Learning Optical Flow using Deep Neural Networks

Deepti Preta Gouthaman
New York University
dg3781@nyu.edu

Nobel Dang
New York University
nd2100@nyu.edu

Shivansh Sharma
New York University
ss14890@nyu.edu

Abstract

In this report, we have trained a variation of RAFT–Recurrent All Pairs Field Transforms for Optical Flow (Teed and Deng, 2020). The aim is to solve the problem of optical Flow using Deep Neural Networks. Optical flow is a vector representation that quantifies the displacement of an element between two frames in a video. RAFT obtains features per pixel, and constructs correlation volumes for all pairs of pixels. This is then iteratively updated to perform lookups on the correlation volumes. RAFT displays high efficiency in inference time, training time, and parameter count. The code is available at https://github.com/shiva2096/ optical_flow_with_RAFT

1 Introduction

Optical flow refers to the apparent motion of individual pixels in the images. It only pertains to the image plane (or at pixel level) and not to the real 3D world. The early solutions for optical flow trace back to Lucas-Kanade (Lucas and Kanade, 1981) and Horn-Schnuck (Horn and Schunck, 1980), who define optical flow as a small smooth field with brightness consistency assumptions.

Earlier, the implementations of Convolutional Networks were unknown to solve this problem due to pixel-level predictions. But starting from FlowNet (Fischer et al., 2015), the convolutional neural networks have become an efficient key to solving the problem. Optical flow is applied in areas where the object's motion information is crucial. It is commonly found in video editors for compression, stabilization, and slow motion. Optical flow also finds its application in Action Recognition tasks and real-time tracking systems. Optical flow estimations only track the displacement of objects from one frame to the next and not the real-world actual motion of objects. Hence, if the camera moves during the video, optical flow estimation also occurs against the background.

The main challenge faced by an optical flow problem is regularization which imposes prior on the plausibility of motion and the data that encourages the alignment of visually similar image regions. Alternatives to the traditional methods are deep learning methods like FlowNet (Fischer et al., 2015) which is the first CNN approach for calculating Optical Flow. In this approach, the network is trained directly to predict the flow.

Deep learning is significantly faster than traditional methods in terms of interface time. One of the main challenges imposed for further research is to design architectures that show great improvement in terms of performance, training, and generalization. This is where Recurrent All-Pairs Field Transforms a new deep network architecture for optical flow comes into play.

2 Approach

RAFT (Teed and Deng, 2020) model can be separated into two divisions: Encoder and iterator. The former is used to extract latent feature maps of input images and the latter is used to predict a sequence of flow steps. The iterator is similar to the RNN structure and is implemented as a Convolution gated recurrent unit module with shared parameters. Raft supports and updates a single flow field at high resolution, unlike the traditional systems where the flow is estimated at low resolution at first and then refined to high resolution. RAFT is divided into three stages which are explained below.

2.1 Feature Extraction

The input contains two consecutive frames. To extract features from these two images, we used two CNNs with shared weights. The feature extractor in RAFT uses instance normalization while the context network uses batch normalization. The Context network's feature map will be used later in the recursive blocks. This approach is like FlowNet-

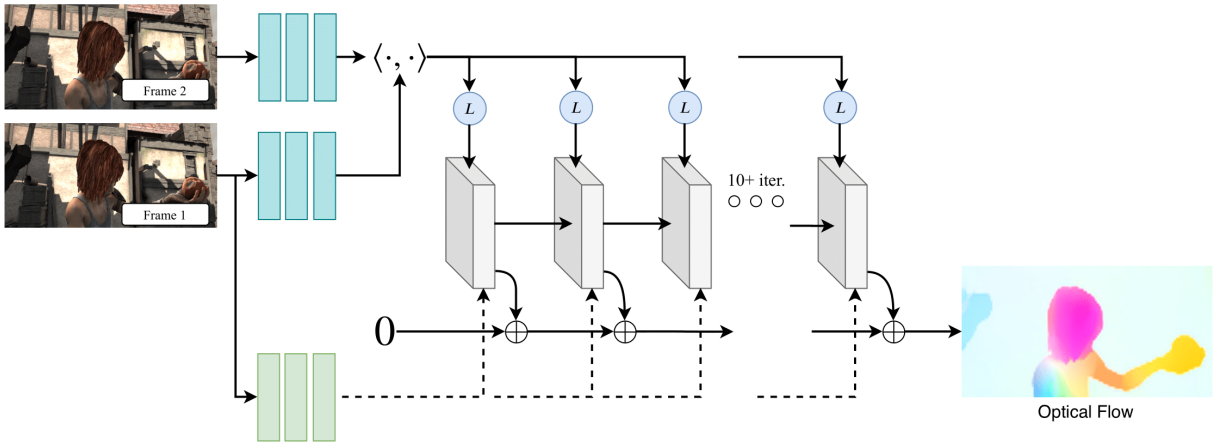

Figure 1: RAFT consists of 3 main components: (1) A feature encoder that extracts per-pixel features from both input images, along with a context encoder that extracts features from only I1. (2) A correlation layer which constructs a 4D W × H × W × H correlation volume by taking the inner product of all pairs of feature vectors. The last 2-dimensions of the 4D volume are pooled at multiple scales to construct a set of multi-scale volumes. (3) An update operator which recurrently updates optical flow by using the current estimate to look up values from the set of correlation volumes.

