Andrej Tibenský
Ata Korkusuz
Julio Vega Sanchez

Title: A Comprehensive replication of the RAFT project

Introduction

In this blog post, we attempt to replicate the cutting-edge RAFT project. This groundbreaking research has made significant strides in the domain of optical flow estimation, garnering well-deserved attention from the AI community. However, as with any breakthrough technology, replication is a crucial step in verifying its credibility, robustness, and overall impact. Our goal is to follow the guidelines laid out in the original paper and present a detailed account of our findings, so that the reader can be confident of the inner workings of RAFT.

The RAFT project, which stands for Recurrent All-Pairs Field Transforms, has revolutionized the field of optical flow estimation. This approach leverages a combination of feature encoders using pyramids, grid mapping and GRU, resulting in a system that outperforms existing methods in both accuracy and efficiency. The significance of this breakthrough cannot be overstated, as it has the potential to transform the landscape of applications such as video analysis, robotics, and autonomous vehicles.

Our replication effort aims to provide a comprehensive exploration of the RAFT project. To achieve this, we'll dissect the original paper and delve into its inner workings, revealing the key components that make RAFT so effective. Our goal is to create an environment where fellow researchers, enthusiasts, and practitioners can learn from our experiences, and confidently engage with this exciting technology.

Before diving into the depths of our replication, it is essential to acknowledge the efforts of the original researchers and their contributions to the advancement of AI technology. The original RAFT project was proposed by Zachary Teed and Jia Deng in their paper, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow." Their work laid the foundation for this remarkable innovation.

Model Overview

The RAFT model consists of the ‘Feature /Context Encoder’ and the ‘Update Block’. In our reproduction, the contents of the Update Block was split into its components for better understandability. The initial part of the ‘Update Block’ is also called the update block by us and any further references to it will refer to just the initial part with the convolutional layers before the GRU.

