_{pred})$ and ground truth data $V_{gt} = (\Delta x_{gt}, \Delta y_{gt})$:

$$L = ||V_{gt} - V_{calc}||_1 =$$

$$\sqrt{(\Delta x_{gt} - \Delta x_{calc})^2 + (\Delta y_{gt} - \Delta y_{calc})^2}$$

4.1.2 RAFT

RAFT consists of three main components:

1. a feature encoder that extracts a feature vector for each pixel from both images, along with a context encoder that extracts features from only img_1.
2. a correlation layer that constructs $4DW \times H \times W \times H$ correlation volume by taking the inner product of all pairs of feature vectors, with subsequent pooling to produce lower resolution volumes at multiple scales
3. a recurrent GRU-based update operator that takes the correlation volume, the context features, and the flow prediction to iteratively update the flow field initialized at zero

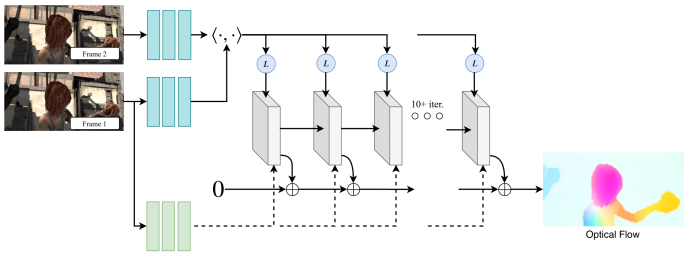


Figure 5. RAFT architecture

as shown by Fig. 5. In this paper, we will use RAFT-S, the light version of RAFT, for computational efficiency. RAFT-S has less number of parameters since it has less number of channels in the extractor, bottleneck residual units, and only one 3×3 convolution GRU cells [13].

Loss Function: All upsampled recurrent cell outputs create a sequence of the Optical Flow predictions $\mathbf{f}_1, \ldots, \mathbf{f}_N$. The total loss is the sum of losses on each recurrent block output between the ground truth and the upsampled prediction:

$$L = \sum_{i=1}^{N} \gamma^{i-N} ||gt - \mathbf{f}_i||_1, \quad \gamma = 0.8$$

Each loss function is defined as the L1 distance between ground truth and prediction, which is the same loss with FlowNet [13].

4.2. Our Contribution: Deconvolutional Upsampling for FlowNet

FlowNet decoder results in a predicted flow for which the resolution is still 4 times smaller than the input. The final result of the model is the *bilinear upsampling* of this output to the size of the imput images. Here, bilinear upsampling uses all nearby pixels to calculate the pixel's value, performing linear interpolation first in one direction, and then again in the other direction.

One of the key architectural difference of RAFT is that it maintains a single fixed flow field at high resolution throughout the architecture. Hence, it never needs to upsample.

Inspired by this, we proposed an alternative upsampling scheme for FlowNet. The prediction process in Flownet traditionally involves taking the decoder output, passing it through an additional convolutional layer and then using bilinear upsampling. Our change combines these two last



Figure 6. KITTI



Figure 7. Sintel

steps, taking the decoder output and applying a transposed convolution to simultaneously extract features and upsample the output. Hopefully, this will provide a more granular predicted flow that can achieve better performance as it uses trainable weights to upsample the flow.

We used existed codebase for FlowNet model as well as utility functions for some parts of file loading, data augmentation, and metrics calculation. We developed the new FlowNet architecture based on the existed codebase and wrote code for data augmentation, visualization, model training and testing.

4.3. Datasets and Features

4.3.1 KITTI 2015

KITTI was captured by driving around the mid-size city of Karlsruhe, in rural areas and on highways. Compared to the stereo 2012 and flow 2012 benchmarks, this KITTI dataset comprises dynamic scenes for which the ground truth has been established in a semi-automatic process [4].

