UNIVERSITY OF CALIFORNIA SAN DIEGO

Learning Augmentation Policy Schedules for Unsupervised Depth Estimation

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Electrical and Computer Engineering (Machine Learning and Data Science)

by

Aman Raj

Committee in charge:

Professor Dinesh Bharadia, Chair
Dr Gaurav Bansal
Professor Sujit Dey
Professor Truong Nguyen

2020

The thesis of Aman Raj is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California San Diego

2020

DEDICATION

To my father and mother, and my fiancée.

EPIGRAPH

Dare to be free, dare to go as far as your thought leads, and dare to carry that out in your life.

—Vivekananda

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This thesis is impossible without the many motivating discussions with my advisor Professor Dinesh Bharadia and Co-advisor Dr Gaurav Bansal. I would also like to extend my warm gratitude to the committee members for their valuable suggestions and feedback. Finally, I would like to thank colleagues and fellow students at UC San Diego with whom I have closely collaborated with.

VITA

2016	B. Tech. in Electronics and Communication Engineering, Delhi Technological University, Delhi
2020	M. S. in Electrical and Computer Engineering(Machine Learning and Data Science), University of California San Diego

PUBLICATIONS

Yue Meng, Yongxi Lu, Aman Raj, Samuel Sunarjo, Rui Guo, Tara Javidi, Gaurav Bansal, and Dinesh Bharadia, "SIGNet: Semantic Instance Aided Unsupervised 3D Geometry Perception", *in 32nd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

ABSTRACT OF THE THESIS

Learning Augmentation Policy Schedules for Unsuperivsed Depth Estimation

by

Aman Raj

Master of Science in Electrical and Computer Engineering (Machine Learning and Data Science)

University of California San Diego, 2020

Professor Dinesh Bharadia, Chair

This thesis represents a novel approach to augment data for unsupervised depth estimation. Data augmentation is an effective method to improve the performance of neural network models. However, such data augmentation strategies need to be hand-designed for the task for which the network is trained. In this work, we focus on learning data augmentation strategies from data itself using population-based training. We show the effectiveness of this approach in unsupervised learning setting for depth estimation task in monocular videos on KITTI dataset.

Chapter 1

Introduction

Sensors are the heart of an autonomous driving system. Such systems operate in a wide variety of settings which offer it's own set of challenges. For example an urban setting as shown in Figure 1.1 has challenges like pedestrian, cyclist, slow moving traffic etc. Self-driving cars perceive the environment using an array of sensors such as Radar, Lidar, and Camera. Cameras are widely used as they offer very dense information of the scene at a very cheap price compared to sensors like LIDAR. Depth estimation from camera feed is core of tasks such as 3D bounding box detection. Therefore, generating high quality depth from monocular color videos is very attractive as it can either complement or in some cases replace the expensive LIDAR. However, performance of camera is highly dependent on ambient lighting conditions, and hence can perform poorly in situations like shadow. Moreover, current state of the art depth estimation methods suffer badly in challenging weather conditions such as fog, rain, snow and speed blur. Some examples are shows in Figure 1.2. There are many opensource datasets to evaluate depth estimation methods for autonomous driving application, out of which KITTI is most widely used.

Current state of the depth estimation methods on KITTI [14][21][7] are based on supervised learning and rely heavily on ground truth labels. But, obtaining high quality ground truth is a formidable task as it requires expensive sensors and careful calibration procedures. Also,

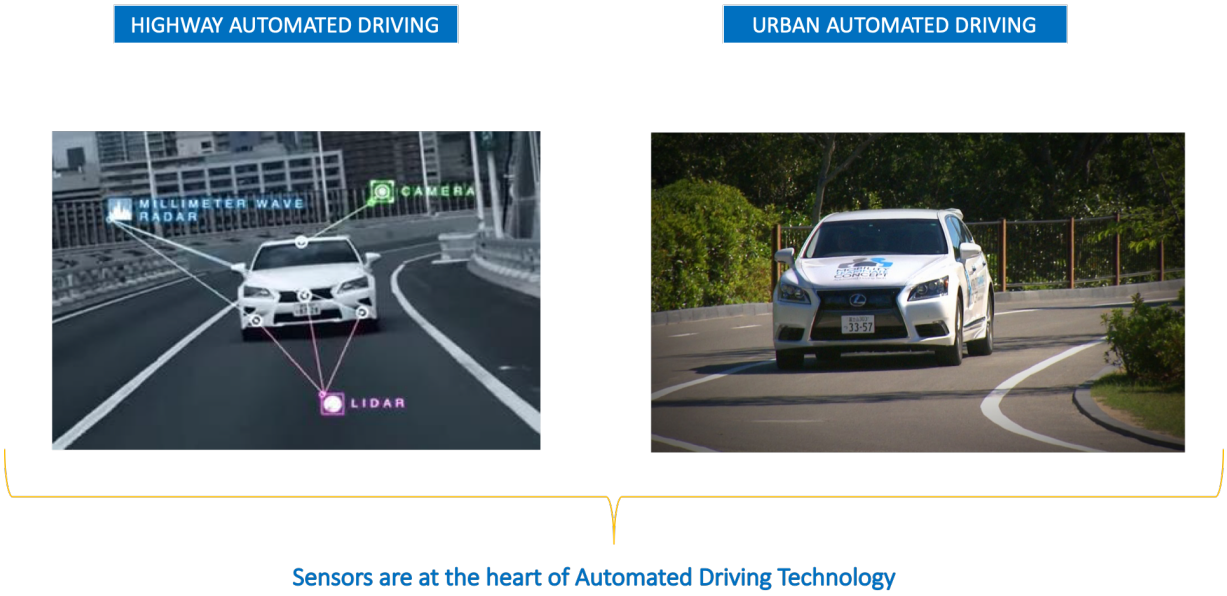

Figure 1.1: Example of two different settings for automated driving vehicles.

Figure 1.2: Examples of some real world bad weather conditions where depth estimation is very challenging.

these learned strategies are not transferable to other datasets that lack ground truth annotations. In order to overcome such limitations, there are many recent research works that use unsupervised learning techniques on KITTI [17][2][9][33][32] [25] to solve depth estimation problem using only RGB inputs. These methods take input monocular RGB video and estimate depth as a primary task with ego-motion estimation as an auxiliary task and then a spatial constraint is applied to generate learning signal. Since, such methods focus on autonomous driving applications our work is inspired from their setup. An Unsupervised learning setup doesn't help in making depth estimation robust to bad weather conditions. One way is to augment the color data input to the depth estimation model with additional prior knowledge about the environment such as

using semantic segmentation map to provide semantic consistency as guidance. This is very well explored in our previous work [17]. The other way is learn to augment color data from itself, generating wide variety of scenarios including weather conditions. In this work we explore this direction and seek to understand the effectiveness of learning data augmentations strategies for unsupervised depth estimation in self-driving scenario. We also devise novel data augmentations that make our depth estimation robust to bad weather conditions and evaluate on real word scenes. Our contributions are summarized in Section 1.1.

1.1 Contributions

Although our contributions are designed for depth estimation, we think it can be also used for other tasks that requires the geometrical structures in images to remain intact after application of data augmentation operations. In summary, our main contributions are listed below:

- Design and implement a search method that jointly optimizes network parameters with hyperparamters of data augmentation policies for depth estimation task in unsupervised setting.
- Demonstrate the benefit of learning data augmentation policy schedules over fixed augmentation strategy or random augmentation policy schedule.
- Propose a set of novel data augmentation operations that imitate various challenging weather conditions in an autonomous driving scenario. We also introduce a new version of KITTI Eigen split with such augmentations that can be used for bench-marking unsupervised depth estimation methods to measure their robustness towards weather conditions.

