Monocular Depth Estimation Based on Convolutional Neural Networks

Dawei Wang*
University of Michigan
wdwdawei@umich.edu

Ruiyi Wang*
University of Michigan
ruiyiw@umich.edu

Siyi Chen*
University of Michigan
siyich@umich.edu

Yiyang Qiu*
University of Michigan
yiyangq@umich.edu

1. Introduction

1.1. Background

Depth estimation is a crucial task designated to reconstruct the spatial structure of a scene, recover 3D scene geometry and obtain depth information from 2D images. Depth estimation is fundamental to many real-life applications, aiding tasks such as navigation and motion planning of robots, 3D scene reconstruction, and augmented reality.

There are prior works on depth estimation based on stereo images or motion, where local correspondence suffices for estimation given well-studied methods[10]. However, obtaining depth information from a single RGB image is less straightforward than stereo images. In many cases such as robotics and autonomous driving, only monocular images are available. With the advancements in deep learning and convolutional neural networks, it is possible to effectively estimate depth of a scene using only single RGB images.

1.2. Related work

1.2.1 Monocular Depth Estimation

Monocular depth estimation aims to predict depth of pixels from a single 2D image. Early works proposed approaches using multi-scale information and directly regressing over pixels for depth estimation[3]. Recently, with the use of Deep Convolutional Neural Networks (abbr. as DCNNs hereafter), some methods were able to estimate the continuous depth map, which is formulated as a continuous conditional random field (CRF) learning problem[8, 2]. To improve such model, Xu et al. added a structured attention model which can robustly fuse features derived from multiple scales[14]. Another line of research aims at taking advantage of DCNNs focused on applying standard DCNNs designed for image classification as feature extractors, and upsampling effectively to restore resolution of the produced dense maps. ResNet-50[6] and denseNet-169[1] were used as pre-trained model for depth estimation task and achieved high performance with fewer computational power. Deviating from prior approaches for monocular depth estimation, Fu et al. proposed a new method reconstructing the problem as an ordinal regression task rather than a continuous depth mapping problem. In this paper, a space-increasing discretization (SID) is introduced, which increases the robustness by allowing the prediction of larger depth value to accept larger error instead of over-strengthening the effect of large values[4].

Research on monocular depth estimation that focused more on real-time inference rather than high quality also points out a popular direction. In applications involving embedded platform, an efficient and light network architecture is needed. The work proposed by Wofk et al. achieved similar accuracy as the prior state-of-the-art models while reducing the computational complexity by applying a simple encoder-decoder architecture and pruning the network further[13].

1.2.2 Transfer Learning

Transfer learning improves model performance and reduces computational cost by utilizing useful information on a large scale from a well-defined task and fine-tuning the model in use. Transfer learning is often expressed through the use of pre-trained models, among which VGG, ResNet, Inception V3 and denseNet are popular. As is investigated in the work proposed by Zamir et al., transfer learning has achieved high performance in tasks including 3D keypoint detection and Z-buffering, which is closely related to 3D scene reconstruction[15]. Transfer learning is recently applied in depth estimation using encoder-decoder structure, where the encoder is initialized using denseNet-169 pre-trained on ImageNet[1]. Such approach simplifies the model architecture and reduces the trainable parameters while outperforming state-of-the-art benchmarks[1].

2. Approach

2.1. Architecture

An overview of the architecture of our model is shown in Figure 1. The input image is first passed through an encoder to extract features, which is a backbone model pretrained with ImageNet. The learned features are passed to the decoder, which gradually upsamples the image and finally outputs the depth prediction for each pixels.

2.1.1 Encoder

Transfer learning is applied on the encoder with all layers frozen. By transfer learning, we can utilize high-performance models originally trained for image classification and use the models as an expressive feature extractor. We adapted 2 models: mobile net v2 [9] and squeeze net [5] as the backbone model and the comparison will be discussed in section 3. The last classification layer of the backbone model is removed so the encoder outputs various features learned from the image.

2.1.2 Decoder

The goal of the decoder is to estimate the depth of the image given the learned features outputted from the encoder. The decoder is composed of several decoder blocks. Each block first applies $\times 2$ bilinear upsampling to the input, and concatenate with the output from the corresponding layer in the encoder. The merged input is then passed through 2 convolutional layers, each with a filter size of 3, stride size 1 and padding size 1. Finally, we use a leaky Relu activation to add non-linearity. The leaky Relu is only applied to the second convolutional layer[1].