4.3.2 MPI Sintel

Sintel is derived from open-source 3D animated short film Sintel. The dataset has 23 different scenes. It obtains ground truth from rendered artificial scenes with special attention to realistic image properties. For this paper's purposes, we use the clean version of the images which does not include blur and atmospheric effects. The labels provide dense ground truth for very long sequences with large motions. The images have resolution of 1024 pixels and 8-bit per channel [1].

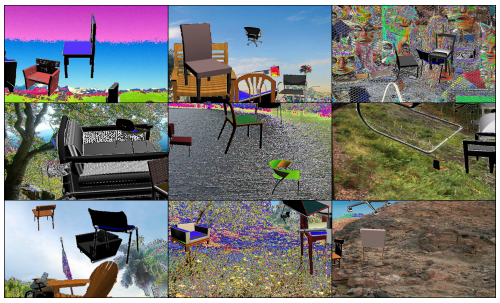


Figure 8. Flying Chairs

4.3.3 Flying Chairs

Trading in realism for quantity, Flying Chairs is a common synthetic dataset used. It consists of random background image pairs from Flickr on which segmented images of chairs are overlayed. These data have little in common with the real world but have arbitrary amounts of samples with custom properties [3].

4.3.4 Flying Things

FlyingThings3D is a synthetic dataset which consists of everyday objects flying along randomized 3D trajectories. Instead of realism, it was intended for generating orders of magnitude more data than any existing option, without running a risk of repetition or saturation [10]. The number of image pairs for training in each of these datasets is displayed in the following table.

Table 1. Dataset Sizes

Dataset	Size
KITTI	200
MPI Sintel	1,041
Flying Chairs	22,872
Flying Things	25,000

4.4. Our Contribution: Data Augmentation

A widely used strategy to improve generalization of neural networks is data augmentation. Even though the Flying Chairs dataset is fairly large, we still used augmentations to avoid overfitting in training. Specific data transformation and augmentation we used:

- Cropping all the images to be the same size and normalizing the pixels
- Randomly applying horizontal and vertical flips to images
- Randomly stretching the images to zoom in or out
- Random applying asymmetric jitter to images which changes their *brightness, contrast, saturation, and hue*

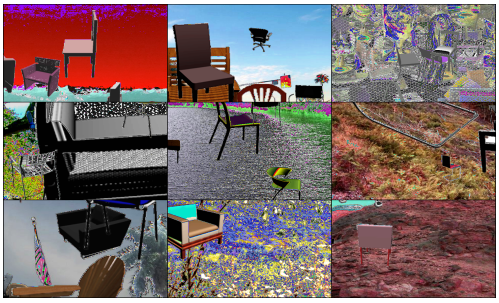


Figure 9. Flying Chairs with Data Augmentation

- Randomly erasing some section of the image (this was only applied to img_2 in the image pairs)

For each transformation, the flow is updated accordingly. The hyperparamaters we used for data augmentation are the same parameters used for the training dataset of the state of the art RAFT model [13]. Fig. 9 shows a sample of images after data augmentation. The effects of these transformations could be seen by comparing Fig. 8 and 9.

5. Experiments and Results

5.1. Experiments

To explore the performance of the models with different training methods, we pre-trained FlowNet and RAFT with 0, 500, and 2000 image pairs from Flying Chairs and fine-tuned on 520 image pairs from Sintel, with and without data augmentation respectively. We therefore did 12 different experiments on different training methods.

Our pretraining had 100 epochs on the sampled data and our fine-tuning had 20 epochs. We used a mini-batch size of 5 due to the memory constraints of our GPU. All other training hyperparameters were the same from their respective papers [8] [13].

The image pairs used from Sintel dataset for fine-tuning were the same throughout the experiments. We tested the models on the remaining 520 images in Sintel and all 200 images in KITTI. Experiment results can be found in table 2 to table 6 from the next sections.