1.2 Summary of Results

- Our proposed method is competitive and even beats current state of the art MonoDepth2 [9] with a small margin on KITTI depth estimation benchmark using the split by Eigen [6].
- Proposed data augmentation operations to imitate weather conditions help to increase robustness of depth estimation of our method in real world challenging bad weather conditions.
- Learning augmentation policy schedules shows consistent superior performance over using a fixed augmentation policy for depth estimation task.

Chapter 2 discusses related work of supervised and unsupervised depth estimation methods. We also describe a set of work that learns data augmentation strategies for classification tasks on COCO.

Chapter 3 first explain our unsupervised depth estimation setup, then we introduce various augmentation operations used in this work. Here, we introduce and describe in detail our augmentation operations that simulate challenging weather conditions. We also ponder over the necessity of learning augmentation policy schedules and the setting to learn the same and discuss complexity issues associated with it.

Chapter 4 finally present our finding and comparison with state of the art, through various experiments and ablation studies we establish the benefits of learning policy schedules over fixed policy schedule. We also show robustness of our method on weather simulation as well as some real world bad weather examples.

Chapter 5 discusses conclusion and some future directions of this work.

Chapter 2

Related Work

Most of the recent supervised methods on KITTI for depth estimation use CNN model and generate supervisory signal using ground truth depth from LIDAR. In the pioneering work by Eigen in [6], a coars to fine network was introduced that utilizes a scale-invariant loss to improve the accuracy of the estimated depth map. [7] introduces a deep ordinal regression based network for depth estimation that reports the current state of the art on KITTI eigen split. [29] use the skip-connection based CNN that fuses low-spatial resolution depth map in deeper layers with high-spatial resolution depth map in lower layers. All these methods requires ground truth depth information while training. In the absence of ground-truth depth, various unsupervised learning methods use photo-metric image reconstruction loss as supervisory signal to train deep learning models for depth estimation. [33] proposed a fully differential learning approach that jointly predicts depth and ego-motion using separate networks. [25] additionally handle object motion in the scene by predicting a motion mask, however their online version achieves superior performance by disable this feature. [12] improved depth by training on stereo pairs and extracts features from left-right images using a siamese network. [32] decompose the motion in scene into rigid and non-rigid components, depth estimation branch of network models rigid motion and optical flow estimation branch models non-rigid motions. However,

they don't report any performance gains when both branches are trained jointly. Works by the authors of [31][30] enforce geometrical constraints on predicted depth as well as surface normals and edge to encourage sharp depth discontinuities. [2] uses pre-computed instance segmentation masks, to separately handle object motion from ego-motion of scene, they report very competitive performance only after online-refinement. [17] achieves large improvement on depth and flow prediction of dynamic objects in scene by using pixel-wise and instance segmentation of scene as additional prior knowledge. Our unsupervised depth estimation backbone uses the novelties introduced in GeoNet [32] and MonoDepth2 [9].

Almost all supervised and unsupervised depth estimation methods use some form of data augmentations strategies. Commonly used augmentation operations are horizontal flip, translation, random scale and crop. Augmentation puts a strong regularization effect on the model training and has shown to improve the accuracy of modern classifiers by increasing its generalization ability. On MNIST, augmentation strategies like elastic distortions across scale, position, and orientation have been applied to achieve state of the art results [23] [3] [22][26]. GANs have been also used to augment data on the fly [1][19][20][24][34]. Recent works aim to learn optimal data augmentation strategies from data itself. AutoAugment [4], uses reinforcement learning to determine a sequence of augmentation operations along with magnitude and probability associated with each such operation. PBA [10] proposed a new algorithm to learn augmentation policy schedules at significantly low computational cost that leverages population based training [11]. Similar to PBA, [15] proposed hyperparameter optimization to learn optimal policies using Tree-structured Parzen Estimator (TPE). [35], finds optimal data augmentation policies for object detection task. While all of the above methods have worked on either classification problems or objection detection problem in supervised learning setting, in this work we seek to find optimal data augmentation strategies and thoroughly investigate its potential for unsupervised depth estimation in self-driving scenes. We also propose, a set of augmentation operations that aims at making depth estimation robust in challenging situation such as various weather conditions.

Chapter 3

Methodology

3.1 Unsupervised Depth Estimation

We would like to take a digression and introduce our unsupervised depth estimation setup in this chapter. In nutshell, it takes input a short video sequence and predict 3D geometry information such as depth and relative camera poses. This information is then utilized to project 2D frames to 3D space, followed by 3D alignment and re-projection back to 2D to reconstruct one of the neighboring frames in the sequence. The frame reconstruction error is then used to train the model.

As shown in Figure 3.1 input to the method is a sequence of n RGB frames where $n \geq 3$, and from neighboring source images a target image is reconstructed. If I_t denotes target image, and I_s each source image, the model predicts depth for target image D_t and relative camera pose between target and source image $\hat{T}_{t\to s}$ and minimizes the photo-metric re-projection error L_p given by:

$$L_p = \sum_s pe(I_t, \hat{I}_s)$$

$$where \quad \hat{I}_s = I_s\langle proj(D_t, T_{t\to s}, K)\rangle \quad and \quad s \in \{t-1, t+1\}$$

