CS172 Computer Vision I:
Single image depth estimation

Ziqi Gao
2018533193
gaozq@shanghaitech.edu.cn

Abstract

In this homework, an single image depth estimation task is performed. Eigen et al.'s NIPS 2014 paper named Depth Map Prediction from a Single Image using a Multi-Scale Deep Network is reproduced in this homework.

1. Introduction

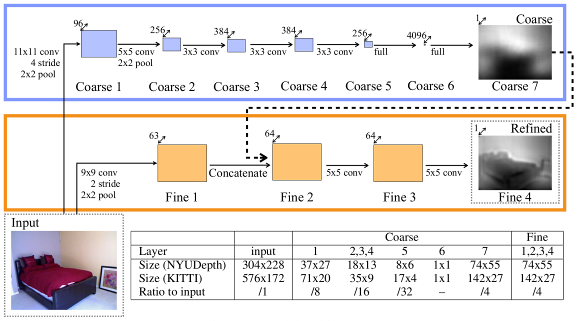
As a fundamental problem in computer vision, depth estimation shows the geometric relations within a scene, which leads to improvement in some recognition tasks including autonomous driving, robotics, 3D reconstruction, etc. Many depth estimation techniques are based on stereo vision, but there are many scenarios that require monocular image depth estimation. However, estimating depth forom a single monocualr image is an ill-posed problem due to a lot of difficulties in techniques. In this homework, Eigen et al.'s paper is reproduced, which is a CNN based model that uses 2 networks: a global coarse one and a local fine one to address the task. The experiment is carried out on NYU Depth V2 dataset.

2. Algorithm

2.1. Network

As shown in Figure 1, the model first predicts the depth of the scene at a global level and then refines within local regions by a fine-scale network. The global, coarse-scale network contains five feature extraction layers of convolution and max-pooling, followed by two fully connected layers. The local, fine-scale network's input is the coarse one's output, which goes through three convolutional layers.

2.2. Loss Function



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Figure 1. Model Architecture

$$L(y, y^*) = \frac{1}{n}\sum_i d_i^2 - \frac{\lambda}{n^2}\left(\sum_i d_i\right)^2$$

Figure 2. Loss function

The traing loss is based on a scaled invariant error function as shown in figure 2. y and y* here are predicted depth and ground truth of depth, respectively; di is defined as the difference between the logarithm of the ith predicted depth and ground truth.

2.3. Data augmentation

I augment the training data with random online transformations.

- rotation: Input and target are rotated by +- 5 degrees.
- translation: Input and target are randomly cropped to 304x228.
- flips: Input and target are horizontally flipped with 0.5 probability.

3. Experiment setup

I did the experiment on NYU Depth V2. RGB images are downsampled by half, from 640x480 to 320x240. The optimizer is SGD with different parameters setting in global and local networks and different layers of a network, which are set exactly same as that in the paper.

Data parallel is implemented for faster training on G-Cluster.

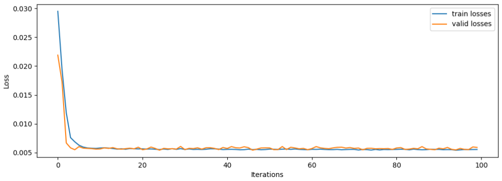


Figure 3. Global Loss during training

4. Implementation and Result

4.1. Code Repository

- net.py: a Python file containing implementation of coarse and fine network, namely GlobalCoarseNet and LocalFineNet.
- util.py: a Python file containing some helper functions like loading datasets and plotting loss and results.
- train.py: a python file containing the network training.
- loss.py: a python file containing the loss function.
- model: a repository containing the training result of global and local networks, respectively.
- nyu: a repository that is supposed to have NYU Depth V2 datatset.

4.2. How To Run

After configuring the model repository, run python training.py and model can be trained and visualized. Cuda is needed for speeding up the experiment.

4.3. Hyperparameters

As specifed in train.py, I chose epochs_num = 100 and rate_factor = 0.1 for training after testing for several times. For a simple implementation, epochs_num = 10 and rate_factor = 1 or 0.1 is also okay. Other parameters that are specified in the paper like batch_size are set to the same values as those in the paper.

4.4. Result

Acoording to figure 3 and figure 4, traing loss and validation loss is small and don't change much after the 20th iteration, and the error of validation set does not increase so there is no overfitting. The depth result is too ugly to be shown.

5. Some random thought

Maybe using the output from local network and refeeding it into a new local network and training it is a good idea? Maybe it will cost some problems too.

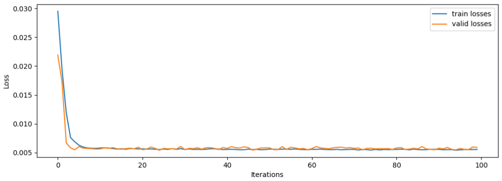


Figure 4. Local Loss during training