Monocular Depth Estimation via Ensemble Deep Learning

Chinchuthakun Worameth

Department of Transdisciplinary Science and Engineering
Tokyo Institute of Technology

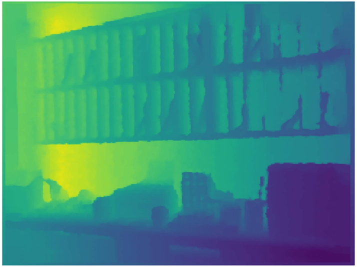
July 22, 2021

Table of Contents

What is monocular depth estimation?

- Predict depth information/generating a corresponding **depth map** from images or a video sequence of a single camera
- Provide a cost, space, and energy efficient alternative to existing depth sensors, e.g. LiDARs which are large and have high power consumption, especially in small robotic platforms.

Raw image



Depth map

Table of Contents

Monocular Depth Estimation at a glance

Deep learning approaches are classified into 3 main types:

- **Supervised**: Use ground truth depth maps and frame the problem as a regression one
- **Unsupervised**: Exploit geometric constraints between frames in monocular video sequence
- **Semi-supervised**: Basically unsupervised approach with known transformation between frames

Current challenges in the field include, but not limited to, the followings:

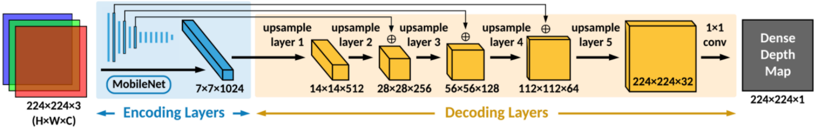
- **Accuracy**: as a matter of course, especially in unsupervised methods
- **Transferability**: train on one dataset, test on another
- **Real-time Performance**: lightweight and low-latency networks

FastDepth: Fast Monocular Depth Estimation on Embedded Systems

This paper proposes a lightweight decoder for monocular depth estimation with details summarized as follows:

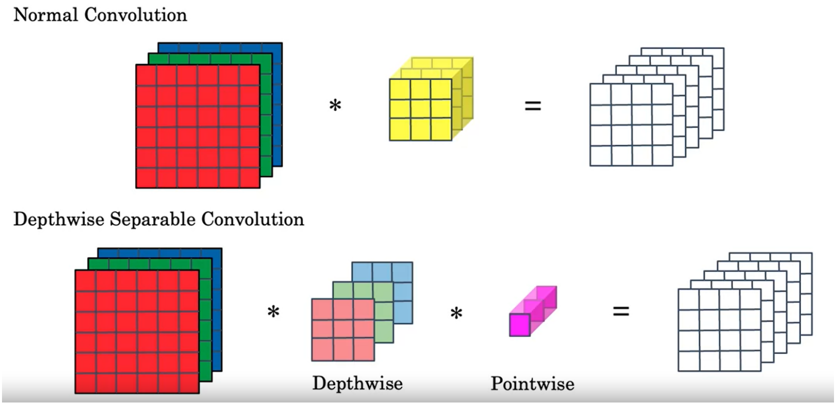
- Contains only **depthwise separable convolutions**
- When decoding, interpolate **after** convolution instead of before

While it has low-latency and smaller size (even smaller after network pruning), its accuracy is naturally worse than state of the arts.



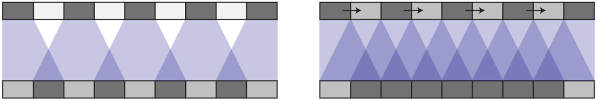
FastDepth Architecture [1]

Supplementary #1: Depthwise Separable Convolution

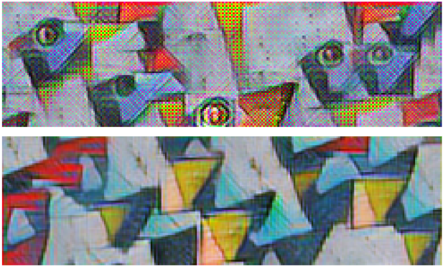


Computational cost comparison

Supplementary #2: Why Interpolation + Convolution instead of Transpose convolution

