Towards Continual, Online, Self-Supervised Depth

Muhammad Umar Karim Khan

Abstract—Although depth extraction with passive sensors has seen remarkable improvement with deep learning, these approaches may fail to obtain correct depth if they are exposed to environments not observed during training. Online adaptation, where the neural network trains while deployed, with self-supervised learning provides a convenient solution as the network can learn from the scene where it is deployed without external supervision. However, online adaptation causes a neural network to forget the past. Thus, past training is wasted and the network is not able to provide good results if it observes past scenes. This work deals with practical online-adaptation where the input is online and temporally-correlated, and training is completely self-supervised. Regularization and replay-based methods without task boundaries are proposed to avoid catastrophic forgetting while adapting to online data. Effort has been made to make the proposed approach suitable for practical use. We apply our method to both structure-from-motion and stereo depth estimation. We evaluate our method on diverse public datasets that include outdoor, indoor and synthetic scenes. Qualitative and quantitative results with both structure-from-motion and stereo show superior forgetting as well as adaptation performance compared to recent methods. Furthermore, the proposed method incurs negligible overhead compared to fine-tuning for online adaptation, proving to be an adequate choice in terms of пластичность, стабильность and applicability. The proposed approach is more inline with the artificial general intelligence paradigm as the neural network learns continually with no supervision. Source code is available at https://github.com/umarKarim/cou_sfm and https://github.com/umarKarim/cou_stereo.

Index Terms—Continual learning, structure-from-motion, stereo, depth, disparity

I. INTRODUCTION

DEEP learning mimics the human behavior from certain perspectives, though much is yet to be achieved. The results of deep learning reflect our behavior much better compared to classical algorithms, with Catastrophic forgetting [1] and online adaptation being two specific examples. Neural networks recall recent information much better compared to the distant past. Similarly, neural networks require special mechanisms to retain past information.

Extracting depth of a scene is critical for scene perception. Depth serves a wide variety of applications from computational photography to navigation. Active sensors such as LIDAR or time-of-flight are capable of providing results at depth resolutions better than the human vision system. However, these have numerous challenges such as low spatial resolution, interference, and high power consumption. Efforts have been made towards passive sensors replicating the human vision system. For example, stereo cameras, structure-from-motion (SfM) and monocular methods replicate binocular vision, head motion and object-size information for depth extraction, respectively.

Despite the widely-acclaimed success of deep learning, generalization towards unseen samples has proven to be a major problem. This problem is usually tackled by training over a dataset that covers as much of the possible samples as possible. For example, MiDaS [2] adopts this strategy and produces great results for a wide variety of scenes. However, the possibility of an edge case still exists, which can prove to be catastrophic. Furthermore, any small addition to the training data requires performing retraining over the whole dataset again. Thus, despite deep learning approaches outperforming classical methods in depth extraction, their generalization to unseen samples is a challenge yet to be resolved.

A different approach towards generalization is to allow the neural network to be always in an active, learning state. In other words, the neural network learns about the scene where it is deployed while not forgetting past information. This approach not only allows generalization to new domains but also does not require retraining. Thus, edge devices can autonomously learn without being connected to a centralized system.

The above paradigm shift comes with numerous challenges. Directly learning new tasks causes catastrophic forgetting of past tasks. Thus, the performance may improve on the new domain but the system will have relearn once it is used in the past domain. Also, in conventional training, samples are passed as a batch and repeatedly used with shuffling to train the neural network. However, in the above setting, only one sample is available at a time and that cannot be used repeatedly to train the neural network. Furthermore, task boundaries are not available, i.e., there is no supervision about when a new scene starts. Finally, it is not practical to provide labels for training for every or any sample.

Numerous notable approaches have been proposed for the above challenges separately. These include continual, online, task-free continual, and self-supervised learning. However, little effort has been made towards solutions that can tackle all these problems together. Furthermore, evaluations have been limited to small and simple datasets in general.

This paper aims towards providing a unified solution to the above challenges for depth extraction. Our key contributions are as follows.

- A novel regularization and replay-based method for online learning of depth while avoiding catastrophic forgetting is proposed.
- Practical constraints such as unsupervised learning, no task (scene) labels, no test-time oracle (where task information is available at test time), temporally-correlated input, synchronous operation and fixed memory are con-
Experiments are conducted on both SfM and stereo, which show the plasticity (online adaptation) and stability (forgetting) performance under the same settings on challenging datasets.

Although this work is limited to depth extraction, the principles proposed here can be used in any downstream self-supervised task. Successful implementation of the proposed method on edge devices provides a step closer to artificial general intelligence, where devices can learn and make decisions on their own while deployed.

The rest of the paper is structured as follows. Section II describes the related work in depth estimation, continual and online learning, and continual learning methods for depth estimation. Section III presents the proposed method for soft task-boundaries. The proposed regularization and replay approaches are discussed in Sections IV, V, and VI. Our choice of neural networks is discussed in Section VII. Section VIII details the experimental results and the paper is concluded in Section IX.

II. RELATED WORK

A. Depth Estimation

Self-supervised training for stereo-based depth estimation is based on reconstructing one stereo image (or depth map) from the other [3]. [4]. Similar approaches have been proposed for monocular or SfM approaches [5], [6], where the next or previous frame is reconstructed using depth and pose estimated by neural networks. [7] proposes excluding the occluded regions in training. Wasserstein distance is used in [8] for depth estimation and 3D object detection. Authors in [9] propose a self-supervised scheme to estimate the depth and 3D scene flow simultaneously. Multi-camera systems have been proposed in [10], [11] to increase depth range. [12] proposes a method to measure confidence of self-supervised depth estimation. In [13], authors use classical stereo approaches to improve performance of monocular depth estimation schemes. In [14], authors propose features for self-supervised depth estimation that are robust to the changes in domain. SLAM and monocular depth estimation are used together to improve each other in [15]. In [16], stereo depth is estimated by binary classification. Removing camera rotations [17] and textured patches [18] have been used for depth estimation of indoor scenes. Authors in [19] distill information from a network which only provides relative depth of objects in a scene for depth estimation of indoor scenes. [20] provides a method for self-supervised depth estimation from gated images. DiffNet [21] uses the HRNet [22] proposed for the task of semantic segmentation for self-supervised depth estimation and shows superior performance on the KITTI [23] dataset.

B. Continual and Online Learning

In continual learning, a neural network learns new tasks while not forgetting about the old ones. Broadly, these methods use distillation [24], [25], regularization [1], [26], replay [27], [28] or their combination [29]. Expansion-based methods can be considered special form of regularization. Regularization-based methods penalize changes to important weights or add new computational nodes to the neural network with a new task. Replay-based schemes feed past data to neural networks while learning the new task. The effect of different training regimes on catastrophic forgetting has been analyzed in [30]. [31] proposes a continual learning method with constant memory, no test-time oracle and without task boundaries. In [32], fast adaptation and recall are achieved for continual learning. [33] proposes to determine the instance of an input sample and use it to train the network accordingly. Knowledge transfer in similar tasks is dealt with in [34]. [35] deals with continual learning where new tasks may contain samples from old tasks. [36] gates layers based on the current task and also predicts tasks at inference. [37] proposes an online learning method for detecting interesting scenes for mobile robots. [38] proposes a method to perform continual learning with temporally-correlated data streams and [39] uses the loss curve to define task boundaries. [40] and [41] use network features to partially update the neural network for fast operation.

Although approaches have been proposed for continual learning with replay under numerous constraints, there are not methods that satisfy all the constraints mentioned in this work. Authors of [42] propose a method of continual and online learning by using samples for replay that are supposed to show the worst performance if the network is updated with the current online samples. They estimate the loss for numerous
past samples with each online batch (up to 50 for MNIST). In other words, they require multiple forward passes while online training. This may not be a problem with small networks but is not feasible for a practical online training network. Authors also propose a replay-based approach where the replay buffer is filled with samples such that the replay samples are as varied as possible. However, their method is also not feasible for real-time operation because it requires computing the dot-product between the gradients of the online batch and at least a subset of the replay buffer with every online batch. Authors of divide the network into Representation Learning and Prediction Learning sub-networks. They train such that the features learned by Representation Learning allow forward transfer and avoid forgetting for online batches. Their method, however, would also require multiple forward and backward passes in an inner loop for a single batch of online data. and are methods for task-free continual learning. These methods are based on expanding the neural network. In practise, the number of tasks is unknown beforehand. In other words, there is no limit on the computational complexity of these systems, making them less feasible for practical applications.

