Supervised Monocular Depth Estimation via Stacked Generalization

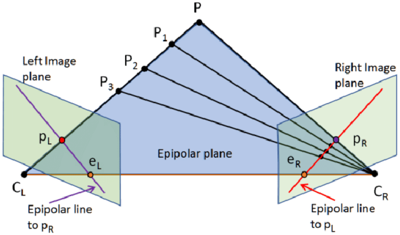
Chinchuthakun Worameth

December 28, 2021

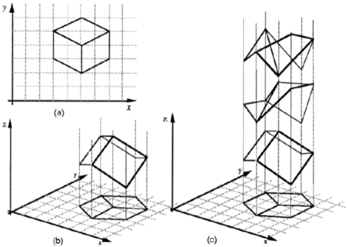
Yamashita Laboratory
Department of Transdisciplinary Science and Engineering
School of Environment and Society
Tokyo Institute of Technology

Depth Estimation

- **Active** $\rightarrow$ Depth sensors based on wave reflection
- **Passive** $\rightarrow$ Use images from different perspectives to predict **depth map**
  - **Stereo (2), Multiview (>2)** $\rightarrow$ Near-perfect approximation via **epipolar geometry**
  - **Monocular (1)** $\rightarrow$ **Ill-posed problem**
- Monocular approach offers a cost, space, and energy efficient alternative



Epipolar geometry [1]

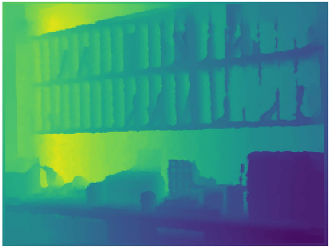


Many-to-one relationship of 3D scenes and projections [2]

Monocular Depth Estimation

- Human perceive depth **subconsciously**
  - Difficult to mathematically describe the process
- **Deep learning** approaches
  - **Supervised** → Use ground truth depth maps
  - **Unsupervised / Semi-supervised** → Use geometric constraints in video

Original RGB image [3]



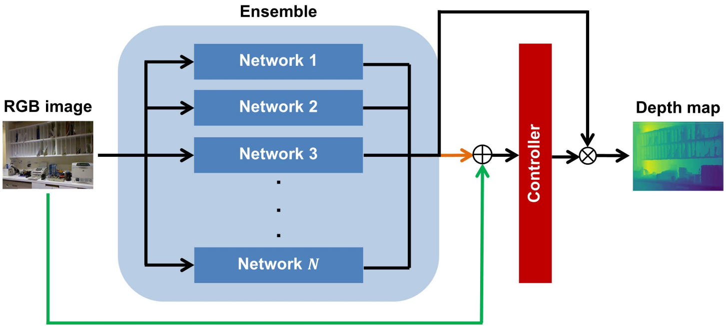
Corresponding depth map [4]

Related Works

- **Monocular Depth Estimation**
  - Novel architectures based on **Convolutional Neural Network (CNN)** or **Visual Transformer**
  - **Multi-tasking** (usually with image segmentation)
  - **Domain adaptation**
  - **Lightweight Network** for fast inference
  - and so on...
- **Ensemble Deep Learning**
  - Approaches that combine predictions from **base learners** and generate a better final output (hopefully)
  - **Stacked Generalization** linearly combines predictions by using **meta-learner**
  - **While it has been adopted in many tasks, there is no application in monocular depth estimation according to a recent survey** [5]

Research Objective

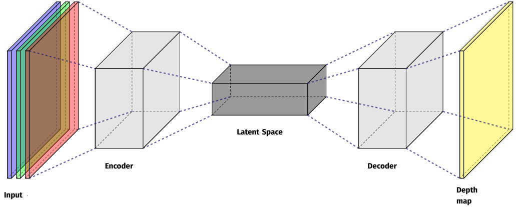
- Study SG in monocular depth estimation
- Compare performance of different SG setups with **simple average (baseline)**



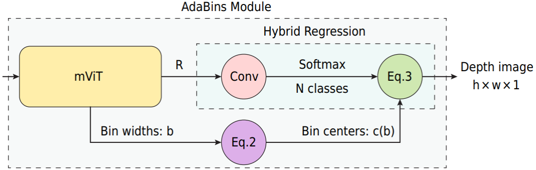
Overview of ensemble architecture

Methodology: Base learners

- Adopt 3 SOTAs as base learners
  - **Adabins** [6], **BTS** [7], **LDRN** [8]
- Every architecture employs **Encoder-Decoder framework**
- Adabins also adds a **visual transformer** module to learn depth distribution
- Input is an **RGB image** $X \in \mathbb{R}^{H \times W}$. Output is a depth map of the same size.