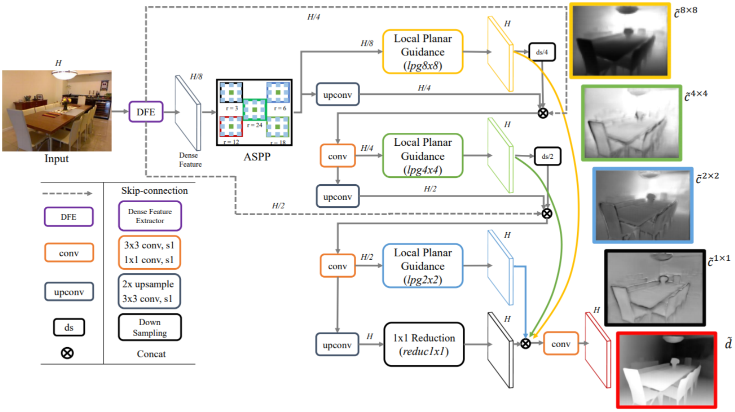


Coverage of Transpose Conv (left) and Conv + Int (right)



Checker board patterns

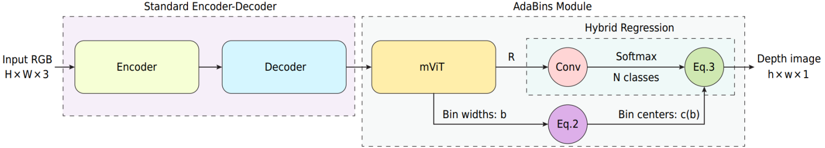
BTS: From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation



BTS architecture [2]

AdaBins: Depth Estimation using Adaptive Bins

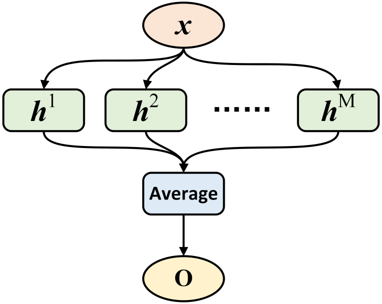
- Rephrase the problem to multi-class classification one by quantizing depth values into bins to obtain discrete depth values
- Bin width is automatically adjusted for each image by a visual-transformer network (ViT)
- Final depth values are computed by a linear combination of bin centers to suppress discretization artifacts



Adabins architecture [3]

Ensemble Deep Learning

Ensemble deep learning refers to an approach of combining prediction of multiple neural networks to make final decision. While it has been applied in different tasks, e.g. image classification and text summarization, a recent survey indicates no application in monocular depth estimation.



Fusion and Voting Paradigm

Table of Contents

Objective

Currently, aiming to:

- Demonstrate prospect application of ensemble deep learning in Monocular Depth Estimation by reaching (approximately) the same level of performance with the **downgraded** version of the original models, **consisting of less than half of parameters**.

If possible:

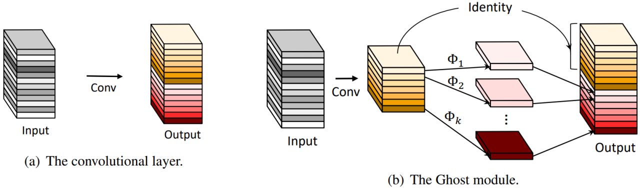
- Exceed the original model's performance (This is likely not possible since **both models are one of the most recent SOTAs in KITTI and NYU V2 dataset**)

Proposed Method: Overview

- Consists of Adabins [3] and BTS [2]
- Partly inspired by FastDepth [1], Adabins and BTS are simplified (reduce number of parameters) by
  - Use GhostNet [4], pretrained by ImageNet, as the encoder
  - When upsampling feature maps, interpolate after convolution
  - Substitute convolutions by depthwise seperable convolutions
- Final prediction is the mean of output depth map of each model
- Number of parameters:
  - **Original BTS**: 47.0M
  - **Original Adabins**: 78.2M
  - **Proposed Ensemble**: 26.3M

Supplementary #3: GhostNet: More Features from Cheap Operations [4]

- Based on observation that the output feature maps of convolutional layers often contain much redundancy
- Generate some feature maps through usual convolution. Then, apply **linear operations** to generate more feature maps
- **GhostNet** is obtained by arranging **ghost modules** in a structure similar to MobileNetV2



Ghost module [4]