C. Continual and Online Learning of Depth

Numerous authors have worked towards the domain adaptation problem for depth estimation. In and , authors propose a method to adapt from synthetic datasets to real datasets. proposed an adversarial approach for online adaptation across datasets. In , the authors propose using conventional stereo matching algorithms for domain adaptation. To speed up the process, authors propose modular adaptation in . Authors in propose a meta-learning objective to quickly adapt to new scenes. Their work has been advanced in , where the authors develop an approach to perform online adaptation without catastrophic forgetting. uses replay at test time. proposes using two networks; one for adapting to the current scene while the other to generalize over past scenes. Note that , and perform adaptation on test data. Furthermore, resets the neural network for every test sequence.

D. This Work

Methods have been proposed that deal with the individual challenges listed in Fig. however, we could not find methods which deal with all these challenges together. Methods that continually learn depth either present results of forgetting or adaptation in separate experiments, thus, the plasticity-stability compromise cannot be fully understood. Generally, similar datasets are used to evaluate online adaptation from one dataset to another and test data adaptation is performed. In this work, both adaptation and forgetting are analyzed over different datasets without adaptation over the test data.

III. SOFT TASK-BOUNDARIES

Unlike general continual learning research, the task boundaries are unknown, which is more practical. Different scenes can be treated as different tasks. For example, day-time images represent a different task compared to night images. Similarly, indoor scenes represent a different task compared to outdoor scenes. Furthermore, a task can be further sub-divided into sub-tasks. For example, indoor scenes can be divided into classroom and kitchen scenes, and outdoor scenes can be divided into residential and highway scenes. The proposed approach is applicable to any classification of tasks.

The profile of the loss can provide us with valuable hints towards task boundaries. A converged loss is expected to remain stable over a period of time. A significant change in the loss value may indicate that the task has changed.

Let be a random variable indicating the loss value at time . It is assumed

\[ L(t) \sim N(\mu_t, \sigma_t), \]

with and the corresponding mean and standard deviation, respectively. It can be derived (see Appendix)

\[ -\log P(T(t) = T(t-1)) = \frac{1}{2} D_t(L(t-1), L_t(x_t; \theta_t)) + c, \]

where is the task at time , . represent the network parameters and is the squared Mahalanobis distance given by

\[ D_t(L(t-1), L_t(x_t; \theta_t)) = \frac{(L_t(x_t; \theta_t) - \mu_{t-1})^2}{\sigma_{t-1}^2}. \]

In detail, the log-probability of a new task at time depends on how far the current observed loss is from the loss distribution of . The parameters are updated iteratively as

\[ \mu_t = \mu_{t-1} + \alpha_L(L_t - \mu_{t-1}) \]

and

\[ \sigma_t^2 = \sigma_{t-1}^2 + \alpha_L ((L_t - \mu_t)^2 - \sigma_{t-1}^2), \]

where is a constant. The loss profile has been used to define task boundaries in the past as well. However, uses plateus and peaks of the loss profile to determine a task boundary whereas here just the Mahalanobis distance is used. Furthermore, they use hard task-boundaries whereas here the Mahalanobis distance is used as a representation of the probability of the task boundary. In other words, soft task-boundaries are used here.

IV. REGULARIZATION WITH SOFT TASK-BOUNDARIES

Regularization for continual learning limits changes to the network architecture or parameters across tasks to retain information about previous tasks. Memory-aware synapses (MAS) is a known approach, which assigns importance to parameters based on how changing them changes the output for a given input. A sample is passed through the network at the beginning of every task and the gradient determines the importance of parameters. This creates numerous problems. First, hard task-boundaries are required. Second, an additional forward-backward pass is performed at the end of each task.
In this work, inspiration is taken from neural compression techniques, where parameters close to zero are compressed [56]. This is quite intuitive as a parameter with larger magnitude will affect the output more than a parameter with smaller magnitude. Thus, the importance of the \(i\)-th parameter is given by

\[
\Omega_{i,t} = |\theta_{i,t-1}|,
\]

where \(\theta_i\) is the \(i\)-th parameter of the network. The regularization loss then becomes

\[
L^{(reg)}(\theta) = \sum_i \Omega_{i,t} |\theta_i - \theta_{i,t-1}|,
\]

and the combined loss is

\[
L^{(tot)}(x,t,\theta) = L_t(x_t;\theta) + \gamma D_t L^{(reg)}(\theta),
\]

where \(\gamma\) is a constant. In detail, the method is more conservative about changing the important parameters if the chances of a new task are higher. The proposed approach assists in both remembering past tasks as well as forward transfer, does not require hard task boundaries and does not require additional forward-backward passes. Note that \(\theta_i\) represents the current value of the parameters whereas \(\theta\) represents the decision variable.

V. REPLAY WITH SOFT TASK-BOUNDARIES

Let \(T\) represent the current task. A general representation of the overall loss function for the task \(T\) with regularization is

\[
L^{(ov)}_T = L^{(curr)}_T + \gamma L^{(prev)}_T.
\]

Minimizing \(L^{(curr)}_T\) and \(L^{(prev)}_T\) is equivalent to improving performance on the current and previous tasks, respectively. The loss on the previous task can be similarly given as

\[
L^{(prev)}_{T-1} \equiv L^{(ov)}_{T-1} = L^{(curr)}_{T-1} + \gamma L^{(prev)}_{T-1}.
\]

Thus, the contribution of the \(n\)-th previous task to the current loss function is \(\gamma^n\). Since \(\gamma\) is generally set to less than one, it is expected that the performance over distant tasks will be poorer compared to recent tasks.

To resolve this issue, we use sample replay which assigns equal importance to all past tasks. Samples are stored in the replay memory if \(D_t\) is greater than one. By using this approach, difficult samples are saved from which the network can learn more [58]. The storage capacity is limited to 1.5GB [40]. If at full capacity, the new sample replaces a randomly-chosen old sample.

The choice of storing original images to memory is based on numerous factors. Replay with a generative network [27] is an option. For classification problems, generative replay may work well as classification is based on abstract features. Depth extraction, on the other hand, is based on pixel-level disparities which are difficult to recreate. Furthermore, generative replay requires significant amount of additional computational power. Some authors [40] propose storing compressed intermediate features and their corresponding target labels for partially updating the neural network. However, storing target labels is not possible here. First, such an approach requires storing the depth maps as targets. Second, the generated depth maps may not be accurate enough to be used as targets. Replay with SfM has been used in [53], however, their method is solely focused on adaptation; they do not limit replay memory; they adapt on test data; and they do not update the replay buffer while training online. The proposed method, on the other hand, focuses on both adaptation and forgetting; has limited replay memory; does not adapt on test data; and updates the replay buffer while training online. The last point is important as it allows the proposed method to remember depth of older scenes observed during online training.

The replay and online samples are input to the neural network with the same probability. By this the paradigm of spaced repetition [57] is followed where difficult examples are taught to a learning system spaced over time.

VI. REPLAY VS REGULARIZATION

Regularization requires additional memory and computations to maintain the importance of the parameters. Replay does not require significant amount of computational power, however, it needs space to store past samples. Therefore, replay can be used when sufficient storage is available and read/write is fast. Both the proposed regularization and replay are used together in this work.

VII. SELF-SUPERVISED DEPTH EXTRACTION

The proposed approaches are applied to both self-supervised stereo and SfM. In the following, the subscript \(t\) is skipped for brevity.