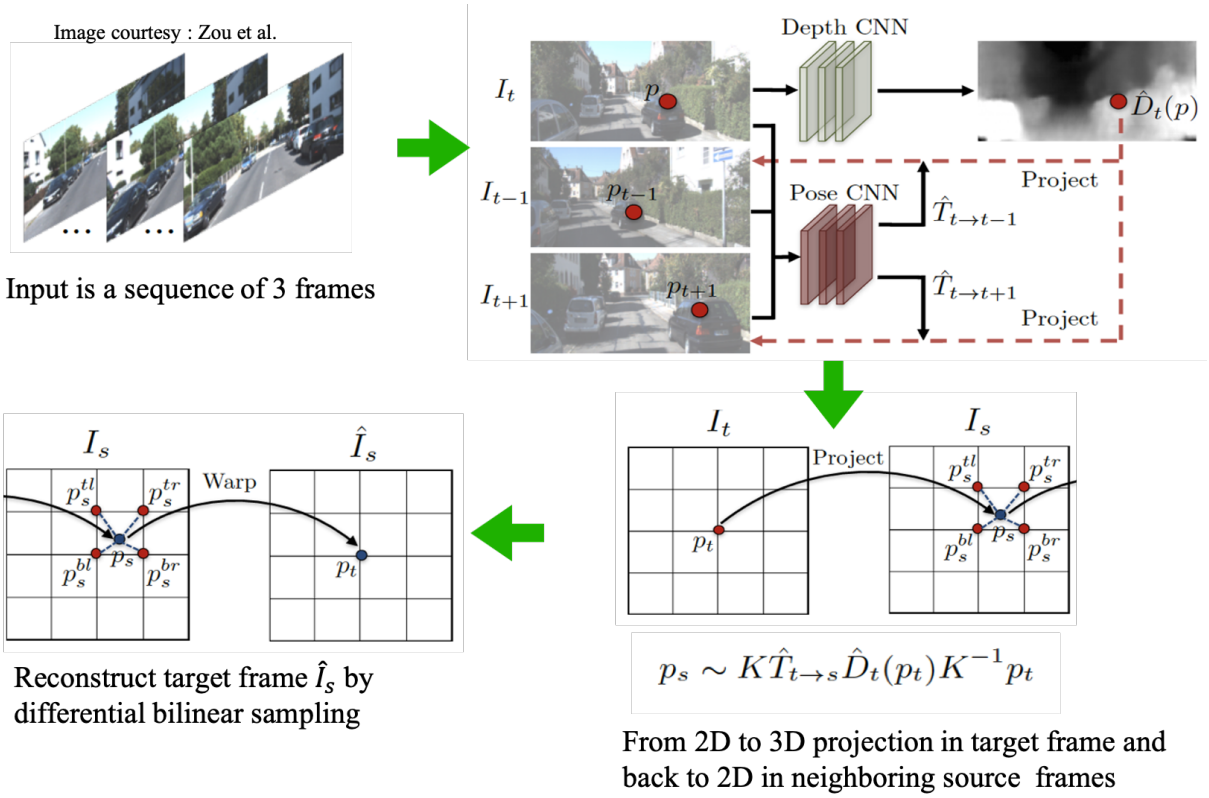

Figure 3.1: Overview of setup for learning unsupervised depth

Here, pe is called photo-metric reconstruction error, K is camera intrinsic matrix, $proj()$ is resulting 2D coordinates of projected depth D_t in source frame I_s, and $\langle\rangle$ denotes differential bilinear sampling operation. Final pe computation contains a weighted sum of $L1$ error between pixels and SSIM (structural similarity index) [28] given by:

$$pe(I_t, \hat{I}_s) = \frac{\alpha}{2}(1 - SSIM(I_t, \hat{I}_s) + (1-\alpha)\|I_t - \hat{I}_s\|_1$$

where $\alpha = 0.85$. In order to encourage sharp depth discontinuities based on image gradients, standard practice is to include edge-aware depth smoothness loss L_s given by:

$$L_s = \sum_{p_t} |\nabla D_t(p_t)|.(e^{-|\nabla I(p_t)|})^T$$

where $|.|$ elementwise absolute operation, ∇ is vector differential operator and T denotes transpose of gradients. Pixels that are visible in target image but occluded in source images, can contaminate the loss by giving large $L1$ errors, MonoDepth2 [9] handles this problem by taking minimum photo-metric error at each pixel over all source images.

$$L_p = \min_s pe(I_t, \hat{I}_s)$$

Unsupervised monocular training for depth assumes a static scene and moving camera, when theses assumptions are violated, the performance suffers greatly. MonoDepth2 [9] introduces automasking term that keeps loss from only those pixels where re-projection error between warped image $\hat{I}_s$ and target image I_t is lower than that between source image I_s and target image I_t. This is achieved by a masking terming μ:

$$\mu = [\min_s pe(I_t, \hat{I}_s) < \min_s pe(I_t, I_s)]$$

where $[\quad]$ is inversion bracket. All losses are computed at multiple scales, with number of scales set to 4. Liked [9], we upsample the predicted depth at lower scales to original input resolution before applying bilinear image warping operation. The final loss is given by : $L = L_p + \lambda L_s$, where $\lambda = 0.5$. Our DepthCNN is uses a ResNet50 based encoder and PoseCNN implementation is taken from GeoNet [32].

3.2 Data Augmentation

We give a review of augmentation operations that we have identified for our unsupervised depth estimation task. Since, our depth estimation require that geometrical structures in image should remain intact after augmentation, so we discard operations such as Shear, Translation, or Rotation that changes geometry of image and mostly focus on color operations. We also introduce

new augmentation operations that we discuss in Section 3.2.2. While augmenting a training sample we make sure to use same operation for all images in the sequence. For operations that scales or crop the image, we accordingly change the camera intrinsic matrix K for the frames in given sequence.

3.2.1 Traditional Data Augmentation

From PBA work [10], we adapt following color operations: `Brightness`, `Color`, `Invert`, `Sharpness`, `Posterize`, `Solarize`, `Equalize`, `AutoContrast`, `Cutout`, `Contrast` and call this set as *Trad-10*. We also introduce some additional new operations such as: `Blur`, `Smooth`, `EdgeEnhance`, `FlirLR`, `ScaleCrop` and call this set as *Trad-5*. All these operations are largely derived from transformations in Python Image Library (PIL).

3.2.2 Weather based Data Augmentation

In order to make our depth estimation for self-driving cars robust to various kinds of weather conditions, we introduce new augmentation operations which imitates weather conditions like rain, snow, fog and blur caused by high speed. Examples of such operations applied on an input image is shown in Figure 3.2. We call this set as *Weather*. These operations are implemented using Opencv and PIL. We define the algorithmic implementation of these operations below:

- *Rain* : We know since rainy weather is dark in general, so we first darken the image by reducing its brightness to 80% of it original value. Then we generate some random lines having width 1 pixel, length 15 pixels and slant $\alpha \in [-20, 20]$ and set its color to RGB value $(200, 200, 200)$. Next, we overlay these lines on darken image and apply a normalized box filter of kernal size 4×4. During search the parameter α is treated as magnitude hyperparamter.

- *Snow* : We set a threshold $\tau = 255(\frac{\alpha}{2} + \frac{1}{3})$, where $0 < \alpha <= 0.5$. For all the pixels in

image where its value is less than τ, increase lightness by a factor of 2. During search, the parameter α is treated as magnitude hyperparamter. The equation is obtained empirically on KITTI with the aim to allow only a certain percentage of pixels to be changed.

- *Fog* : Generate some random points with coordinates (x, y), treat these points as center and draw circles with varying radius and the apply blur. During search the width of circumference parameter α is treated as magnitude hyperparamter.

- *SpeedBlur* : We apply a custom 2D filter of size 5×5 on right and left side of image where top half of filter is set to zero and bottom half is set to one. We only learn probability hyperparameter for this operation. Filter is obtain empirically on KITTI.

Figure 3.2: Examples of augmentation operations applied to imitate weather conditions.

3.3 Learning Data Augmentation Policy Schedules

3.3.1 What are Augmentation Policy Schedules?

Data augmentation policy is basically a list of *(op, prob, mag)* where *op*: augmentation operation, *prob* : probability of applying operation and *mag* : magnitude with which operation is applied. During training at any epoch X number of operations are chosen and applied to a training sample.

$$eg.[(Brightness, 0.5, 8), (Rotate, 0.2, 8), (Translate, 0.7, 10), \dots]$$

Data augmentation policy schedule is a list of data augmentation policies, where policy at index i specifies augmentation policy to be used for i^{th} epoch of training.

3.3.2 Why Augmentation Schedules?

A classical approach of data augmentation is to select a set of operations and apply all of them on an example during training with some probability. However, as the number of operations increases, it becomes computationally expensive to apply all of them on a training example. Most recent works [15][4][35] tries to learn a fixed augmentation policy which aims to maximize a reward attribute eg. validation accuracy, chosen for the selected task. However, augmentation operations that can reduce generalization error of classifier during later epochs of training is not necessarily a good function during initial epochs. This intuition leads to the idea of learning a schedule of augmentation policies rather than a fixed policy.

