## 1. Accessing the data

In this exercise, we will use two common optical flow datasets: FlyingThings and Sintel. They are provided under `/project/cv-ws2122/shared-data1` and should be accessible.

## 2. Optical Flow

Optical flow is the task of estimating the motion between two consecutive frames and it is a regression task since the output is a motion field $(u, v)$ for every pixel of the input image. In this exercise, we will use the common FlowNet which has a U-Net architecture and consists of an encoder and a decoder. The encoder takes the two input images and learns a feature representation of them, while the decoder learns to construct the optical flow field for all pixels of the image, see https://arxiv.org/pdf/1504.06852.pdf for more details.

Given the true labels, we can train the network in a supervised manner. Obtaining optical flow ground truth is feasible for synthetic datasets like FlyingChairs and FlyingThings. However, in real dataset, this is very expensive (i.e., we need a motion field for every pixel of the image). Although we can train networks on synthetic datasets, we are more interested in the performance of the network on real data. This is an issue commonly known as the gab between synthetic and real domains.

It is a common practice to train first on synthetic dataset to learn the general concepts of finding correspondence between the two frames and later fine-tune on a real dataset. In this exercise, we already provide a pretrained model that has been train on the synthetic dataset FlyingThings. The main objective of the exercise is to explore two different approaches during the finetuning phase: supervised and unsupervised training.

**Todo:** Implement the function to estimate the evaluation metric functions (*aepe* and *pointwise_ epe*) between two optical flows under *lib/metrics.py*.

**Todo:** Test the pretrained model on both Sintel and FlyingThings:

```
python eval.py --output /your/output/directory --model FlowNetC --dataset FlyingThings3D --restore
/path/to/chkpt/checkpoint-model-iter-000600000.pt
```

**Todo:** Finetune the pretrained model on the Sintel dataset by running:

```
python train.py --output /your/output/directory --model FlowNetS --dataset Sintel --restore
/path/to/chkpt/checkpoint-model-iter-000600000.pt --completed_iterations 600000 --iterations
610000
```

**Todo:** Implement the unsupervised loss function which consists of two terms: data term (so-called photometric) and smoothness term (under *lib/loss.py*) as follows:

$$L_{photometric}(u, v; I(x, y, t), I(x, y, t+1)) = \sum_{i,j} p(I(i, j, t) - I(i + u_{i,j}, j + v_{i,j}, t+1)) \tag{1}$$

$$L_{smoothness}(u, v) = \sum_{j} \sum_{i} [p(u_{i,j} - u_{i+1,j}) + p(u_{i,j} - u_{i,j+1}) + p(v_{i,j} - v_{i+1,j}) + p(v_{i,j} - v_{i,j+1})] \tag{2}$$

$$p(x) = (x^2 + \epsilon^2) \tag{3}$$

For more details about these loss terms, please check the following: https://arxiv.org/pdf/1608.05842.pdf

**Todo:** Finetune the pretrained model on the Sintel dataset using your implementation of the unsupervised loss.

**Todo:** Compare the two resulting models by testing them on both Sintel and FlyingThings using (1). Explain your observation.

**This assignment will be discussed on 9.12. 13:00-14:00 CET.**