D4D: An RGBD Diffusion Model to Boost Monocular Depth Estimation

Lorenzo Papa, Paolo Russo, and Irene Amerini, *Member, IEEE*

***Abstract*— Ground-truth RGBD data are fundamental for a wide range of computer vision applications; however, those labeled samples are difficult to collect and time-consuming to produce. A common solution to overcome this lack of data is to employ graphic engines to produce synthetic proxies; however, those data do not often reflect real-world images, resulting in poor performance of the trained models at the inference step. In this paper we propose a novel training pipeline that incorporates Diffusion4D (D4D), a customized 4-channels diffusion model able to generate realistic RGBD samples. We show the effectiveness of the developed solution in improving the performances of deep learning models on the monocular depth estimation task, where the correspondence between RGB and depth map is crucial to achieving accurate measurements. Our supervised training pipeline, enriched by the generated samples, outperforms synthetic and original data performances achieving an RMSE reduction of (8.2%, 11.9%) and (8.1%, 6.1%) respectively on the indoor NYU Depth v2 and the outdoor KITTI dataset.**

***Index Terms*— Computer vision, diffusion models, deep learning, monocular depth estimation, generation.**

I. Introduction

DEEP learning has achieved astonishing results in several research fields encouraging its fast growth in all of its aspects, from the study of neural network structure to its optimization. In computer vision and image processing, it has gained significant success in tasks like object detection, depth estimation, and semantic segmentation [1]. However, the increasing size and capacity of neural network architectures require the availability of a huge amount of labeled training data, which are often missing or difficult to collect. This issue led researchers to focus on several techniques to reduce the data requirements, such as unsupervised [2] or self-supervised [3] learning strategies, with the objective of categorizing unlabeled or partially labeled data. However, unsupervised learning is intrinsically more complex than (data-driven) supervised learning due to the lack of labeled output samples. Another possible solution could be the use of AI-based methodologies [4] to automatically generate realistic samples and data augmentation techniques [5] exploited to increase the diversity of training data. Nevertheless, the latter techniques are usually constrained by the mathematical transformations that can be used to modify original images while preserving their information. Moreover, the automatic generation of realistic samples has been typically attributed to variational autoencoders (VAEs) and generative adversarial networks (GANs), which lack of samples' variety and details. Differently, a commonly used solution to generate novel datasets is based on synthetic rendering such as Unity® [6] and Unreal Engine® [7] frameworks. Unfortunately, those technologies often fail to provide realistic data, lacking of many realistic features such as accurate light reflections, camera artifacts, and noisy data. As a result, the data distribution of real samples will differ from synthesized ones, despite many works have been proposed to address the problem via domain adaptation and randomization approaches [8], [9], [10].

The lack of a large amount of ground truth data is particularly significant in the case of dense prediction applications, such as depth estimation, where RGB images and corresponding depth maps are required to perform the task. This situation is likely related to the difficulties and highly time-consuming procedures needed to collect congruent RGB and depth data. Such issues are not limited to calibration and alignment procedures between cameras and depth sensors but are also related to unfilled depth maps captured with LiDAR devices and the wide range of possible scenarios. Even if many RGBD datasets have been proposed [11], most of them include less than $50K$ real-world samples such as NYU Depth v2 (NYU) [12] and KITTI [13] datasets. In contrast, millions of labeled samples are available for other computer vision tasks such as image classification (ImageNet [14]) and object detection (COCO [15]). Consequently, the objective of this paper is to automatically generate realistic RGBD samples in order to increase the amount of training data while improving the deep learning model's performances, aiming to overcome the limits of data augmentation and synthetically created samples. Our proposed solution, named Diffusion4D (D4D), is based on *denoising diffusion probabilistic models* (DDPMs) [16], [17], a score-based generation techniques that have shown outstanding results in the creation of high-fidelity images [18]. Our strategy focuses on a custom 4-channels DDPM to capture the intrinsic information presents in real indoor and outdoor RGBD samples in order to generate

Manuscript received 29 May 2023; revised 25 August 2023, 2 November 2023, 15 December 2023, and 16 February 2024; accepted 20 May 2024. Date of publication 22 May 2024; date of current version 30 October 2024. This study has been partially supported by the Italian Ministry of Enterprises and Made in Italy (Ministero delle Imprese e del Made in Italy - MIMIT) with the project PDMI under the agreements for innovation in the automotive sector D.M. 31/12/2021 and DD 10/10/2022. This article was recommended by Associate Editor J. Zhang. *(Corresponding author: Lorenzo Papa.)*

The authors are with the Department of Computer, Control and Management Engineering, Sapienza University of Rome, 00185 Rome, Italy (e-mail: papa@diag.uniroma1.it; paolo.russo@diag.uniroma1.it; amerini@diag.uniroma1.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2024.3404256.

Digital Object Identifier 10.1109/TCSVT.2024.3404256

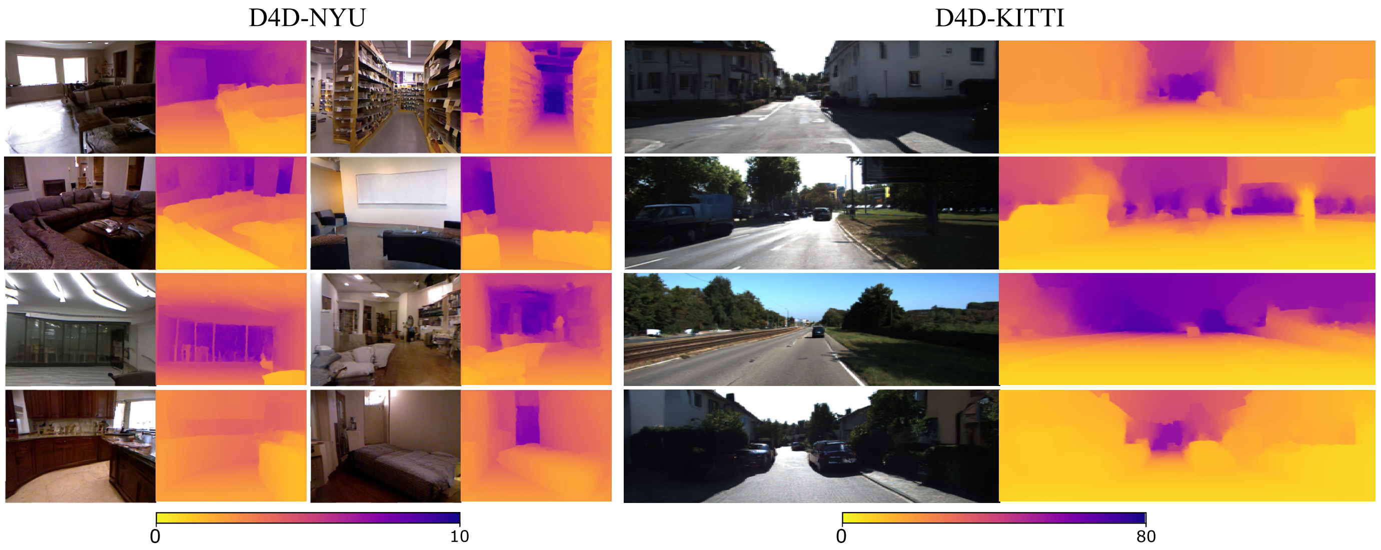


Fig. 1. D4D generated RGBD samples based on the indoor NYU Depth v2 (right) and the outdoor KITTI (left) datasets. The images are scaled to match the aspect ratio of the original samples. The depth maps are converted in RGB format with a perceptually uniform colormap for a better view, while the two bottom colorbars emphasize the depth data distribution (in meters) over the generated samples.

realistic RGB images and corresponding depth maps while improving the data diversity between training samples. D4D introduces customized architecture configurations which are based on 4-channels samples, fine-tuned loss functions, and diffusion schedules. The designed models are used to drive the learning procedure of the DDPM to generate (unconditioned[1]) heterogeneous variations of the original RGBD dataset. Exploiting the characteristic of DDPMs based on the principle of non-equilibrium statistical physics, our aim is to extract key features of real RGBD samples during the forward (inference) process; subsequently, during the backward (generative) phase, the model generates realistic variations of original data obtained merging previously learned features. Therefore, we do not target the production of highly photo-realistic images rather than coherent samples where RGB values and depth distances are correlated as in real-world; some examples are shown in Figure 1. Furthermore, to demonstrate the effectiveness of generated RGBD samples, we apply D4D in a novel supervised training pipeline to tackle the monocular depth estimation (MDE) [19] task, a dense prediction task consisting of estimating a per-pixel distance map given a single RGB image as input.

The main contributions of this work are summarized as follows: **1)** We design a customized 4-channels diffusion model to generate realistic RGBD samples. **2)** We incorporate D4D-generated data into a novel training pipeline to boost MDE models' performances. **3)** We demonstrate the effectiveness of the proposed training strategy to tackle the MDE task over four reference MDE models. In particular, we focus on three convolution neural networks (CNN) and one hybrid vision transformer (hViT), which are respectively DenseDepth [20], FastDepth [21], SPEED [22], and METER [23]. We identify those architectures in order to provide a general overview of the adaptability of the proposed solution over various MDE architectures; precisely, in Section V, we will report the quantitative and qualitative estimation error reduction achieved with the employment of the D4D training pipeline over both indoor and outdoor scenarios. Furthermore, we report some additional experiments on two efficient ViT architectures proposed in [24]. Subsequently, we show the superior performances of generated samples in three settings: **3.1)** When the training of MDE models is performed without the original dataset. **3.2)** When compared against synthetic datasets, such as SceneNet RGB-D [25] and SYNTHIA-SF [26] datasets. **3.3)** In generalization performances on the indoor DIML/CVL RGB-D [27] test dataset in blind conditions. **4)** Finally, we created two new datasets, namely D4D-NYU and D4D-KITTI, each dataset refers to the original one (NYU, KITTI) and it is internally divided according to the generation resolution used. The datasets collect D4D-generated RGBD samples at a variety of resolutions, ranging from 64×48 pixels to 320×240 pixels. We hope that such datasets could be further exploited to improve the performances of MDE architectures and other depth-based tasks. The project page and generated datasets are publicly available at the following link `https://github.com/lorenzopapa5/Diffusion4D`.

This paper is organized as follows: Section II reviews some previous works related to the topics of interest. Section III describes the proposed D4D method and the overall training pipeline in detail. Experiments and hyper-parameters are discussed in Section IV, while Section V reports the qualitative and quantitative improvements achieved by the chosen MDE model with the use of D4D generated samples. Some final considerations and future applications are provided in Section VI.