Corr architecture, where we extract features from two images separately.

2.2 Visual Similarity

The inner product of all pairs of feature maps is calculated as a visual similarity. The result will provide us with a 4D tensor which is also referred a correlation volume. This offers us vital information about small and large pixel movements. In the other deep learning technique FlowNetCorr, a patch-wise correlation is used. Similarly, in RAFT we calculated the all-pairs correlation between two features without any fixed window size. The correlation volume C between the two features is computed below.

$$\mathbf{C}\left(g_\theta\left(I_1\right), g_\theta\left(I_2\right)\right) \in \mathbb{R}^{H \times W \times H \times W},$$

$$C_{ijkl} = \sum_h g_\theta\left(I_1\right)_{ijh} \cdot g_\theta\left(I_2\right)_{klh}$$

We had constructed a 4-layer correlation pyramid by pooling the last two dimensions of the correlation volume with kernel sizes 1, 2, 4, and 8. The 2 dimensions of the first 3 layers is represented in Figure 2. The correlation pyramid gives more information on the small and large displacements because it is used to create image similarity on the variation scale. This is usually done to make abrupt movements more noticeable.

2.3 Iterative update

The iterative update contains a series of GRU (Gated Recurrent Unit) units that combine all the data calculations computed from before. GRU units encompass trainable convolution layers with shared weights. To predict the flow update the hidden state outputted by the GRU is passed through two convolution layers. The network outputs optical flow at 1/8 resolution. We upsample the optical flow to full resolution by taking the full resolution flow at each pixel to be the convex combination of a 3x3 grid of its coarse resolution neighbors. A classical GRU cell scheme is represented below:

$$\begin{aligned}
z_t &= \sigma\left(\mathrm{Conv}_{3\times3}\left(\left[h_{t-1}, x_t\right], W_z\right)\right) \\
r_t &= \sigma\left(\mathrm{Conv}_{3\times3}\left(\left[h_{t-1}, x_t\right], W_r\right)\right) \\
\tilde{h}_t &= \tanh\left(\mathrm{Conv}_{3\times3}\left(\left[r_t \odot h_{t-1}, x_t\right], W_h\right)\right) \\
h_t &= \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}$$

where x_t is the concatenation of flow, correlation, and context features previously defined. We also experiment with a separable ConvGRU unit, where we replace the 3×3 convolution with two GRUs: one with a 1×5 convolution and one with a 5×1 convolution to increase the receptive field without significantly increasing the size of the model.

3 Experiments

We evaluated RAFT (Teed and Deng, 2020) on Sintel (Butler et al., 2012) dataset. Here we have directly trained the model on the Sintel dataset,

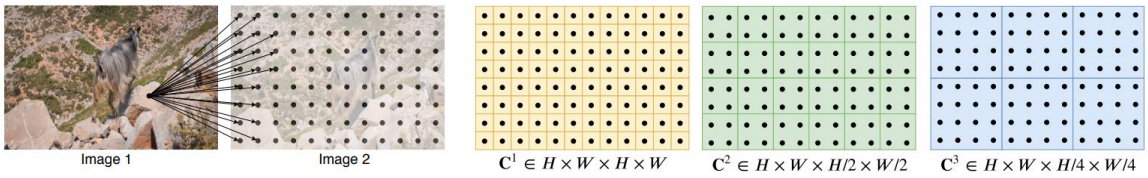

Figure 2: Building correlation volumes. Here we depict 2D slices of a full 4D volume. For a feature vector in I1, we take take the inner product with all pairs in I2, generating a 4D W ×H ×W ×H volume (each pixel in I2 produces a 2D response map). The volume is pooled using average pooling with kernel sizes 1, 2, 4, 8.

and not following the previous works where they train first on FlyingChairs (Fischer et al., 2015) and FlyingThings (Mayer et al., 2016). This is because of the time constraints.

RAFT is implemented in PyTorch (Paszke et al., 2017). All modules are initialized from scratch with random weights. During training, we use the AdamW (Loshchilov and Hutter, 2019) optimizer and clip gradients to the range $[-1, 1]$. Unless otherwise noted, we evaluate after 32 flow updates on Sintel. For every update, $\Delta f + f_k$, we only backpropgate the gradient through the Δf branch, and zero the gradient through the f_k branch as suggested by (Hofinger et al., 2020).