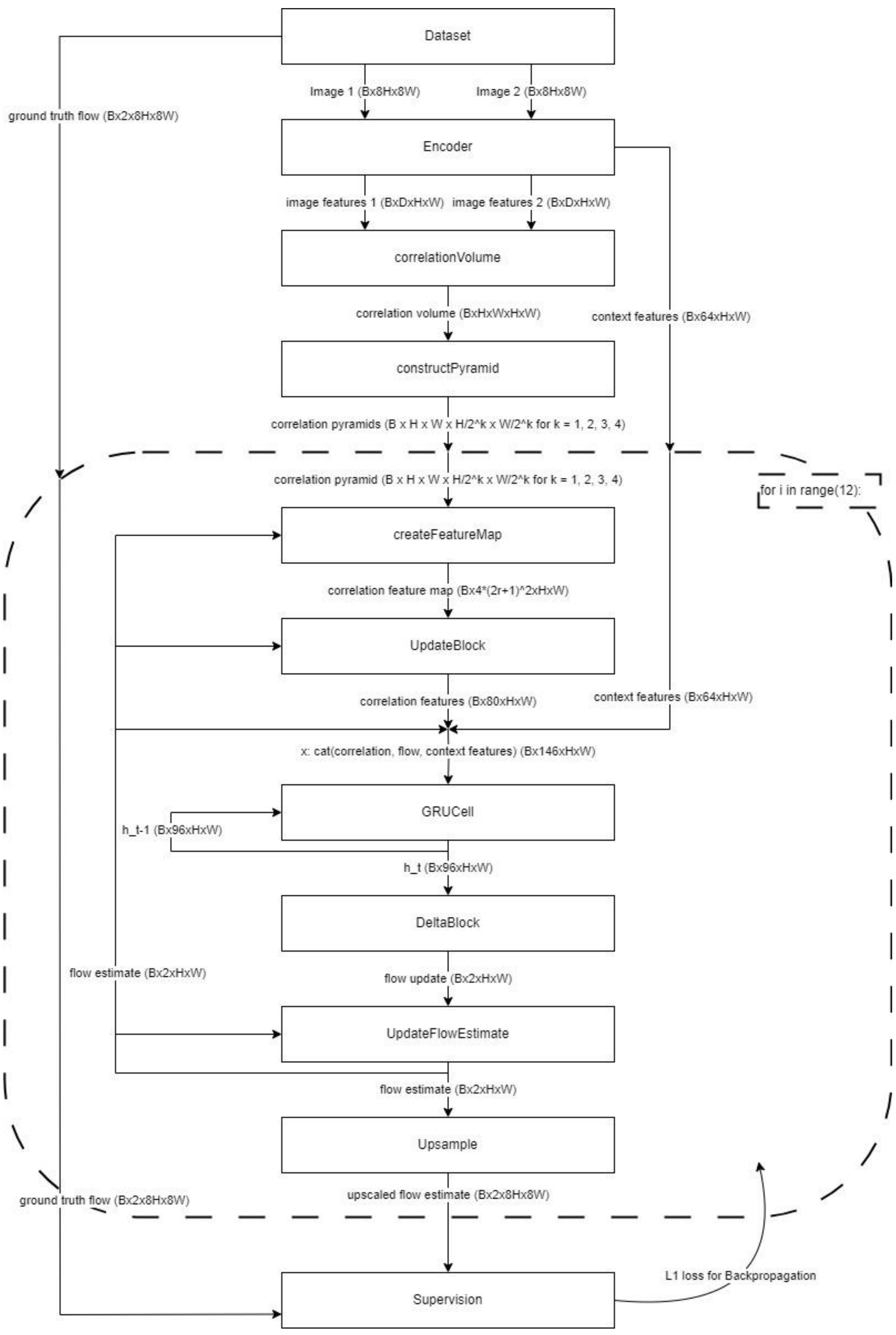

Dataset
Image 1 (Bx8Hx8W)
Image 2 (Bx8Hx8W)
ground truth flow (Bx2x8Hx8W)
Encoder
image features 1 (BxDxHxW)
image features 2 (BxDxHxW)
correlationVolume
correlation volume (BxHxWxHxW)
context features (Bx64xHxW)
constructPyramid
correlation pyramids (B x H x W x H/2^k x W/2^k for k = 1, 2, 3, 4)
correlation pyramid (B x H x W x H/2^k x W/2^k for k = 1, 2, 3, 4)
for i in range(12):
createFeatureMap
correlation feature map (Bx4*(2r+1)^2xHxW)
UpdateBlock
correlation features (Bx80xHxW)
context features (Bx64xHxW)
x: cat(correlation, flow, context features) (Bx146xHxW)
GRUCell
h_t-1 (Bx96xHxW)
h_t (Bx96xHxW)
DeltaBlock
flow estimate (Bx2xHxW)
flow update (Bx2xHxW)
UpdateFlowEstimate
flow estimate (Bx2xHxW)
Upsample
ground truth flow (Bx2x8Hx8W)
upscaled flow estimate (Bx2x8Hx8W)
Supervision
L1 loss for Backpropagation

Figure 1. RAFT reproduction architecture diagram for iteration = 12

Our reimplementation of the model contains the following sections:

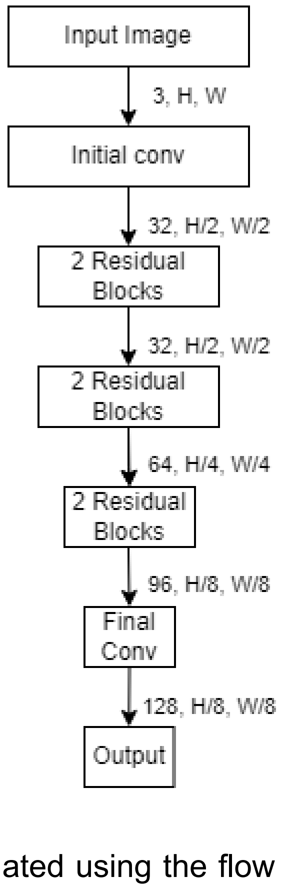
- ‘Feature/Context Encoder
  - Residual Units
  - Encoder
  - Unfold Decoder

- Computing Visual Similarity
    - Correlation Volume
    - Construct Pyramid
    - Create Feature Map
    -
- Iterative Updates
    - Update Block
    - GRU Cell
    - Delta Block

The sub components are called under the main RAFT forward function in the correct order to predict flow from two consecutive frames. Figure 1. shows the interaction and work flow of the separate components. The input and output variables with their dimensions are also given.

