

## 1 Overview

The goal of this assignment is to implement a deep neural network for optical flow estimation. To achieve this goal, you are guided to first implement the most simple network (Sec. 3) and loss function (Sec. 4). Then you are required to implement some more complicated but practical modules (Sec. 5 & 6) to improve the accuracy. Finally, an open challenge (Sec. 7) is left for you to express your creative idea.

Before the journey, Sec. 2 introduces the background related to optical flow for preparation.

## 2 Background

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. It finds multiple applications in various video tasks, such as video coding, video inpainting and so on. An example is shown in Fig. 1.



Figure 1: Two consecutive frames  $I_0$ ,  $I_1$  and their optical flow  $I_{0 \rightarrow 1}$ . The optical flow is visualized with a color-coding  $C$ . Color values indicate the 2D movement direction.

For two consecutive frames  $I_0$  and  $I_1$  with size  $H \times W \times 3$ , their optical flow  $I_{0 \rightarrow 1}$  has size  $H \times W \times 2$ . For a point with coordinate  $(x, y)$ ,  $I_{0 \rightarrow 1}(x, y) = (\Delta x, \Delta y)$  indicates that the point  $(x, y)$  at  $I_0$  moves  $(\Delta x, \Delta y)$  pixels to the point  $(x + \Delta x, y + \Delta y)$  at  $I_1$ .

### 2.1 Reference

1. [https://en.wikipedia.org/wiki/Optical\\_flow](https://en.wikipedia.org/wiki/Optical_flow)
2. <http://sintel.is.tue.mpg.de/>

## 3 Task 1: Implement FlowNet Encoder

This section aims to implement a deep convolutional neural network called FlowNet. FlowNet is proposed in 2016 to firstly estimate optical flow with a fully convolutional network. FlowNet shows impressive performance compared with previous methods. Since then, deep learning has become more and more popular in optical flow calculation.

In this task, you are required to implement the encoder of FlowNet, as shown in Fig. 2.

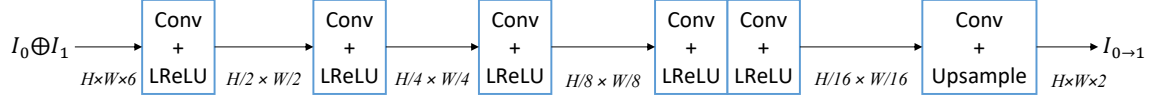


Figure 2: Architecture of FlowNet.  $\oplus$  denotes concatenation on the channel dimension. ‘LReLU’ denotes LeakyReLU.

### 3.1 To Do List:

Implement FlowNet-Encoder. (40 points)

`torch.nn.Conv2d`, `torch.nn.LeakyReLU`, `torch.nn.Interpolate` can be directly used in your implementation. You can set ‘stride=2’ in `torch.nn.Conv2d` for feature downsample and ‘scale\_factor=2’ in `torch.nn.Interpolate` for upsample. **No** other modules can be used.

You can choose different stride, channels number, kernel size, padding for each layer to boost the performance. Note that depthwise convolution or group convolution must not be used.

Your code for FlowNet-Encoder should be put in ‘networks/FlowNetE.py’.

The FLOPs of the model should be less than 2300M and parameters should be less than 5M. **All other models in this project should meet the constraints.**

### 3.2 Reference

1. <https://pytorch.org/docs/stable/nn.html>
2. <https://arxiv.org/abs/1504.06852>

## 4 Task 2: Loss Function

For the estimated optical flow  $I_{0 \rightarrow 1}$  and ground-truth optical flow  $\hat{I}_{0 \rightarrow 1}$ , the standard evaluation metric EPE (end point error) is deinfed as

$$\mathcal{L}(I_{0 \rightarrow 1}, \hat{I}_{0 \rightarrow 1}) = \frac{1}{H \times W} \sum_{(x,y)} \|I_{0 \rightarrow 1}(x, y) - \hat{I}_{0 \rightarrow 1}(x, y)\|_2. \quad (1)$$

EPE measures the derivation pixels between the ground-truth vectors and estimated vectors for optical flow. It can also be directly used as the loss function for the FlowNet.

