CSci 5563 Assignment 3

Luis Guzman

Friday March 12, 2021

This assignment focuses on the task of depth prediction using a single image. I implemented two different methods using PyTorch: the first is a simple encoder/decoder with three convolutional layers each, and the other is the method presented in the paper by Eigen et al. "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network". The full network architecture is shown in Figure 1.

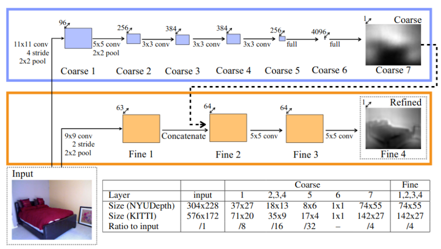

Figure 1: The network architecture of Eigen et al.

In order to train these networks, I implemented two loss functions. L_d measures the error of the predicted depth image. We use the L1 loss function and a mask, which ignores invalid depth pixels in the ground truth image.

$$L_{\mathrm{d}} = \frac{1}{|\mathbf{M}|} \sum_{i}^{B} \sum_{\mathbf{u} \in [0,W) \times [0,H)} \| \mathbf{M}_{\mathbf{u}}^{i} (\mathbf{D}_{\mathbf{u}}^{i} - f_{\mathbf{u}}(\mathbf{I}^{i})) \|_1, \qquad \mathbf{M}_{\mathbf{u}}^{i} = \begin{cases} 1 & \mathbf{D}_{\mathbf{u}}^{i} > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{D}$ is the ground-truth depth and $f(\mathbf{I})$ is the predicted depth. The next loss component, L_c, measures the loss of the predicted normals. To calculate the normals from the predicted depth image, we use

$$\mathbf{X}_{\mathbf{u}} = d\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \qquad \widehat{\mathbf{n}}_{\mathbf{u}} = \frac{(\mathbf{X}(u+1,v) - \mathbf{X}(u,v)) \times (\mathbf{X}(u,v+1) - \mathbf{X}(u,v))}{\|(\mathbf{X}(u+1,v) - \mathbf{X}(u,v)) \times (\mathbf{X}(u,v+1) - \mathbf{X}(u,v))\|},$$

where K is the camera intrinsic matrix and d is the predicted depth value at pixel (u,v). Figure 2 shows the normal images that I calculated from the ground truth depth images.

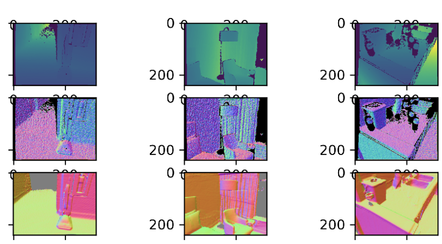

Figure 2: Demonstration of the normal estimation (middle) compared to the ground truth (bottom).

Next, we calculate L_c as

$$L_c = \frac{1}{|\mathbf{M}|} \sum_{i}^{B} \sum_{\mathbf{u} \in [0,W) \times [0,H)} \mathbf{M}_{\mathbf{u}}^{i} (1 - |\hat{\mathbf{n}}_{\mathbf{u}}^{\mathrm{T}} \mathbf{N}_{\mathbf{u}}^{i}|).$$

The final loss function is $L = L_d + \lambda L_c$ where we choose $\lambda = 0.1$.

Figure 3 shows the training and validation loss for the simple network and Figure 4 shows the results. Since this network does not give sensible normal estimation, I chose to train it using only L_d. Poor performance is expected for this network due to the simple architecture. The outputs are blurry since the decoder network upsamples the output over 16x. In 15 epochs, the Simple network L_d converged to 0.5 for both the training and the validation set.

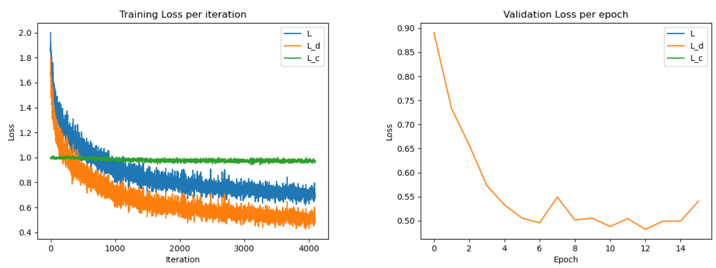

Figure 3: The training loss (left) and validation loss (right) of the Simple network