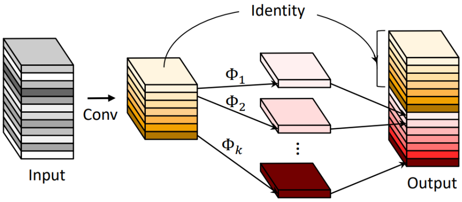
Encoder-decoder framework



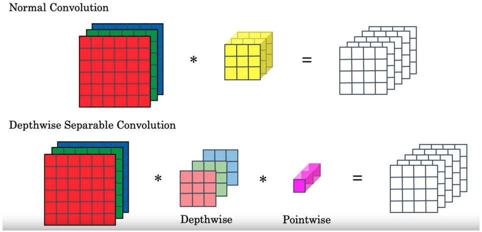
Visual Transformer module in [6]

Methodology: Base learners (2)

- Some modifications to reduce models' size and latency
  - Employing **GhostNet** [9] as the encoder for all architectures.
  - Replacing convolution with **depthwise separable convolution** in [6] and [7] except for atrous convolution layers.
  - Interpolating **after** convolution instead of before in [6], as described in [10].



(b) The Ghost module.
Building block of GhostNet [9]



Normal convolution and depthwise separable convolution [11]

Methodology: Meta-learner

- For a N-ensemble, output is a **pixel-wise coefficient tensor** $\mathbf{W} \in \mathbb{R}^{N \times H \times W}$
- Final prediction $\mathbf{D}^* \in \mathbb{R}^{H \times W}$ is calculated by

$$\mathbf{D}^* = \sum_{i=1}^{N} \left(\mathbf{W} \odot \left[\mathbf{D}^{(1)}; \ldots; \mathbf{D}^{(N)} \right]^T \right)(i, :, :)$$

  where $\mathbf{D}^{(i)}$ is the prediction from i-th base learner
- Consider 4 design aspects for ablation studies
  - Train base learners and controller **simultaneously** or **sequentially**?
  - **Freeze** base learners' parameters or **fine-tune** them?
  - Train with **RGB images** or **predictions from base learners**? Or **both of them**?
  - Are performance consistent among **different encoders**?

- **Pixel-wise depth loss** [12]
  - For mitigating pixel-wise difference in every module

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

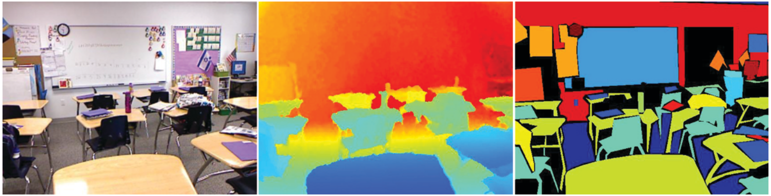
where $y_i^2 = \log(d_i) - \log\left(d_i^*\right)$

- **Bichamfer Loss** [6]
  - For encouraging distribution of bin centers to follow that of ground truth in **Adabins** [6]

$$\mathrm{BC}(c(b), D) = \sum_{x \in c(b)} \min_{y \in D} \|x - y\|_2^2 + \sum_{y \in D} \min_{x \in c(b)} \|x - y\|_2^2$$

Experiment

- Implemented in **PyTorch** [13] and trained on laboratory server (10 GTX 1080 Ti GPUs) via **distributed learning**
- Employ **data augmentation** in [6] to avoid overfitting
- Evaluated on the official split of **NYU Depth V2 dataset** [3] with standard metrics from [12]



A sample of raw image, preprocessed depth, and labeled from NYU Depth V2 dataset [3]

Result and Discussion

- Different base learners
- Same base learner, same dataset
- Same base learner, CV-like dataset

	Variant	#Params	*higher is better*			*lower is better*				
			$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	REL	Sq REL	RSME	RSME log	log10
MOD	Adabins	17.2M	0.813	0.965	0.992	0.146	0.106	0.498	0.178	0.060
	BTS	8.9M	0.855	0.973	0.994	0.120	0.075	0.435	0.156	0.052
	LDRN	14.9M	0.831	0.967	0.993	0.130	0.085	0.455	0.167	0.056
SAME	BTS #1	8.9M	0.862	0.973	0.999	0.120	0.074	0.427	0.155	0.052
	BTS #2		0.855	0.972	0.993	0.121	0.077	0.434	0.157	0.053
	BTS #3		0.852	0.973	0.993	0.121	0.076	0.438	0.157	0.053
	BTS #4		0.856	0.973	0.994	0.119	0.073	0.431	0.155	0.052
	BTS #5		0.856	0.972	0.994	0.121	0.075	0.431	0.155	0.052
CV-BTS	BTS #1	8.9M	0.854	0.973	0.994	0.121	0.076	0.437	0.157	0.053
	BTS #2		0.852	0.971	0.994	0.122	0.076	0.439	0.158	0.053
	BTS #3		0.853	0.972	0.993	0.121	0.077	0.442	0.158	0.053
	BTS #4		0.856	0.973	0.993	0.122	0.078	0.433	0.156	0.052
	BTS #5		0.855	0.972	0.993	0.122	0.077	0.438	0.157	0.053
CV-LDRN	LDRN #1	14.9M	0.840	0.970	0.993	0.128	0.082	0.450	0.164	0.055
	LDRN #2		0.842	0.968	0.992	0.130	0.084	0.451	0.165	0.055
	LDRN #3		0.841	0.969	0.993	0.128	0.081	0.445	0.163	0.055
	LDRN #4		0.851	0.971	0.993	0.125	0.081	0.437	0.159	0.054
	LDRN #5		0.836	0.968	0.992	0.131	0.086	0.456	0.166	0.056