Model Details

Figure 2. Feature encoder



To replicate the feature encoder, we used the same architecture and hyperparameters as described in the paper's small model. This feature encoder was originally based on the ResNet architecture, shown in Figure 2, with an initial convolution, 3 pairs of Residual blocks, each at ½ the previous resolution, and a final convolution. The feature encoder was used to create the two feature maps at ⅛ resolution with 128 features. Our context encoder had the same architecture with an output of 160 features.

The next step was to construct the pyramid by performing pooling on the feature maps. The **constructPyramid** function takes in the input feature map **C** and applies average pooling to generate a pyramid of size **[C1, C2, C3, C4]**. This pyramid is then used as input to the **createFeatureMap** function, along with the flow field estimate **fe**.

The **createFeatureMap** function generates the feature map for four levels, each level with their respective dimensions. The function first loops through each level of the pyramid and calculates the local grid for each feature map. The grid is calculated using the flow field estimate **fe**, which is first permuted to have the correct dimensions. The **delta** tensor is generated using the **radius** argument, and creates two grids of the correct size, one for horizontal and one for vertical movement. By combining the **delta** and permuted **fe** tensor we create a local grid for every pixel in the flow estimate.

The grid is normalized so that it matches the current level height and width. The **grid** is then used for bilinear interpolation in which the feature maps are generated using the correlations from the correlation pyramid and the grid that indicates which pixels to sample. The feature maps are concatenated along the channel dimension and the final tensor is returned as the feature map **F**.

During the iterative part of the network the **UpdateBlock** applies two 2d convolutions to the current flow estimate of filter sizes 64 and 32 respectively. Similarly a 2d convolution of filter size 96 is applied to the correlation feature map before concatenating the output to the flow output and applying one final 2d convolution of filter size 80. The final output is the correlation features.

The **GRUCell** takes as input the previous hidden signal, which is initialized from the context features, and the concatenation of the correlation features, context features and the current flow estimate. The GRU cell has its fully connected layers replaced with convolutions instead. The gate values are given by the following equations:

$$z_t = \sigma(\mathrm{Conv3x3}([h_{t-1}, x_t], W_z))$$
$$r_t = \sigma(\mathrm{Conv3x3}([h_{t-1}, x_t], W_r))$$
$$\tilde{h}_t = \tanh(\mathrm{Conv3x3}([r_t \odot h_{t-1}, x_t], W_h))$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

where x_t is the concatenated input and h_t is the hidden input. GRU finally outputs the next hidden value and the separately calculated mask.

DeltaFBlock does two convolutions of filter size 128 and 2 on the gru hidden output to calculate the change on the flow estimate required. The downscaled flow estimate is then recalculated by adding the delta to it. The Iterative part of the network from the Update block until here is looped over for a set number of iterations. The iteration number is chosen as 12 in the paper and in our reproduction.

Finally, we upsample the flow back to full resolution. For this, we use the method described by the paper which first uses two convolutional layers to predict a 9x8x8 channel mask at 1⁄8 resolution. Next it unfolds the flow using the built-in torch unfold function. Finally, the upsampler runs softmax on the mask and calculates the value of each coordinate in the full resolution flow prediction to be the sum of the 9 surrounding coordinates in the ⅛ flow prediction.

Results

Split	EPE	1px	3px	5px
Clean (Full)	1.47	0.90	0.96	0.97
Final (Full)	2.69	0.85	0.92	0.94
Clean (Replication)	5.11	0.64	0.80	0.85
Final (Replication)	5.41	0.62	0.79	0.84

Table 1. Results of both models on the Sintel dataset

We have tested a pre-trained version of the full model which was trained on the Flying Things dataset to have a comparison to our replicated model. We have evaluated it on the SIntel dataset. As we can see from Table 1, the full model has a very low loss and is very close to the ground truth. This can also be seen from the predicted flow in Figure 3.

We have also tested our replication on the Sintel dataset. As we can see, although the results are worse than the full model, they are still comparable. They could be closer if we implemented the full model or if we trained on the Flying Things dataset for more steps. Figure 3 also compares the flow predictions of both models on a frame from the Sintel dataset. From this we can see that our replication is comparable with some small noise.