Proposed Method: Loss functions

Similar to [3], **pixel-wise depth loss** and **bin-center density loss** are used to mitigate pixel-wise difference and encourage the distribution of bin centers to follow the distribution of depth values in the ground truth, respectively. Since [2] also uses pixel-wise depth loss, both sub-networks are trained simultaneously using a loss function defined by:

$$L_{\text{total}} = L_{\text{pixel}} + L_{\text{bins}}$$

$$L_{\text{pixel}} = \alpha \sqrt{\frac{1}{N}\sum_{i=1}^{N} y_i^2 - \frac{\lambda}{N^2}(\sum_{i=1}^{N} y_i)^2}$$

where $y_i^2 = \log(d_i) - \log(d_i^*)$ and d_i^* is ground truth depth

$$L_{\text{bins}} = \textbf{BiChamfer}(c(b), D) + \textbf{BiChamfer}(D, c(b))$$

Note that $\lambda = 0.85$ and $\alpha = 10$ are used as same as the original Adabins [3]

Evaluation: Datasets and Data Augmentation

NYU Depth V2 and KITTI, are used to evaluate the performance:

- **NYU Depth V2**: indoor scenes, acquired from a RGB-D camera. Training data are extracted in the same manner as [2] and [3]
- **KITTI**: outdoor scenes, acquired from LiDARs

Similar to [2] and [3], the following augmentation are applied to avoid overfitting:

- **Random horizontal flipping** with probability of 0.5
- **Random contrast, brightness, and color adjustment** in a range of [0.9, 1.1] with probability of 0.5
- **Random crop** of size 416×544 for NYU V2 and 352×704 for KITTI
- **Random rotation** of degree in ranges of [-2.5,2.5] for NYU V2 and [-1,1] for KITTI

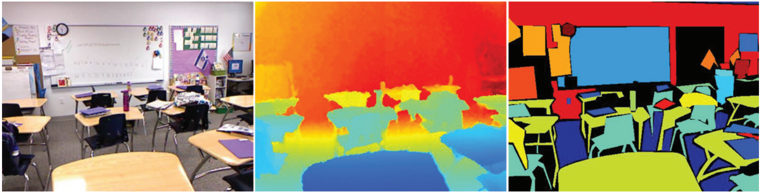
Supplementary #4: KITTI dataset

- Consists of 56 different outdoor scenes
- Official split is 28 for training and 28 for testing
- Constructed by stereo image pairs and ground truth depth obtained from LIDAR

A sample of raw image from KITTI dataset

Supplementary #5: NYU Depth V2 dataset

- Consists of 464 different indoor scenes
- Official split into is for training and 215 for testing
- Constructed by monocular video sequences of scenes and ground truth of depth captured by an RGB-D camera



A sample of raw image, preprocessed depth, and labeled from the dataset

Evaluation: Metrics

Standard metrics used in previous works include:

- **Threshold Accuracy**: % of d_i s.t. $\max\left(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}\right) = \delta <$ threshold, usually threshold $= 1.25, 1.25^2, 1.25^3$
- **Average Relative Error (REL)**: $\frac{1}{N}\sum\left(\frac{|d_i - d_i^*|}{d_i^*}\right)$
- **Root Mean Squared Error (RSME)**: $\sqrt{\frac{1}{N}\sum(d_i - d_i^*)^2}$
- **Average** $\log_{10}$ **Error**: $\frac{1}{N}\sum|\log_{10}(d_i) - \log_{10}(d_i^*)|$

Additionally, another two standard metrics are included for KITTI dataset:

- **Squared REL (Sq REL)**: $\frac{1}{N}\sum\frac{\|d_i - d_i^*\|}{d_i^*}$
- **RSME of logarithm (RSME log)**: $\sqrt{\frac{1}{N}\sum\|\log d_i - \log d_i^*\|^2}$

Proposed Method: Implementation

- Implemented in **Pytorch**, trained in a Laboratory's server (Rat) using distributed training.
- **AdamW optimizer** [5] together with **1-cycle policy** [6] are used to train the model for fast convergence
- Maximum learning rate is determined from **lr range test** [7]. **Batch size** is set to as large as possible, limited only by physical memory, to enable larger learning rate, as suggested by [6].
- Other hyperparameters are tuned via **grid search** and **random search** together with **Hyperband** [8].
- Training and hyperparameter tuning are monitored by **Weights and Biases** platform