3.3.3 Learning Schedules

In order to learn augmentation policy schedules we use Population Based Augmentation (PBA) introduced in [10]. It leverages Population Based Training [11] to learn the best augmentation policy for each epoch of training. PBT [11], jointly optimise a population of models and

their hyperparameters to maximise performance on a given task. In order to learn a schedule we choose a subset of training set from KITTI Eigen split[5] that has 5,000 samples where each sample is a sequence of 3 frames. During search we jointly optimize model parameters and their hyperparametrs in an asynchronous fashion and outputs augmentation policy schedules. Similar to [10], our algorithmic implementation of PBA uses PBT scheduler in Ray[18] and uses a population of 8 child models, where the probability *prob* and magnitude *mag* values of augmentation operations are treated as a set of hyperparameters. We discard the trained child models after PBA search and use the learned policy schedule of child model that gave lowest error for full scale training. Full scale training means training on original size of train set with augmentation policy schedule. Our choice of using PBA to learn augmentation policy schedules over other algorithms is solely based on its efficiency as it offers the lowest computational cost compared to other methods.

3.3.4 Search Space Complexity

Algorithm 1 discusses the modified version of augmentation template from PBA, that our work uses, where we tune the hyperparameter *count* for different experiments on KITTI. This algorithm takes an augmentation policy as input and apply upto 3 augmentations in sequence. Our augmentation policy is a set of operations where each operation is associated with two probability and magnitude values. If we choose N number of operations, this give us $2N$ operation-probability-magnitude tuples, and in total $4N$ hyperparameters. Each hyperparameter is either probability having 11 possible discrete values (0 to 10) or magnitude having 10 possible discrete values (0 to 9). Therefore each policy is sampled from a search space that has $(10 \times 11)^{2N}$ possibilities. We experiment with different values of N, and discuss our finding in Chapter 4. We now present the steps involved in policy search in PBA to complete our discussion about it.

Working of PBA algorithm can described as follows:

Step: An epoch of gradient descent on training set.

Eval: Evaluation of each trial on validation set.

Ready: A trial is deemed ready to go through exploit-and explore process after every step.

Exploit: The process in which a trial in bottom 50% of population clones the weights and hyperparameters of a trial in the top 50% of population. We use Abs Rel (absolute relative error) value on test set to divide the population in top and bottom halves.

Explore: Algorithm 2 explains the exploration function where for each hyperparameter we either uniformly re-sample from all possible values or perturb the original value.

Algorithm 1: Our modified version of PBA augmentation policy template [10].

Input: data x, **policy** p, [list of parameters *(op, prob, mag)*];
Shuffle parameters ;
Set *count* = [0, 1, 2, 3] with probability [p_1, p_2, p_3, p_4] where $\sum_i p_i = 1$;
for *(op, prob, mag) in p* **do**
    if *count = 0* **then**
        break;
    end
    if *random(0,1) < prob* **then**
        count = count - 1 ;
        x = op(x, mag) ;
    end
end
return *x*

Algorithm 2: The PBA explore function from [10].

Input: Params p, list of augmentation hyperparameters ;

for *param in p* **do**

  if *random(0,1)* $<$ *0.2* **then**

    Resample *param* uniformly from domain;

  else

    amt = [0,1,2,3] uniformly at random ;

    if *random(0, 1)* $<$ *0.5* **then**
      param = param + amt

    else
      param = param - amt

    end

    Clip *param* to stay in domain

  end

end

Chapter 4

Experiment and Analysis

To highlight the benefits that learning data augmentation policy schedule brings to unsupervised depth estimation, we designed the experiments similar to MonoDepth2 [9]. First, we show our best model's performance on test set of KITTI dataset [8] in Table 4.2 comparing it with previous works in similar setting. Next, through ablation studies in Table 4.3 we show contribution of augmentation operations introduced in Section 3.2. We also introduced a new version of KITTI eigen test set with simulated weather conditions, and evaluate and compare our models in Table 4.4 and Table 4.5. We also do a study to reduce complexity of search space in the hope of finding better solution for policy schedules and report our findings in Table 4.6.

4.1 Implementation Details

KITTI dataset is divided into train set having 40238 samples (3 frame sequences) and test set (single image) having 697 samples as per Eigen split [6]. For PBA search our population contains 8 child models and we randomly sample 5000 sequences from train set and use *Abs Rel* error as metric to minimize. Search is performed for 35 epochs and requires 8 Nvidia 1080Ti GPUs. During full scale training, we load best policy and training needs only one GPU. We use Adam optimizer [13], with initial learning rate of 2e-4 and learning rate decay multiplier 0.1 after

every 15 epochs, we train our models for 35 epochs.

4.2 Monocular Depth Evaluation on KITTI

We use the learned data augmentation policy schedule to augment the image sequences during training and adopt scale normalization as suggested in [27]. The normalized plot of probability of different augmentation operations learned in policy schedule is shown in the Figure 4.1, and their corresponding magnitudes are shown in Figure 4.2. This policy schedule is optimized to improve generalization error on KITTI, for example applying `color` operation with higher probability and magnitude is beneficial in epoch range 15-20 than compared to epoch range 30-35. We also visualize some examples of augmented image from learned policy schedule during different epochs of training in Figure 4.3.

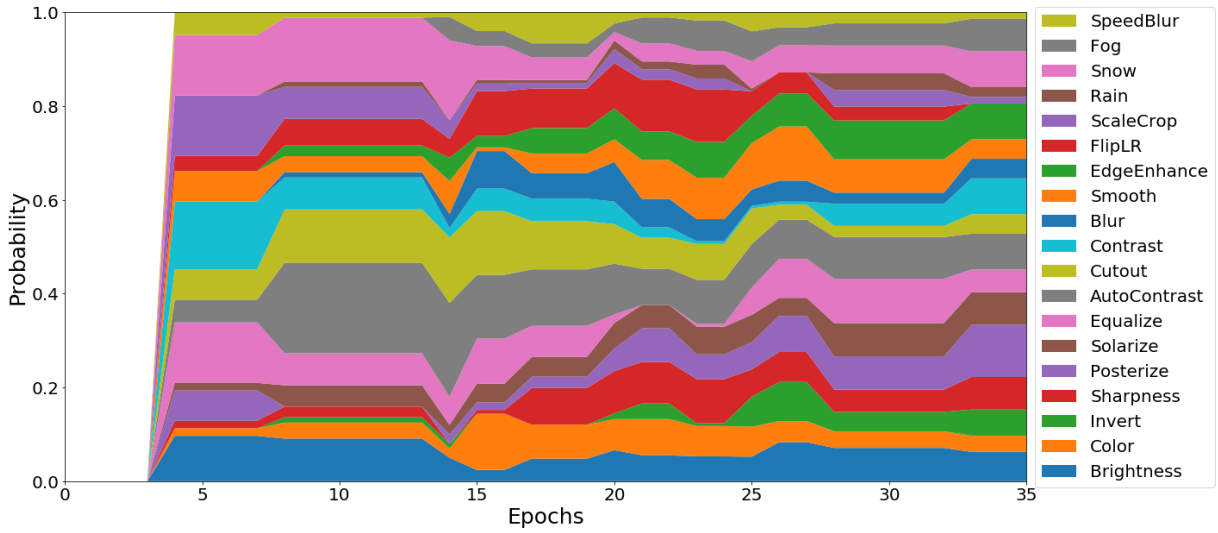

Figure 4.1: Normalized plot of probability parameters of different augmentation operations over epochs. The distribution flattens out towards the end of training.