### 4.1 Todo List:

Implement EPE loss function. **No** PyTorch function can be used. (10 points)

Your code for the loss function should be put in ‘losses.py’. After completing this part, the reference EPE is 6.35 pixels.

## 5 Task 3: Refinement Module

Optical flow estimation is a 2D dense prediction task. Directly upsample the prediction (see Fig. 2) will make the results over-smoothing. A typical technique to increase detail information is feature refinement, as shown in Fig. 3.

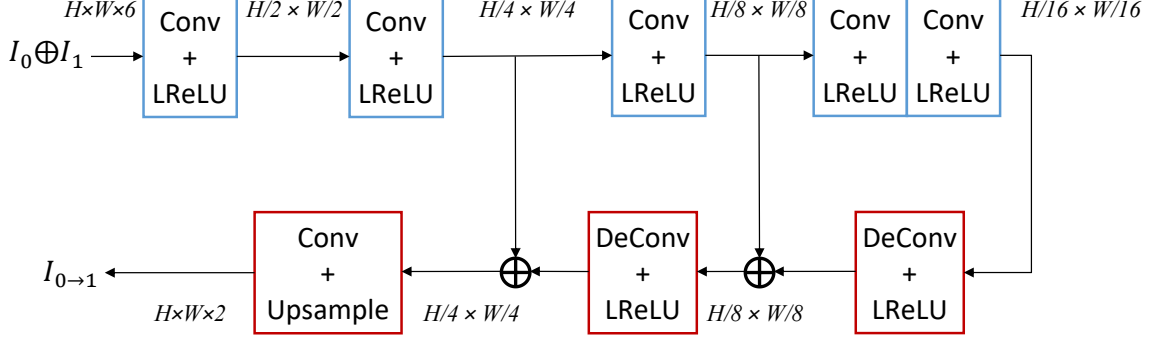


Figure 3: FlowNet with refinement module.

The deep-level feature maps are up-scaled with DeConvolution and merged with shallow-level features via concatenation. The shallow-level features contain richer texture information, which can be used to refine the optical flow prediction.

### 5.1 Todo List:

Implement the refinement module. (20 points)

`torch.nn.ConvTranspose2d` can be directly used to implement the DeConvolution operation. You can set 'stride=2' in `torch.nn.ConvTranspose2d` for feature upsample. The computational cost should be under 150M FLOPs.

Your code for FlowNet-Encoder-Refine should be put in 'networks/FlowNetER.py'. After completing this part, the reference EPE is 6.05 pixels.

### 5.2 Reference

1. <https://arxiv.org/abs/1504.06852>
2. <https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html>

## 6 Task 4: Multi-scale Optimization

In the previous pipeline, the loss function is only calculated on the final stage (see Sec. 4). In fact, the predicted optical flow can be optimized in different stages, as shown in Fig. 4. In this way, the estimated optical flow is refined explicitly.

With multi-scale prediction  $I_{0 \rightarrow 1}$ ,  $I'_{0 \rightarrow 1}$  and  $I''_{0 \rightarrow 1}$ , the multi-scale loss is defined as

$$\mathcal{L}_{ms} = \mathcal{L}(I_{0 \rightarrow 1}, \hat{I}_{0 \rightarrow 1}) + w' \mathcal{L}(I'_{0 \rightarrow 1}, \hat{I}'_{0 \rightarrow 1}) + w'' \mathcal{L}(I''_{0 \rightarrow 1}, \hat{I}''_{0 \rightarrow 1}), \quad (2)$$