5.2. Primary Metrics

The standard error measure for the Optical Flow task is called **EPE (End-point error)** and defined as the Euclidean distance between ground-truth $V_{gt} = (\Delta x_{gt}, \Delta y_{gt})$ and calculated $V_{calc} = (\Delta x_{calc}, \Delta y_{calc})$ optical flow values for each pixel in the image for the case of the dense optical flow estimation. The End-point error, used as a metric both for FlowNet and RAFT, is given by:

$$EPE = ||V_{gt} - V_{calc}||_2 =$$

$$\sqrt{(\Delta x_{gt} - \Delta x_{calc})^2 + (\Delta y_{gt} - \Delta y_{calc})^2}$$

In addition, in our analysis, **1px**, **3px**, and **5px** metrics will indicate the *percentage of pixels* that has EPE error less than 1, 3 and 5 respectively.

The metric **F1** is the *percentage of pixels* that are considered to be *outlier*. An outlier pixel is defined as a pixel whose EPE (End-point error) is more than 3 and *relative* EPE is more than 0.5. Relative EPE is calculated by dividing each pixel *p*'s EPE by the sqaure root of the sum of EPE scores for all pixels on the same column as pixel *p*.

5.3. Quantitative Results

5.3.1 Effect of Data Augmentation in Training FlowNet and RAFT on Datasets of Different Sizes

In tables 2, 3, 4, and 5 *size* gives the number of image pairs we used from Flying Chairs for training, while *DA* stands for Data Augmentation.

Table 2. Evaluating Flownet on Sintel

Size	DA	EPE	1px	3px	5px	F1
0	no	10.18	23.27	51.73	63.36	48.27
	yes	10.43	30.72	56.04	67.26	43.97
500	no	7.88	46.42	68.29	75.81	31.70
	yes	7.74	37.49	65.50	75.03	34.48
2000	no	6.79	**54.39**	**73.79**	**79.88**	**26.18**
	yes	**6.47**	47.69	71.24	79.03	28.72

Table 3. Evaluating Flownet on KITTI

Size	DA	EPE	1px	3px	5px	F1
0	no	8.40	52.22	80.49	83.48	19.51
	yes	**7.71**	**80.09**	**83.20**	**84.21**	**16.80**
500	no	14.66	17.51	30.35	39.43	70.00
	yes	13.76	21.30	34.20	42.32	65.80
2000	no	15.68	18.05	29.80	38.63	70.20
	yes	14.77	19.40	32.17	41.90	67.82

5.4. Analysis

From our results, we can see that even with limited amounts of pre-training data, dataset size is absolutely crucial. In almost every single case, increasing data size led to better performance from the model. The exception is FlowNet on KITTI, which could be due to the fact that the KITTI and Sintel datasets are much more similar than KITTI and Flying Chairs are, so pre-training on such an out of distribution dataset actually hurts results.

Table 4. Evaluating RAFT on Sintel

Size	DA	EPE	1px	3px	5px	F1
0	no	45.02	3.45	20.16	33.59	79.83
	yes	18.37	3.24	23.91	40.29	78.09
500	no	28.67	8.48	29.33	37.57	70.66
	yes	8.70	15.73	59.57	75.26	40.38
2000	no	38.78	14.39	32.51	42.54	67.49
	yes	**6.51**	**44.73**	**80.59**	**87.05**	**19.15**

Table 5. Evaluating RAFT on KITTI

Size	DA	EPE	1px	3px	5px	F1
0	no	70.93	1.23	8.34	15.72	91.66
	yes	18.94	2.73	17.43	31.66	82.57
500	no	37.93	2.90	15.46	23.57	84.54
	yes	**14.06**	8.71	33.74	46.23	66.25
2000	no	88.58	12.64	19.55	23.10	80.45
	yes	17.87	**15.97**	**35.53**	**45.02**	**64.44**

FlowNet also seems to be more prone to overfitting on training data, as evidenced by Fig. 14 in the Appendix. Although Flownet usually achieves better losses on the training data, RAFT does slightly better at test time.

Another interesting result was that data augmentation led to universally better performance when testing on an out-of-distribution dataset (KITTI). However, for FlowNet on Sintel, data augmentation made almost no difference to performace.

