arXiv:2403.08368v1 [cs.CV] 13 Mar 2024

METER: a mobile vision transformer architecture for monocular depth estimation

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

***Abstract*—Depth estimation is a fundamental knowledge for autonomous systems that need to assess their own state and perceive the surrounding environment. Deep learning algorithms for depth estimation have gained significant interest in recent years, owing to the potential benefits of this methodology in overcoming the limitations of active depth sensing systems. Moreover, due to the low cost and size of monocular cameras, researchers have focused their attention on monocular depth estimation (MDE), which consists in estimating a dense depth map from a single RGB video frame. State of the art MDE models typically rely on vision transformers (ViT) architectures that are highly deep and complex, making them unsuitable for fast inference on devices with hardware constraints.**

Purposely, in this paper, we address the problem of exploiting ViT in MDE on embedded devices. Those systems are usually characterized by limited memory capabilities and low-power CPU/GPU. We propose METER, a novel lightweight vision transformer architecture capable of achieving state of the art estimations and low latency inference performances on the considered embedded hardwares: NVIDIA Jetson TX1 and NVIDIA Jetson Nano. We provide a solution consisting of three alternative configurations of METER, a novel loss function to balance pixel estimation and reconstruction of image details, and a new data augmentation strategy to improve the overall final predictions. The proposed method outperforms previous lightweight works over the two benchmark datasets: the indoor NYU Depth v2 and the outdoor KITTI.

***Index Terms*—Deep learning, embedded device, monocular depth estimation, vision transformer**

I. Introduction

Acquiring accurate depth information from a scene is a fundamental and important challenge in computer vision, as it provides essential knowledge in a variety of vision applications, such as augmented reality, salient object detection, visual SLAM, video understanding, and robotics [1]–[3]. Depth data is usually captured with active depth sensors as LiDARs, depth cameras, and other specialised sensors capable of perceiving such information by perturbing the surrounding environment, e.g. through time-of-flight or structured light technologies. These sensors have several disadvantages, including unfilled depth maps and restricted depth ranges, as well as being difficult to integrate into low-power embedded devices. In addition, we also need to consider the power consumption in the case of hardwares with low-resource constraints.

On the contrary, passive depth sensing systems based on deep learning (DL) could potentially overcome all the active depth sensor limitations. Moreover, in some settings such as indoor or hostile environments, where the use of small robots and drones could introduce additional constraints, the presence of a single RGB camera offers an effective and low-cost alternative to such traditional setups. The monocular depth estimation (MDE) task consists in the prediction of a dense depth map from a video frame with the use of DL algorithms, where the estimation is computed for each pixel.

Recent MDE models aim at enabling depth perception using single RGB images on deep vision transformer (ViT) architectures [4]–[6], which are generally unsuitable for fast inference on low-power hardwares. Instead, well-established convolutional neural networks (CNN) architectures [7], [8] have been successfully exploited on embedded devices with the goal of achieving accurate and low latency inferences. However, ViT architectures demonstrate the advantage of a global processing by obtaining significant performance improvements over fully-CNNs. In order to balance computational complexity and hardware constraints, we propose to integrate the two architectures by fusing transformers blocks and convolutional operations, as successfully exploited in classification and object detection [9], [10] tasks.

This paper presents METER, a *MobilE vision TransformER* architecture for MDE that achieves state of the art results with respect to previous lightweight models over two benchmark datasets, i.e. NYU Depth v2 [11] and KITTI [12]. METER inference speed will be evaluated on two embedded hardwares, the 4GB NVIDIA Jetson TX1 and the 4GB NVIDIA Jetson Nano. To improve the overall estimation performances, we focus on three fundamental components: a specific loss function, a novel data augmentation policy and a custom transformer architecture. The loss function is composed of four independent terms (quantitative and similarity measurements) to balance the architecture reconstruction capabilities while highlighting the image high-frequency details. Moreover, the data augmentation strategy employs a simultaneous random shift over both the input image and the dense ground truth depth map to increase model resilience to tiny changes of illumination and depth values.

The proposed network exploits a hybrid encoder-decoder structure characterized by a ViT encoder, which was inspired by [9] due to its fast inference performances. We focus on the transformer structure in order to identify and to improve the blocks with the highest computational cost while optimizing the model to extract robust features. In addition, we designed a novel lightweight CNN decoder to limit the amount of

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

Manuscript received Month Day, 2022; revised Month Day, 2022.

Copyright © 20xx IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.