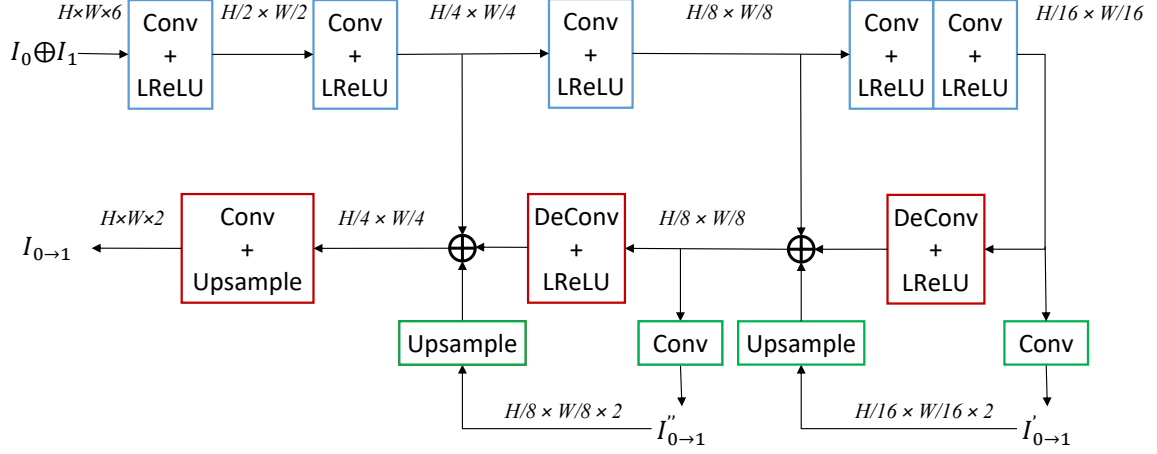


Figure 4: Architecture of FlowNet with multi-scale loss optimization.

where  $w', w'' \in [0, 1]$  are hyper-parameter to balance different loss. Since the prediction at the last stage is the most important.

## 6.1 Todo List:

Implement the new architecture and multi-scale loss function. (20 points)

For weight  $w', w''$ , you can set as constant values, functions regarding training epochs, or something reasonable else.

Your code for FlowNet-Encoder-Refine-Multiscale should be put in 'networks/FlowNetERM.py'. Your code for the multi-scale loss should be put in 'losses.py'. After completing this part, the reference EPE is 5.85 pixels.

## 6.2 Reference

1. <https://arxiv.org/abs/1504.06852>

# 7 Task 5: Open Challenge

In this task, you are required to conduct more techniques on optical flow estimation. The lower EPE value, the better. Any other tricks can be plugged in your algorithm, like data augmentations, dilated convolution or two-branch encoders.

## 7.1 Todo List:

Improve your algorithm with any tricks to make the EPE lower than 5.6 pixels. (10 points)

Your code for this part should be put in the 'open\_challenge' folder.

This is an open question, you can change the input transformation, network, and losses anyway you want. **Not** allowed to change the training epochs or learning policy. The only constrain is FLOPs and parameters of model. Read the following papers for inspiration.

## 7.2 Reference

1. <https://arxiv.org/abs/1612.01925>
2. <https://arxiv.org/abs/1606.00915>
3. <https://arxiv.org/abs/2003.12039>

## 8 General Specification

You are required to complete all functionalities according to the specification with Python. Skeleton programs are given. Do not change the files' name or the functions' name. You are **NOT** allowed to implement your program in another language (e.g. Matlab/C++) and then initiate system calls or external library calls in Python. Your source codes will be compiled and PERUSED, and the object code tested! You should not add new files.

The folder's name, filename and code can not contain any Chinese.

- **NO PLAGIARISM!**

You are free to design your algorithm and code your implementation, but you should not "borrow" codes from your classmates. If you use an algorithm or code snippet that is publicly available or use codes from your classmates or friends, be sure to DECLARE it in the comments of your program. Failure to comply will be considered as plagiarism.

## 9 Submission Guidelines

The folder you hand in must contain the following:

- **report.txt** - containing anything about the project that you want to tell the TAs, including a brief introduction of the usage of the codes.
- **code/** - directory containing all your code for this assignment, which is expected to have five .py files as shown in the skeleton.

Rename the folder as <your student ID>-Asgn4, and compress it into <your student ID>-Asgn4.zip, and upload it to blackboard system. (For example, 1155123456-Asgn4/ and 1155123456-Asgn4.zip, pay attention to the name.)

Please read the guidelines CAREFULLY. If you fail to meet the deadline because of a submission problem on your side, marks will still be deducted. The late submission policy is as follows:

- 1 day late: -20 marks
- 2 days late: -40 marks
- 3 days late: -100 marks

**Pay attention to the format before. 10% deduction for every wrong format(file name, function name, etc.).**