A. Self-Supervised Stereo

Depth extraction with stereo cameras is based on disparity estimation across rectified left and right images. The relationship between left and right images is given by

\[
I_t(x + d(x,y), y) = I_r(x, y),
\]
where $I_l$, $I_r$, and $d$ denote the left image, right image, and disparity map, respectively. A network is trained to implicitly learn the disparity by reconstructing the right image. I.e.,

$$d = \mathcal{F}_d(I_l, I_r; \theta),$$

which reconstructs the right image as

$$I'_r = f_w(I_l; d),$$

where $\mathcal{F}_d$ represents the disparity neural network and $f_w$ represents the warping function followed by bilinear interpolation \[59\]. The reconstruction loss is then given by

$$L = L_p(I'_r, I_r) + \beta_{ssim}L_{ssim}(I'_r, I_r) + \beta_sL_s(d, I_l),$$

where $L_p$, $L_{ssim}$, and $L_s$ are pixel-wise, SSIM and smoothness losses, respectively. $\beta_{ssim}$ and $\beta_s$ are constants. Refer to \[4\], \[6\] for a discussion on these constants and losses. The approach is shown in Fig. 3.

Numerous techniques have been proposed to improve the performance of self-supervised stereo. For example, \[4\] proposes to reconstruct both the left and right images, and \[60\] uses cycle GANs. Since in this work the emphasis is on online operation, only the right image is reconstructed and GANs are not used.

### B. Self-Supervised Structure-from-Motion

For estimating depth using SfM, two separate networks are used. The first generates the disparity map while the second generates the camera matrix with two frames as input, one as reference and the other as target. The target frame is reconstructed from the reference frame, disparity and camera matrix using the warping function \[59\]

$$I'_{tar} = f_w^{(c)}(I_{ref}; d, c),$$

where $f_w^{(c)}$ represents the warping function \[59\], $d$ is the disparity, and $c$ is the camera matrix. The approach is shown in Fig. 3. The current frame and previous frame are the target and reference, and vice versa. Eq. \[14\] is used to compute the loss between the reconstructed and original target frame. Geometric loss is used as in \[61\].

### C. Network Architecture

The network architecture for the disparity and pose generators are shown in Fig. 4 and Fig. 5, respectively. The disparity generator has an architecture similar to the UNet \[62\] with an encoder and decoder. More specifically, the disparity generator is composed of a ResNet18 encoder \[58\] and a convolutional decoder. The network architecture is similar to the networks used in the past for monocular depth-estimation \[4\], \[6\], \[61\]. There are a couple of key differences. For stereo disparity estimation, we concatenate the left and the right images before passing them to the disparity generator network. Furthermore, we use a relatively small image resolution of $320 \times 256$ to reduce computational complexity. The pose generator also includes the ResNet18 encoder followed by multiple convolutional layers. However, unlike \[61\] and majority of SfM-based deep learning approaches, the pose network learns both the camera intrinsics and extrinsics \[63\]. This is more practical as the intrinsics are not obtained beforehand and the method is camera agnostic. In the rest of the paper, these networks with the proposed subtle changes will be termed as Proposed.

### VIII. Experimental Results

Since the proposed approach is for online learning and not based on the conventional approach of training followed by
testing, therefore, some terminology and evaluation protocols are discussed first.

A. Datasets
For experiments, three datasets are used: KITTI [23], Virtual KITTI [64], and rectified New York University (NYU) v2 [65], [66] datasets. The Eigen test split [67] is used with KITTI. For the Virtual-KITTI dataset, the first 90% frames of every sequence are used for training and the last 10% are for evaluation. Refer to supplementary material for more details. These datasets are very different from each other, thereby allowing better evaluation of the proposed approach.

B. Terminology
Both the plasticity (adaptation) and stability (remembrance) performance of the proposed method are evaluated at both the dataset and scene levels. Training Dataset Performance (TDP) shows the results on the dataset over which online training has been performed, whereas Non-training Dataset Performance (NDP) shows the results on the other dataset. TDP shows how well the method is adapting to new scenes as well as how well the network remembers the scenes previously observed during online training. NDP shows how well the network remembers the data observed during pre-training.

C. Training and Evaluation Protocol
Online training is preceded by pre-training or warmup [40]. Pre-training and online training data are composed of different scenes. Pre-training provides the base model for online training. For online training, data is sequentially passed without shuffling. Thus, temporal correlation exists which is more practical and challenging [38]. Proposed results are averaged over three runs of online training. The evaluation results are obtained by averaging the results of all frames of the test data. Further details are in the supplementary. The test data is not used for learning unlike [52] and [51] but for evaluation only. Thus, there is no test-time oracle.

| Mode | SFM | Stereo |
|------|-----|--------|
| Online Train Net. | TDP | NDP | TDP | NDP |
| K [21] FT | 0.2214 | 0.5746 | 0.1932 | 0.1996 |
| K [21] Prop. | 0.1904 | 0.2265 | 0.2098 | 0.1670 |
| K Prop. | 0.1916 | 0.2694 | 0.2123 | 0.1902 |
| K Prop. | 0.1895 | 0.3504 | 0.1920 | 0.1980 |
| K Prop. | 0.1580 | 0.1911 | 0.1962 | 0.1685 |
| N,vK Prop. | 0.1543 | 0.1992 | 0.1825 | 0.1660 |
| N,vK Prop. | 0.3243 | 0.3088 | 0.1860 | 0.2401 |
| N,vK Prop. | 0.2243 | 0.1695 | 0.1803 | 1.4442 |
| N,vK Prop. | 0.2379 | 0.1995 | 0.1765 | 0.2190 |
| N,vK Prop. | 0.2430 | 0.3336 | 0.1991 | 0.2090 |
| N,vK Prop. | 0.1912 | 0.1586 | 0.1834 | 0.3177 |
| N,vK Prop. | 0.1872 | 0.1624 | 0.1653 | 0.1770 |

The following metrics have been used in this work.

- **RMSE**: \( \sqrt{\frac{1}{N} \sum_{x} (d(x) - d^*(x))^2} \)
- **Absolute Relative**: \( \frac{1}{N} \sum_{x} \frac{|d(x) - d^*(x)|}{d(x)} \)
- **\( \delta(a) \)**: \( \frac{1}{N} \sum_{x} 1 \{ \max \left( \frac{d^*(x)}{d(x)}, \frac{d(x)}{d^*(x)} \right) < a \} \)

where \( x, d \) and \( d^* \) represent the pixel location, obtained depth and ground-truth depth, respectively. Check supplementary for detailed results on other metrics as well.

D. Setup
The same hyper-parameters and optimizer are used in all experiments. The Adam optimizer [68] is used with the same hyper-parameters and optimizer are used in all experiments. The Nvidia GTX 1080Ti GPU is used and frames are resized to \( 320 \times 256 \).

E. Performance Evaluation
The first set of experiments are based on SFM, where pre-training is performed on roughly half of the KITTI and NYU datasets, and online training is performed on the other half of either KITTI or NYU. The results for the absolute-relative metric are shown in Table I. Refer to supplementary for results on other metrics. For comparison, we include results of the COMODA [53] continual learning approach, where samples from dataset used for pre-training are randomly used for replay with equal probability as the online-training samples. We apply COMODA and the proposed method of continual learning to the DiffNet [21] network and the proposed network architecture described in previous section. Online training on one dataset severely degrades performance on the other dataset if it is directly fine-tuned. However, by using the proposed approach of regularization and replay the neural network remembers information from the dataset it is not being trained on. Furthermore, plasticity is not degraded, rather, it is improved. This should come as no surprise as regularization improves forward transfer of information and replay exposes the network to difficult examples. The performance with the DiffNet architecture is generally poorer compared to

| Method | Model | RMSE ↓ | Abs. Rel. ↓ | \( \delta(1.25) \) ↑ | \( \delta(1.25) \) ↑ |
|--------|-------|--------|-------------|----------------|----------------|
| FT | [5] | 6.5248 | 0.2070 | 0.7041 | 0.8806 |
| FT | [61] | 5.6528 | 0.1735 | 0.7743 | 0.9140 |
| L2A [5] | [5] | 6.3804 | 0.1937 | 0.7221 | 0.8980 |
| L2A [5] | [61] | 5.5000 | 0.1692 | 0.7881 | 0.9197 |
| LPF [5] | [5] | 6.1090 | 0.1794 | 0.7307 | 0.9126 |
| LPF [5] | [61] | 5.4452 | 0.1505 | 0.7990 | 0.9325 |
| FT Prop. | 7.4267 | 0.2575 | 0.6106 | 0.8441 |
| Prop. | 5.4739 | 0.1313 | 0.8345 | 0.9455 |
| Prop. | 7.147 | 0.2158 | 0.6690 | 0.9455 |
| Prop. | 5.3554 | 0.1392 | 0.8251 | 0.9433 |

*Shows that fog and rain are included in online training. Best and second best are bold and underlined, respectively.
the proposed architecture. It is seen that COMODA slightly outperforms the proposed method in the NDP. This shows that COMODA shows similar or better performance on the non-training dataset observed during pre-training. However, the proposed method outperforms COMODA on the TDP, which considers both the online-adaptation and the performance on previously observed scenes in the dataset used for online training. The qualitative results with online training with different continual learning strategies and network architectures are shown in Fig. 8. The results clearly demonstrate that the fine-tuning does degrade the output drastically. Furthermore, the proposed method of continual learning with the proposed architecture generally produces superior results compared to other alternatives.