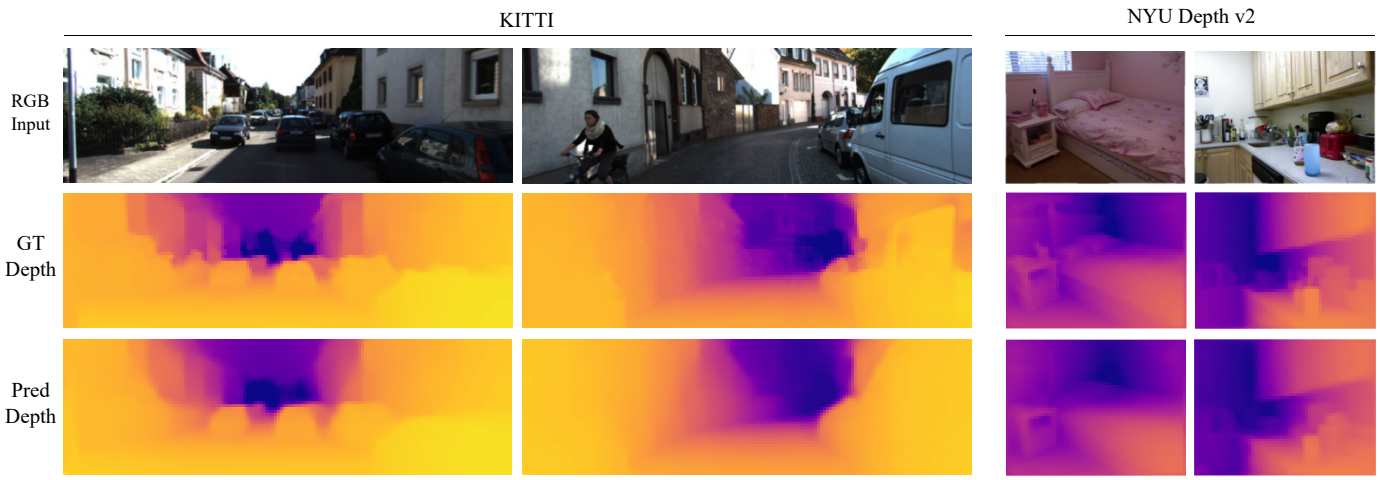

Fig. 1: METER depth map predictions (third-row) over the KITTI and NYU Depth v2 datasets. GT depth maps are resized to match METER output resolution. The depth maps are converted in RGB format with a perceptually uniform colormap (Plasma-reversed) extracted from the ground truth (second-row), for a better view.

operations while improving the reconstruction process. Furthermore, we propose three different METER configurations; for each variant, we reduce the number of trainable parameters at the expense of a slight increase of the final estimation error. Figure 1 shows several METER depth estimations for both indoor and outdoor environments.

Moreover, to the best of our knowledge, METER is the first model for the MDE task that integrates the advantage of ViT architectures in such lightweight DL structures under low-resource hardware constraints. The main contributions of the paper are summarized as follows:

- We propose a novel lightweight ViT architecture for monocular depth estimation able to infer at high frequency on low-resource (4GB) embedded devices.
- We introduce a novel data augmentation method and loss function to boost the model estimation performances.
- We show the effectiveness and robustness of METER with respect to related state of the art MDE methods over two benchmark datasets, i.e. NYU Depth v2 [11] and KITTI [12].
- We validate the models through quantitative and qualitative experiments, data augmentation strategies and a loss function components, highlighting their effectiveness.

This paper is organized as follows: Section II reviews some previous works related to the topics of interest. Section III describes the proposed method and the overall architecture in detail. Experiments and hyper-parameters are discussed in Section IV, while Section V reports the results and a quantitative analysis of METER with respect to other significant works. Some final considerations and future applications are provided in Section VI.

II. Related Works

In this section, we report state of the art related works on monocular depth estimation, grouped as follows: fully CNN-based methods are covered in Section II-A, ViT-based approaches in Section II-B and lightweight (CNN) MDE methods in Section II-C.

A. CNN-based MDE methods

Fully convolutional neural networks based on encoder-decoder structures are commonly used for dense prediction tasks such as depth estimation and semantic segmentation. In the seminal work of Eigen et al. [13] it is presented a CNN model to handle the MDE task by employing two stacked deep networks to extract both global and local informations. Cao et al. present [14] and [15] two works based on deep residual networks to solve the MDE defined as a classification task, respectively, over absolute and relative depth maps. Alhashim et al. [16] propose DenseDepth, a network which exploits transfer learning to produce high-resolution depth maps. The architecture is composed of a standard encoder-decoder with a pre-trained DenseNet-169 [17] as backbone and a specifically designed decoder. Gur et al. [18] present a variant of the DeepLabV3+ [19] model where the encoder is composed of a ResNet [20] and of an *atrous* spatial pyramidal pooling while introducing a Point Spread Function convolutional layer to learn depth informations from defocus cues. Recently, Song et al. [21] propose LapDepth, a Laplacian pyramid-based architecture composed of a pretrained ResNet-101 encoder and a Laplacian pyramid decoder that combined the reconstructed coarse and fine scales to predict the final depth map.

However, those methods, which often rely on deep pre-trained encoders and high-resolution images as input, are unsuitable for inferring on low-resource hardwares. In contrast, we propose a lightweight architecture that takes advantage of transformers blocks to balance global feature extraction capabilities and the overall computational complexity of convolutional operations.