II. Related Work

The task of producing new samples from an existing data collection is known as generation. There are two basic generation methodologies: unconditioned, in which the samples are generated from noise (i.e., Gaussian noise), and conditioned,

[1]The unconditioned generation techniques are identified by the absence of additional input data.

in which the samples are generated in response to a given input, e.g., text prompts and images. In AI-based approaches, this task is usually tackled through VAEs, GANs, and the recent DDPMs, deep learning techniques commonly based on convolutional and transformer operations. Many aspects in developing models for generating realistic images have been studied and improved during these years, such as conditioning the output with ad-hoc input variables as well as speeding up the process by working on the efficiency and inference frequency. Zhu et al. [28] propose DM-GAN, a text-conditioned architecture able to improve the quality of generated samples based on information prompts. Karras et al. [29] focus on an augmentation solution for training a GAN model under limited data constraints. Cai et al. [30] propose a deep convolutional GAN solution to generate synthetic data to tackle the imbalanced problem of training datasets for crash prediction scenarios. Zhao et al. [31] integrate and optimize the computational complexity of transformer architectures into a GAN-based approach in order to produce high-resolution images.

Furthermore, generative models have also been widely applied to handle the image translation task, in which an input image from one domain is translated (mapped) to another one while preserving the content of the given image. An example is provided by Zhu et al. [32] with CycleGAN, where the authors mainly focus on a cycle consistency loss to enhance the overall generation performances. Russo et al. [33], inspired by [32], introduce a class consistency loss for cross-domain classification tasks. Moreover, Tang et al. [34] propose to guide the translation process through an attention mechanism in order to achieve high-fidelity images, whereas Torbunov et al. [35] improve CycleGAN performances by incorporating transformers layers as the generator. Similarly to previous related works and closer to our application scenario, Du et al. [36] present a specific domain shift model to extract depth maps from RGB images. This work has been motivated by the limited amount of labeled data provided in existing RGBD datasets,

Recently, DDPMs [17], a powerful new family of deep generative models have been proposed. Such architectures are based on two Markov chains: a forward chain that perturbs input data to noise and a reverse chain that translates noise to data. Ho et al. [16] demonstrate DDPM capabilities in computer vision applications for the generation of high-quality images. Moreover, Dhariwal et al. [37] shows that such models are able to achieve superior performances than GANs to handle image synthesis. However, those architectures require substantial computational resources to be trained; consequently, Rombach et al. [38] propose a latent diffusion model that can be trained on limited computational resources proposing to integrate the Markovian structure into the latent space of a pretrained autoencoder network. Contrarily, Peebles and Xie et al. [39] replace the commonly-used U-Net [40] with transformer modules improving the generation capabilities while increasing the computational complexity.

In contrast to such AI-based approaches, another popular solution for the generation of (potentially unlimited) samples is based on the extraction of frames and associated ground truth data from virtual environments, i.e., generated via graphical engines such as Unity®, Unreal Engine® and the most recent NVIDIA Isaac Sim™ (Replicator) [41]. Those technologies often fail to provide realistic data, lacking artifact information commonly present in real-world images, resulting in poor performance at the inference step. Synthetic datasets, generated with graphic engines, have been widely employed in the MDE task. Zou et al. [42] use the synthetic SYNTHIA datasets as a pre-training strategy to improve depth estimation performances on autonomous driving scenarios, while Chen et al. [43] employ the synthetic SceneNet dataset to increase the number of training samples and the model's generalization performances. Contrarily, Xian et al. [44] propose to estimate pseudo-depth data trained on relative depth datasets to improve the model's generalization in real-world scenarios. The work also underlies the presence of a domain gap between synthetic and real data, as well as the need for domain adaptation techniques to efficiently use synthesized samples.

Consequently, based on similar motivation of [36] and [44], in this paper, we integrate in a novel training pipeline a custom 4-channels DDPM in order to generate realistic RGBD samples for both indoor and outdoor contexts and improve the estimation performances of MDE approaches while overcoming the limitations introduced by graphical engines. To the best of our knowledge, no previous works propose a similar solution to improve a dense prediction task; a detailed description of the proposed training pipeline is following reported.

III. Methodology

This section describes the proposed pipeline for generating RGBD samples with the D4D model. As mentioned in Section I, one of the primary bottlenecks in the MDE task, a computer vision application where a dense depth map is predicted from a single RGB image, is the lack of a large amount of training data. Therefore, the proposed training pipeline aims to improve the estimation performances of well-known MDE architectures by generating RGBD samples learned from real-world 4-channels (images) data distribution. We report a graphical representation in Figure 2; as can be seen, the pipeline is divided into three stages described below.

Stage 1: The first phase is characterized by widely employed preprocessing techniques. More in detail, we select as training datasets the NYU for the indoor scenarios and the KITTI for the outdoor ones, both of which are composed of real-world RGBD samples. Furthermore, the pixel values of the training samples are normalized into the $[0, 1]$ range and rescaled to the working model resolution. Consequently, the image's height and width are scaled (resized with a bilinear interpolation process) to the working resolutions of the compared architectures used in Stage 3, such as 640×480 (DenseDepth), 224×224 (FastDepth), and 256×192 (SPEED and METER). This choice will influence (in Stage 2) the generation resolution of D4D model at inference time.

Stage 2: The second phase is devoted to generating realistic samples; precisely, we leverage our custom DDPM to produce 4-channels samples based on the original training data. Before introducing our generation strategy, let us briefly review some basic concepts necessary to better understand DDPMs, highlighting the motivations that led us to develop

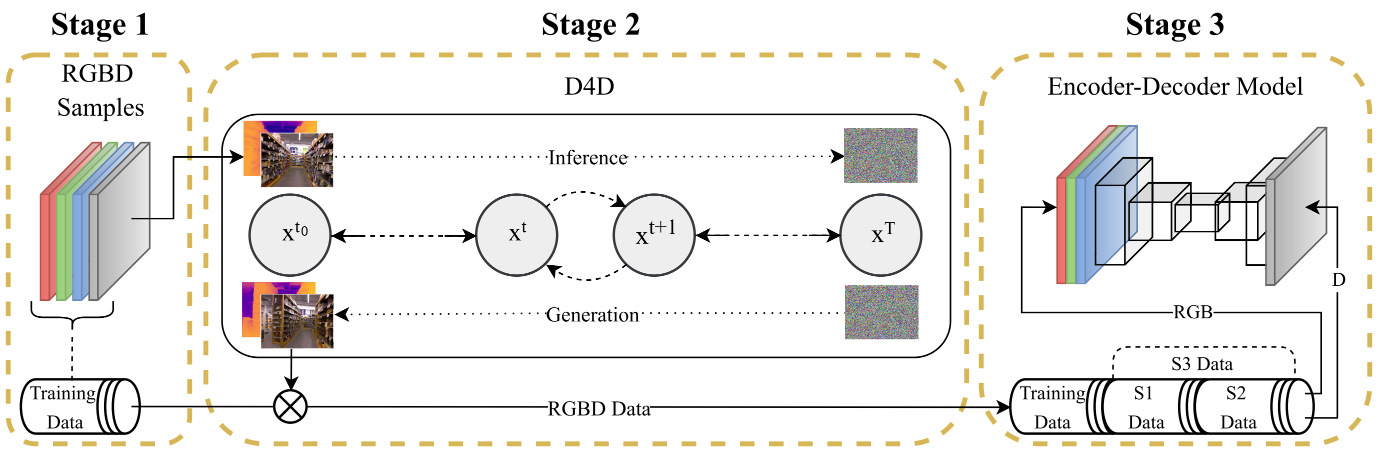


Fig. 2. Graphical representation of the introduced training pipeline. **Stage 1** shows the pre-processing operations applied on 4-channels samples extracted from the original training dataset. **Stage 2** emphasizes the training and unconditioned generation processes of D4D model. **Stage 3** depicts the training procedure of a generic encoder-decoder MDE network by highlighting how the RGBD training samples are composed.

the proposed solutions. DDPMs, inspired by non-equilibrium statistical physics, exploit the reduction of the input data distribution into a well-known one, in our case, the Gaussian distribution. This process, known as forward diffusion (inference), is then reversed (generation) to restore input data distribution. This procedure is commonly defined in literature as *highly flexible* and *tractable* since the model can potentially represent unlimited data distributions. According to this behavior, the straightforward baseline idea of this paper is to use a DDPM to learn the distribution of RGBD data from real-world benchmark datasets during the forward phase. As a result, during the generation phase, D4D could produce multiple realistic 4-channels variations of original ground-truth data by combining previously extracted features.

Therefore, we introduce some basic knowledge about diffusion model methodologies by focusing on the main parameters that would impact D4D generation performance. More in detail, diffusion models are characterized by forward and reverse procedures. The training process of our diffusion model is principally driven by the cost function $L(\cdot,\cdot)$ and the diffusion rate β. The first function, usually a $L1(\cdot,\cdot)$ (mean-absolute) or $L2(\cdot,\cdot)$ (mean-squared) loss, is computed between the input data distribution $q(x^{t_0})$ and the generated one $p(x^{t_0})$ to fit the DDPM data distribution $\pi(y)$, which usually represents a Gaussian distribution. At the forward phase, the diffusion rate, as defined in [17], drives the Markov diffusion kernel $t_\pi(y|y';\beta_t)$ with $t=[t_0;T]$ steps, to make the distribution $\pi(y)$ analytically tractable, while the reverse phase is trained to describe the same trajectory, but in a reverse way; we report the two procedures in the following equations.

$$forward \rightarrow q(x^t) = q(x^{t_0})\Pi_{t_0}^{T} t_\pi(x|x';\beta_t) \quad (1)$$

$$reverse \rightarrow p(x^t) = \pi(x^T)\Pi_{t_0}^{T} t_\pi(x'|x) \quad (2)$$

Moreover, the configuration of the diffusion rate is fundamental for its final performances; in [16] and [17] authors set a *linear* β variance ranging from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$ with $T = 1000$. In contrast, in [45], authors propose to improve diffusion models with a reparametrization of the generation process variance, i.e., replacing the linear schedule with a squared *cosine* to prevent abrupt changes of noise levels. This choice leads to a slower forward process with $T = 4000$ steps while increasing reconstructed image details.