The number of decoder blocks depend on the encoder. For mobile net, there are 4 pooling layers in the encoder so the decoder is composed of 4 decoder blocks. For squeeze net, the ender downsamples the image 3 times so there are 3 decoder blocks in the decoder.

2.1.3 Skip Connections

In the encoder, after each downsampling layer, we obtain lower resolution images and may lose some high resolution details. During decoding, the model needs to recover these details from the low resolution inputs. To help recover these details, we add skip connections from the encoder to the decoder. After upsampling in the decoder, the output is concatenated with the output from the encoder with the same spatial size, before passing into convolutional layers.

2.2. Data Augmentation

Two basic data augmentation methods are used, horizontal flipping with a probability of 0.5 and channel swapping with a probability of 0.5 [1], though the paper applies channel swapping with a probability of 0.25.

In general, for monocular depth estimation, rotation as well as many other data augmentation methods do not improve the prediction results. After experimenting on encoder-decoder structures, we perform two kinds of experiments on data processing, including normalization and random gray-scaling.

2.3. Loss function

Standard loss functions for optimization of regression problems include $\mathcal{L}_2$ loss and $\mathcal{L}_1$ loss. Apart from the standard loss functions, other types of loss are studied over the years, such as loss from optical flow and motion[11], the reverse Huber loss [7], and so on.

Experiments of different loss functions would be another interesting topic to study. However, in the paper, we use the weighted sum of three loss functions as the net loss[1]:

$$L(y, \hat{y}) = \lambda L_{depth}(y, \hat{y}) + L_{grad}(y, \hat{y}) + L_{SSIM}(y, \hat{y}). \tag{1}$$

$L_{depth}(y, \hat{y})$ is the point-wise $\mathcal{L}_1$ loss between the predicted and ground truth depth values:

$$L_{\text{depth}}(y, \hat{y}) = \frac{1}{n} \sum_{p}^{n} \left| y_p - \hat{y}_p \right|. \tag{2}$$

$L_{grad}(y, \hat{y})$ is the point-wise $\mathcal{L}_1$ loss between the gradient $\mathbf{g}$ of the predicted and ground truth depth values:

$$L_{grad}(y, \hat{y}) = \frac{1}{n} \sum_{p}^{n} \left| \boldsymbol{g}_{\mathbf{x}}\left(y_p, \hat{y}_p\right) \right| + \left| \boldsymbol{g}_{\mathbf{y}}\left(y_p, \hat{y}_p\right) \right|. \tag{3}$$

$L_{SSIM}(y, \hat{y})$ is constructed based on the Structural Similarity (SSIM), which compares local patterns of pixel intensities[12]:

$$L_{SSIM}(y, \hat{y}) = \frac{1 - SSIM(y, \hat{y})}{2} \tag{4}$$

where SSIM is defined as:

$$\text{SSIM}(x, y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)},$$

where μ is the average, σ^2 is the variance, σ_{xy} is the covariance, c_1 and c_2 are two variables to stabilize the division with weak denominator[12].

Another raised problem is the large loss values of pixels with larger depth may cause huge overestimation of deeper pixels. One solution is to use the inverse depth for calculating loss functions [11, 1], where the reciprocal of the depth