Note that these results should not be compared with methods that follow the paradigm of training followed by testing. With KITTI in conventional setting, the network is trained over the KITTI training data for multiple epochs and then evaluated over the test data. In the current scenario, the network is pre-trained over half of KITTI and NYU datasets for multiple epochs, and then trained over the other half of the KITTI dataset for one epoch only during online training.

For evaluation over stereo, pre-training is performed over half of the KITTI and virtual KITTI datasets, and online training on the other half of either KITTI or virtual KITTI datasets. The results are shown in Table I. Again we see a significant improvement is seen in performance with the proposed method. Qualitative results are shown in Fig. 8. We note that the proposed method does outperform competing methods, however, the performance of competing methods is not as poor as in the case of SfM. This is an interesting result. We believe that this is because in case of SfM the networks are trained to estimate disparity from a single image. Thus, the network is required to memorize the size-to-depth relationship of different objects in the scene. Once the scene changes, new objects appear which have a very different size-to-depth relationship compared to previously observed objects, resulting in the networks perform poorly. Stereo-based approach, however, learns to estimate disparity from left and right images, and only uses size-to-depth relationship where disparity estimation is not possible, for example, textureless or occluded regions. The disparity estimation process remains the same across scenes, therefore, stereo-based depth estimation is not as severely affected by observing new scenes as SfM.

To further analyze the performance of these methods, we show the results of different methods and networks on individual scenes from the online training datasets after online training in Fig. 8. We see that the proposed network architecture trained with the proposed method for continual learning shows better performance on individual scenes observed sequentially during training with both SfM and stereo. This is because the proposed approach not only remembers scenes from pre-training but also has mechanisms to remember older scenes observed during online training.

We conduct another experiment to validate the performance of the proposed method. In [2], the authors use SfM where they first pre-train the model on the KITTI dataset over multiple epochs and then online train over the virtual KITTI dataset for a single epoch. They also perform test-time adaptation, i.e., they use the sequences used for testing to learn while testing. We also test our method under their experimental settings and the results are shown in Table I. However, test-time adaptation is not performed with the proposed method. Using the complete video sequence in testing provides useful context, which maybe taken as test-time oracle. Unlike [2], results where the challenging fog and rain sequences of the virtual KITTI dataset are included in the online training are also included. The results show that the proposed method outperforms competing methods and provides a larger percentage increase compared to fine-tuning.

F. GPU Performance

One important aspect worth attention is the overhead incurred by the proposed method compared to fine tuning. To analyze this, we trace the GPU performance with different metrics for 5 seconds while online training. The results for SfM and stereo are shown in Fig. 9 and 10, respectively. All the methods for continual learning with the proposed network architecture provide 12fps and 17fps for SfM and stereo, respectively, which is higher than the frame rate of 10 of the KITTI dataset. The fps goes down to 4fps and 5fps with DiffNet [21] in the same order. This shows that although there is still room for
Fig. 7. Qualitative results with SfM over KITTI and NYU datasets with different networks and methods for continual learning. Best seen in color.

improvement, there is no overhead in terms of speed with the proposed method for continual learning. The figures show that there is only a slight increase in the GPU memory consumption with the proposed approach for continual learning. The maximum GPU memory required with the proposed method for continual learning and the proposed network architecture is 3.8GB, which can be easily accommodated on low-end GPUs for deep learning such as NVIDIA GeForce GTX 1650 Ti Mobile. For all other metrics, the proposed method for continual learning of depth has the same resource utilization as not using any method for continual learning at all. The power consumption with the proposed network architecture is higher compared to the power consumption with DiffNet. This is because the proposed network architecture operates at a higher frame rate compared to DiffNet.
Fig. 8. Qualitative results with stereo over KITTI and virtual KITTI datasets with different networks and methods for continual learning. Best seen in color.

IX. CONCLUSION

A method is presented for learning depth of a scene while deployed and not forgetting about the past. Novel regularization and replay methods are proposed to this end. The method learns depth from a stream of monocular or stereo video-frames, and avoids catastrophic forgetting. Experiments show that the proposed method has superior performance with both SfM and stereo. This work is a step towards autonomous scene-perception without any supervision.

APPENDIX

Based on (1), we can write

\[ P(T(t) = T(t - 1)) = N(L_t; \mu_{t-1}, \sigma_{t-1}) \]  

or

\[ P(T(t) = T(t - 1)) = \mathcal{N}(L_t; \mu_{t-1}, \sigma_{t-1}) \]
Fig. 9. Hardware performance with different networks and methods for continual learning while training SfM. Memory and GPU utilization show the percent of time over the past sample period when memory read/write was performed and a kernel was executed on the GPU, respectively.

Fig. 10. Hardware performance with different networks and methods for continual learning while training stereo. Memory and GPU utilization show the percent of time over the past sample period when memory read/write was performed and a kernel was executed on the GPU, respectively.

\[ P(T(t) = T(t-1)) = \frac{1}{\sqrt{2\pi\sigma_{t-1}}} e^{-\frac{(L_t - \mu_{t-1})^2}{2\sigma_{t-1}^2}}. \]  

and

\[ C = \sqrt{2\pi\sigma_{t-1}}. \]  

Thus

\[ P(T(t) = T(t-1)) = \frac{\exp(B)}{C}. \]  

Let

\[ B = \frac{-(L_t - \mu_{t-1})^2}{2\sigma_{t-1}^2}. \]  

Taking logarithm of both sides
\[
\log(P(T(t) = T(t-1)) = \log \left( \frac{\exp(B)}{C} \right). \tag{21}
\]

Using the properties of \(\log\), and substituting \(A\) and \(B\), we can write
\[
-\log(P(T(t) = T(t-1)) = \frac{1}{2} \left( \frac{L_t - \mu_{t-1}}{\sigma_{t-1}} \right)^2 + \log(\sqrt{2\pi}\sigma_{t-1}). \tag{22}
\]

REFERENCES

[1] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska et al., “Overcoming catastrophic forgetting in neural networks,” Proceedings of the national academy of sciences, vol. 114, no. 13, pp. 3521–3526, 2017.

[2] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer,” arXiv preprint arXiv:1907.01341, 2019.

[3] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, “Unsupervised cnn for single view depth estimation: Geometry to the rescue,” in European conference on computer vision. Springer, 2016, pp. 740–756.

[4] C. Godard, O. Mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 270–279.

[5] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “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.

[6] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, “Digging into self-supervised monocular depth estimation,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3828–3838.

[7] J. L. GonzalezBello and M. Kim, “Forget about the lidar: Self-supervised depth estimators with med probability volumes,” Advances in Neural Information Processing Systems, vol. 33, 2020.

[8] D. Garg, Y. Wang, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, “Wasserstein distances for stereo disparity estimation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 14402–14413.

[9] L. Tiwari, P. Ji, Q.-H. Tran, B. Zhuang, S. Anand, and M. Chandraker, “Pseudo rgb-d for self-improving monocular slam and depth prediction,” in European Conference on Computer Vision. Springer, 2020, pp. 457–455.

[10] R. A. Badki, A. Troccoli, K. Kim, J. Kautz, P. Sen, and O. Gallo, “Bi3d: Stereo depth estimation via binary classifications,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1600–1608.