RAFT, on the other hand, is drastically improved by data augmentation in all cases in our experiments. This can also be observed from the training losses of RAFT when data augmentation is applied, see Fig. 15 in the Appendix. RAFT models overfit to Flying Chairs dataset way less than FlowNet during training both with data augmentation .

Thus, an important takeaway is that while RAFT is the state of the art method when trained on a large corpus of pre-training data (like the entire 22k flying chairs dataset), RAFT and FlowNet achieve similar performance under their respective optimal training conditions in data-constrained environments.

5.4.1 Testing FlowNet with a New Architecture

Table 6 shows the performance of the new architecture of FlowNet trained with the same parameters as described in previous experiments. Since data augmentation was shown to make little difference on FlowNet, we didn't apply data augmentation to the training data here. We compared the performance of this new model with that of the original one, both on Sintel data. In table 6, *New* corresponds to our architecture whereas *Old* corresponds to the original FlowNet model.

Table 6. Evaluating FlowNet with a new Upsampling Architecture on Sintel

Size	Model	EPE	1px	3px	5px	F1
0	New	**9.28**	**27.16**	**55.44**	**68.19**	**44.56**
	Old	10.18	23.27	51.73	63.36	48.27
500	New	7.88	**46.52**	**67.96**	75.48	32.03
	Old	7.88	46.42	68.29	**75.81**	**31.70**
2000	New	**6.80**	54.25	73.38	79.65	26.57
	Old	6.79	**54.39**	**73.79**	**79.88**	**26.18**

According to the results, the architecture change yields no significant change in performance from the model. Two models achieve nearly identical scores, though the new architecture does slightly better with less data and the traditional FlowNet does slightly better with more pretraining data.

5.5. Qualitative Results and Analysis

In qualitative analysis, we observed how well the best-trained models in our experiments before were able to generalize to diverse sets of examples. For this, we sampled image pairs from 3 diverse synthetic datasets given by 3DFlyingThings dataset: Driving, FlyingThings, and Monkaa [10].

The FlowNet model (results shown in Fig. 10) is pre-trained on 2000 Flying Chairs image pairs and fine-tuned on 520 Sintel image pairs with data augmentation and has the best performance based on our quantitative results. In Fig. 10, the three columns show the data and results sampled from Driving, Flying Things, and Monkaa datasets.

To understand the effect of data augmentation on FlowNet, Fig. 11 shows the flow results that had the most significant difference between training with and without data augmentation.

Results from RAFT model in Fig. 12 show that data augmentation improves optical flow estimation for the training setting. For example, output for driving (first row) is definitely more similar to Ground Truth when data augmentation is applied. Similar observation can be made for FlyingThings and Monkaa in second and third row.

Comparing prediction results in Fig. 10 and Fig 12., we see that RAFT outperforms FlowNet, especially when data augmentation is used in this data-constrained environment. Lastly, Fig. 13 shows the flow results from RAFT with no pre-training and only fine-tuning.

Comparing Fig. 11 with Fig. 13, we can see that, when pre-training is not possible, only with fine-tuning, FlowNet

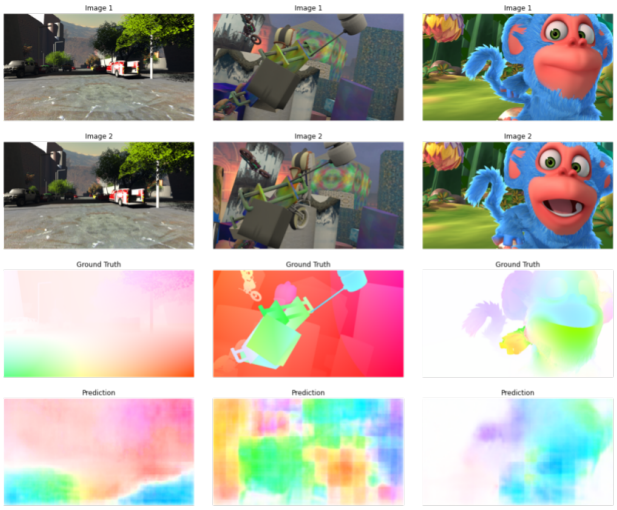


Figure 10. The input, ground truth, and output for images from Driving, FlyingThings, and Monkaa datasets respectively using FlowNet