In the evaluation phase, ground truth depth maps were generated by projecting LIDAR points to image plane. As per [32], we clip our depth estimation in range 0.001m and 80m and calibrate the scale by medium number of ground truth. Table 4.1 shows the definition of

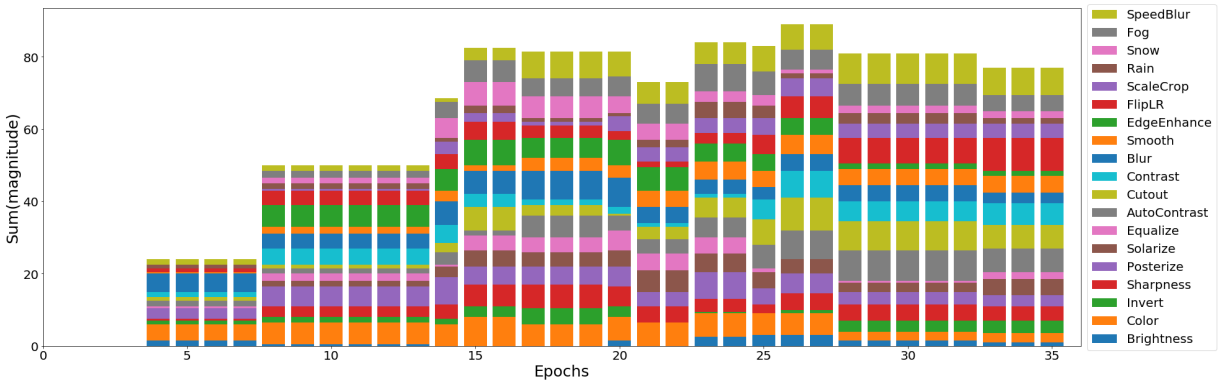

Figure 4.2: Plot of magnitude parameters of augmentation operations, it increases rapidly in the initial phase of training, and later becomes stable. Since we have two magnitude values for each operation, we take their average for visualization.

evaluation metrics that we use. We compare our best method *Ours* with current state of the art unsupervised depth estimation methods in Table 4.2 where our method beats the current state of the art MonoDepth2 [9] by a small margin. We also trained a model with randomly generated fixed policy called *Ours(random policy)* and compare it with our method *Ours* that uses learned policy schedule in Table 4.2. Results show learning policy schedules achieves higher performance than a fixed random policy over all metrics. For random method we do four trials and report average values.

Table 4.1: Definition of evaluation metrics as per [6] which is commonly used to evaluate depth estimation on KITTI. Here y^* is ground truth depth and y is predicted depth.

Threshold: $\%of y_i, s.t. max(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}) = \delta < thr$	RMSE(linear): $\sqrt{\frac{1}{\lvert T\rvert}\sum_{y\in T}\lvert\lvert y_i - y_i^*\rvert\rvert^2}$
Abs Relative difference: $\frac{1}{\lvert T\rvert}\sum_{y\in T}\lvert y - y^*\rvert / y^*$	RMSE(log): $\sqrt{\frac{1}{\lvert T\rvert}\sum_{y\in T}\lvert\lvert logy_i - logy_i^*\rvert\rvert^2}$
Squared Relative difference: $\frac{1}{\lvert T\rvert}\sum_{y\in T}\lvert\lvert y - y^*\rvert\rvert^2 / y^*$	

4.3 Ablation Studies

How many augmentation operations are necessary ? Search space complexity increases exponentially with number of operations as discussed in Chapter **??**. We performed an

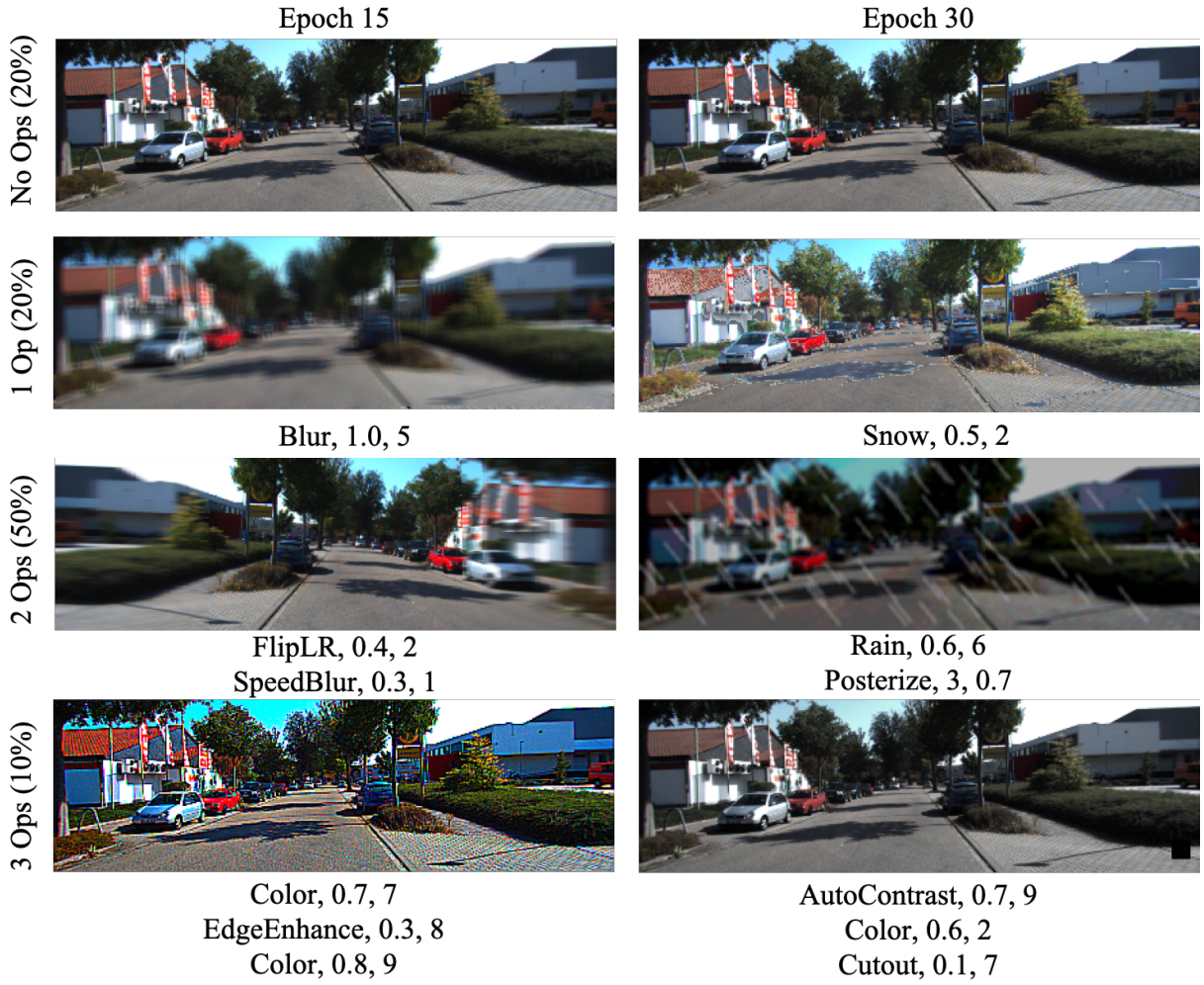

Figure 4.3: Examples of images showing which augmentation operations were applied during training with what probability and magnitude. These parameters vary at different epochs of training learned as policy schedule.

ablation study to tune the number of augmentation operations for KITTI, results in Table 4.3 shows our method *Ours-10AugW* can achieve performance close to our best model with less operations and still beats state of the art MonoDepth2. This effectively reduces the expanse of search space and can help us to find better solutions.