[11] J. Zhou, Y. Wang, K. Qin, and W. Zeng, “Moving indoor: Unsupervised video depth learning in challenging environments,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8618–8627.

[12] Z. Yu, L. Jin, and S. Gao, “P2 net: Patch-match and plane-regularization for unsupervised indoor depth estimation,” in Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16. Springer, 2020, pp. 206–222.

[13] Z.-Y. Wu, J. Wang, M. Hall, U. Neumann, and S. Su, “Towards practical self-supervised monocular indoor depth estimation,” arXiv preprint arXiv:2112.02036, 2021.

[14] A. Walia, S. Walz, M. Bijelic, F. Mannan, F. Julca-Aguilar, M. Langer, W. Ritter, and F. Heide, “Gated2gated: Self-supervised depth estimation from gated images,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 2811–2821.

[15] H. Zhou, D. Greenwood, and S. Taylor, “Self-supervised monocular depth estimation with internal feature fusion,” 2021.

[16] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mei, M. Tan, X. Wang et al., “Deep high-resolution representation learning for visual recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 43, no. 10, pp. 3349–3364, 2020.

[17] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[18] Z. Li and D. Hoiem, “Learning without forgetting,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 12, pp. 2935–2947, 2017.

[19] E. Fini, S. Lathuilière, E. Sangineto, M. Nabi, and E. Ricci, “Continuous learning under extreme memory constraints,” in European Conference on Computer Vision. Springer, 2020, pp. 720–735.

[20] R. Aljundi, F. Babiloni, M. Eloiseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 139–154.

[21] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continuous learning with deep generative replay,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 2994–3003.

[22] G. Gupta, K. Yadav, and L. Paull, “La-manl: Look-ahead meta learning for continual learning,” 2020.

[23] P. Pan, S. Swaroop, A. Immer, R. Eschenhagen, R. Turner, and M. E. Khan, “Continuous deep learning by functional regularisation of memorable past,” 2020.

[24] S. I. Mirzadeh, M. Farajtabar, R. Pascanu, and H. Ghasemzadeh, “Understanding the role of training regimes in continual learning,” 2020.

[25] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, “Dark experience for general continual learning: a strong, simple baseline,” in 34th Conference on Neural Information Processing Systems (NeurIPS 2020), 2020.

[26] M. Caccia, P. Rodriguez, O. Ostanenko, F. Normandin, M. Lin, L. Page-Caccia, I. H. Laradji, I. Rish, A. Lacoste, D. Vázquez et al., “Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning,” Advances in Neural Information Processing Systems, vol. 33, 2020.

[27] H.-J. Chen, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, “Mitigating forgetting in online continual learning via instance-aware parameterization,” Advances in Neural Information Processing Systems, vol. 33, 2020.

[28] Z. Ke, B. Liu, and X. Huang, “Continual learning of a mixed sequence of similar and dissimilar tasks,” Advances in Neural Information Processing Systems, vol. 33, 2020.

[29] J. He, R. Mao, Z. Shao, and F. Zhu, “Incremental learning in online scenario,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13926–13935.

[30] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and F. Normandin, “Conditional channel gated networks for task-aware continual learning,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3931–3940.

[31] C. Wang, W. Wang, Y. Qiu, Y. Hu, and S. Scherer, “Visual memorability for robotic interestingness via unsupervised online learning,” in European Conference on Computer Vision (ECCV). Springer, 2020.

[32] A. Chrysakis and M.-F. Moens, “Online continual learning from imbalanced data,” in International Conference on Machine Learning. PMLR, 2020, pp. 1952–1961.

[33] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, “Task-free continual learning,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11254–11263.

[34] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, “Remind forget: Inducing forgetting in online continual learning via instance-aware parameterization,” in European Conference on Computer Vision. Springer, 2020, pp. 466–483.
[41] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, “Latent replay for real-time continual learning,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 10203–10209.

[42] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, “Online continual learning with maximal interfered retrieval,” Advances in Neural Information Processing Systems, vol. 32, pp. 11,849–11,860, 2019.

[43] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, “Gradient based sample selection for online continual learning,” Advances in Neural Information Processing Systems, vol. 32, pp. 11,816–11,825, 2019.

[44] K. Javed and M. White, “Meta-learning representations for continual learning,” in Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, pp. 1820–1830.

[45] D. Rao, F. Visin, A. A. Rusu, Y. W. Teh, R. Pascanu, and R. Hadsell, “Continual unsupervised representation learning,” arXiv preprint arXiv:1910.14481, 2019.

[46] S. Lee, J. Ha, D. Zhang, and G. Kim, “A neural dirichlet process mixture model for task-free continual learning,” in International Conference on Learning Representations, 2019.

[47] C. Zheng, T.-J. Cham, and J. Cai, “T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 767–783.

[48] Q. Sun, G. G. Ye, Y. Tang, and C. Zhao, “Learn to adapt for monocular depth estimation,” arXiv preprint arXiv:2203.14005, 2022.

[49] A. Tonioni, M. Poggi, S. Mattoccia, and L. D. Stefano, “Unsupervised adaptation for deep stereo,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1605–1613.

[50] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. D. Stefano, “Real-time self-adaptive deep stereo,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 195–204.

[51] A. Tonioni, O. Rahnama, T. Joy, L. D. Stefano, T. Ajanthan, and P. H. Torr, “Learning to adapt for stereo,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9661–9670.

[52] Z. Zhang, S. Lathuilière, E. Ricci, N. Sebe, Y. Yan, and J. Yang, “Online depth learning against forgetting in monocular videos,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4494–4503.

[53] Y. Kuznietsov, M. Proesmans, and L. Van Gool, “Comoda: Continuous monocular depth adaptation using past experiences,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 2907–2917.

[54] N. Vödisch, D. Cattaneo, W. Burgard, and A. Valada, “Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning,” arXiv preprint arXiv:2203.01578, 2022.

[55] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149), vol. 2. IEEE, 1999, pp. 246–252.

[56] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” arXiv preprint arXiv:1608.08710, 2016.

[57] H. Amiri, T. Miller, and G. Savova, “Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks,” in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 2401–2410.

[58] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[59] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, “Spatial transformer networks,” arXiv preprint arXiv:1506.02025, 2015.

[60] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe, “Unsupervised adversarial depth estimation using cycled generative networks,” in 2018 International Conference on 3D Vision (3DV). IEEE, 2018, pp. 587–595.

[61] J.-W. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, “Unsupervised scale-consistent depth and ego-motion learning from monocular video,” arXiv preprint arXiv:1908.10553, 2019.

[62] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.

[63] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, “Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8977–8986.

[64] Y. Cabon, N. Murray, and M. Humenberger, “Virtual kitti 2,” 2020.

[65] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in European conference on computer vision. Springer, 2012, pp. 746–760.

[66] J.-W. Bian, H. Zhan, N. Wang, T.-J. Chan, C. Shen, and I. Reid, “Unsupervised depth learning in challenging indoor video: Weak rectification to rescue,” arXiv preprint arXiv:2006.02708, 2020.

[67] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” arXiv preprint arXiv:1406.2283, 2014.

[68] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.
The datasets are divided into two parts: Pre-training and Online training. Scenes are assigned to pre-training and online training such that the number of scenes remains roughly the same in both.

A. SfM Experiments

1) KITTI Dataset: The KITTI dataset is already divided into multiple categories based on the location of the vehicle. Majority of these scenes are from the residential category. It is very difficult to have a reasonable division of the KITTI dataset based on categories. Thus, residential sequences were divided into two parts: residential_1 and residential_2.