Based on the just introduced description on diffusion model methodologies and influenced by the loss function formulation commonly employed in the MDE task [20], [46], where the learning process usually relies on multiple loss functions focused on contours, fine details, and images as a whole, we design D4D with a similar behavior. Precisely, the proposed strategy would combine two configurations of loss functions and beta scheduler setups in order to ensure diversity and consistency in the generated RGBD samples. The combination of diversity and consistency of the generated samples, which are combined into the training set, act as a powerful and realistic data augmentation schema, which is able to increase the generalization capabilities of our network, resulting in a lower testing error as shown in the Results section. More in detail, we propose a merging strategy based on two complementary configurations, namely S1 and S2, that are able to generate realistic samples with various data distributions in order to enhance the overall depth estimation performances of well-known MDE models. In the first configuration (S1), the model focuses on creating realistic images mainly composed of constant or gradually increasing depth distances. As a result, we develop S1 with a slow convergence behavior, i.e., characterized by an $L1$ loss function to mitigate the error during the training process, and a linear diffusion rate (β) [16], [17] leading the model to a faster forward process with the constant addition of noisy data. Moreover, by defining with $\mathcal{P}$ the set of pixels, for any pixel $p \in \mathcal{P}$, the S1 configuration can be formalized as reported in Equation 3.

$$S1 : L1 = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} ||x_p - y_p||_1, \quad \beta = linear \quad (3)$$

In contrast, in the second configuration (S2), we look for generated images that are rich in detail with stronger distance variations. Consequently, we implement S2 with a slower forward process better focusing on details and objects in the images, i.e., a cosinusoidal diffusion scheme (β) [45]

combined with a $L2$ loss function to achieve a fast convergence of the learning system. Moreover, by defining with $\mathcal{P}$ the set of pixels, for any pixel $p \in \mathcal{P}$, the S2 configuration can be formalized as reported in Equation 4.

$$S2 : L2 = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} ||x_p - y_p||_2^2, \quad \beta = cosine \tag{4}$$

Finally, the proposed configuration (S3) is composed by merging the generated RGBD samples from S1 and S2. We opted to set the number of steps T equal to 1000 as a trade-off between training time and image photorealism. Under these settings, S3 effectively encompasses a wide range of possible RGB and depth data distributions while balancing the convergence speed and the diffusion rate of the 4-channels DDPM. Moreover, by defining with $s1$ and $s2$ the set of generated RGBD data, respectively, from S1 and S2 configurations, the proposed strategy can be summarized as follows:

$$S3 = (s1 \cup s2) \ where \begin{cases} S1 : \{loss : L1, & \beta : linear\} \\ S2 : \{loss : L2, & \beta : cosine\} \end{cases} \tag{5}$$

We conclude this stage by merging the generated RGBD samples with the original training data in order to create a unique augmented training set. Furthermore, because DDPM has a significant computing cost during the training and generation stages, we perform all of the operations described in this step offline.

Stage 3: Following the proposed training pipeline, in the last phase, we employ the novel augmented training set to tackle the MDE task. Precisely, we employ the RGB images and respective depth maps to train commonly used encoder-decoder architectures, which are represented as transparent blocks in Figure 2; in particular, we focus on DenseDepth, FastDepth, SPEED, and METER, which are typically deep and shallow architectures commonly used in the MDE task. We chose these models due to their different working resolutions, architectural components, and estimation capabilities in order to demonstrate the effectiveness of D4D-generated samples at different scales and performances. This final phase is fundamental for demonstrating the efficacy of the proposed training pipeline and for quantitatively measuring the attained improvement.

IV. Experimental Setup

In this section, we describe hyperparameter setups of trained architectures and evaluation metrics used to compare their performances. The proposed method is implemented on the PyTorch framework [47]. To generate new samples with the D4D procedure, we employ two benchmark MDE datasets, i.e., NYU Depth v2 and KITTI, following the Eigen et al. [48] split strategy. NYU and KITTI are respectively composed of around $(50K, 23K)$ training and $(654, 652)$ test samples at a resolution of $(640 \times 480, 1242 \times 375)$ and a maximum depth range of (10, 80) meters. Furthermore, to compare the performances achieved by generated samples with respect to synthetic ones (Figure 2, Stage 3), we use the SceneNet dataset for the indoor scenario and the SYNTHIA-SF for the outdoor one. We use a subset of $300K$ samples for the first dataset and the entire training set for the second one, composed of $3K$ samples. Finally, we use the 503 samples of the DIML test dataset to show the generalization performances on an unseen set of data. Moreover, following the training pipeline outlined in the previous section, we describe the hyperparameters and evaluation metrics used in this paper.

In Stage 2, we train each configuration (S1 and S2) at different image resolutions ranging from 64×48 pixels to 320×240 pixels on NYU and KITTI datasets. The DDPM layers are initialized as described in [16] and [49]. We train D4D for 150 epochs with a batch size ranging from 256 to 16 depending on the image resolution on an NVIDIA A100 SXM4. We use Adam as optimizer with decoupled weight decay [50] of 1×10^{-2}, a learning rate equal to 1×10^{-4} and a decay of 1×10^{-1} after 100 and 125 epochs. Following common practice we set remaining hyperparameters as $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$.

In Stage 3, we train all the compared MDE models (DenseDepth, FastDepth, SPEED, and METER) with the following hyperparameter setting: we use Adam optimizer configuration as before with a learning rate equal to 1×10^{-3} and a decay of 1×10^{-1} every 20 epochs for a total of 80 epochs on an NVIDIA RTX 3090. Furthermore, we initialized the convolutional kernels as suggested in respective papers [20], [21], [22], [23] and trained/tested the MDE architectures with original input-output model resolutions, i.e., $(640 \times 480, 320 \times 240)$, $(224 \times 224, 224 \times 224)$, $(256 \times 192, 64 \times 48)$ and $(256 \times 192, 64 \times 48$ or $640 \times 192, 160 \times 48)$[2] respectively for DenseDepth, FastDepth, SPEED, and METER. The training procedure is further enriched using the strategy proposed in [20] with the addition of the random crop. Finally, we evaluate the trained models following the evaluation metrics introduced in [48]: root mean squared error (RMSE, in meters [m]), mean absolute error (MAE, in meters [m]), absolute relative error (Abs_{Rel}), and accuracy values such as δ_1, δ_2 and δ_3. Moreover, for any pixel $p \in \mathcal{P}$, we define its ground truth depth map as y_p while $\hat{y}_p$ is the predicted one. Those evaluation metrics are formally defined in the following equations.

$$RMSE = \sqrt{\frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} ||y_p - \hat{y}_p||^2} \tag{6}$$

$$MAE = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} |y_p - \hat{y}_p| \tag{7}$$

$$Abs_{Rel} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \frac{|y_p - \hat{y}_p|}{y_p} \tag{8}$$

For estimating the accuracy values $\delta_{z \in \mathbf{N}}$ with $z \in [1, 3]$, a threshold (thr) is commonly set to 1.25^z while the set of

[2]Differently to the other compared CNN architecture, METER has different image resolutions between the indoor and outdoor scenarios (same height but different width).

TABLE I

QUANTITATIVE EVALUATION OF DIFFERENT MDE ARCHITECTURES AND CONFIGURATIONS. THE ORIGINAL SAMPLES ARE TAKEN FROM NYU DATASET (THIRD COLUMN, NYU = 50K), THE *Synthetic* SAMPLES ARE FROM SCENENET, WHILE THE GENERATED SAMPLES (*Add*) ARE FROM D4D-NYU. THE PROPOSED S3 CONFIGURATION IS IN BOLD, WHILE THE OPTIMAL STRATEGY FOR EACH COMPARED MODEL IS HIGHLIGHTED IN GRAY

Model	Configuration	NYU [K]	*Add* [K]	*Res* [pix]	RMSE↓ [m]	MAE↓ [m]	Abs$_{Rel}$↓	δ_1↑	δ_2↑	δ_3↑
DenseDepth	S0	50	0	-	0.5021	0.3663	0.1445	0.8087	0.9507	0.9846
	Synthetic	50	50	320 × 240	0.4882	0.3438	0.1367	0.8199	0.9583	0.9888
	Synthetic	50	150	320 × 240	0.4713	0.3487	0.1358	0.8251	0.9634	0.9910
	S1	50	50	320 × 240	0.4575	0.3352	0.1294	0.8379	0.9640	0.9899
	S2	50	50	320 × 240	0.4598	0.3354	0.1273	0.8390	0.9667	0.9921
	S3	50	50	320 × 240	0.4568	0.3368	0.1327	0.8340	0.9659	0.9912
	S3	50	100	320 × 240	0.4480	0.3262	0.1236	0.8499	0.9693	0.9923
	S3	50	50	256 × 192	0.4788	0.3513	0.1340	0.8241	0.9614	0.9912
	S3	50	100	256 × 192	0.4578	0.3364	0.1286	0.8376	0.9672	0.9917
FastDepth	S0	50	0	-	0.5714	0.4317	0.1751	0.7535	0.9374	0.9820
	Synthetic	50	100	320 × 240	0.5468	0.4122	0.1617	0.7747	0.9450	0.9858
	Synthetic	50	300	320 × 240	0.5198	0.3883	0.1519	0.7948	0.9533	0.9870
	S1	50	100	256 × 192	0.5029	0.3741	0.1455	0.8058	0.9586	0.9892
	S2	50	100	256 × 192	0.5313	0.3995	0.1600	0.7775	0.9454	0.9869
	S3	50	100	256 × 192	0.4980	0.3678	0.1414	0.8119	0.9603	0.9901
	S3	50	50	320 × 240	0.5132	0.3810	0.1467	0.8014	0.9553	0.9886
	S3	50	100	320 × 240	0.5103	0.3802	0.1492	0.7903	0.9507	0.9865
SPEED	S0	50	0	-	0.5638	0.4275	0.1676	0.7601	0.9357	0.9836
	Synthetic	50	100	320 × 240	0.5606	0.4247	0.1657	0.7605	0.9404	0.9857
	Synthetic	50	300	320 × 240	0.5542	0.4217	0.1633	0.7696	0.9496	0.9864
	S1	50	100	256 × 192	0.5170	0.3877	0.1482	0.7948	0.9549	0.9897
	S2	50	100	256 × 192	0.5216	0.3943	0.1486	0.7905	0.9565	0.9912
	S3	50	100	256 × 192	0.4982	0.3712	0.1430	0.8054	0.9610	0.9911
	S3	50	50	320 × 240	0.5132	0.3870	0.1494	0.7973	0.9559	0.9885
	S3	50	100	320 × 240	0.5001	0.3767	0.1441	0.8090	0.9587	0.9903
METER	S0	50	0	-	0.5112	0.3854	0.1439	0.8138	0.9577	0.9876
	Synthetic	50	100	320 × 240	0.4893	0.3675	0.1446	0.8130	0.9592	0.9890
	Synthetic	50	300	320 × 240	0.4957	0.3709	0.1446	0.8150	0.9574	0.9882
	S1	50	100	256 × 192	0.4649	0.3471	0.1353	0.8320	0.9685	0.9915
	S2	50	100	256 × 192	0.4760	0.3584	0.1388	0.8202	0.9660	0.9923
	S3	50	100	256 × 192	0.4574	0.3390	0.1290	0.8357	0.9667	0.9924
	S3	50	50	320 × 240	0.4669	0.3495	0.1334	0.8303	0.9673	0.9923
	S3	50	100	320 × 240	0.4615	0.3447	0.1320	0.8350	0.9695	0.9928

pixel $\mathcal{P}_z^*$ is defined as follows:

$$\mathcal{P}_z^* = \left\{ p \in \mathcal{P} \ s.t. \max\left(\frac{y_p}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}\right) < thr^z \right\} \tag{9}$$