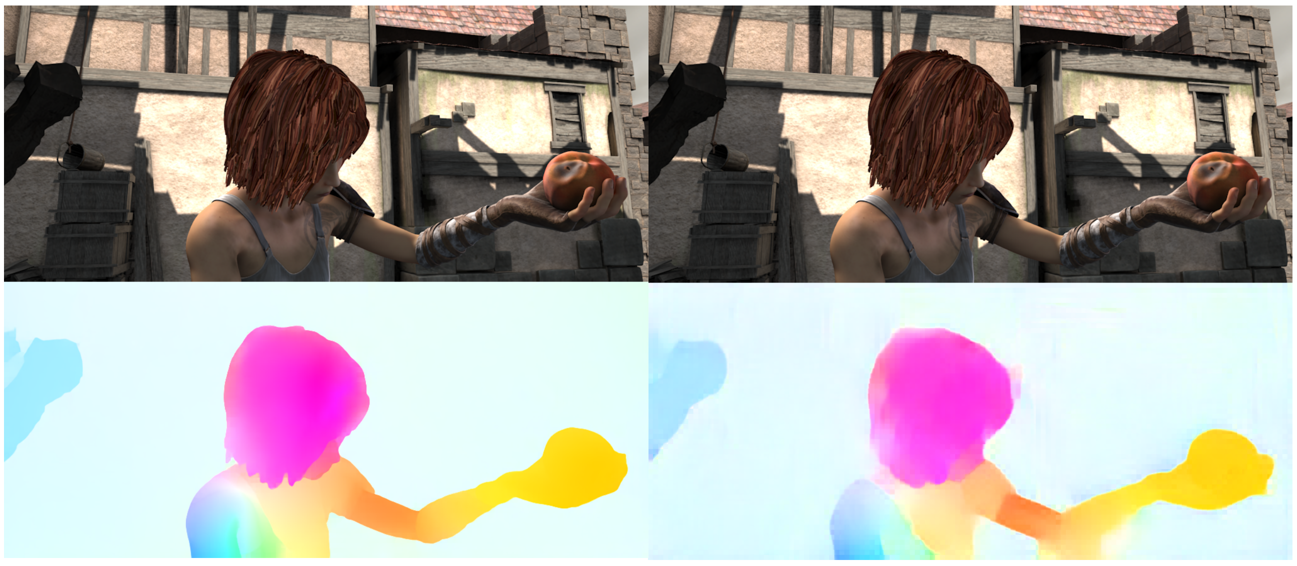


Figure 3. Flow prediction of both models (original left, reproduction right)

Conclusion

The reproduction process has shown that RAFT is still a cutting edge-model with consistent performance as presented in the original paper. We were able to reproduce the RAFT model with a comparable albeit lower performance. Ablation studies are comprehensive and well made so there is not much room to test changing inside the model.

The model still does have some small issues that makes reproducibility and interpretability relatively harder. The training time is really long. Even though the model does reach a “good enough” state before the number of steps indicated in the paper, the full training procedure for the full number of steps takes a lot of time. There is also no good way of tracking the progress of training other than displaying flow estimates. Loss values are too volatile during the epochs and even in between epochs. The combination of having to train for a long time and having to output the flow estimation for that training session just to check if the reproduction has worked can get really tiring due to how inefficient it is. Some parts of the model such as the encoder/decoder not being able to be tested individually ads to this trial and error type of troubleshooting required.

There are also some issues regarding the presentation of the model in the paper. Some model details are not explicitly given in the paper and are needed to be found from the source code or are needed to be recalculated (convolution input sizes, mask). ”Creation of the feature map” implementation is also not clearly explained. A faster implementation exists in source code but the logic for it is not explained in the paper.

Overall, reproducing the paper allowed for spotting these problems that are not apparent while just reading through the paper. Reproduction forces the reader to follow each step of the implementation by heart while requiring some domain and programming language (library, framework) specific information for the model to function. This in turn has revealed the design decision taken during the creation of the RAFT model in our project.

Task Distribution

All members focused on the **full reimplementation** with **reproduction** and **evaluation** especially on the parts where the paper was vague about the details.

Assignee	**Task**
Julio	Correlation lookup, Update Block, Delta Block
Andrej	Feature extraction, Flow upsampling, Model Training
Ata	Calculating correlation volume C, Correlation pyramid, GRU Cell