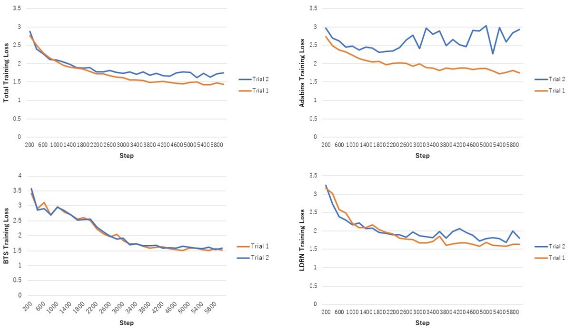
Result and Discussion (2)

- Should train sequentially
- Freezing is more stable
- Both inputs are better (?)
- Performances are consistent (?)

	Variant	#Params	*higher is better*			*lower is better*				
			$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	REL	Sq REL	RSME	RSME log	log10
MOD	Baseline	-	0.8573	0.9758	0.9949	0.1215	0.0735	0.4245	0.1533	0.0517
	I-F-G	4.3M	0.8591	0.9753	0.9946	0.1191	0.0720	0.4261	0.1529	0.0513
	O-F-G	4.3M	0.8602	0.9749	0.9943	0.1186	0.0722	0.4240	0.1527	0.0511
	IO-F-G	4.3M	0.8607	0.9751	0.9945	0.1190	0.0718	0.4240	0.1525	0.0512
	IO-F-M2	3.7M	0.8589	0.9747	0.9942	0.1191	0.0727	0.4270	0.1532	0.0514
	IO-F-D161	30.3M	0.8607	0.9749	0.9944	0.1184	0.0717	0.4223	0.1523	0.0510
SAME	Baseline	-	0.8658	0.9759	0.9948	0.1161	0.069	0.4153	0.1491	0.0503
	IO-F-G	4.3M	0.8663	0.9761	0.9948	0.116	0.0689	0.414	0.1489	0.0503
	IO-F-D161	30.3M	0.8661	0.976	0.9948	0.1162	0.0693	0.4144	0.149	0.0503
CV-BTS	Baseline	-	0.8644	0.9761	0.9948	0.1171	0.0707	0.4204	0.1504	0.0508
	IO-F-G	4.3M	0.8649	0.976	0.9948	0.1171	0.0707	0.4194	0.1503	0.0508
	IO-F-D161	30.3M	0.8654	0.9762	0.9947	0.1169	0.0706	0.418	0.1501	0.0507
CV-LDRN	Baseline	-	0.8596	0.9749	0.9944	0.1206	0.0724	0.4202	0.1529	0.0516
	IO-F-G	4.3M	0.8594	0.9743	0.9941	0.1208	0.0744	0.4226	0.1538	0.0517
	IO-F-D161	30.3M	0.8570	0.9741	0.9940	0.1206	0.0741	0.4252	0.1546	0.0521

Result and Discussion (3)

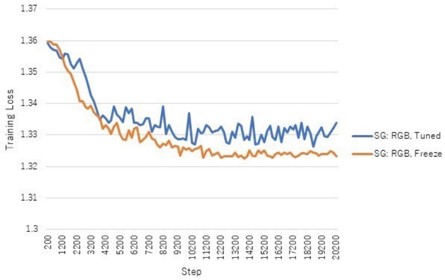
- Train base learners & meta-learner together is fast, but yields worse models
  - Need **hyperparameter tuning** specific to base learners to ensure convergence
  - Even then, reproducibility is not guaranteed → inappropriate for practical use

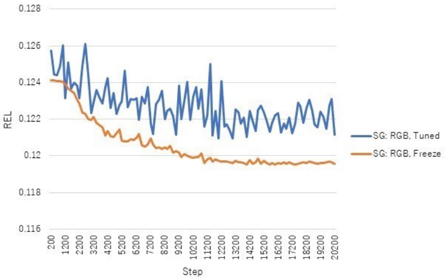


Results are non-reproducible (not even close!)