B. ViT-based MDE methods

Vision Transformers [22] gain popularity for their accuracy capabilities thanks to the attention mechanism [23] that simultaneously extract information from the input pixels and their inter-relation, outperforming the translation-invariant property of convolution. In dense prediction tasks, ViT architectures

share the same encoder-decoder structure that has significantly contributed to face many CNN vision-related problems. Bhat et al. [5] have been the first to handle the MDE task with ViT architectures by proposing Adabins: it uses a minimized version of a vision transformer structure to adaptively calculate bins width. Ranftl et al. [4] investigate the application of ViT proposing DPT, a model composed of a transformer-CNN encoder and a fully-convolutional decoder. The authors show that ViT encoders provide finer-grade predictions with respect to standard CNNs, especially when instantiated with a large amount of training data. Yun et al. [24] improves $360°$ monocular depth estimation methods with a joint supervised and self-supervised learning strategies taking advantage of non-local DPT. Recently, Li et al. [25] design MonoIndoor++, a framework that takes in account the main challenges of indoor scenarios. Kim et al. [26] propose GLPDepth, a global-local transformer network to extract meaningful features at different scales and a Selective Feature Fusion CNN block for the decoder. The authors also integrate a revisited version of CutDepth data augmentation method [27] which is able to improve the training process on the NYU Depth v2 dataset without needing additional data. Li et al. propose DepthFormer [6] and BinsFormer [28], where the first one is composed of a fully-transformer encoder and a convolutional decoder interleaved by an interaction module to enhance transformer encoded and CNN decoded features. Differently, in BinsFormer the idea of the authors is to use a multi-scale transformer decoder to generate adaptive bins and to recover spatial geometry information from the encoded features.

Instead of following the recent trend of high-capacity models, we propose a novel lightweight ViT architecture that is able to achieve accurate, low latency depth estimations on embedded devices.

C. Lightweight MDE methods

The models reported so far are not suitable for embedded devices due to their size and complexity. For this reason, developing lightweight architectures could be a solution to perform inference on constrained hardwares as shown in [29], [30]. To provide a clearer overview of those approaches we also provide the frames per second (fps) published in the original papers that focus on inference frequency, remarking that they are not comparable due to the different tested hardwares. Poggi et al. [31] propose PyD-Net, a pyramidal network to infer on CPU devices. The authors use the pyramidal structure to extract features from the input image at different levels, which are afterwards upsampled and merged to refine the output estimation. Such model achieves less than 1 fps on an ARM CPU and almost 8 fps on an Intel i7 CPU. Spek et al. [32] present CReaM, a fully convolutional architecture obtained through a knowledge-transfer learning procedure. The model is able to achieve real-time frequency performances (30 fps) on the 8GB NVIDIA Jetson TX2 device. Wofk et al. [8] develop FastDepth, an encoder-decoder architecture characterized by a MobileNet [33] pre-trained network as backbone, and a custom decoder. Furthermore, the authors show that pruning the trained model guarantees a boost of inference frequency at the expense of a small increment of the final estimation error. FastDepth achieves 178 fps on the 8GB NVIDIA Jetson TX2 device. Recently, Yucel et al. [34] propose a small network composed by the MobileNet v2 [33] as encoder and FBNet x112 [35] as decoder, trained on an altered knowledge distillation process; the model achieves 37 fps on smartphone GPU. Papa et al. [7] design SPEED, a separable pyramidal pooling architecture characterized by an improved version of the MobileNet v1 [36] as an encoder and a dedicated decoder. This architecture exploits the use of depthwise separable convolutions, achieving real-time frequency performances on the embedded 4GB NVIDIA Jetson TX1 and 6 fps on the Google Dev Board Edge TPU.

As previously mentioned, all those lightweight MDE works are designed over fully-convolutional architectures. In contrast to previous methodologies, METER exploits a lightweight transformer module in three different configurations, achieving state of the art results over the standard evaluation metrics.

III. Proposed Method

This section outlines the design of METER, the proposed lightweight monocular depth estimator. In particular, in Section III-A, we provide a detailed architecture analysis for both encoder and decoder modules, in Section III-B we describe the proposed loss function and in Section III-C the employed augmentation policy.

A. METER architecture

The vision transformer architecture has demonstrated outstanding performances in a variety of computer vision tasks, usually relying on deep and heavy structures. On the other hand, to reduce the computational cost of such models, lightweight CNN usually relies on convolutional operations with small kernels (i.e. 3x3, 1x1) or on particular techniques such as depthwise separable convolution [37]. Based on those statements, we design an hybrid lightweight ViT characterized by convolutions with small kernels and as few transformers blocks as possible reducing the computational impact in the overall structure. Motivated by this, in the following, we present METER: a *MobilE vision TrasformER* architecture characterized by a lightweight encoder-decoder model designed to infer on embedded devices. METER encoder redesign computational demanding operations of [9] to improve the inference performances while maintaining the feature extraction capabilities. The high-level features extracted from the encoder are then fed into the decoder through the skip-connections to recover the image details. The proposed fully convolutional decoder has been structured to upsample the compact set of encoder high-level features while enhancing the reconstruction of the image details to obtain the desired output depth map (i.e. a per-pixel distance map). A graphical overview of the architecture is reported in Figure 2 while the number of channels employed in the different METER configurations, METER S, METER XS, and METER XXS are reported in Table I. The number of trainable parameters of the three proposed networks consist of $3.29M$, $1.45M$, and $0.71M$, respectively.