How to measure robustness of our method in bad weather conditions ? We generate a version of KITTI test set that simulates weather conditions and evaluate our methods on it. SSIM (structural similarity index) between original and augmented version is used to measure the

Table 4.2: Monocular depth estimation results on KITTI 2015 [8] by the split of Eigen [6]. **D** stand for supervised depth estimation methods and **M** stands for monocular unsupervised depth estimation methods. For fair comparison, we use *MonoDepth2* model checkpoint from github(*mono_no_pt*_640*x*192) and evaluate using per image median scaling.

Method	Train	Error-related metrics				Accuracy-related metrics		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen [6]	D	0.203	1.548	6.307	0.282	0.702	0.890	0.957
Liu [16]	D	0.201	1.584	6.471	0.273	0.680	0.898	0.967
DORN [7]	D	**0.072**	**0.307**	**2.727**	**0.120**	**0.932**	**0.984**	**0.994**
Zhou [33]	M	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Yang [31]	M	0.182	1.481	6.501	0.267	0.725	0.906	0.963
GoeNet [32]	M	0.149	1.060	5.567	0.226	0.796	0.935	0.975
DF-Net [36]	M	0.150	1.124	5.507	0.223	0.806	0.933	0.973
DDVO [27]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974
Struct2depth '(M)' [2]	M	0.141	1.026	5.291	0.215	0.816	0.945	**0.979**
MonoDepth2 [9]	M	0.132	1.057	5.150	0.210	**0.844**	**0.947**	0.977
Ours(random policy)	M	0.134	0.951	5.210	0.210	0.828	0.944	0.978
Ours	M	**0.129**	**0.920**	**5.089**	**0.205**	0.840	**0.947**	0.978

Table 4.3: Ablation study of depth estimation performance gains on KITTI Eigen split due to different augmentation operations used in learning policy schedule.

Method	Trad-10	Trad-5	Weather	Error-related metrics				Accuracy-related metrics		
				Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
MonoDepth2 [9]				0.132	1.057	5.150	0.210	**0.844**	0.947	0.977
Ours-10Aug	✓			0.131	0.926	5.202	0.208	0.834	0.945	0.978
Ours-15Aug	✓	✓		0.133	0.962	5.299	0.212	0.828	0.943	0.977
Ours-10AugW	✓		✓	0.130	0.937	5.193	0.208	0.837	0.945	0.978
Ours	✓	✓	✓	**0.129**	**0.920**	**5.089**	**0.205**	0.840	**0.947**	**0.978**

loss of information after augmentation, we keep only those samples where $SSIM \geq 0.40$. This gives us $694/697$ images from original test set in case of applying single weather operation and $641/797$ in case of applying double weather operation. For both cases, we apply augmentation with probability 90%, this helps us to create a test set that has mix of original images also. Results show in Table 4.4 and Table 4.5 shows learning policy schedule shows consistent gain over a fixed policy. We also generate some qualitative results using our best method and show them in Figure 4.4 where we are able to recover depth discontinuities blurred regions.

Can we reduce complexity of search space ? As we discussed previously reducing search space complexity can help to find better solution for policy schedules, leading to better generalization error. We explore following ways to reduce search space complexity and show the

Table 4.4: Comparison on weather simulated version of KITTI Eigen test set when a **single weather operation** is applied. After augmentation we keep only those samples where $SSIM \geq 0.40$ between original and augmented version.

Method	Error-related metrics				Accuracy-related metrics		
	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Ours-10Aug	0.149	1.097	5.540	0.227	0.798	0.930	0.972
Ours-15Aug	0.148	1.091	5.548	0.227	0.799	0.933	0.973
Ours-10AugW	**0.137**	0.993	5.341	0.215	**0.823**	0.941	0.976
Ours	0.138	**0.991**	**5.262**	**0.214**	0.821	**0.942**	**0.977**
Ours(random policy)	0.141	1.011	5.354	0.216	0.816	0.940	**0.977**

Table 4.5: Comparison on weather simulated version of KITTI Eigen test set dataset when **double weather operations** are applied in sequence. After augmentation we keep only those samples where $SSIM \geq 0.40$ between original and augmented version.

Method	Error-related metrics				Accuracy-related metrics		
	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Ours-10Aug	0.161	1.122	5.789	0.240	0.768	0.921	0.968
Ours-15Aug	0.157	1.178	5.702	0.235	0.779	0.928	0.972
Ours-10AugW	**0.141**	**1.030**	5.433	**0.219**	**0.812**	0.937	**0.976**
Ours	0.144	1.043	**5.368**	**0.219**	0.811	**0.938**	**0.976**
Ours(random policy)	0.145	1.068	5.454	0.220	0.808	0.937	**0.976**

results in Table 4.6:

1. Modify the search and learn only magnitude parameter for operations. *(Ours-mag-only)*
2. Despite of learning two different sets of parameters for an operation, learn only one set of parameters. *(Ours-one-ops)*
3. Choose top 10 operations from our best method in Table 4.2 based on their probability contribution in the normalized plot. *(Ours-reduce)*

Our method *(Ours-one-ops)* can achieve performance close to our best method with lesser complexity.

How does our method perform in real world settings ? First we show some qualitative results on challenging images from KITTI test set in Figure 4.5 that has problems like occlusion due to shadow which is a common in driving scene. We go a step ahead, and evaluate robustness

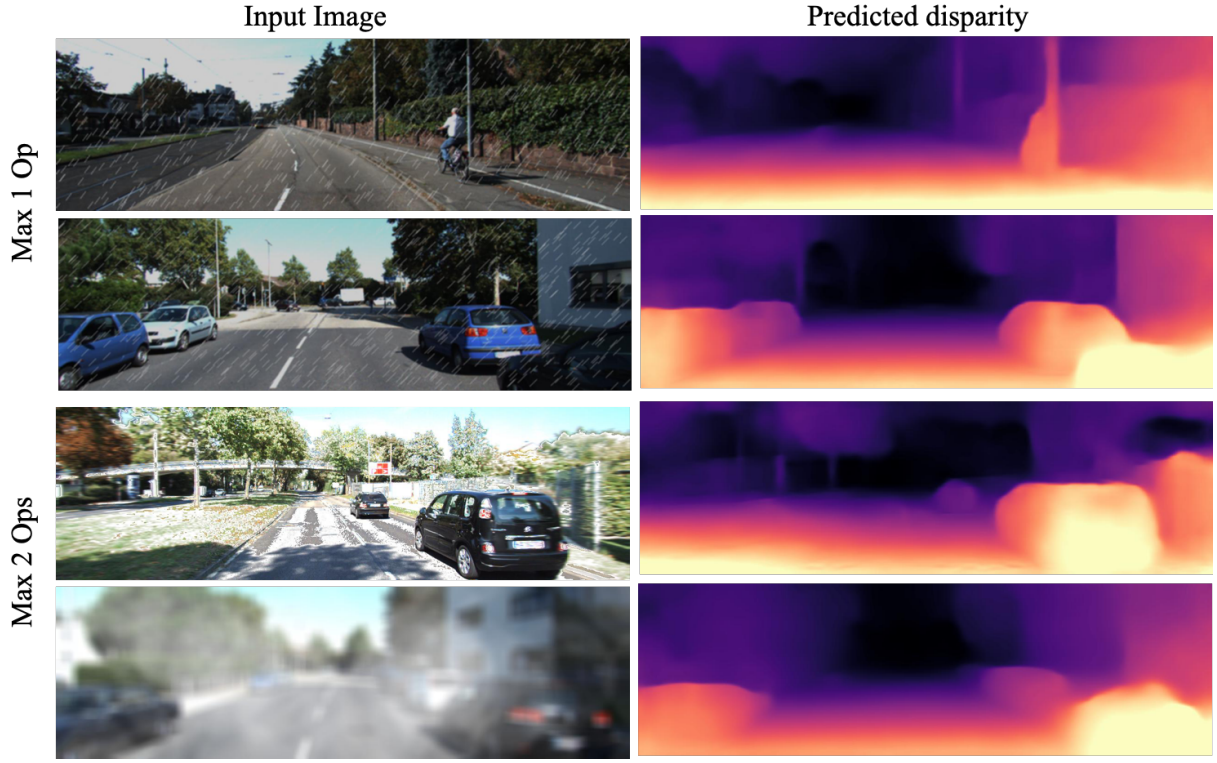

Figure 4.4: Examples of disparity estimation on samples from weather simulated version of KITTI Eigen test set using our best method *Ours* from Table 4.2

of our method on some challenging real world weather conditions. We samples some images from google images, and predict on those, results shown in Figure 4.6. As we can see we are able to achieve better depth estimation of vehicles compared to methods that doesn't use any weather based augmentations. It also suggest, by using operations that simulate weather conditions, we are able to better performance in good weather (KITTI test set) as well as bad weather (Figure 4.6).