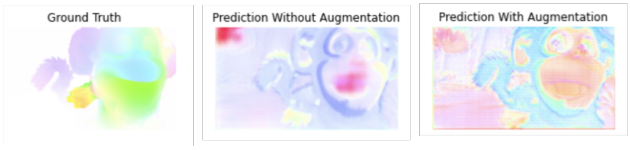


Figure 11. Ground Truth and Predictions with and without Data Augmentation of fine-tuned FlowNet on Sintel with no pre-training

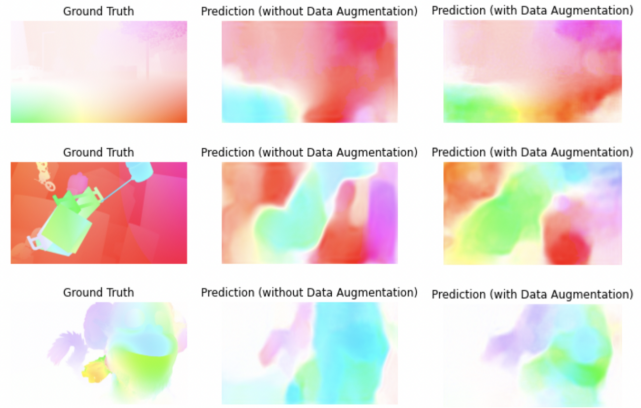


Figure 12. RAFT pre-trained on 2000 image pairs on Flying Chairs and fine-tuned on 520 image pairs Sintel with and without Data Augmentation

concentrates on the edges and segmentation that appear in the image pairs while RAFT focuses more on the flow between them.



Figure 13. Ground Truth and Predictions with and without Data Augmentation of fine-tuned RAFT on Sintel with no pre-training

6. Conclusion and Future Work

In our experiments, we found that data size is crucial to training optical flow models in data-constrained environments. We compared two models–RAFT and FlowNet–and found that they have similar performance under data-constrained environments overall, but show different performances under different specific conditions: data augmentation helps models generalize better, however, it improves RAFT significantly more than it improves FlowNet. For future work, it would be helpful to conduct similar experiments on different optical flow models (PWC-net, IRR, etc.) to explore how different models perform in data-constrained environments and how conditions including data augmentation and training data size effect their performance.

We also tested our new architecture for Flownet and compared its performance with that of the original. For future work, different aspects of the latter models could be singled out and implemented on FlowNet to understand which feature is the most crucial for improving optical flow model performances under data-constrained environments.

7. Appendices

We visualized the training loss during the training loop of each training method we experimented with for each model. 0, 500, 2000 indicate number of Flying Chairs image pairs the models were pre-trained on.

We grouped the training loop losses based on the model they belong to, whether this model was implemented with data augmentation, and how much pre-training data was used respectively.