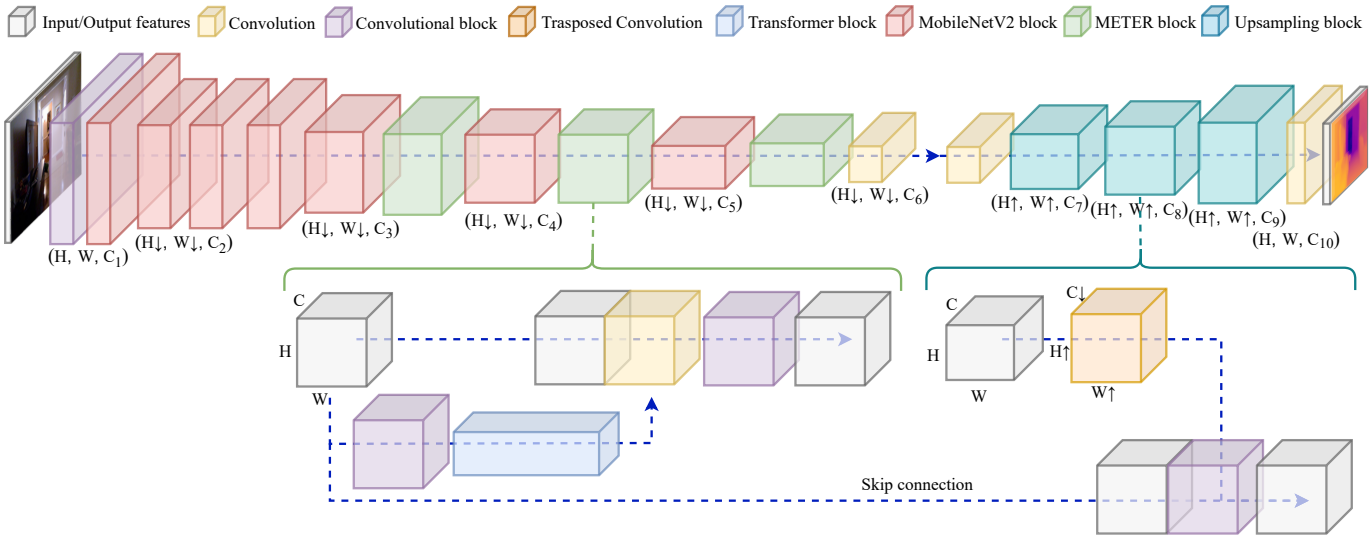

Fig. 2: Overview of METER encoder-decoder network structure. The processing flow, i.e. the sequence of operations and the skip-connection, is represented with a blue dashed arrow. The (H, W, C) format refers to the input-output spatial dimensions, while the ↑ and ↓ refers to the feature resolution upsampling and downsampling.

TABLE I: Number of channels (C_i) used in METER configurations.

Channels	METER S	METER XS	METER XXS
C_1	16	16	16
C_2	32	32	16
C_3	64	48	24
C_4	128	80	64
C_5	160	96	80
C_6	320	192	160
C_7	128	128	64
C_8	64	64	32
C_9	32	32	16
C_{10}	16	16	8

METER encoder exploits a modified version of MobileViT network due to its light structure demonstrated in [9]. As can be noticed in Figure 2, METER presents a hybrid network composed of convolutional MobileNetV2 blocks (red) and transformers blocks (green). The MobileViT blocks with the highest computational cost, i.e. the ones composed of cascaded transformers and convolution operations, have been identified and replaced with new modules (METER blocks). Such modules are able to guarantee low latency inference while tuning the entire structure to minimize the final estimation error. Along the lines of [9], we propose three variants of the same encoder architecture with decreasing complexity and computational cost namely S, XS, and XXS.

The proposed METER block (green in Figure 2) is composed by three feature extraction operations, two Convolutional blocks composed by a 3×3 convolution and a point-wise one (purple) and a second 1×1 convolution (yellow) interleaved by a single transformer block (blue). Such module computes an unfold operation to apply the transformer attention on the flattened input patches while reconstructing output feature map with an opposite folding operation, as described in [9]. Moreover, in order to apply an attention mechanism to the encoded features, the input of METER block (gray) has been concatenated with the output of the transformer and fed to the previous 1×1 convolution layer. When compared with MobileViT architecture, characterized by four convolutions operations and a number of cascaded transformers blocks, the proposed design allows to reduce the computational cost of the overall model while producing an accurate estimation of the depth (as will be shown in Section V-B).

Finally, we halved the number of output encoder features (channel C_6) and we replaced the MobileViT SiLU non linearity function with the ReLU. Despite the fact that SiLU activation function is differentiable at every point[1], it does not ensure better performance, likely due to the depth-data distribution.

METER decoder is designed with a fully convolutional structure to enhance the estimation accuracy and the reconstruction capabilities while keeping a limited number of operations. As can be seen in Figure 2, the decoder consists of a sequence of three cascaded upsampling blocks (light blue) and two convolutional layers (yellow) located at the beginning and at the end of the model. Each upsampling block is composed by a sequence of upsampling, skip-connection and feature extraction operations. The upsampling operation is performed by a transposed convolutional layer (orange) which doubles the spatial resolution of the input. Then, a Convolutional block (purple) is used for feature extraction; the skip-connection (dashed blue arrow) linking METER encoder-decoder modules allows to recover image details from the encoded feature maps.

B. The balanced loss function

The standard monocular depth estimation formulation consider as loss function the per-pixel difference between the ith ground truth pixel y_i and the predicted one $\hat{y}_i$. However,

[1]Unlike the SiLU, the ReLU activation function is non-differentiable at zero.

as reported in literature [16], [38], [39] several modifications have been proposed to improve the convergence speed and the overall depth estimation performances. In particular, the addition of different loss components focuses on refinement of fine details in the scenes, like object contours.