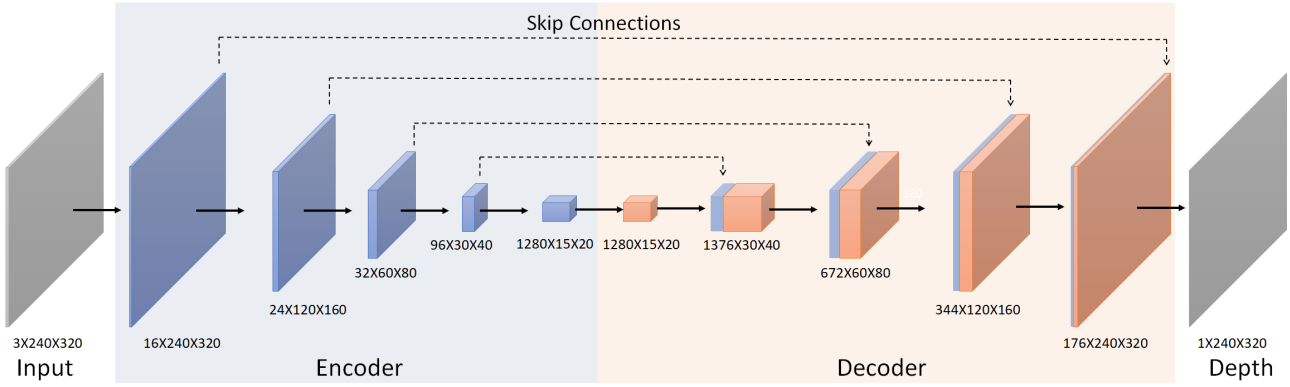

Figure 1. **Model Architecture** with mobile net as the backbone. The encoder is a pretrained mobile net v2 with the classification layer removed. The decoder consists of blocks that applies upsampling, concatenating and convolution. Dimensions of the features are given as #channels $\times$ height $\times$ width.

is used instead. However, considering the data set used in this paper is NYU2, which is a collection of indoor scenes without much significant differences in depth, we decide not to use the inverse depth as an experiment.

2.4. Training procedure

The model is trained using the NYU2 data set. The data set is shuffled and then divided into training, validation, and testing set by a fraction of $79:20:1$.

First, we train the model with three Encoder-Decoder pairs of different structures as presented in the architecture section: the Mobile net v2 pair and the Squeeze net pair. Each model is trained with 7 epochs. The number of epochs may be experimented with methods like early stopping.

Then based on the evaluation on both the accuracy and efficiency of the model performance, we choose the Mobile net v2 to conduct further experiments on data augmentation methods with a subset of 2000 images. The experiments include data normalization and random grayscaling.

3. Experiments

3.1. Quantitative evaluation metrics

There are six measures suggested in [1]:

- average relative error (rel):

$$\frac{1}{n}\sum_{p}^{n}\frac{|y_p - \hat{y_p}|}{y_p}$$

- root mean squared error (rms):

$$\sqrt{\frac{1}{n}\sum_{p}^{n}(\frac{y_p - \hat{y_p}}{y_p})^2}$$

- average ($\log_{10}$) error:

$$\frac{1}{n}\sum|\log_{10}(y_p) - \log_{10}(\hat{y_p})|$$

- threshold accuracy (δ_i): % of y_p such that $\max(\frac{y_p}{\hat{y_p}}, \frac{\hat{y_p}}{y_p}) = \delta < thr$ for $thr = 1.25, 1.25^2, 1.25^3$

Here y_p is a pixel in the depth image (ground truth), $\hat{y_p}$ is a pixel in the predicted depth map, and n is the total number of pixels in an image. In our test, we used all of the six metrics to measure one model's performance against the other. As the names suggest, the first three metrics are "errors", so they are expected to be low for a good model, while for the other three, as they are "accuracy", they should be high for a good model.

3.2. Encoder-Decoder Evaluation

We have implemented two kinds of encoders and their corresponding decoders: Mobile net v2 and Squeeze net. Table 1 shows the metrics of these two networks. As we

	Rel	RMS	$\log_{10}$
Mobile net v2	0.228	0.337	0.088
Squeeze net	0.412	0.575	0.151
	δ_1	δ_2	δ_3
Mobile net v2	0.661	0.891	0.968
Squeeze net	0.405	0.690	0.867

Table 1. Mobile net v2 versus Squeeze net

can see, for Rel, RMS, and log_{10} errors, Mobile net v2 has a much lower value than that of the squeeze net, while for the threshold accuracy metrics, it has a slightly higher value than that of the squeeze net. Therefore, we choose Mobile net v2 as a good encoder for our model.

3.3. Visualization

We have selected several images to be tested by our models, and visualization of the depth prediction results with the

original image as well as ground truth is shown in Figure 2. The model with mobile net v2 as the encoder performs better than the squeeze net as visualized. We also include an image that 'breaks' our model as shown in the 4th row. We can see that the model predicts different depth for the rug and the floor.