We train RAFT using one RTX 8000 GPUs. We train on Sintel for 25k iterations with a batch size of 5.

3.1 Sintel

The main difference from the original implementations was to train the model directly on the Sintel dataset (Butler et al., 2012). The regular method suggests to train first on FlyingChairs (Fischer et al., 2015), then on FlyingThings (Mayer et al., 2016), and then use this pretrained model to fine-tune on Sintel. However have skipped this step in order to same time and computation. Even then the results are surprisingly good.

We train our model using the schedule and then evaluate on the Sintel dataset (Butler et al., 2012) using the train split for validation. Results are shown in Table 1 and Figure 3. We tried batch sizes of 3,5 and 8, and number of iterations of 10k, 15k and 25k. The best results came out to be with batch size of 5 and iterations of 25k. However the results can be further improved by increasing the number of iterations. Training for 25k iteration took 6 hours in the RTX GPU.

Method	Sintel(clean)	Sintel(final)
FlowNetS	4.5	5.45
FlowNetC	4.31	5.87
PWC-Net	2.55	3.93
PWC-Net-ft-final	2.02	**2.08**
Ours(RAFT)	**1.86**	2.57

Table 1: Comparison of validation EPE on Sintel

4 Results

With 25k iterations and batch size set to 5, we acheived the EPE (end-point-error) of 1.86 on clean validation set, and 2.56 on final validation set. The results are fairly good considering the number of iterations we have trained the model on. Results are also shown in Table 1.

Figure 3: Left image shows the result on a sample frame from Sintel data set. Right image shows moving cars with still background.

From the left image in figure 3 we can see the model in action on Sintel dataset. Key observation is that the face of the character was not clearly identified, and the separation from the background is also not visible in the complex parts of the image. However the results are siginificantly pronounced in the right image where simpler objects like cars are moving on a still background.

From the left image in figure 4 we can see the model is pretty good at identifying moving objects with still background and is able in show the inten-

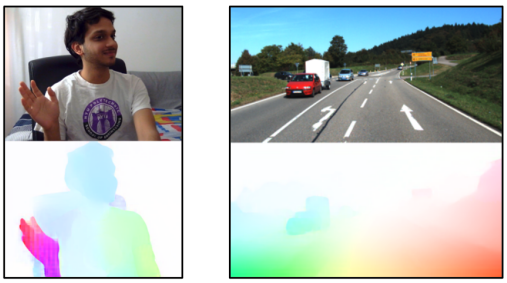

Figure 4: Left image shows the result on hand movement on a person. Right image shows moving cars with moving background.

sity of movements. The image is on a person, who has most of the movement on his hand, and RAFT is able to identify that with a dark color. However the results are siginificantly different in the right image where moving cars are captured on the from a moving car, and thus the objects that are far away (left side) shows less movement as compared to the closer objects (right side).

5 Conclusion

We have replicated RAFT—Recurrent All-Pairs Field Transforms—a new end- to-end trainable model for optical flow. RAFT is unique in that it operates at a single resolution using a large number of lightweight, recurrent update operators. Our method achieves great accuracy given the less number of training iterations and less time required to train the overall model. The trained model is able to identify moving objects not only from Sintel dataset (Butler et al., 2012), but also able to differentiate objects in images of human and cars from KITII dataset (Geiger et al., 2013). This method is efficient in terms of inference time, parameter count, and training iterations.

This method is great alternative when performing object segmentation on moving objects. The best part is, we dont even need to train the model on specific labeled dataset, like humans or cars. Just using the Sintel dataset, we are able to acheive amazing results.

Acknowledgments: We have referenced parts of code from (Teed and Deng, 2020).

References

D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. 2012. A naturalistic open source movie for optical flow evaluation. *European Conf. on Computer Vision (ECCV) 2012*.

P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. 2015. Flownet: Learning optical flow with convolutional networks. *IEEE International Conference on Computer Vision (ICCV), Dec-2015*.

A Geiger, P Lenz, C Stiller, and R Urtasun. 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research 32*.

Markus Hofinger, Samuel Rota Bulò, Lorenzo Porzi, Arno Knapitsch, Thomas Pock, and Peter Kontschieder. 2020. Improving optical flow on a pyramid level. *Computer Vision – ECCV 2020 - 16th European Conference, 2020, Proceedings*.

Berthold K.P. Horn and Brian G. Schunck. 1980. Determining optical flow. *Massachusetts Institute of Technology*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Published as a conference paper at ICLR 2019*.

Bruce D. Lucas and Takeo Kanade. 1981. An iterative image registration technique with an application to stereo vision (darpa). *Carnegie-Mellon University*.

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

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS 2017 Workshop*.

Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. *Computer Vision – ECCV 2020 - 16th European Conference, 2020, Proceedings*.