Result and Discussion (4)

- Fine-tuning base learners when training meta-learner yields **training loss fluctuation**
- Unfortunately, no improvements in evaluation metrics





Fluctuations in training loss and an evaluation metric

References i

[1] D. Chotrov, Z. Uzunova, Y. Yordanov, and S. Maleshkov, “Mixed-reality spatial configuration with a zed mini stereoscopic camera,” in *FDIBA Conference Proceedings, vol. 2, 2018*, vol. 2, 11 2018, pp. 21–24.

[2] P. Sinha and E. Adelson, “Recovering reflectance and illumination in a world of painted polyhedra,” in *1993 (4th) International Conference on Computer Vision*, 1993, pp. 156–163.

[3] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760.

[4] C. Chaijirawiwat, “Monocular Depth Estimation via Transfer Learning and Multi-Task Learning with Semantic Segmentation,” Bachelor’s thesis, Tokyo Institute of Technology, Tokyo, Jul. 2019.

[5] M. A. Ganaie, M. Hu, M. Tanveer, and P. N. Suganthan, “Ensemble deep learning: A review,” *CoRR*, vol. abs/2104.02395, 2021. [Online]. Available: https://arxiv.org/abs/2104.02395

[6] S. F. Bhat, I. Alhashim, and P. Wonka, “Adabins: Depth estimation using adaptive bins,” *CoRR*, vol. abs/2011.14141, 2020. [Online]. Available: https://arxiv.org/abs/2011.14141

[7] 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. [Online]. Available: http://arxiv.org/abs/1907.10326

[8] M. Song, S. Lim, and W. Kim, "Monocular depth estimation using laplacian pyramid-based depth residuals," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[9] 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. [Online]. Available: http://arxiv.org/abs/1911.11907

[10] 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. [Online]. Available: http://arxiv.org/abs/1903.03273

[11] "Convolutional neural networks." [Online]. Available: https://www.coursera.org/learn/convolutional-neural-networks/home/welcome.

[12] 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. [Online]. Available: http://arxiv.org/abs/1406.2283

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

Data Augmentation

- Data augmentation refers to techniques to prevent overfitting by generating more (**possible**) training examples from original data
- Follow data augmentation techniques described in [6]:
  - **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
  - **Random rotation** of degree in a range of $[-2.5, 2.5]$

Evaluation Metrics

- Follow evaluation metrics described in [12]:

$$\text{ThreAcc}\left(\hat{\mathbf{d}}, \mathbf{d}; \delta\right) = \frac{|S_\delta|}{N} \times 100\%$$

$$\text{RMSE}\left(\hat{\mathbf{d}}, \mathbf{d}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{d}_i - d_i\right)^2}$$

$$\text{REL}\left(\hat{\mathbf{d}}, \mathbf{d}\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\hat{d}_i - d_i\right|}{d_i}$$

$$\text{RMSElog}\left(\hat{\mathbf{d}}, \mathbf{d}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\hat{d}_i - d_i\right\|^2}$$

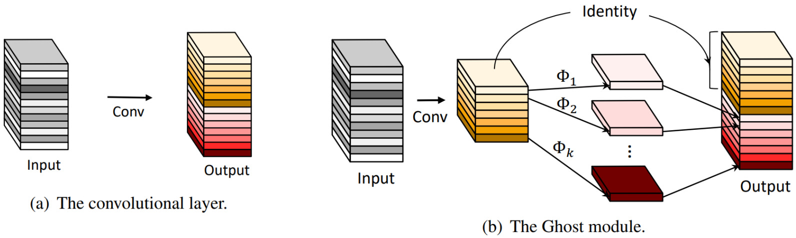
$$\text{SqREL}\left(\hat{\mathbf{d}}, \mathbf{d}\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left\|\hat{d}_i - d_i\right\|}{d_i}$$

$$\text{log10}\left(\hat{\mathbf{d}}, \mathbf{d}\right) = \frac{1}{N}\sum_{i=1}^{N}\left|\log_{10}(\hat{d}_i) - \log_{10}(d_i)\right|$$

where $S_\delta = \left\{d_i \middle| \max\left(\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}\right) < \delta \text{ and } i \leq N\right\}$ and $\delta = 1.25, 1.25^2, 1.25^3$

GhostNet [9]

- 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 ghost modules arranged in a structure similar to **MobileNetV2**



Traditional convolutional layer and The proposed Ghost module [9]