Derived from [38], [39], we propose a *balanced loss function* (BLF) to weight the reconstruction loss through the $L_{depth}(y_i, \hat{y}_i)$ and $L_{SSIM}(y_i, \hat{y}_i)$ components with the high-frequency features taken into account by the $L_{grad}(y_i, \hat{y}_i)$ and the $L_{norm}(y_i, \hat{y}_i)$ losses. The BLF $L(y_i, \hat{y}_i)$ mathematical formulation is reported in Equation 1, where $\lambda_1, \lambda_2, \lambda_3$ are used as scaling factors.

$$L(y_i, \hat{y}_i) = L_{depth} + \lambda_1 L_{grad} + \lambda_2 L_{norm} + \lambda_3 L_{SSIM} \quad (1)$$

In detail, the loss $L_{depth}(y_i, \hat{y}_i)$ in Equation 2 is the point-wise L1 loss computed as the per-pixel absolute difference between the ground truth y_i and the predicted image $\hat{y}_i$.

$$L_{depth}(y_i, \hat{y}_i) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \quad (2)$$

The $L_{grad}(y_i, \hat{y}_i)$ and the $L_{norm}(y_i, \hat{y}_i)$ losses reported respectively in Equation 3 and Equation 4 are designed to penalize the estimation errors around the edges and on small depth details. The $L_{grad}(y_i, \hat{y}_i)$ loss computes the Sobel gradient function to extract the edges and objects boundaries.

$$L_{grad}(y_i, \hat{y}_i) = \frac{1}{n} \sum_{i=1}^{n} (\nabla_x(|y_i - \hat{y}_i|) + \nabla_y(|y_i - \hat{y}_i|)) \quad (3)$$

We report with ∇ the spatial derivative of the absolute estimation error with respect to the x and y axes.

The $L_{norm}(y_i, \hat{y}_i)$ loss, reported in Equation 4, calculates the cosine similarity [40] between the ground truth and the prediction.

$$L_{norm}(y_i, \hat{y}_i) = \frac{1}{n} \sum_{i=1}^{n} \left(1 - \frac{\langle n_{\hat{y}_i}, n_{y_i} \rangle}{\sqrt{\langle n_{\hat{y}_i}, n_{\hat{y}_i} \rangle} \sqrt{\langle n_{y_i}, n_{y_i} \rangle}} \right) \quad (4)$$

We identify with $\langle n_{y_i}, n_{\hat{y}_i} \rangle$ the inner product of the surface normal vectors n_{y_i} and $n_{\hat{y}_i}$ computed for each depth map i.e. $n_z = [-\nabla_x(z), -\nabla_y(z), 1]^T$ with $z = [y_i, \hat{y}_i]$.

The last component $L_{SSIM}(y_i, \hat{y}_i)$ loss, Equation 5, is based on the mean structural similarity ($SSIM$) [41]. Similarly to [16], [39] we add this function to improve the depth reconstruction and the overall final estimation.

$$L_{SSIM}(y_i, \hat{y}_i) = 1 - SSIM(y_i, \hat{y}_i) \quad (5)$$

In conclusion, the proposed BLF balances the image reconstruction L_{depth}, the image similarity L_{SSIM}, the edge reconstruction L_{grad} and the edge similarity L_{norm} losses. The impact of each loss will be quantitatively evaluated in Section V-C.

C. The data augmentation policy

Deep learning architectures and especially Vision Transformer need a large amount of input data to avoid overfitting of the given task. Those models are typically trained on large-scale labelled datasets in a supervised learning strategy [4]. However, gathering annotated images is time-consuming and labour-intensive; as result, the data augmentation (DA) technique is a typical solution for expanding the dataset by creating new samples. In the MDE task, the use of DA techniques characterized by geometric and photometric transformations are a standard practice [5], [16]. However, not all the geometric and image transformations would be appropriate due to the introduced distortions and aberrations in the image domain, which are also reflected on the ground-truth depth maps.

With METER we propose a data augmentation policy based on commonly used DA operations while introducing a novel approach named *shifting strategy*. In particular we consider as *default* augmentation policy the use of the vertical flip, mirroring, random crop and channels swap of the input image as in [16] to make the network invariant to specific color distributions. The key idea is to combine the *default* augmentation policy with the *shifting strategy* augmentation, based on two simultaneous transformations applied respectively to the input image and to the ground truth depth map. The first one applies a color (*C*) shift to the RGB input images, while the second one is a depth-range (*D*) shift, which consists of adding a small, random positive or negative value to the depth ground truth. The mathematical formulation of the computed transformations are following reported; we refer with rgb_{un} and rgb_{aug} respectively the unmodified and the augmented input for RGB images and with d_{un} and d_{aug} the unmodified and the augmented depth map.