Finally, the accuracy values can be expressed as reported in Equation 10.

$$\delta_{z\in\mathbf{N}, z\in[1,3]} = \frac{|\mathcal{P}_z^*|}{|\mathcal{P}|} \tag{10}$$

V. EXPERIMENTS

In this section, we show the effectiveness of the proposed pipeline in terms of improvements obtained over the four chosen MDE models. The first performed analysis is computed with respect to indoor and outdoor D4D-generated datasets, i.e., when selected models are trained by adding the D4D-NYU and D4D-KITTI datasets. Subsequently, we investigate the effects of the different resolutions and amounts of RGBD data generated by D4D on the trained models. We conclude this section by analyzing the generalization performances on an unseen test dataset DIML/CVL RGB-D (DIML), the estimation improvement over efficient variants of METER architecture and with an analysis of similarity distances over probabilistic distributions. We compare the obtained results with respect to S1, S2, S3, a baseline configuration (S0), i.e., when the models are trained on original datasets (NYU and KITTI), as well as an alternative augmentation schema based on synthetic datasets (*Synthetic*).

A. Indoor Results

The first analysis is performed on D4D-NYU dataset under different configurations (Si with $i = [0, 3]$ and *Synthetic*), settings (NYU $= 50K$ or NYU $= 0$), number of generated samples *(Add)* and D4D resolutions (*Res*). These training combinations have been taken in order to show how the presence of the original dataset and the generation resolution of the samples influence the estimation performances of chosen models. Precisely, we report the same tests over the four chosen reference MDE models with and without the original datasets (NYU), respectively in Table I and Table II, in order to understand differences, similarities, and respective quantitative improvement obtained when using generated samples, i.e., how much D4D mimic original samples or how much those generated samples differs from original one. Generally speaking, we noticed that the proposed merging strategy (S3) has superior estimation performances in indoor scenarios with respect to all the compared configurations. Based on the achieved results, we derive that the closer the generation resolution of the samples is to the input resolution of the trained model, the better the estimation

TABLE II

QUANTITATIVE EVALUATION OF DIFFERENT MDE ARCHITECTURES AND CONFIGURATIONS. THE *SYNTHETIC* SAMPLES ARE FROM SCENENET WHILE THE GENERATED SAMPLES (*ADD*) ARE FROM D4D-NYU WHILE NO NYU (ORIGINAL) SAMPLES ARE USED (THIRD COLUMN, NYU = 0K). THE PROPOSED S3 CONFIGURATION IS IN BOLD, WHILE THE OPTIMAL STRATEGY FOR EACH COMPARED MODEL IS HIGHLIGHTED IN GRAY

Model	Configuration	NYU [K]	*Add* [K]	*Res* [pix]	RMSE↓ [m]	MAE↓ [m]	Abs$_{Rel}$↓	δ_1↑	δ_2↑	δ_3↑
DenseDepth	S0	0	0	-	-	-	-	-	-	-
	Synthetic	0	50	320 × 240	1.1034	0.8648	0.4298	0.4123	0.6886	0.8465
	Synthetic	0	150	320 × 240	1.0383	0.8292	0.4019	0.4197	0.7228	0.8825
	S1	0	50	320 × 240	0.5559	0.4250	0.1736	0.7549	0.9373	0.9821
	S2	0	50	320 × 240	0.6087	0.4767	0.1931	0.6619	0.9297	0.9773
	S3	0	50	320 × 240	0.5306	0.4030	0.1580	0.7755	0.9489	0.9873
	S3	0	100	320 × 240	0.5301	0.4003	0.1578	0.7754	0.9490	0.9873
	S3	0	50	256 × 192	0.5473	0.4163	0.1654	0.7654	0.9446	0.9866
	S3	0	100	256 × 192	0.5398	0.4096	0.1597	0.7720	0.9469	0.9877
FastDepth	S0	0	0	-	-	-	-	-	-	-
	Synthetic	0	100	320 × 240	1.1169	0.9779	0.4538	0.3866	0.6903	0.8621
	Synthetic	0	300	320 × 240	1.0852	0.9051	0.4167	0. 4247	0.7275	0.8817
	S1	0	100	256 × 192	0.5709	0.4319	0.1768	0.7543	0.9412	0.9839
	S2	0	100	256 × 192	0.5952	0.4569	0.1845	0.7047	0.9292	0.9842
	S3	0	100	256 × 192	0.5502	0.4165	0.1730	0.7649	0.9464	0.9877
	S3	0	50	320 × 240	0.5735	0.4397	0.1756	0.7468	0.9389	0.9844
	S3	0	100	320 × 240	0.5651	0.4343	0.1721	0.7473	0.9394	0.9854
SPEED	S0	0	0	-	-	-	-	-	-	-
	Synthetic	0	100	320 × 240	1.2278	1.0606	0.5424	0.3159	0.6279	0.8290
	Synthetic	0	300	320 × 240	1.1635	0.9827	0.4732	0.3923	0.6850	0.8532
	S1	0	100	256 × 192	0.5833	0.4430	0.1687	0.7493	0.9385	0.9857
	S2	0	100	256 × 192	0.6003	0.4646	0.1779	0.6875	0.9224	0.9825
	S3	0	100	256 × 192	0.5590	0.4260	0.1622	0.7665	0.9438	0.9874
	S3	0	50	320 × 240	0.5803	0.4482	0.1735	0.7456	0.9352	0.9852
	S3	0	100	320 × 240	0.5694	0.4379	0.1674	0.7439	0.9423	0.9862
METER	S0	0	0	-	-	-	-	-	-	-
	Synthetic	0	100	320 × 240	1.2242	1.0100	0.4319	0.3688	0.6770	0.8547
	Synthetic	0	300	320 × 240	1.0480	0.8556	0.3837	0.4468	0.7403	0.8909
	S1	0	100	256 × 192	0.5445	0.4140	0.1636	0.7679	0.9474	0.9863
	S2	0	100	256 × 192	0.5905	0.4574	0.1837	0.7180	0.9322	0.9851
	S3	0	100	256 × 192	0.5370	0.4075	0.1577	0.7711	0.9510	0.9886
	S3	0	50	320 × 240	0.5778	0.4465	0.1709	0.7729	0.9366	0.9862
	S3	0	100	320 × 240	0.5368	0.4125	0.1602	0.7686	0.9491	9887

results, although the error difference is small (e.g., 2.2% of the RMSE in the DenseDepth case). This finding, based on the best D4D generation resolution, has been used in the experiments listed below and will be further investigated in the following ablation studies. Moreover, we observe that by doubling the amount of generated data with respect to the original training dataset (from $50K$ to $100K$), the proposed configuration (S3) outperforms the baseline configuration (S0) and the *Synthetic* datasets with an RMSE reduction equal to (10.8%, 4.9%) on DenseDepth, (14.7%, 9.7%) on FastDepth, (11.6%, 11.1%) on SPEED and (10.5%, 6.5%) on METER. Furthermore, when trained only on D4D-NYU (NYU = 0), S3 is able to achieve better performances than S0 in the case of FastDepth and SPEED, while slightly worse for DenseDepth and METER. Contrarily, the synthetic RGBD data performs poorly without the original training dataset. These results demonstrate the ability of D4D-generated samples to mimic real-world samples. To summarize, the overall average percentage improvement obtained with the proposed training pipeline, computed with respect to the baseline configuration over the evaluation metrics used, is equal to 7.3%, 9.6%, 8.2%, and 6.2% respectively for DenseDepth, FastDepth, SPEED, and METER.

Finally, to have a complete understanding of the obtained improvement, we report in Figure 3 a qualitative comparison of the estimation performances of the DenseDepth model under the compared configurations, i.e., Si with $i = [0, 3]$ and *Synthetic*. Based on predicted depth maps and related difference maps[3] reported for each configuration, we note that DenseDepth, in the synthetic configuration, produces the highest estimation error (more than $100cm$) with respect to compared setups. Contrarily, S3 is the only configuration with an error range less than $80cm$ (demonstrated by darker difference map in Figure 3). Furthermore, we notice that all the compared predicted depth maps have well-defined contours. However, in the reported case, the proposed configuration (S3) is able to correctly estimate distances in the situation where all the others fail, i.e., where the scene distance varies rapidly (e.g., behind a wall); we highlight this area on the difference map with a dashed red rectangle.

B. Outdoor Results

Along with the previous findings, the proposed method (S3) achieves notable estimation improvements also in the outdoor scenario, especially when the D4D generation resolution is close to the MDE model input resolution. We report in Table III the results obtained by the selected MDE models when trained on KITTI dataset and in combination with D4D-KITTI or the synthetic SYNTHIA-SF dataset. Precisely,

[3]The difference map is computed as a per pixel-difference between predicted ($\hat{y}$) and expected (y) depth map.