- Training frames: 44764
- Testing frames: 697
- Pre-training
  - Scenes: Road, residential_1
  - Frames: 24920
- Online Training
  - Scenes: Residential_2, city, campus
  - Frames: 19844

2) NYU Dataset:

- Training frames: 69383
- Testing frames: 654
- Pre-training
  - Scenes: From basement to indoor_balcony
  - Frames: 29596
- Online Training
  - Scenes: The rest
  - Frames: 39787

B. Stereo Experiments

1) KITTI Dataset:

- Training frames: 41888
- Testing frames: 697
- Pre-training
  - Scenes: Road, residential_1
  - Frames: 21696
- Online Training
  - Scenes: Residential_2, city, campus
  - Frames: 20192

2) vKITTI Dataset:

- Training frames: 21620
- Testing frames: 2100
- Pre-training
  - Scenes: 15-deg-left, 15-deg-right, clone, fog, morning
  - Frames: 9580
- Online Training
  - Scenes: 30-deg-left, 30-deg-right, overcast, rain, sunset
  - Frames: 9580

C. Test Frames Categories

For vKITTI, the directory from which the frame is taken gives us the test frame category. The text file here was used for first finding the sequence to which the frame belongs, and then find the category to which the sequence belongs using the KITTI website. For NYU, the .mat file provided here was used to find the categories of the test frames.

Let \( N \) be the total number of pixels in a frame used for evaluation and \( x \) be the pixel index. Let \( d \) and \( d^* \) represent the obtained and the ground truth distance, respectively. Also, \( 1\{.\} \) is the indicator function, which returns 1 if the condition in the parentheses is true, otherwise returns 0.

- **RMSE:** \[ \sqrt{\frac{1}{N} \sum_x (d(x) - d^*(x))^2} \]
- **Absolute Relative:** \[ \frac{1}{N} \sum_x |d(x) - d^*(x)| \]
- **Square Relative:** \[ \frac{1}{N} \sum_x (d(x) - d^*(x))^2 \]
- **Log RMSE:** \[ \frac{1}{N} \sum_x (\log(d(x)) - \log(d^*(x))) \]
- **\( \delta(a) \):** \[ \frac{1}{N} \sum_x 1 \left\{ \max \left( \frac{d^*(x)}{d(x)}, \frac{d(x)}{d^*(x)} \right) < a \right\} \]

The results in the manuscript are given for the whole dataset as well as domains within the dataset. These are described as follows.

D. Training Dataset Performance (TDP) Results

The results on the test data of the dataset over which training is performed. For example, if training is performed on KITTI then TDP results show the results over the test data of KITTI dataset after online training is complete.

E. Non-training Dataset Performance (NDP) Results

The results on the test data of the dataset other than the one over which training is performed. For example, if training is performed on KITTI then NDP results show the results over the test data of NYU dataset after online training is complete.

It is noted that 320 × 256 .jpg images are approximately 20kB in size. This is the resolution used in our experiments. The number of frames stored for replay did not exceed 1500 in any of our experiments.

- **KITTI + NYU SfM Experiments:** The storage space required to maintain the pre-training dataset is approximately 1.04GB. An additional 12086 frames can be maintained for replay.
- **KITTI + vKITTI SfM Experiments:** The storage space required to maintain the pre-training dataset (KITTI) is approximately 0.85GB. An additional 16939 frames can be maintained for replay.
- **KITTI + vKITTI Stereo Experiments:** The storage space required to maintain the stereo pre-training dataset is approximately 1.19GB. An additional 8045 stereo images can be maintained for replay.

The effect of regularization and replay is shown in Table III. The table shows that the combination of replay and regularization provide the overall best results.
| Meth. | Data | TDP  | NDP  |
|-------|------|------|------|
| FT    | KITTI| 7.3054 | 11805 |
| Reg.  | KITTI| 6.6603 | 0.6955 |
| Rep.  | KITTI| 5.9583 | 0.6782 |
| Prop. | KITTI| 5.8085 | 0.6603 |
| FT    | NYU  | 0.7688 | 10.2060 |
| Reg.  | NYU  | 0.7010 | 8.5240 |
| Rep.  | NYU  | 0.6368 | 7.3262 |
| Prop. | NYU  | 0.6270 | 5.9958 |

**TABLE III**

RMSE under with different components of the proposed approach for a single run. FT, Reg., Rep. and Prop. stand for fine tuning, regularization, replay and proposed method, respectively. KITTI + NYU was used for pre-training.
### TABLE IV
**Absolute Relative of SfM-based depth estimation. FT stands for fine tuning.**

| Online Train | Method | Network | TDP     | NDP     |
|--------------|--------|---------|---------|---------|
| KITTI FT     | DiffNet | [21]    | 0.2214 ± 0.0000 | 0.5746 ± 0.0000 |
| KITTI Prop.  | DiffNet | [21]    | 0.1916 ± 0.0024 | 0.2694 ± 0.0213 |
| KITTI [53]   | DiffNet | [21]    | 0.1904 ± 0.0140 | 0.2265 ± 0.0056 |
| KITTI FT     | Prop.   |         | 0.1895 ± 0.0000 | 0.3504 ± 0.0000 |
| KITTI Prop.  | Prop.   |         | 0.1543 ± 0.0034 | 0.1952 ± 0.0021 |
| KITTI [53]   | Prop.   |         | 0.1580 ± 0.0064 | 0.1911 ± 0.0010 |
| NYU FT      | DiffNet | [21]    | 0.3243 ± 0.0000 | 0.3088 ± 0.0000 |
| NYU Prop.   | DiffNet | [21]    | 0.2379 ± 0.0030 | 0.1995 ± 0.0176 |
| NYU [53]    | DiffNet | [21]    | 0.2243 ± 0.0016 | 0.1695 ± 0.0013 |
| NYU FT     | Prop.   |         | 0.2430 ± 0.0000 | 0.3336 ± 0.0000 |
| NYU Prop.  | Prop.   |         | 0.1872 ± 0.0058 | 0.1624 ± 0.0044 |
| NYU [53]   | Prop.   |         | 0.1912 ± 0.0003 | 0.1586 ± 0.0037 |

### TABLE V
**RMSE of SfM-based depth estimation. FT stands for fine tuning.**

| Online Train | Method | Network | TDP     | NDP     |
|--------------|--------|---------|---------|---------|
| KITTI FT     | DiffNet | [21]    | 7.9542 ± 0.0000 | 2.2098 ± 0.0000 |
| KITTI Prop.  | DiffNet | [21]    | 6.3588 ± 0.1508 | 0.8783 ± 0.0733 |
| KITTI [53]   | DiffNet | [21]    | 6.4767 ± 0.5267 | 0.7731 ± 0.0125 |
| KITTI FT     | Prop.   |         | 7.3054 ± 0.0000 | 1.1805 ± 0.0000 |
| KITTI Prop.  | Prop.   |         | 5.8085 ± 0.1168 | 0.6603 ± 0.0015 |
| KITTI [53]   | Prop.   |         | 6.2494 ± 0.1864 | 0.6629 ± 0.0057 |
| NYU FT      | DiffNet | [21]    | 1.0214 ± 0.0000 | 10.7680 ± 0.0000 |
| NYU Prop.   | DiffNet | [21]    | 0.7942 ± 0.0051 | 6.6515 ± 0.4935 |
| NYU [53]    | DiffNet | [21]    | 0.7660 ± 0.0115 | 6.3401 ± 0.3011 |
| NYU FT     | Prop.   |         | 0.7688 ± 0.0000 | 10.2060 ± 0.0000 |
| NYU Prop.  | Prop.   |         | 0.6270 ± 0.0115 | 5.9958 ± 0.0847 |
| NYU [53]   | Prop.   |         | 0.6727 ± 0.0025 | 6.2292 ± 0.1704 |