The *C* shift augmentation, applied on RGB images, is composed of two consecutive steps. In the first operation we apply a gamma-brightness transformation (rgb_{gb}), as reported in Equation 6, where β and γ are respectively the brightness and gamma factors that are randomly chosen into a value range experimentally defined between $[0.9, 1.1]$.

$$rgb_{gb} = \beta * (rgb_{un})^{\gamma} \quad (6)$$

Then, the color augmentation transformation reported in Equation 7 is applied, where I is an identity matrix of $H \times W$ resolution and η is a scaling factor that is randomly chosen into a value range empirically set between $[0.9, 1.1]$.

$$rgb_{aug} = rgb_{gb} * (I_{H \times W} * \eta) \quad (7)$$

The *D* shift augmentation, Equation 8, is made up of a random positive or negative value summed to the ground-truth depth maps (d_{un}). The random value, with a range of $[-10, +10]$ centimeters for the indoor dataset and $[-10, +10]$ decimeters for the outdoor one, is uniformly applied to the whole depth map.

$$d_{aug} = d_{un} + S_{H \times W} \quad (8)$$

In Figure 3 we report a sample frame before and after the application of the proposed strategy with the minimum and the maximum shift values. To emphasise the impact of the *D* shift, we focus on a narrow portion of the original depth map (in a distance range between 150 and 300 centimeters) by applying a perceptually uniform colormap and highlighting the minimum and maximum depth intervals through the associated color bars. The reported frames show that the depth with

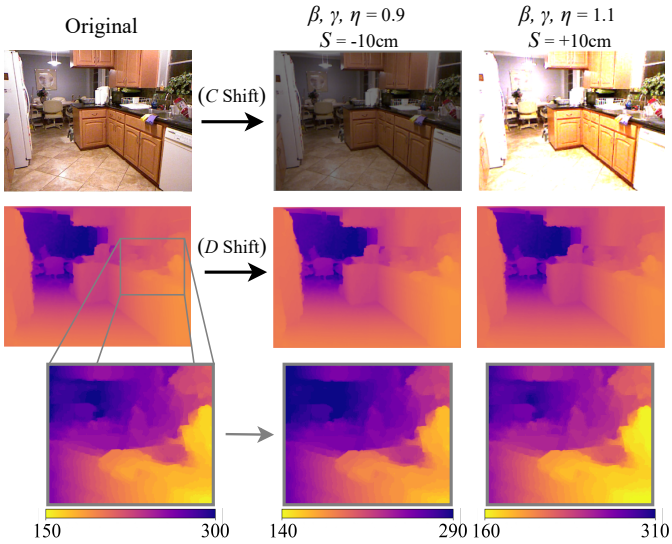

Fig. 3: Illustration of an augmented sample with the proposed *shifting strategy*. The shifting factors (β, γ, η, and S) are set as their maximum and minimum values, i.e. $\{0.9, -10\}$ and $\{1.1, +10\}$ respectively. The min/max depth ranges for the regions of interest are given through the respective colored bars.

the positive displacement ($+10$ centimeters) has a lighter colormap, while the depth with the negative displacement (-10 centimeters) has a darker one; this effect is emphasised by the colormap of the original distance distribution.

The introduced depth-range shift augmentation, along with the color and brightness shift and the commonly used transformations, leads to better final estimations as will be shown in Section V-D providing also invariance to color and illumination changes.

IV. Experimental Setup

This section gives a detailed description of the experimental setup, including training hyper-parameters, benchmark datasets and evaluation metrics respectively in Sections IV-A, IV-B, and IV-C.

A. Training hyper-parameters

METER has been implemented using PyTorch[2] deep learning API, randomly initializing the weights of the architectures. All the models have been trained from scratch using the AdamW optimizer [42] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $wd = 0.01$ and an initial learning rate of 0.001 with a decrement of 0.1 every 20 epochs. We use a batch size of 128 for a total of 60 epochs. For the *balanced loss function* we empirically choose the scaling factors $\lambda_1 = 0.5$ and $\lambda_2, \lambda_3 = \{1, 10, 100\}$ depending on the unity of measure used for the predicted depth map, i.e. meters, decimeters or centimeters. We apply a probability of 0.5 for all the random transformations set in the data augmentation policy.

[2]Code and corresponding pre-trained weights are made publicly available at the following GitHub repository:
https://github.com/lorenzopapa5/METER

B. Benchmark datasets

The datasets used to show the performance of METER are NYU Depth v2 [11] and KITTI [12], two popular MDE benchmark datasets for indoor and outdoor scenarios.

NYU Depth v2 dataset provides RGB images and corresponding depth maps in several indoor scenarios captured at a resolution of 640×480 pixels. The depth maps have a maximum distance of 10 meters. The dataset contains $120K$ training samples and 654 testing samples; we used for training the $50K$ subset as performed by previous works [5], [16]. The input images have been downsampled at a resolution of 256×192.

KITTI dataset provides stereo RGB images and corresponding 3D laser scans in several outdoor scenarios. The RGB images are captured at a resolution of 1241×376 pixels. The depth maps have a maximum distance of 80 meters. We train our network at a input resolution of 636×192 on Eigen et. al [13] split; it is composed of almost $23K$ training and 697 testing samples. Similarly to [21], due to the low density depth maps, we evaluate the compared models in the cropped area where point-cloud measurement are reported.