References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 1, 4, 9

[2] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015. 1, 2, 3, 9

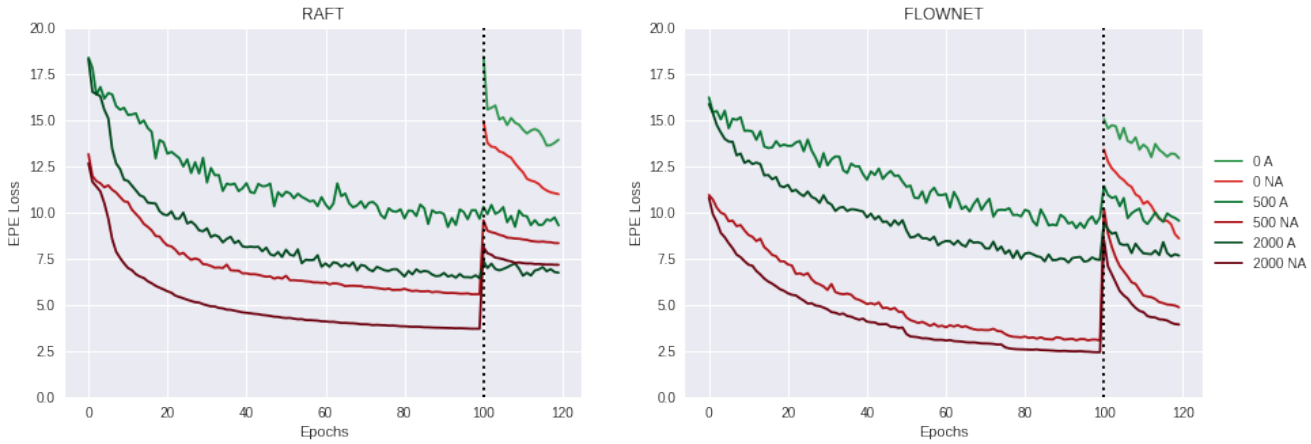


Figure 14. Training Losses by Model

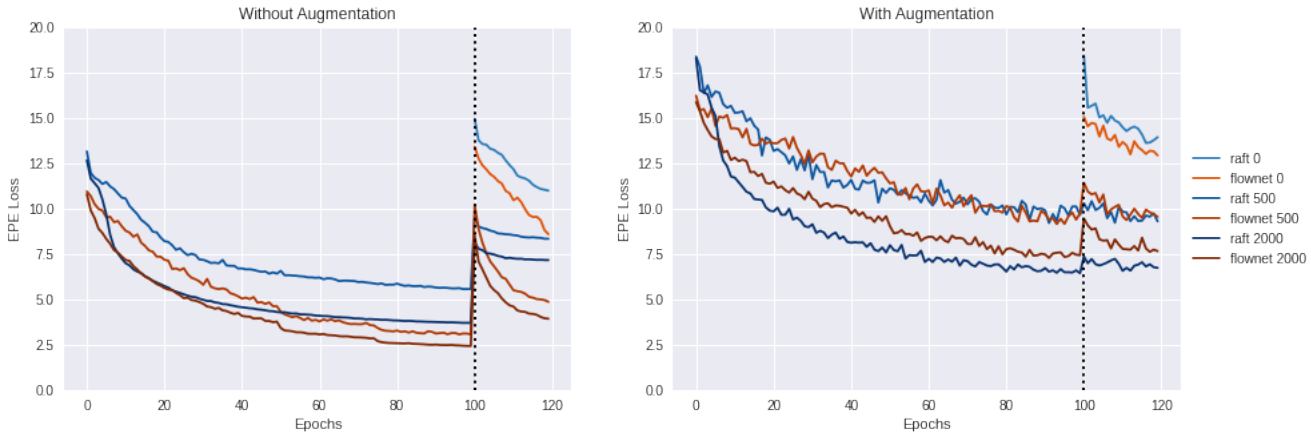


Figure 15. Training Losses by Augmentation

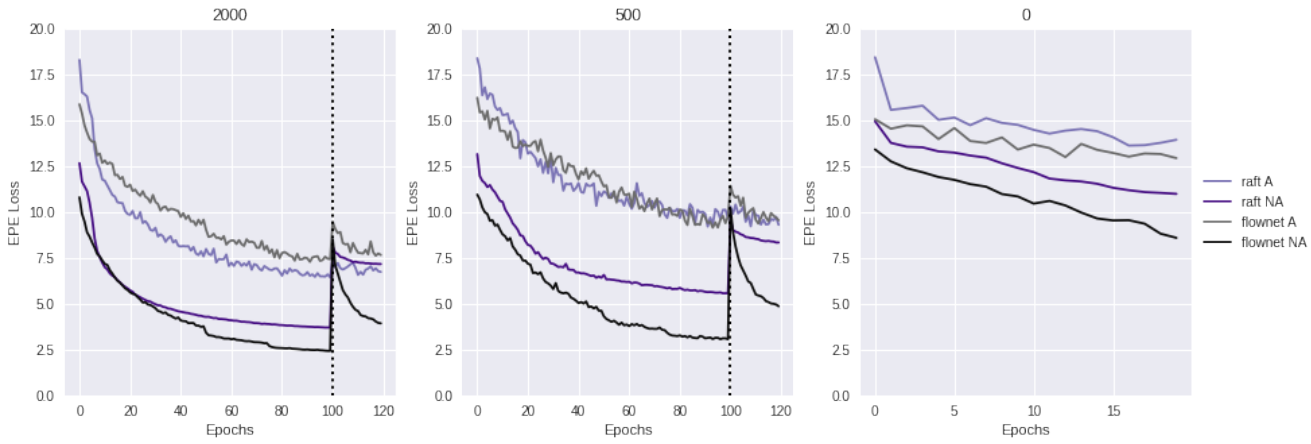


Figure 16. Training Losses by Pre-train Size

[3] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 4, 9

[4] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset, 2013. 4, 9

[5] Berthold K. P. Horn and Brian G. Schunck. Determining

optical flow. *Artif. Intell.*, 17:185–203, 1981. 2, 9

[6] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8981–8989, 2018. 2, 9

[7] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5747–5756, 2019. 2, 9

[8] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017. 2, 5, 9

[9] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard I. Hartley. Learning to estimate hidden motions with global motion aggregation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9752–9761, 2021. 2, 9

[10] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134. 4, 6, 9

[11] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2720–2729, 2017. 2, 9

[12] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 2, 9

[13] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *ArXiv*, abs/2003.12039, 2020. 1, 2, 3, 5, 9

[14] Zhigang Tu, Wei Xie, Jun Cao, Coert Van Gemeren, Ronald Poppe, and Remco C. Veltkamp. Variational method for joint optical flow estimation and edge-aware image restoration. *Pattern Recognit.*, 65:11–25, 2017. 1, 9

[15] Pavan K. Turaga, Rama Chellappa, and Ashok Veeraraghavan. Advances in video-based human activity analysis: Challenges and approaches. *Adv. Comput.*, 80:237–290, 2010. 9