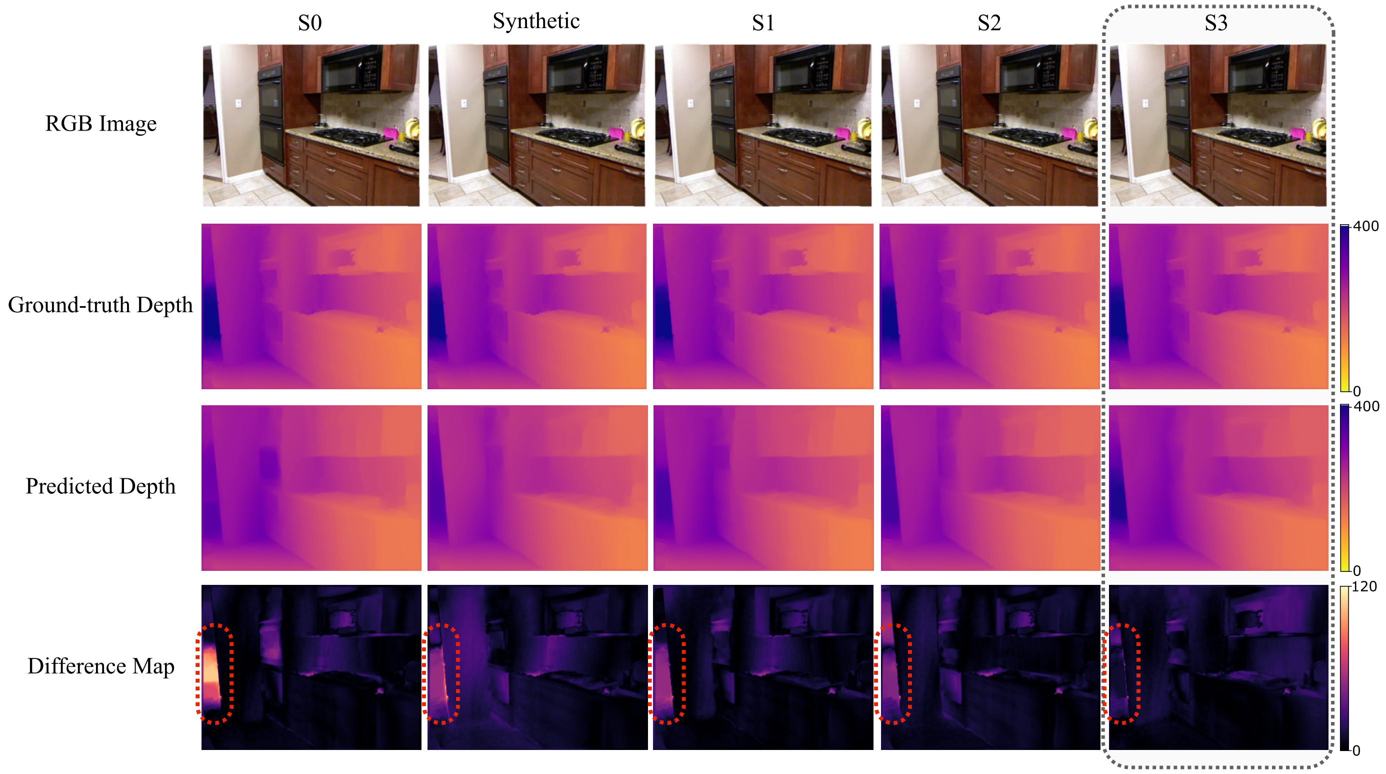


Fig. 3. **Indoor results.** Qualitative analysis of the estimated prediction obtained with DenseDepth method. The model has been tested on NYU (indoor) dataset. S0 is the baseline setup, i.e., when the MDE model is trained only on the NYU dataset. In *Synthetic* setup, DenseDepth has been trained over NYU and a $50K$ subset from the SceneNet dataset. In Si with $i = [1, 3]$, as described in Section III, DenseDepth has been trained over NYU and $50K$ samples taken from our proposed D4D-NYU datasets generated at a resolution of 320×240. The Difference Map is computed as a per pixel-difference between predicted ($\hat{y}$) and expected depth (y), while the reported colorbars are used to emphasize the depth/error range in centimeters (cm).

TABLE III

QUANTITATIVE EVALUATION OF DIFFERENT MDE ARCHITECTURES AND CONFIGURATIONS. THE ORIGINAL SAMPLES ARE TAKEN FROM KITTI DATASET, THE *SYNTHETIC* SAMPLES ARE FROM SYNTHIA-SF WHILE THE GENERATED SAMPLES (*ADD*) ARE FROM D4D-KITTI. THE PROPOSED S3 CONFIGURATION IS IN BOLD, WHILE THE OPTIMAL STRATEGY FOR EACH COMPARED MODEL IS HIGHLIGHTED IN GRAY

Model	Configuration	KITTI [K]	*Add* [K]	*Res* [pix]	RMSE↓ [m]	MAE↓ [m]	Abs$_{Rel}$↓	δ_1↑	δ_2↑	δ_3↑
DenseDepth	S0	23	0	-	5.2099	3.1749	0.1417	0.7991	0.9475	0.9840
	Synthetic	23	3	1940×1080	5.2982	3.2499	0.1448	0.7871	0.9458	0.9856
	S1	23	50	320×240	5.1284	3.0221	0.1341	0.8057	0.9546	0.9882
	S2	23	50	320×240	5.1437	3.0539	0.1349	0.7989	0.9533	0.9869
	S3	23	50	320×240	4.9636	2.9874	0.1294	0.8168	0.9580	0.9892
	S3	23	50	256×192	5.1478	3.1324	0.1337	0.8058	0.9542	0.9883
FastDepth	S0	23	0	-	6.1884	3.9174	0.1910	0.7147	0.9088	0.9684
	Synthetic	23	3	1940×1080	6.1257	3.8100	0.1895	0.7184	0.9182	0.9764
	S1	23	50	256×192	5.9277	3.6774	0.1854	0.7286	0.9240	0.9781
	S2	23	50	256×192	5.9417	3.6994	0.1884	0.7292	0.9223	0.9777
	S3	23	50	256×192	5.6310	3.5062	0.1682	0.7551	0.9316	0.9804
	S3	23	50	320×240	5.8244	0.3613	0.1759	0.7374	0.9290	0.9792
SPEED	S0	23	0	-	5.3957	3.0473	0.1480	0.7797	0.9387	0.9841
	Synthetic	23	3	1940×1080	5.4219	3.1233	0.1565	0.7574	0.9307	0.9808
	S1	23	50	256×192	5.2321	2.9477	0.1409	0.7890	0.9445	0.9848
	S2	23	50	256×192	5.0945	2.8758	0.1401	0.7980	0.9476	0.9857
	S3	23	50	256×192	4.9828	2.8017	0.1337	0.8104	0.9521	0.9878
	S3	23	50	320×240	5.2640	3.0663	0.1437	0.7823	0.9421	0.9839
METER	S0	23	0	-	4.8398	2.7284	0.1278	0.8153	0.9462	0.9859
	Synthetic	23	3	1940×1080	5.2139	3.0725	0.1468	0.7753	0.9428	0.9847
	S1	23	50	256×192	4.8961	2.7206	0.1275	0.8118	0.9512	0.9864
	S2	23	50	256×192	4.7908	2.8271	0.1456	0.7840	0.9450	0.9845
	S3	23	50	256×192	4.7288	2.6833	0.1308	0.8155	0.9533	0.9875
	S3	23	50	320×240	4.7519	2.6780	0.1314	0.8083	0.9503	0.9857

the maximum RMSE reduction with respect to S0 and the *Synthetic* dataset is obtained by tripling the amount of training data, and it is equal to (4.7%, 6.3%) on DenseDepth, (9.1%, 8.1%) on FastDepth, (8.3%, 8.8%) on SPEED, and (2.3%, 9.3%) on METER. However, we cannot rule out that further improvements could be obtained by greatly increasing

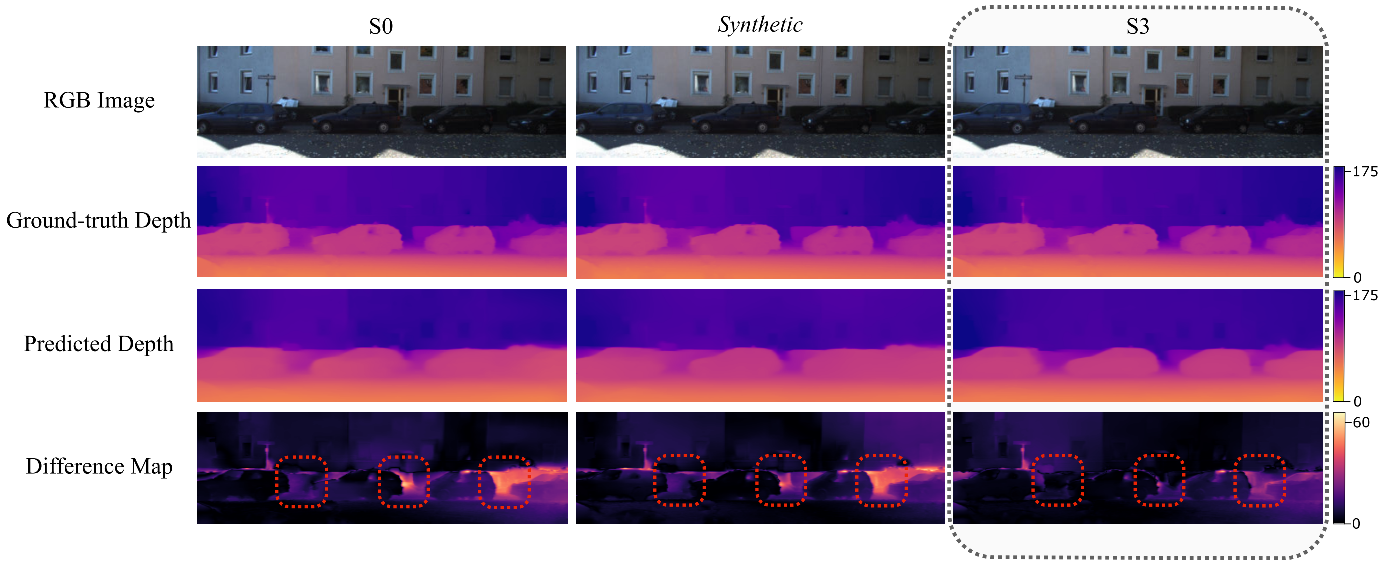


Fig. 4. **Outdoor results.** Qualitative analysis of the estimated prediction obtained with DenseDepth method. The model has been tested on KITTI (outdoor) dataset. S0 is the baseline setup, i.e., when DenseDepth is trained only on KITTI dataset. In *Synthetic* setup, the model has been trained over KITTI and SYNTHIA-SF datasets. In the proposed configuration (S3), the model has been trained over KITTI and $50K$ samples taken from our proposed D4D-KITTI datasets generated at a resolution of 320×240. The Difference Map is computed as a per pixel-difference between predicted ($\hat{y}$) and expected depth (y), while the reported colorbars are used to emphasize the depth/error range in decimeters (dm).

the number of generated samples. Summarizing, the overall average percentage improvement achieved with the proposed training pipeline, when compared with S0, is equal to 4.0%, 6.7%, 5.7%, and $\simeq$ 1.0% respectively, for DenseDepth, FastDepth, SPEED, and METER. The latter results obtained for the hViT architecture are most likely attributed to the D4D generation resolution. Consequently, similar to the indoor scenario, we expect comparable RMSE reductions to the CNN architectures in the case of images generated at the same working resolution of METER. These results confirm the soundness of D4D for increasing the performances of any kind of MDE model. Finally, we report in Figure 4 a qualitative comparison for the estimation performances of the DenseDepth model in S0, *Synthetic*, and S3 configurations. Based on the reported predictions and associated difference maps, we noticed that the maximum depth error for all the configurations is in between $(50, 60)dm$. However, the proposed setup (S3) predicts object edges and overall distances more precisely than the other configurations; we highlight these areas on the difference map with three dashed red circles (the darker is the area the better).