Table 4.6: Comparison on test set of KITTI Eigen split of various methods with complexity variations in search space where $N = 19$ and $n \approx N/2$.

Method	Complexity	Error-related metrics				Accuracy-related metrics		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
MonoDepth2 [9]	-	0.132	1.057	5.150	0.210	**0.844**	0.947	0.977
Ours-mag-only	$(10)^{2N}$	0.134	0.957	5.182	0.209	0.832	0.945	0.978
Ours-one-ops	$(10 \times 11)^{N}$	0.130	0.931	5.194	0.207	0.836	0.947	0.978
Ours-reduce	$(10 \times 11)^{2n}$	0.131	0.944	5.159	0.208	0.835	0.946	0.978
Ours	$(10 \times 11)^{2N}$	**0.129**	**0.920**	**5.089**	**0.205**	0.840	**0.947**	**0.978**

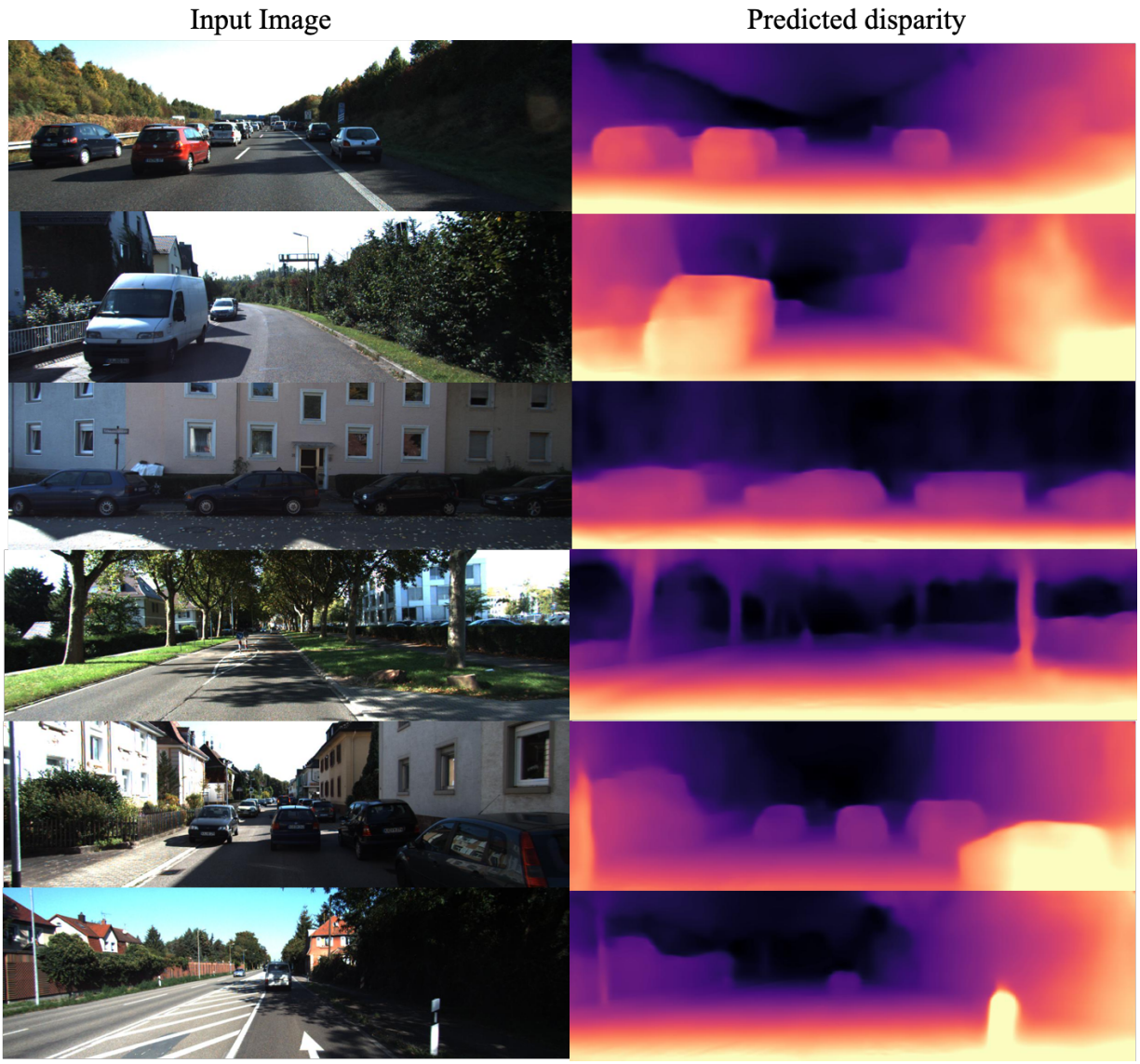

Figure 4.5: Examples of disparity estimation on samples from KITTI Eigen test set using our best method *Ours* from Table 4.2