C. Performance evaluation

We quantitatively evaluate the performance of METER using common metrics [13] in the monocular depth estimation task: the root-mean-square error (RMSE, in meters [m]), the relative error (REL), and the accuracy value δ_1, respectively reported in Equations 9, 10, and 11. We remind that y_i is the ground truth depth map for the ith pixel while $\hat{y}_i$ is the predicted one, n is the total number of pixels for each depth image, and thr is a threshold commonly set to 1.25.

$$RMSE = \sqrt{\frac{1}{|n|} \sum_{i \in n} ||y_i - \hat{y}_i||^2} \tag{9}$$

$$REL = \frac{1}{|n|} \sum_{i \in n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{10}$$

$$\delta_1 = \frac{1}{|n|} \sum_{i \in n} \max\left(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}\right) < thr \tag{11}$$

Moreover, we compare the different models through the number of multiply-accumulate (MAC) operations and trainable parameters. METER has been tested on the low-resource embedded 4GB NVIDIA Jetson TX1[3] and the 4GB NVIDIA Jetson Nano[4] that have a power consumption of $10W$ and $5W$ respectively. Those devices are equipped with an ARM CPU and a 256-core NVIDIA Maxwell GPU[5] for the TX1 and a 128-core for the Nano. The inference speed reported in Section V are computed as frame-per-second (fps) on a single image averaged over the entire test dataset.

[3]https://developer.nvidia.com/embedded/jetson-tx1
[4]https://developer.nvidia.com/embedded/jetson-nano
[5]https://developer.nvidia.com/maxwell-compute-architecture

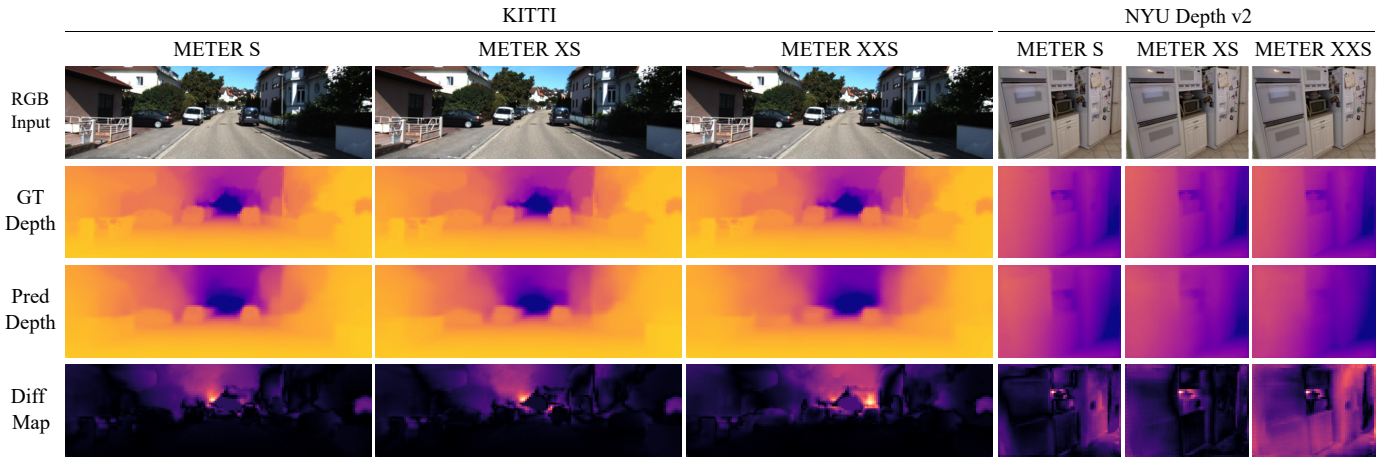

Fig. 4: A graphical comparison among METER (S, XS, XXS) configurations. For a better visualization, we apply to depth images and difference maps uniform colormaps with the same depth range. Precisely, in the ground truth (GT) and predicted depth maps (Pred) a lower color intensity corresponds to further distances, while in the difference map (Diff $= |\text{GT} - \text{Pred}|$) a lower color intensity corresponds to a smaller error.

TABLE II: Comparison with state of the art lightweight methods on the two benchmark datasets. The best scores are in bold and second best are underlined; the - represents a value which is not reported in the original paper.

Models	NYU				KITTI				Parameters [M]
	RMSE↓ [m]	REL↓	δ_1 ↑	MAC [G]	RMSE↓ [m]	REL↓	δ_1 ↑	MAC [G]	
CReaM [32]	0.687	0.190	0.704	-	-	-	-	-	-
PyD-Net (50) [31]	-	-	-	-	6.253	0.262	0.759	-	1.9
PyD-Net (200) [31]	-	-	-	-	6.030	0.153	0.789	-	1.9
FastDepth [8]	0.579	-	0.772	3.210	-	-	-	-	3.9
M.Net v2 + FBNet [34]	0.564	-	0.790	-	-	-	-	-	2.6
SPEED [7]	0.566	0.158	0.783	0.552	5.191	0.181	0.770	1.403	2.6
METER S	**0.471**	**0.134**	**0.831**	0.975	**4.603**	**0.126**	**0.829**	2.432	3.3
METER XS	0.522	0.154	0.793	0.579	4.671	0.128	0.827	1.444	1.4
METER XXS	0.580	0.174	0.744	0.186	5.157	0.156	0.782	0.464	0.7

V. Results

In this section, we report the results obtained with METER on the two evaluated datasets, NYU Depth v2 and KITTI, described in the previous Section IV-B. In Section V-A METER is compared with lightweight, state of the art related works in terms of the metrics described in Section IV-C; then, we report multiple ablation studies to emphasize the individual contribution of each METER component. In particular, Section V-B is related to the architecture structure, while Sections V-C and V-D analyze respectively the effect of each element of the proposed *balanced loss function* and of the *shifting strategy* used for data augmentation. Finally, in SectionV-E, we provide an example of METER application in a real-case scenario.