Result and Discussion (so far)

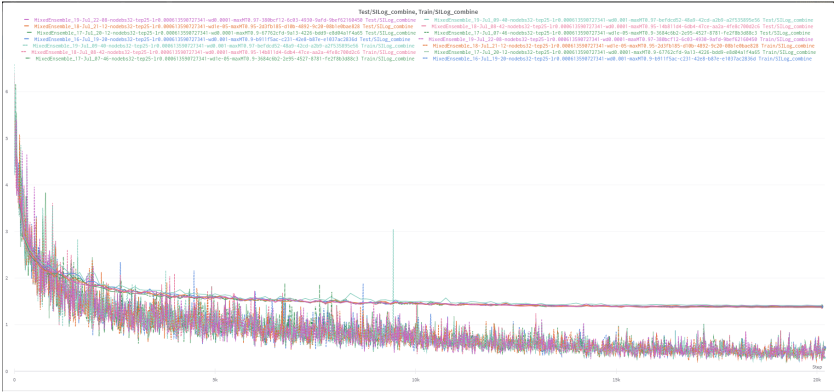
- The model is evaluated on pre-defined center cropping, without other transformations, as described in [9]
- At test time, the final output is the average of an image's prediction and the prediction of its mirror image which is commonly used in previous work such as [3]

Metrics	BTS	Adabins	Proposed
Alpha1	0.885	0.903	0.851
Alpha2	0.978	0.984	0.973
Alpha3	0.994	0.997	0.993
REL	0.110	0.103	0.122
RMSE	0.392	0.364	0.450
log10	0.047	0.044	0.053

Current best performance of proposed method on NYU Depth V2 dataset

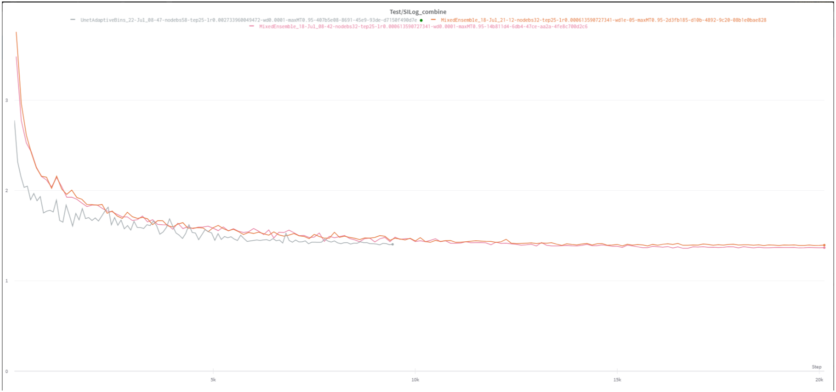
Result and Discussion (so far)

- Varying momentum and learning rate did not yield significant improvement
- Does not seem to be overfitting
- So, what wrong? Global optima? Local optima? Saddle point?



Result and Discussion (so far)

- Increasing batch size and learning rate yields faster convergence
- Currently training, but probably no significant improvement, again...



Future Works

- **Implementation**
  - Continue tuning hyperparameters
  - Train and evaluate on KITTI dataset
- **Design**
  - Explore new strategies to regularize the training
  - Continue tuning hyperparameters

References

[1] D. Wofk, F. Ma, T. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," *CoRR*, vol. abs/1903.03273, 2019.

[2] J. H. Lee, M. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *CoRR*, vol. abs/1907.10326, 2019.

[3] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," *CoRR*, vol. abs/2011.14141, 2020.

[4] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," *CoRR*, vol. abs/1911.11907, 2019.

[5] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *CoRR*, vol. abs/1711.05101, 2017.

[6] L. N. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," *CoRR*, vol. abs/1708.07120, 2017.

[7] L. N. Smith, "No more pesky learning rate guessing games," *CoRR*, vol. abs/1506.01186, 2015.

[8] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Efficient hyperparameter optimization and infinitely many armed bandits," *CoRR*, vol. abs/1603.06560, 2016.

[9] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *CoRR*, vol. abs/1406.2283, 2014.