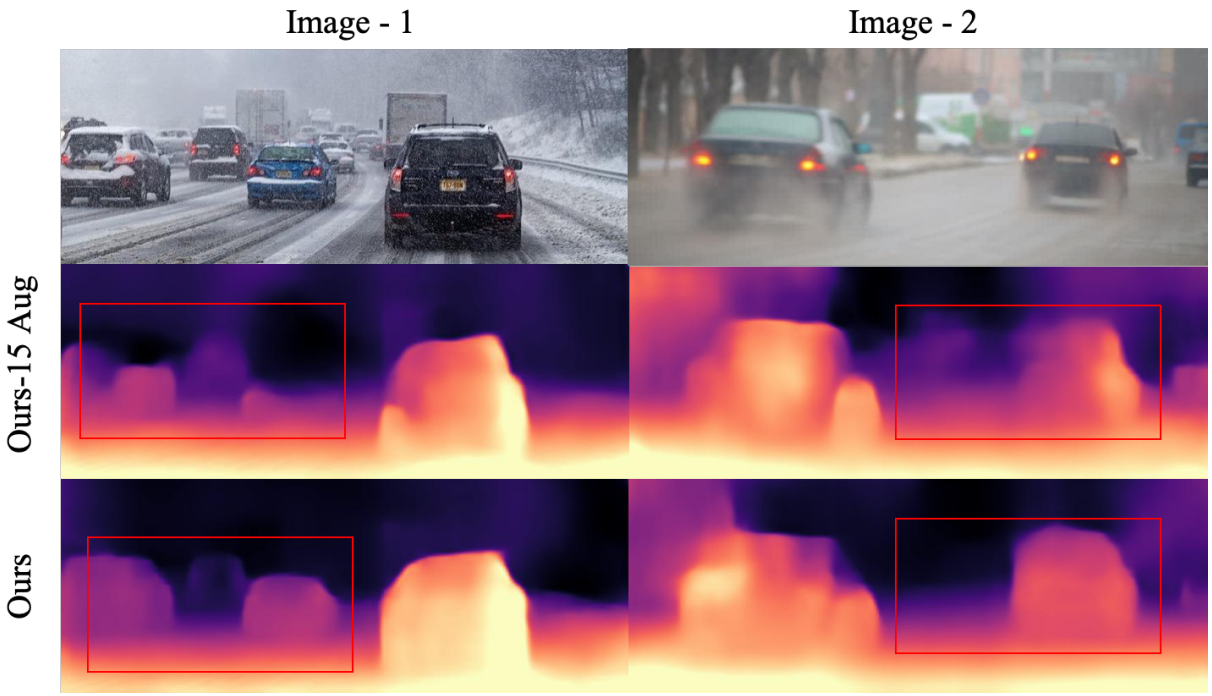

Figure 4.6: Examples of disparity estimation on samples taken from google images that show real world weather conditions using our best method *Ours* from Table 4.2. *Ours-15 Aug* doesn't use any weather based augmentation operations. We are able to recover depth of vehicles as well as detect crisp depth discontinuities.

Chapter 5

Conclusion and Future Directions

In this work, we investigated the idea of learning data augmentation strategies from data itself to make depth estimation from camera sensor robust for autonomous driving application. We used the setup of depth estimation in monocular videos in unsupervised setting to investigate this idea. We also proposed data augmentations to imitate weather conditions that increases robustness of our depth estimation network. Our method is competitive and even beats current state of the art with a small margin on KITTI benchmark. In future work we would like to investigate similar data augmentation strategies for semantic segmentation, where getting pixel wise annotation is very expensive and such methods can aid to provide free training examples at no extra human cost. Also, we think data augmentation strategies can be learned at object level too. For instance, we can design augmentation operations that can add virtual objects like introducing different models of cars in the scene to make classifier aware of new class of objects present in real world scenarios.

Bibliography

[1] ANTONIOU, A., STORKEY, A., AND EDWARDS, H. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).

[2] CASSER, V., PIRK, S., MAHJOURIAN, R., AND ANGELOVA, A. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 8001–8008.

[3] CIREGAN, D., MEIER, U., AND SCHMIDHUBER, J. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition* (2012), IEEE, pp. 3642–3649.

[4] CUBUK, E. D., ZOPH, B., MANE, D., VASUDEVAN, V., AND LE, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2019), pp. 113–123.

[5] EIGEN, D., AND FERGUS, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 2650–2658.

[6] EIGEN, D., PUHRSCH, C., AND FERGUS, R. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems* (2014), pp. 2366–2374.

[7] FU, H., GONG, M., WANG, C., BATMANGHELICH, K., AND TAO, D. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2002–2011.

[8] GEIGER, A., LENZ, P., AND URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), IEEE, pp. 3354–3361.

[9] GODARD, C., MAC AODHA, O., FIRMAN, M., AND BROSTOW, G. J. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 3828–3838.

[10] HO, D., LIANG, E., STOICA, I., ABBEEL, P., AND CHEN, X. Population based augmentation: Efficient learning of augmentation policy schedules. *arXiv preprint arXiv:1905.05393* (2019).

[11] JADERBERG, M., DALIBARD, V., OSINDERO, S., CZARNECKI, W. M., DONAHUE, J., RAZAVI, A., VINYALS, O., GREEN, T., DUNNING, I., SIMONYAN, K., FERNANDO, C., AND KAVUKCUOGLU, K. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).

[12] KHAMIS, S., FANELLO, S., RHEMANN, C., KOWDLE, A., VALENTIN, J., AND IZADI, S. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *The European Conference on Computer Vision (ECCV)* (September 2018).

[13] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[14] LEE, J. H., HAN, M.-K., KO, D. W., AND SUH, I. H. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326* (2019).

[15] LIM, S., KIM, I., KIM, T., KIM, C., AND KIM, S. Fast autoaugment. In *Advances in Neural Information Processing Systems* (2019), pp. 6662–6672.

[16] LIU, F., SHEN, C., LIN, G., AND REID, I. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence 38*, 10 (2015), 2024–2039.

[17] MENG, Y., LU, Y., RAJ, A., SUNARJO, S., GUO, R., JAVIDI, T., BANSAL, G., AND BHARADIA, D. Signet: Semantic instance aided unsupervised 3d geometry perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 9810–9820.

[18] MORITZ, P., NISHIHARA, R., WANG, S., TUMANOV, A., LIAW, R., LIANG, E., ELIBOL, M., YANG, Z., PAUL, W., JORDAN, M. I., AND STOICA, I. Ray: A distributed framework for emerging {AI} applications. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)* (2018), pp. 561–577.

[19] MUN, S., PARK, S., HAN, D. K., AND KO, H. Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane. *Proc. DCASE* (2017), 93–97.

[20] PEREZ, L., AND WANG, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621* (2017).

[21] REN, H., EL-KHAMY, M., AND LEE, J. Deep robust single image depth estimation neural network using scene understanding. *arXiv preprint arXiv:1906.03279* (2019).

[22] SATO, I., NISHIMURA, H., AND YOKOI, K. Apac: Augmented pattern classification with neural networks. *arXiv preprint arXiv:1505.03229* (2015).

[23] SIMARD, P. Y., STEINKRAUS, D., AND PLATT, J. C. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar* (2003), vol. 3.

[24] SIXT, L., WILD, B., AND LANDGRAF, T. Rendergan: Generating realistic labeled data. *Frontiers in Robotics and AI 5* (2018), 66.

[25] VIJAYANARASIMHAN, S., RICCO, S., SCHMID, C., SUKTHANKAR, R., AND FRAGKIADAKI, K. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804* (2017).

[26] WAN, L., ZEILER, M., ZHANG, S., LE CUN, Y., AND FERGUS, R. Regularization of neural networks using dropconnect. In *International conference on machine learning* (2013), pp. 1058–1066.

[27] WANG, C., MIGUEL BUENAPOSADA, J., ZHU, R., AND LUCEY, S. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2022–2030.

[28] WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing 13*, 4 (2004), 600–612.

[29] XIE, J., GIRSHICK, R., AND FARHADI, A. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision* (2016), Springer, pp. 842–857.

[30] YANG, Z., WANG, P., WANG, Y., XU, W., AND NEVATIA, R. Lego: Learning edge with geometry all at once by watching videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018).

[31] YANG, Z., WANG, P., XU, W., ZHAO, L., AND NEVATIA, R. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665* (2017).

[32] YIN, Z., AND SHI, J. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 1983–1992.

[33] ZHOU, T., BROWN, M., SNAVELY, N., AND LOWE, D. G. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1851–1858.

[34] ZHU, X., LIU, Y., QIN, Z., AND LI, J. Data augmentation in emotion classification using generative adversarial networks. *arXiv preprint arXiv:1711.00648* (2017).

[35] ZOPH, B., CUBUK, E. D., GHIASI, G., LIN, T.-Y., SHLENS, J., AND LE, Q. V. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172* (2019).

[36] ZOU, Y., LUO, Z., AND HUANG, J.-B. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 36–53.