C. Generalization

After showing the efficacy of the proposed solution in the two most common MDE scenarios, we illustrate the generalization performances of DenseDepth in a blind test, i.e., when the model is trained and tested over two different datasets without fine-tuning. In detail, we used the selected model as in previous indoor analysis and tested it on a different real-world dataset (DIML). We report the obtained results in Table IV. It is possible to point out that when the model is trained on S3 configuration, with the same amount of training samples ($Add = 50K$), it outperforms the generalization performances of S0 (NYU). In the case of *Synthetic* (SceneNet), such behavior is evident even when the number of training samples is increased to $150K$. Moreover, using 320×240 pixels as D4D generation resolution, S3 achieves over S0 and *Synthetic* data an RMSE reduction equal to (8.7%, 26.9%) respectively. Furthermore, the increase ($100K$ and $150K$) of D4D-generated samples results in comparable estimation performances with the previously analyzed S3 configuration ($Add = 50K$), as shown in Table IV, which does not justify the time required to produce the additional samples. More in detail, Figure 5 reports a qualitative analysis of DenseDepth model trained on NYU, SceneNet, or D4D-NYU (separately) and tested (without fine-tuning) on the DIML/CVL RGB-D dataset over the compared configurations of Table IV. Based on predicted depth maps and related difference maps for each configuration, it is possible to notice that S3 achieves a lower estimation error than all the other configurations. Precisely, with a maximum distance error of almost $40cm$ with respect to the $100cm$ and $57cm$ achieved by synthetic and baseline (S0) setups. These quantitative and qualitative comparisons demonstrate the superior performances of the proposed D4D-NYU dataset even when testing MDE models on an unseen dataset.

D. Image Resolution

In previous experiments, we showed that the image resolution of D4D-generated samples leads to better depth estimation performances when it is closer to the input image resolution of the trained model. Therefore, we report in Table V a detailed analysis of the effects of D4D-generated resolutions over deep (DenseDepth) and shallow (FastDepth) MDE models. This experiment has been performed on indoor samples (D4D-NYU) with the best parameters' setup, i.e., S3 configuration, NYU $= 50K$, and $Add = 100K$. The previous trend is confirmed since working with a generation resolution significantly different from the model input leads to a noticeable performance decrease, with a maximum difference on

TABLE IV
GENERALIZATION PERFORMANCES OF DENSEDEPTH ON DIML/CVL RGB-D TEST DATASET. THE PROPOSED STRATEGY IS IN BOLD, WHILE THE OPTIMAL CONFIGURATION IS HIGHLIGHTED IN GRAY

Model	Configuration	NYU [K]	*Add* [K]	*Res* [pix]	RMSE↓ [m]	MAE↓ [m]	Abs_{Rel}↓	δ_1↑	δ_2↑	δ_3↑
DenseDepth	S0	50	0	-	0.8723	0.7295	0.1268	0.4466	0.7968	0.9337
	Synthetic	0	50	320 × 240	1.0901	0.8999	0.3738	0.4221	0.7188	0.8800
	Synthetic	0	150	320 × 240	1.0510	0.8721	0.3747	0.4294	0.7248	0.8766
	S1	0	50	320 × 240	0.8443	0.7126	0.2696	0.4876	0.8225	0.9331
	S2	0	50	320 × 240	0.9417	0.7975	0.1432	0.4005	0.7255	0.8943
	S3	0	50	320 × 240	0.7959	0.6660	0.2486	0.5069	0.8381	0.9540
	S3	0	50	256 × 192	0.8142	0.6864	0.2730	0.4998	0.8278	0.9365
	S3	0	100	320 × 240	0.8001	0.6701	0.2522	0.4921	0.8377	0.9537
	S3	0	150	320 × 240	0.7914	0.6623	0.2439	0.5116	0.8421	0.9548

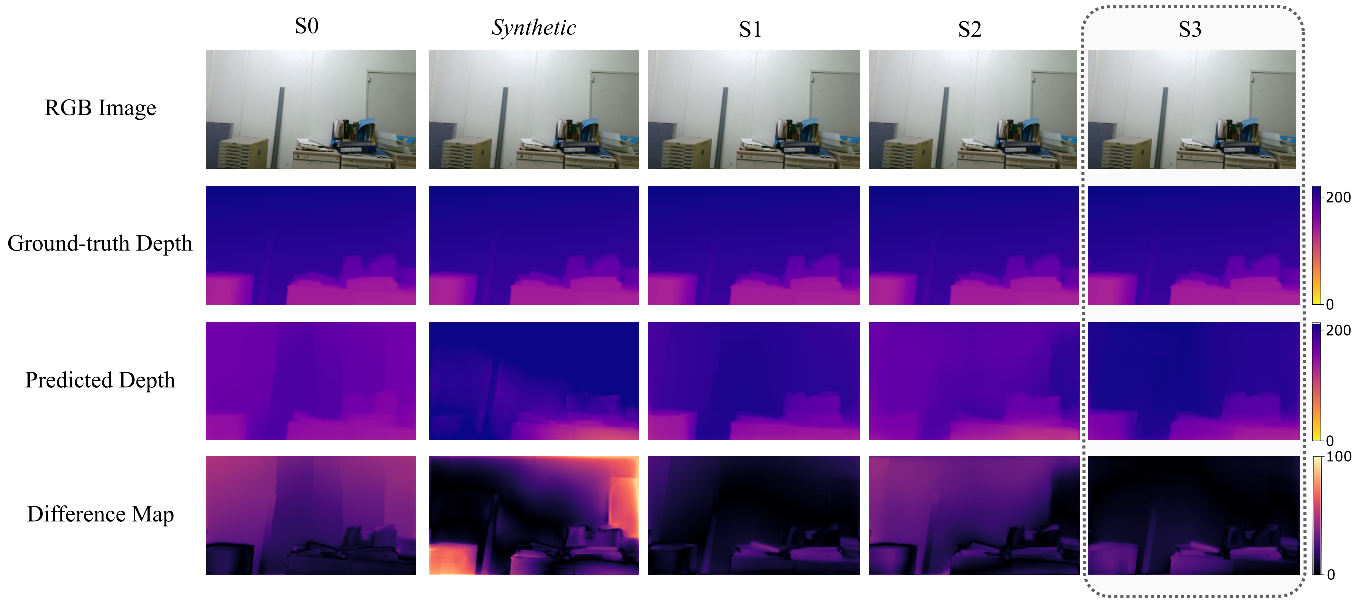


Fig. 5. **Generalization.** Qualitative analysis of the estimated prediction obtained with DenseDepth method. The model has been tested in blind condition (i.e., without fine-tuning) on DIML/CVL RGB-D dataset when trained on a different indoor dataset, i.e., NYU for S0, SceneNet for Synthetic, and D4D-NYU for S1, S2, and S3. The Difference Map is computed as a per pixel-difference between predicted ($\hat{y}$) and expected depth (y), while the reported colorbars are used to emphasize the depth/error range in centimeters (cm).

TABLE V
QUANTITATIVE COMPARISON OF MDE MODELS TRAINED ON SUBSETS OF D4D-NYU GENERATED AT DIFFERENT RESOLUTIONS. THE OPTIMAL VALUES FOR EACH COMPARED MODEL ARE HIGHLIGHTED IN GRAY

Model	*Res* [pix]	RMSE↓ [m]	Abs_{Rel}↓	δ_1↑
DenseDepth	64 × 48	0.5595	0.1595	0.7678
	160 × 120	0.4829	0.1342	0.8258
	256 × 192	0.4578	0.1286	0.8376
	320 × 240	0.4480	0.1236	0.8499
FastDepth	64 × 48	0.5880	0.1758	0.7410
	160 × 120	0.5443	0.1616	0.7816
	256 × 192	0.4980	0.1414	0.8119
	320 × 240	0.5103	0.1492	0.7903

the RMSE equal to (19.9%, 15.3%) and an overall averaged percentage reduction of (17.3%, 12.2%) on DenseDepth and FastDepth. Thanks to this fact, we could keep limited computational requirements needed to generate RGBD samples, avoiding the use of unnecessary high resolutions. Finally, Figure 6 reports a qualitative analysis of FastDepth architecture (other models show similar behavior) when trained on NYU and the proposed D4D-NYU dataset (S3 settings) when its samples are generated at different resolutions ranging from 64 × 48 to 320 × 240 pixels. Based on predicted depth maps and related difference maps for each generation resolution, we qualitatively confirm the fact that the closer the generation resolution of D4D to the input resolution of FastDepth, the better is the estimation for the MDE model. In fact, as noticed, the dataset generated at an image resolution of 256 × 192 pixels, which is closer to FastDepth's input resolution (224 × 224), has a lower error distribution. This can be noticed from the dark region areas that are larger with respect to the other predictions (underlined by the gray dashed rectangle in Figure 6).