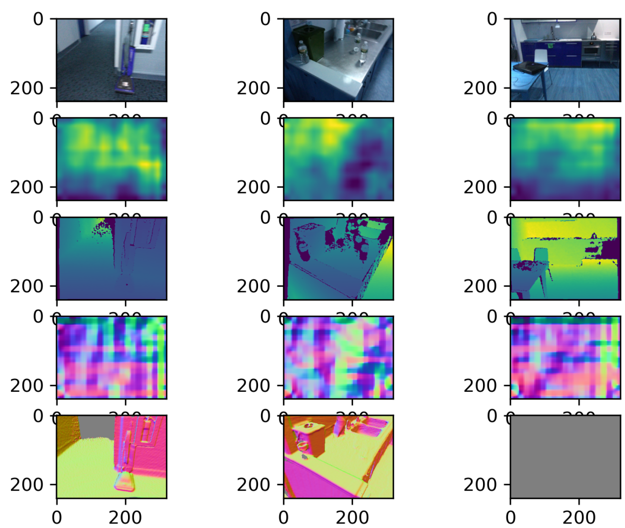

Figure 4: Output examples of the Simple network. From top to bottom: input image, predicted depth, ground truth depth, predicted normals, ground truth normals.

The following figures (5, 6) show the loss and results of the extended network, using only the L_d loss. As expected, the depth prediction fidelity is much improved, and you can see some image features like lines and objects in the predicted depth image and normal image. After 35 epochs, L_d converged to 0.25 for the training set and 0.4 for the validation set.

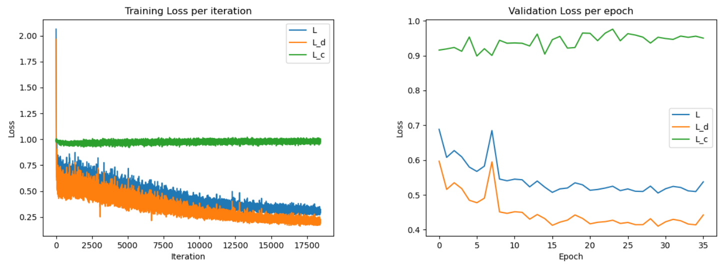

Figure 5: The training loss (left) and validation loss (right) of the Extended network (L_d loss only)

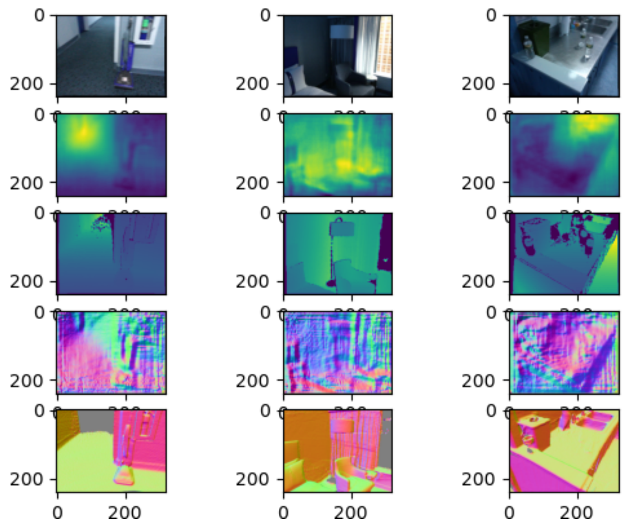

Figure 6: Output examples of the Extended network (L_d loss only). From top to bottom: input image, predicted depth, ground truth depth, predicted normals, ground truth normals.

Unfortunately, adding in the L_c loss function resulted in worse performance in my case. I have checked that my normal estimation is correct (shown in figure 2), so I suspect something else is wrong with my implementation of this part. The L_c loss causes an odd "ripple" artifact that can be seen in figure 7. Overall the extended network outperformed the simple network and gave reasonable results for single-image depth estimation.

Algorithm 1 Training Single View Depth Prediction

```
Construct a training data loader                                  ▷ TinyScanNetDataset
Construct a convolutional neural network                          ▷ ExtendedDepthNet
for Each epoch less than N do
    for Each batch in a epoch do
        Retrieve a batch of data from data loader
        Predict depth and measure prediction error, L_d           ▷ ComputeDepthError
        Measure consistency error, L_c                            ▷ ComputeNormalError
        Back-propagate error
        Update the network weights
    end for
    Save the trained model using torch.save
end for
```

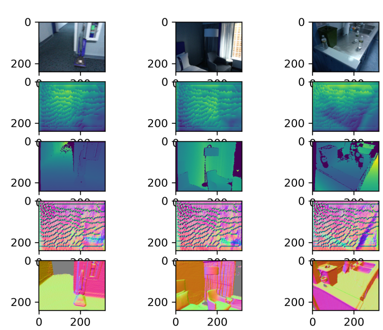

Figure 7: Algorithm (left) and the ripple image using L_c (right)