### TABLE VI
**Square Relative of SfM-based depth estimation. FT stands for fine tuning.**

| Online Train | Method | Network | TDP     | NDP     |
|--------------|--------|---------|---------|---------|
| KITTI FT     | DiffNet | [21]    | 1.9197 ± 0.0000 | 1.9714 ± 0.0000 |
| KITTI Prop.  | DiffNet | [21]    | 1.4904 ± 0.0390 | 0.3246 ± 0.0713 |
| KITTI [53]   | DiffNet | [21]    | 1.4904 ± 0.1844 | 0.2315 ± 0.0122 |
| KITTI FT     | Prop.   |         | 1.5520 ± 0.0000 | 0.5699 ± 0.0000 |
| KITTI Prop.  | Prop.   |         | 1.1420 ± 0.0419 | 0.1748 ± 0.0029 |
| KITTI [53]   | Prop.   |         | 1.2297 ± 0.0744 | 0.1745 ± 0.0027 |
| NYU FT      | DiffNet | [21]    | 0.4248 ± 0.0000 | 3.2486 ± 0.0000 |
| NYU Prop.   | DiffNet | [21]    | 0.2490 ± 0.0052 | 1.5412 ± 0.1497 |
| NYU [53]    | DiffNet | [21]    | 0.2295 ± 0.0046 | 1.3228 ± 0.0281 |
| NYU FT     | Prop.   |         | 0.2432 ± 0.0000 | 3.1929 ± 0.0000 |
| NYU Prop.  | Prop.   |         | 0.1624 ± 0.0095 | 1.2044 ± 0.0390 |
| NYU [53]   | Prop.   |         | 0.1787 ± 0.0010 | 1.2233 ± 0.0544 |
| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI        | FT     | DiffNet | 0.3276 ± 0.0000 | 0.5613 ± 0.0000 |
| KITTI        | Prop.  | DiffNet | 0.2750 ± 0.0055 | 0.3290 ± 0.0149 |
| KITTI [53]   | FT     | Prop.   | 0.2779 ± 0.0243 | 0.3061 ± 0.0026 |
| KITTI        | Prop.  | Prop.   | 0.2927 ± 0.0000 | 0.3921 ± 0.0000 |
| NYU          | FT     | Prop.   | 0.2351 ± 0.0006 | 0.2401 ± 0.0016 |
| NYU [53]     | Prop.  | Prop.   | 0.2477 ± 0.0084 | 0.2396 ± 0.0017 |
| NYU          | FT     | DiffNet | 0.3862 ± 0.0000 | 0.4634 ± 0.0000 |
| NYU [53]     | Prop.  | DiffNet | 0.3088 ± 0.0029 | 0.2804 ± 0.0214 |
| NYU          | FT     | DiffNet | 0.2955 ± 0.0062 | 0.2567 ± 0.0073 |
| NYU [53]     | Prop.  | Prop.   | 0.2296 ± 0.0050 | 0.2398 ± 0.0045 |
| NYU          | FT     | Prop.   | 0.2406 ± 0.0007 | 0.2467 ± 0.0048 |
| NYU [53]     | Prop.  | Prop.   | 0.2351 ± 0.0006 | 0.2401 ± 0.0016 |
| NYU          | FT     | Prop.   | 0.2927 ± 0.0000 | 0.3921 ± 0.0000 |
| NYU [53]     | Prop.  | Prop.   | 0.2477 ± 0.0084 | 0.2396 ± 0.0017 |
| NYU          | FT     | DiffNet | 0.4746 ± 0.0000 | 0.4509 ± 0.0000 |
| NYU [53]     | Prop.  | DiffNet | 0.6199 ± 0.0081 | 0.6835 ± 0.0458 |
| NYU          | FT     | DiffNet | 0.6446 ± 0.0034 | 0.7446 ± 0.0105 |
| NYU [53]     | Prop.  | Prop.   | 0.6063 ± 0.0000 | 0.4050 ± 0.0000 |
| NYU          | FT     | Prop.   | 0.7284 ± 0.0078 | 0.7671 ± 0.0069 |
| NYU [53]     | Prop.  | Prop.   | 0.7055 ± 0.0011 | 0.7688 ± 0.0085 |

TABLE VIII
δ(1.25) OF SFM-BASED DEPTH ESTIMATION. FT STANDS FOR FINE TUNING.

| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI        | FT     | DiffNet | 0.6475 ± 0.0000 | 0.2894 ± 0.0000 |
| KITTI        | Prop.  | DiffNet | 0.7078 ± 0.0040 | 0.5688 ± 0.0299 |
| KITTI [53]   | DiffNet | 0.7037 ± 0.0377 | 0.6373 ± 0.0064 |
| KITTI        | FT     | Prop.   | 0.7110 ± 0.0000 | 0.4462 ± 0.0000 |
| KITTI        | Prop.  | Prop.   | 0.7835 ± 0.0027 | 0.7029 ± 0.0041 |
| KITTI [53]   | Prop.  | Prop.   | 0.7691 ± 0.0157 | 0.7047 ± 0.0066 |
| NYU          | FT     | DiffNet | 0.4746 ± 0.0000 | 0.4509 ± 0.0000 |
| NYU [53]     | Prop.  | DiffNet | 0.6199 ± 0.0081 | 0.6835 ± 0.0458 |
| NYU          | FT     | DiffNet | 0.6446 ± 0.0034 | 0.7446 ± 0.0105 |
| NYU [53]     | Prop.  | Prop.   | 0.6063 ± 0.0000 | 0.4050 ± 0.0000 |
| NYU          | FT     | Prop.   | 0.7284 ± 0.0078 | 0.7671 ± 0.0069 |
| NYU [53]     | Prop.  | Prop.   | 0.7055 ± 0.0011 | 0.7688 ± 0.0085 |

TABLE IX
δ(1.25^2) OF SFM-BASED DEPTH ESTIMATION. FT STANDS FOR FINE TUNING.

| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI        | FT     | DiffNet | 0.8489 ± 0.0000 | 0.5540 ± 0.0000 |
| KITTI        | Prop.  | DiffNet | 0.8930 ± 0.0050 | 0.8328 ± 0.0224 |
| KITTI [53]   | DiffNet | 0.8956 ± 0.0226 | 0.8638 ± 0.0039 |
| KITTI        | FT     | Prop.   | 0.8807 ± 0.0000 | 0.7442 ± 0.0000 |
| KITTI        | Prop.  | Prop.   | 0.9278 ± 0.0011 | 0.9140 ± 0.0029 |
| KITTI [53]   | Prop.  | Prop.   | 0.9180 ± 0.0073 | 0.9130 ± 0.0014 |
| NYU          | FT     | DiffNet | 0.7662 ± 0.0000 | 0.7384 ± 0.0000 |
| NYU [53]     | Prop.  | DiffNet | 0.8572 ± 0.0015 | 0.8892 ± 0.0213 |
| NYU          | FT     | Prop.   | 0.8671 ± 0.0037 | 0.9098 ± 0.0070 |
| NYU [53]     | Prop.  | Prop.   | 0.9220 ± 0.0050 | 0.9243 ± 0.0020 |
| NYU          | FT     | Prop.   | 0.9127 ± 0.0009 | 0.9178 ± 0.0047 |
### TABLE X

Absolute Relative of Stereo-based Depth Estimation. *FT* stands for fine tuning.

| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI FT     | DiffNet | [21]    | 0.1932 ± 0.0000 | 0.1996 ± 0.0000 |
| KITTI Prop.  | DiffNet | [21]    | 0.2123 ± 0.0192 | 0.1902 ± 0.0113 |
| KITTI [53] FT Prop. | DiffNet | [21]    | 0.2098 ± 0.0433 | 0.1656 ± 0.0067 |
| KITTI Prop.  | Prop.   |         | 0.1920 ± 0.0000 | 0.1980 ± 0.0000 |
| KITTI Prop.  | Prop.   | [53]    | 0.1825 ± 0.0037 | 0.1660 ± 0.0037 |
| KITTI Prop.  | Prop.   | [53]    | 0.1962 ± 0.0220 | 0.1685 ± 0.0050 |
| vKITTI FT    | DiffNet | [21]    | 0.1860 ± 0.0000 | 0.2401 ± 0.0000 |
| vKITTI Prop. | DiffNet | [21]    | 0.1765 ± 0.0025 | 0.2190 ± 0.0200 |
| vKITTI [53] FT Prop. | DiffNet | [21]    | 0.1803 ± 0.0067 | 1.4442 ± 1.7665 |
| vKITTI FT    | Prop.   |         | 0.1991 ± 0.0000 | 0.2090 ± 0.0000 |
| vKITTI Prop. | Prop.   |         | 0.1653 ± 0.0038 | 0.1770 ± 0.0066 |
| vKITTI Prop. | Prop.   | [53]    | 0.1825 ± 0.0037 | 0.1660 ± 0.0066 |
| vKITTI Prop. | Prop.   | [53]    | 0.1962 ± 0.0220 | 0.1685 ± 0.0050 |

### TABLE XI

RMSE of Stereo-based Depth Estimation. *FT* stands for fine tuning.

| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI FT     | DiffNet | [21]    | 7.1615 ± 0.0000 | 9.2327 ± 0.0000 |
| KITTI Prop.  | DiffNet | [21]    | 8.3708 ± 2.5778 | 8.0995 ± 0.2314 |
| KITTI [53] FT Prop. | DiffNet | [21]    | 10.4766 ± 5.8117 | 7.4932 ± 0.0850 |
| KITTI FT    | Prop.   |         | 7.5129 ± 0.0000 | 8.8856 ± 0.0000 |
| KITTI Prop.  | Prop.   |         | 6.3309 ± 0.2618 | 7.4888 ± 0.2729 |
| KITTI Prop.  | Prop.   | [53]    | 7.9609 ± 1.7703 | 7.4235 ± 0.0857 |
| vKITTI FT    | DiffNet | [21]    | 10.8090 ± 0.0000 | 11.2852 ± 0.0000 |
| vKITTI Prop. | DiffNet | [21]    | 8.7305 ± 8.2799 | 7.9432 ± 1.6038 |
| vKITTI [53] FT Prop. | DiffNet | [21]    | 8.1609 ± 4.3099 | 64.6970 ± 80.7587 |
| vKITTI FT    | Prop.   |         | 15.5476 ± 0.0000 | 11.3597 ± 0.0000 |
| vKITTI Prop. | Prop.   |         | 7.5368 ± 0.1035 | 6.6097 ± 0.4084 |
| vKITTI Prop. | Prop.   | [53]    | 14.6367 ± 8.4462 | 26.1553 ± 26.3570 |

### TABLE XII

Square Relative of Stereo-based Depth Estimation. *FT* stands for fine tuning.

| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI FT     | DiffNet | [21]    | 2.1568 ± 0.0000 | 2.5450 ± 0.0000 |
| KITTI Prop.  | DiffNet | [21]    | 58.9297 ± 80.4640 | 2.3211 ± 0.4232 |
| KITTI [53] FT Prop. | DiffNet | [21]    | 3934.6596 ± 5560.9138 | 1.7403 ± 0.0675 |
| KITTI FT    | Prop.   |         | 2.4475 ± 0.0000 | 2.3190 ± 0.0000 |
| KITTI Prop.  | Prop.   |         | 2.0111 ± 0.4810 | 1.7943 ± 0.0843 |
| KITTI Prop.  | Prop.   | [53]    | 623.9802 ± 876.0476 | 1.7083 ± 0.0845 |
| vKITTI FT    | DiffNet | [21]    | 4.0448 ± 0.0000 | 10.7284 ± 0.0000 |
| vKITTI Prop. | DiffNet | [21]    | 2.6488 ± 0.9602 | 24.4938 ± 30.5706 |
| vKITTI [53] FT Prop. | DiffNet | [21]    | 5.7252 ± 4.7819 | 280307.3590 ± 396372.1155 |
| vKITTI FT    | Prop.   |         | 17.3129 ± 0.0000 | 18.1648 ± 0.0000 |
| vKITTI Prop. | Prop.   |         | 1.9370 ± 0.1294 | 2.1477 ± 0.3456 |
| vKITTI Prop. | Prop.   | [53]    | 106.3278 ± 144.5834 | 112334.4625 ± 158840.1505 |
### TABLE XIII

**Log RMSE of Stereo-based depth estimation.** FT stands for fine tuning.

| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI        | FT     | DiffNet | 0.2992 ± 0.0000 | 0.3515 ± 0.0000 |
| KITTI        | Prop.  | DiffNet | 0.2999 ± 0.0033 | 0.3223 ± 0.0119 |
| KITTI (53)   | DiffNet | Prop.   | 0.2855 ± 0.0075 | 0.2990 ± 0.0050 |
| KITTI        | FT     | Prop.   | 0.3004 ± 0.0000 | 0.3488 ± 0.0000 |
| KITTI        | Prop.  | Prop.   | 0.2853 ± 0.0023 | 0.2984 ± 0.0083 |
| KITTI (53)   | Prop.   |         | 0.2840 ± 0.0030 | 0.2966 ± 0.0048 |
| vKITTI       | FT     | DiffNet | 0.3086 ± 0.0000 | 0.3307 ± 0.0000 |
| vKITTI       | Prop.  | DiffNet | 0.3152 ± 0.0102 | 0.3070 ± 0.0086 |
| vKITTI (53)  | DiffNet | Prop.   | 0.3040 ± 0.0051 | 0.3156 ± 0.0328 |
| vKITTI       | FT     | Prop.   | 0.3220 ± 0.0000 | 0.3043 ± 0.0000 |
| vKITTI       | Prop.  | Prop.   | 0.2968 ± 0.0053 | 0.2816 ± 0.0064 |
| vKITTI (53)  | Prop.   |         | 0.2953 ± 0.0037 | 0.2853 ± 0.0014 |

### TABLE XIV

**δ(1.25) of Stereo-based depth estimation.** FT stands for fine tuning.

| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI        | FT     | DiffNet | 0.7691 ± 0.0000 | 0.7452 ± 0.0000 |
| KITTI        | Prop.  | DiffNet | 0.7613 ± 0.0089 | 0.7821 ± 0.0117 |
| KITTI (53)   | DiffNet | Prop.   | 0.7892 ± 0.0141 | 0.8143 ± 0.0168 |
| KITTI        | FT     | Prop.   | 0.7652 ± 0.0000 | 0.7296 ± 0.0000 |
| KITTI        | Prop.  | Prop.   | 0.7866 ± 0.0016 | 0.8118 ± 0.0231 |
| KITTI (53)   | Prop.   |         | 0.7901 ± 0.0026 | 0.8209 ± 0.0079 |
| vKITTI       | FT     | DiffNet | 0.7962 ± 0.0000 | 0.7590 ± 0.0000 |
| vKITTI       | Prop.  | DiffNet | 0.7904 ± 0.0149 | 0.7685 ± 0.0101 |
| vKITTI (53)  | DiffNet | Prop.   | 0.7973 ± 0.0031 | 0.7763 ± 0.0063 |
| vKITTI       | FT     | Prop.   | 0.7795 ± 0.0000 | 0.7607 ± 0.0000 |
| vKITTI       | Prop.  | Prop.   | 0.8057 ± 0.0058 | 0.7895 ± 0.0076 |
| vKITTI (53)  | Prop.   |         | 0.8176 ± 0.0120 | 0.7855 ± 0.0046 |

### TABLE XV

**δ(1.25^2) of Stereo-based depth estimation.** FT stands for fine tuning.

| Online Train | Method | Network | TDP       | NDP       |
|--------------|--------|---------|-----------|-----------|
| KITTI        | FT     | DiffNet | 0.9103 ± 0.0000 | 0.8795 ± 0.0000 |
| KITTI        | Prop.  | DiffNet | 0.9134 ± 0.0009 | 0.9030 ± 0.0078 |
| KITTI (53)   | DiffNet | Prop.   | 0.9187 ± 0.0020 | 0.9163 ± 0.0026 |
| KITTI        | FT     | Prop.   | 0.9087 ± 0.0000 | 0.8806 ± 0.0000 |
| KITTI        | Prop.  | Prop.   | 0.9182 ± 0.0005 | 0.9208 ± 0.0057 |
| KITTI (53)   | Prop.   |         | 0.9187 ± 0.0017 | 0.9218 ± 0.0017 |
| vKITTI       | FT     | DiffNet | 0.9152 ± 0.0000 | 0.8930 ± 0.0000 |
| vKITTI       | Prop.  | DiffNet | 0.9041 ± 0.0074 | 0.9088 ± 0.0042 |
| vKITTI (53)  | DiffNet | Prop.   | 0.9183 ± 0.0018 | 0.9141 ± 0.0035 |
| vKITTI       | FT     | Prop.   | 0.9075 ± 0.0000 | 0.9113 ± 0.0000 |
| vKITTI       | Prop.  | Prop.   | 0.9194 ± 0.0059 | 0.9178 ± 0.0019 |
| vKITTI (53)  | Prop.   |         | 0.9235 ± 0.0033 | 0.9178 ± 0.0013 |