Based on the obtained findings, we assume that the just described behavior, due to the different D4D generation resolutions, is caused by the varying feature extraction capabilities of each MDE architecture. More in detail, since each MDE architecture has been developed to work with a specific input resolution, it follows that this parameter defines the quantity of information (pixels) that the model is able to process in order to ensure optimal performance. Consequently, the closer the resolution used to generate samples is to the network's working resolution, the better the performance; in contrast, samples that are larger/smaller than the working resolution of

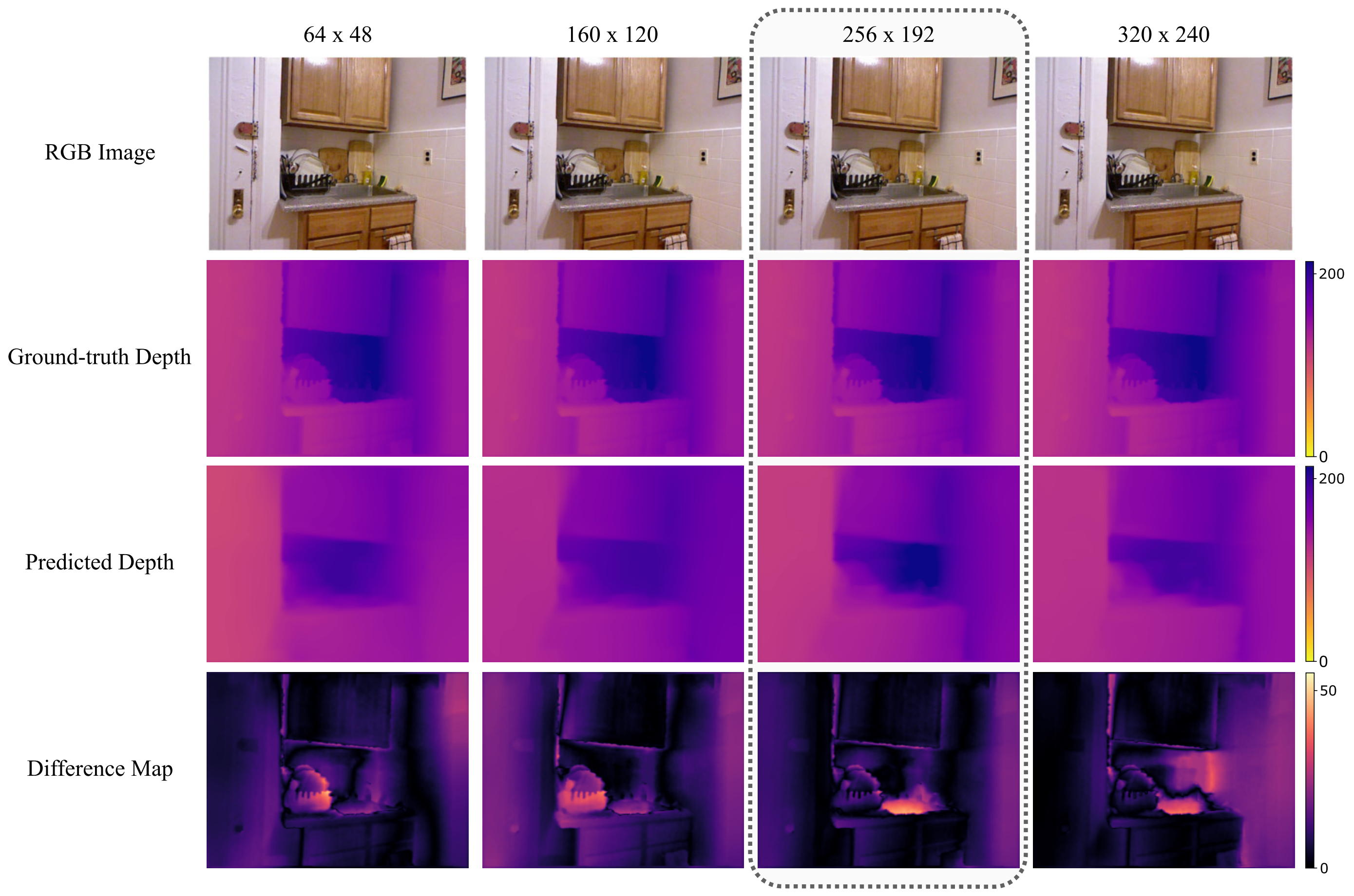


Fig. 6. **Image resolution.** Qualitative analysis of the estimated prediction obtained with FastDepth method. The model has been tested on NYU (indoor) dataset and trained under S3 settings over NYU and D4D-NYU datasets, where its samples have been generated at different resolutions. The Difference Map is computed as a per pixel-difference between predicted ($\hat{y}$) and expected depth (y), while the reported colorbars are used to emphasize the depth/error range in centimeters (cm).

the network will be compressed/expanded, thus resulting in information loss or inaccurate data.

E. Amount of Generated Samples

Once the optimal generation resolution has been analyzed, in this ablation study, we investigate how different amounts of generated samples impact the performance of MDE models. More in detail, we study the behavior of FastDepth architecture when the number of D4D-generated (training) samples varies; precisely, we examine a data range between 0 and 250K RGBD samples generated by D4D-NYU in the optimal S3 configuration at the resolution of 256 × 192 pixels. We report the obtained results in the two compared setups, i.e., with and without the original training dataset (NYU), in Table VI.

Based on the obtained results, it can be noticed that the higher estimation performances are obtained with the addition of $200K$ generated training data ($Add = 200K$). More in detail, we obtain an average RMSE reduction of 7.9% and 4.6% when the best performing model is compared with respect to the other configurations ($Add = i * 50K$ with $i \in [0, 3]$) in the two analyzed scenarios, i.e., when the original training dataset is used (NYU=$50K$) and when it is not considered (NYU=$0K$). Based on the two compared configurations, we can note that when comparing the best-performing setup with the best one ($Add = 100K$) reported in Table I and Table II (also reported in Table VI), the RMSE reduction is limited to 1.2% and 1.9%, respectively. Moreover, when compared to the $Add = 250K$ setups, the $Add = 200K$ ones results in an RMSE reduction of 3.1% (NYU= 50) and 0.9% (NYU= 0). Consequently, we can assume that the $Add = 200K$ setup is FastDepth's best configuration with respect to

TABLE VI
QUANTITATIVE COMPARISON OF FASTDEPTH MODEL TRAINED ON DIFFERENT AMOUNT OF D4D-NYU GENERATED SAMPLES (S3 CONFIGURATION) AT THE RESOLUTION OF 256 × 192 PIXELS. THE BEST VALUES FOR EACH COMPARED SETUP ARE HIGHLIGHTED IN GRAY

Model	NYU [K]	Add [K]	RMSE↓ [m]	Abs_{Rel}↓	δ_1↑
FastDepth	50	0	0.5714	0.1751	0.7535
	50	50	0.5585	0.1643	0.7666
	50	100	0.4980	0.1414	0.8119
	50	150	0.4962	0.1411	0.8121
	50	200	0.4919	0.1406	0.8127
	50	250	0.5076	0.1517	0.7981
	0	0	-	-	-
	0	50	0.5996	0.1746	0.7500
	0	100	0.5502	0.1730	0.7649
	0	150	0.5449	0.1619	0.7665
	0	200	0.5397	0.1607	0.7678
	0	250	0.5444	0.1616	0.7629

TABLE VII

QUANTITATIVE COMPARISON ACROSS EFFICIENT HVIT CONFIGURATIONS. THE ✓ AND ✗ ARE USED TO INDICATE WHEN D4D-GENERATED DATA ARE EMPLOYED

Model	D4D	NYU		KITTI	
		RMSE↓ [m]	δ_1↑	RMSE↓ [m]	δ_1↑
METER	✗	0.5112	0.8138	4.8398	0.8153
	✓	0.4574	0.8357	4.7288	0.8155
MetaM	✗	0.5058	0.8111	4.9493	0.8014
	✓	0.4556	0.8373	4.6714	0.8166
PyraM	✗	0.5196	0.8062	5.2308	0.7652
	✓	0.4944	0.8139	4.9652	0.7737

TABLE VIII

EMBEDDING VECTORS' DISTANCES COMPUTED BETWEEN EACH CONFIGURATIONS (Si WITH $i = [0, 3]$) AND NYU TEST SET. EACH SUBSET COUNTS $50K$ TRAINING SAMPLES

Model	Configuration	RGB		Depth	
		ED	HD	ED	HD
ResNet18	S0	0.1158	0.0771	0.1243	0.0826
	S1	0.2626	0.1720	0.2739	0.1815
	S2	0.2384	0.1597	0.2568	0.1719
	S3	0.3636	0.2403	0.3608	0.2361
EffB4	S0	0.7714	0.4442	1.2708	0.7262
	S1	1.7724	1.0307	2.0593	1.1886
	S2	1.4572	0.8330	1.8667	1.0584
	S3	1.9611	1.1356	2.1222	1.2283

the amount of generated samples; however, when considering the time required to generate a larger number of RGBD data and the limited percentage improvement, we can conclude that $100K$ samples are a good trade-off, ensuring good estimation performance on the NYU dataset while limiting the overall computational time.

F. Additional Experiments on Efficient ViT

Once the main parameters of D4D have been analyzed, we present some additional results on efficient ViT architectures to emphasize the proposed solution's versatility. We outline the following analysis motivated by the practical applicability of MDE models on embedded/mobile devices, which are usually characterized by limited computational powers. In order to infer on such devices, factors like reduced network computational capabilities, number of trainable parameters, or model depth typically result in a reduction of the estimation performances. Consequently, this analysis investigates the percentage boost that D4D is able to achieve when combined with efficient architectures. In particular, we analyze the performance improvement of the proposed pipeline across two efficient METER configurations, namely, Meta-METER (MetaM) and Pyra-METER (PyraM) proposed in [24]. The latter architectures were developed by exploiting the efficiency capabilities of MetaFormer [51] and Pyramid Vision Transformer [52], which aims to reduce/linearize the computational cost of self-attention.

We compare the reported architectures using the same METER's optimal[4] hyperparameters identified in TableI and TableIII respectively for the NYU and KITTI datasets. Based on the obtained results (Table VII), we can note an average percentage RMSE reduction of 6.4% and δ_1 increment of 2.0% when the D4D pipeline is used instead of a standard training pipeline. As a result, it can be noticed that in this scenario, where model learning capabilities are limited with respect to deeper architectures due to computational constraints introduced by embedded devices, the proposed pipeline still provides a good percentage boost for the model's estimation performances.

G. Analysis on Feature Space

We conclude the result section by performing similarity measurements among different configurations on the feature space in order to provide an in-depth explanation of the obtained results. More in detail, we analyze the learning capabilities of D4D configurations (S1, S2, and S3) with respect to the NYU training setup (S0). We extract the visual features characterizing each dataset with two pretrained neural networks (initialized on ImageNet): the ResNet18 [53] and the EfficienNetB4 [54]. This procedure is performed by removing the last classification layer (fully connected) from each respective model. Therefore, a final embedding vector of each dataset is obtained as the average features vector extracted from $50K$ input samples. Subsequently, we compute the distance between the mean of the embedding vectors using two evaluation metrics: the Euclidean distance (ED) and the Hillinger distance (HD) [55]. Table VIII shows such differences computed between the embedding vectors related to each configuration and the NYU test dataset.