[16] Philippe Weinzaepfel, Jérôme Revaud, Zaïd Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. *2013 IEEE International Conference on Computer Vision*, pages 1385–1392, 2013. 2, 9

[17] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I-Chao Chang, and Yan Xu. Maskflownet: Asymmetric feature matching with learnable occlusion mask. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6277–6286, 2020. 2, 9

[15] [2] [12] [13] [6] [8] [11] [7] [17] [9] [14] [5] [16] [4] [1] [3] [10]

Contributions & Acknowledgements

Senem Isik:

- Worked on implementing RAFT
- Researched about architecture improvements for FlowNet
- Wrote testing script to test pre-trained models on KITTI and code for visualizing datasets
- Conducted all experiments with old and new FlowNet architectures and visualized the results
- Wrote most of the report

Shawn Zixuan Kang:

- Did initial literature review
- Worked on implementing FlowNet-S architecture and researched on implementing other models
- Wrote code for visualizing optical flow for models
- Conducted some experiments with RAFT and visualized the results
- Wrote some parts of Experiments, Methods, and Conclusion in the report and helped revise and proofread as a whole

Zander Lack:

- Coded initial data handling scripts
- Implemented data augmentations and training scripts for Sintel and RAFT
- Synthesized training scripts into a single .py file for running experiments
- Conducted experiments with RAFT
- Coded changes to FlowNet architecture
- Visualized model losses
- Wrote Abstract, Motivation, and Analysis of results in the report and helped revise and proofread as a whole

Public Code used:

- GitHub repository for FlowNet
- GitHub repository for RAFT