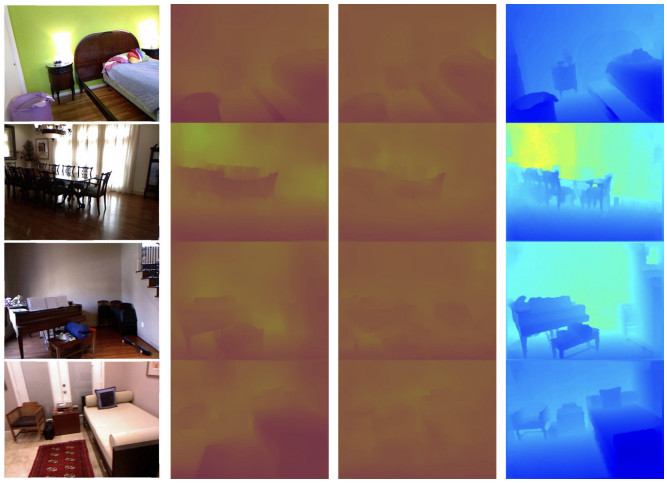

Figure 2. Original image, Mobile net v2, Squeeze, Ground truth

3.4. Data Processing Evaluation

Based on the experiments of encoder and decoder pairs in 3.2, we decide to use Mobile net v2 and perform two experiments on data processing: normalization and data augmentation by grayscale with probability 0.5. Due to training resource limitations, the experiments are conducted on a subset of NYU2 Depth containing 2,000 images.

Normalization The RGB channels of each input image in the training dataset are normalized to have a mean of [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225].

The performance of the model with or without normalization is shown in Table 2, where the Rel, RMS, and log10 errors are generally reduced after normalization and the threshold accuracy increases by a small amount. In conclusion, we state that normalization will reduce the sum of errors and increase the thresholded accuracy, and thus is helpful for depth prediction

Grayscale For each input image in the training dataset, apart from the two basic data augmentation methods, we perform random grayscaling with a probability of 0.5.

The performance of the model with or without random grayscaling is shown in Table 2, where the loss, Rel, RMS and log10 errors are generally reduced after augmentation with grayscaling and the thresholded accuracy increase by a small amount. We conclude that random grayscaling is slightly helpful for our depth prediction.

	Rel	RMS	$\log_{10}$	δ_1
Normalization	0.3175	0.4234	0.1380	0.4176
Grayscale	0.3195	0.4204	0.1379	0.3936
None	0.3501	0.4732	0.1474	0.3974
	δ_2	δ_3	Loss	
Normalization	0.7306	0.9214	45.3805	
Grayscale	0.7364	0.9258	45.5978	
None	0.6989	0.8924	47.1242	

Table 2. Effect of Normalization and Grayscaling

4. Implementation

We use the NYU2d dataset and split it randomly into training, validation and testing datasets by shuffling and creating different csv files with pandas.

The basic data loading, pre-processing and augmentation methods are implemented based on the class implemented in GitHub[1]. The other two pre-processing methods applied are implemented by ourselves: normalization as well as averaging the RGB values of input images with a probability of 0.5.

We build our training framework referring the one of EECS445 project 2. We implement two model structures, where transfer learning is applied to the encoders and decoders are constructed to fit the output sizes of the encoder. The loss functions is implemented based on [1]. The models are trained for 7 epochs using the Adam optimizer with default parameters.

The testing of models with different metrics are implemented by ourselves, and the visualization of predictions given by the trained models is implemented based on [1].

5. Conclusion

In this project, we implemented a decoder-encoder structure to predict depth given a monocular input image. We experimented with mobile net v2 and squeeze net as the backbone encoder and prepossessing techniques including normalization and augmentation. We evaluated the models with 6 quantitative metrics and qualitative analysis. Our results showed that using mobile net as the encoder improves performance and normalizing the images and augmentation lead to small improvements. Due to the limitations of computational resources, we only trained our models for a small number of epochs. The loss of the model had not plateaued when we finished training, so more training can still lead to potential performance improvements.

References

[1] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. 2019.

[2] Amlaan Bhoi. Monocular depth estimation: A survey, 2019.

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network, 2014.

[4] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. 2018.

[5] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size. 2016.

[6] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. 2016.

[7] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. pages 239–248, 2016.

[8] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2024–2039, Oct 2016.

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2019.

[10] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pages 131–140, 2001.

[11] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[12] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[13] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. 2019.

[14] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci. Structured attention guided convolutional neural fields for monocular depth estimation, 2018.

[15] Amir Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. 2018.