Based on reported values, S3 has higher values both for ED and HD rather than other configurations. Moreover, observing the metrics reported in Table I, Table II, and Table IV we noticed that the increasing distances correspond to greater estimation performances. Therefore, without loss of generality, we derive that the higher the distance of the features from the test dataset, the better the performance of the MDE model. We hypothesize that a greater distance corresponds to stronger generalization capabilities due to a more efficient covering of heterogeneous samples.

VI. CONCLUSION

This paper presents a novel training pipeline composed of D4D, a custom 4-channels DDPM to produce realistic RGBD samples used to improve the estimation performances of deep and shallow MDE models. The proposed methodology demonstrates superior performances with respect to synthetically generated datasets in indoor and outdoor scenarios, with an average RMSE reduction equal to 8.2% and 8.1%. Moreover, our solution achieves an RMSE reduction equal to 11.9% and 6.1% with respect to the baseline indoor NYU Depth v2 and outdoor KITTI datasets. We hope that our method, together with the generated datasets (D4D-NYU and D4D-KITTI), will encourage the combined use of DDPM with deep learning architectures to address the lack of labeled training data in a variety of computer vision applications. A key element of the proposed strategy is the use of real-world images to generate novel augmented samples, thus improving the estimation and

[4]For the NYU dataset: configuration S3, *Add*= $100K$, *Res.* 256×192. For the KITTI dataset: configuration S3, *Add*= $50K$, *Res.* 256×192.

generalization of MDE model capabilities for deploying in real-case scenarios.

Our technique is applied to tackle the MDE task, where the generated depth map is crucial to obtain accurate performance. However, the generated RGBD samples could also contribute to other applications, such as monocular SLAM or other computer vision tasks where a fourth (depth) channel can be used to improve standard RGB approaches, as in semantic segmentation [56], human action recognition [57] and object detection [58]. Consequently, in the future, we will further evaluate our method and employ generated samples in different RGBD tasks, study their performances on different architectures, and propose new diffusion architectures specifically tailored for depth data.

References

[1] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3D data: A survey," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–38, Mar. 2018.

[2] M. Usama et al., "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, vol. 7, pp. 65579–65615, 2019.

[3] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 4037–4058, Nov. 2021.

[4] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, Mar. 2022.

[5] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

[6] A. Juliani et al., "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*.

[7] Epic Games. *Unreal Engine*. Accessed: May 20, 2023. [Online]. Available: https://www.unrealengine.com

[8] T. Alkhalifah, H. Wang, and O. Ovcharenko, "MLReal: Bridging the gap between training on synthetic data and real data applications in machine learning," *Artificial Intell. Geosci.*, vol. 3, pp. 101–114, Dec. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666544122000260 x

[9] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.

[10] J. Tremblay et al., "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2018, pp. 969–977.

[11] A. Lopes, R. Souza, and H. Pedrini, "A survey on RGB-D datasets," *Comput. Vis. Image Understand.*, vol. 222, Sep. 2022, Art. no. 103489.

[12] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Computer Vision—ECCV 2012* (Lecture Notes in Computer Science), vol. 7576, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-642-33715-4_54.

[13] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255.

[15] T. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. 13th Eur. Conf. Comput. Vis.* Zürich, Switzerland: Springer, Sep. 2014, pp. 740–755.

[16] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6840–6851.

[17] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2256–2265.

[18] L. Yang et al., "Diffusion models: A comprehensive survey of methods and applications," 2022, *arXiv:2209.00796*.

[19] Y. Ming, X. Meng, C. Fan, and H. Yu, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, May 2021.

[20] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," 2018, *arXiv:1812.11941*.

[21] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "FastDepth: Fast monocular depth estimation on embedded systems," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6101–6108.

[22] L. Papa, E. Alati, P. Russo, and I. Amerini, "SPEED: Separable pyramidal pooling encoder–decoder for real-time monocular depth estimation on low-resource settings," *IEEE Access*, vol. 10, pp. 44881–44890, 2022.

[23] L. Papa, P. Russo, and I. Amerini, "METER: A mobile vision transformer architecture for monocular depth estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 10, pp. 5882–5893, Oct. 2023, doi: 10.1109/TCSVT.2023.3260310.

[24] C. Schiavella, L. Cirillo, L. Papa, P. Russo, and I. Amerini, "Optimize vision transformer architecture via efficient attention modules: A study on the monocular depth estimation task," in *Image Analysis and Processing—ICIAP 2023 Workshops*. Springer, Jan. 2024, pp. 383–394.

[25] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "SceneNet RGB-D: Can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation?" in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2697–2706.

[26] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3234–3243.

[27] J. Cho, D. Min, Y. Kim, and K. Sohn, "DIML/CVL RGB-D dataset: 2M RGB-D images of natural indoor and outdoor scenes," 2021, *arXiv:2110.11590*.

[28] M. Zhu, P. Pan, W. Chen, and Y. Yang, "DM-GAN: Dynamic memory generative adversarial networks for text-to-image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5802–5810.

[29] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 12104–12114.

[30] Q. Cai, M. Abdel-Aty, J. Yuan, J. Lee, and Y. Wu, "Real-time crash prediction on expressways using deep generative models," *Transp. Res. C, Emerg. Technol.*, vol. 117, Aug. 2020, Art. no. 102697. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X20306124

[31] L. Zhao, Z. Zhang, T. Chen, D. Metaxas, and H. Zhang, "Improved transformer for high-resolution GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 18367–18380.

[32] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2223–2232.

[33] P. Russo, F. M. Carlucci, T. Tommasi, and B. Caputo, "From source to target and back: Symmetric bi-directional adaptive GAN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8099–8108.

[34] H. Tang, H. Liu, D. Xu, P. H. S. Torr, and N. Sebe, "AttentionGAN: Unpaired image-to-image translation using attention-guided generative adversarial networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 1972–1987, Apr. 2023.

[35] D. Torbunov et al., "UVCGAN: UNet vision transformer cycle-consistent GAN for unpaired image-to-image translation," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2023, pp. 702–712.

[36] D. Du, L. Wang, H. Wang, K. Zhao, and G. Wu, "Translate-to-recognize networks for RGB-D scene recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11828–11837.

[37] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Red Hook, NY, USA: Curran Associates, 2021, pp. 8780–8794. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf

[38] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 10684–10695.

[39] W. Peebles and S. Xie, "Scalable diffusion models with transformers," 2022, *arXiv:2212.09748*.

[40] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. 18th Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, vol. 9351, Munich, Germany. Cham, Switzerland: Springer, Oct. 2015, pp. 234–241.

[41] NVIDIA. *NVIDIA Isaac Sim*. Accessed: May 20, 2023. [Online]. Available: https://developer.nvidia.com/isaac-sim

[42] Y. Zou, Z. Luo, and J.-B. Huang, "Df-Net: Unsupervised joint learning of depth and flow using cross-task consistency," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 36–53.

[43] W. Chen, S. Qian, and J. Deng, "Learning single-image depth from videos using quality assessment networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5597–5606.

[44] K. Xian, J. Zhang, O. Wang, L. Mai, Z. Lin, and Z. Cao, "Structure-guided ranking loss for single image depth prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 611–620.

[45] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proc. 38th Int. Conf. Mach. Learn.*, Jul. 2021, pp. 8162–8171.

[46] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, "Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1043–1051.

[47] A. Paszke, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[48] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014.

[49] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, "Micro-batch training with batch-channel normalization and weight standardization," 2019, *arXiv:1903.10520*.

[50] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[51] W. Yu et al., "MetaFormer is actually what you need for vision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10819–10829.

[52] W. Wang et al., "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 568–578.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[54] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

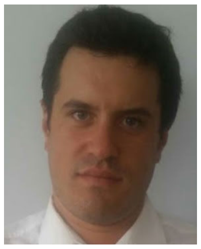
[55] D. Pollard, *A User's Guide to Measure Theoretic Probability* (Cambridge Series in Statistical and Probabilistic Mathematics). Cambridge, U.K.: Cambridge Univ. Press, 2001.

[56] Y. Cheng, R. Cai, Z. Li, X. Zhao, and K. Huang, "Locality-sensitive deconvolution networks with gated fusion for RGB-D indoor semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3029–3037.

[57] Y. Yang, G. Liu, and X. Gao, "Motion guided attention learning for self-supervised 3D human action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 12, pp. 8623–8634, Dec. 2022.

[58] X. Jin, K. Yi, and J. Xu, "MoADNet: Mobile asymmetric dual-stream networks for real-time and lightweight RGB-D salient object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 11, pp. 7632–7645, Nov. 2022.

Lorenzo Papa received the B.S. degree in computer and automation engineering and the M.S. degree in artificial intelligence and robotics from the Sapienza University of Rome, Italy, in 2019 and 2021, respectively. He is currently pursuing the Ph.D. degree in computer science engineering. He collaborates with ALCORLab, Department of Computer, Control, and Management Engineering, Sapienza University of Rome. He is a Visiting Researcher with the School of Electrical and Information Engineering, Faculty of Engineering and Information Technology, The University of Sydney, Australia. His research interests include deep learning, computer vision, and cyber security.



Paolo Russo received the B.S. degree in telecommunication engineering from UniversitÃ degli studi di Cassino, Italy, in 2008, and the M.S. degree in artificial intelligence and robotics and the Ph.D. degree in computer science from the Sapienza University of Rome, Italy, in 2016 and 2020, respectively. From 2018 to 2019, he was a Researcher with Italian Institute of Technology (IIT), Tourin, Italy. He is currently an Assistant Researcher with ALCORLab, DIAG Department, Sapienza University of Rome. His main research interests include deep learning, computer vision, generative adversarial networks, and reinforcement learning.

Irene Amerini (Member, IEEE) received Ph.D. degree in computer engineering, multimedia, and telecommunication from the University of Florence, Italy, in 2010. She is currently an Associate Professor with the Department of Computer, Control, and Management Engineering Antonio Ruberti, Sapienza University of Rome, Italy. Her main research interests include digital image processing, computer vision, and multimedia forensics. She is a member of the IEEE Information Forensics and Security Technical Committee, the EURASIP TAC Biometrics, Data Forensics, and Security, and the IAPR TC6-Computational Forensics Committee.

Open Access funding provided by 'Università degli Studi di Roma "La Sapienza" 2' within the CRUI CARE Agreement