A. Comparison with state of the art methods

In this section, METER is compared with state of the art lightweight models as [7], [8], [31], [32], [34], which are designed to infer at high speed on embedded devices while keeping a small memory footprint (lower than 3GB). This choice is due to the limited amount of available memory in the chosen platforms. Usually a portion of available RAM is reserved for the operating system, thus lowering the overall amount of available space for the model allocation. In particular METER and its variants allocate less than 2.1GB of available memory, a value that does not saturate the hardware's memory and which gives the opportunity to perform other operations on the same device. Moreover, for each compared architecture we also report the number of trainable parameters (in million [M]) and the number of Multiply-And-Accumulate (MAC) operations (in giga [G]).

The results can be found in Table II; as can be noticed, METER outperforms all the other methods on both the datasets. When compared with [7], METER S achieves a boost of 17%, 15%, and 6% respectively for the RMSE, REL and δ_1 metrics over NYU Depth v2 dataset and of 11%, 30% and 7% over KITTI. As before, METER XS achieves superior performances, with a boost of 9%, 10% and 5% over NYU Depth v2 dataset and of 10%, 29% and 7% over KITTI. The last configuration, METER XXS, can still obtain good predictions compared with state of the art models while using just 0.7M trainable parameters and 0.186G MAC operations.

Moreover, in order to assess the frequency performances of such architectures, we choose as baseline models SPEED, due to its accuracy, and FastDepth, which is one of the most popular technique. When tested on the NVIDIA Jetson TX1, such models achieve 30.9 fps and 18.8 fps, while METER S, XS and XXS achieve respectively 16.3 fps, 18.3 fps and 25.8 fps. From these results we can remark that our most accurate model shows similar fps values with respect to FastDepth with

TABLE III: Comparison between the MobileViT [9] and METER encoders over different activation functions (ReLU, SiLU) keeping METER decoder fixed. The fps are measured on the two benchmark hardwares, the NVIDIA Jetson TX1 and the NVIDIA Jetson Nano. In bold the best results for each configuration in terms of RMSE, REL and δ_1.

Encoders	NYU						KITTI						Parameters [M]
	RMSE↓ [m]	REL↓	δ_1 ↑	TX1↑ [fps]	Nano↑ [fps]	MAC [G]	RMSE↓ [m]	REL↓	δ_1 ↑	TX1↑ [fps]	Nano↑ [fps]	MAC [G]	
MobileViT S	0.549	0.168	0.763	13.3	10.5	1.222	4.673	0.128	0.825	5.1	4.1	3.046	5.9
MobilViT ReLU S	0.521	0.153	0.790	13.3	10.5	1.222	4.789	0.140	0.815	5.1	4.1	3.046	5.9
METER SiLU S	0.496	0.145	0.811	16.3	12.0	0.975	4.692	0.134	0.825	5.9	4.8	2.432	3.3
METER S	**0.471**	**0.134**	**0.831**	16.3	12.0	0.975	**4.603**	**0.126**	**0.829**	5.9	4.8	2.432	3.3
MobileViT XS	0.572	0.171	0.754	13.5	13.3	0.815	4.734	0.133	0.822	5.9	5.1	2.032	2.8
MobilViT ReLU XS	0.547	0.158	0.780	13.5	13.3	0.815	4.797	0.137	0.819	5.9	5.1	2.032	2.8
METER SiLU XS	0.539	0.156	0.787	18.3	15.6	0.579	4.727	0.133	0.821	7.2	6.0	1.444	1.4
METER XS	**0.522**	**0.154**	**0.793**	18.3	15.6	0.579	**4.671**	**0.128**	**0.827**	7.2	6.0	1.444	1.4
MobileViT XXS	0.615	0.195	0.715	17.4	16.9	0.472	5.211	0.187	0.761	14.3	10.7	1.180	1.8
MobilViT ReLU XXS	0.588	0.176	0.737	17.4	16.9	0.472	5.210	0.171	0.763	14.3	10.7	1.180	1.8
METER SiLU XXS	0.596	0.180	0.728	25.8	23.2	0.186	5.208	0.165	0.763	20.4	15.1	0.464	0.7
METER XXS	**0.580**	**0.174**	**0.744**	25.8	23.2	0.186	**5.157**	**0.156**	**0.782**	20.4	15.1	0.464	0.7

TABLE IV: Comparison between lightweight decoder architectures keeping METER S encoder fixed. The best scores are in bold.

Decoders	NYU						KITTI						Parameters [M]
	RMSE↓ [m]	REL↓	δ_1 ↑	TX1↑ [fps]	Nano↑ [fps]	MAC [G]	RMSE↓ [m]	REL↓	δ_1 ↑	TX1↑ [fps]	Nano↑ [fps]	MAC [G]	
NNDSConv5 [8]	0.596	0.174	0.685	15.4	11.5	0.869	5.737	0.164	0.677	5.5	4.7	2.166	3.1
NNConv5 [8]	0.562	0.167	0.761	14.6	11.3	1.141	4.895	0.139	0.818	5.6	4.5	2.845	3.6
MDSPP [7]	0.581	0.169	0.694	15.1	11.7	1.004	5.167	0.157	0.760	5.7	4.7	2.503	3.4
METER S	**0.471**	**0.134**	**0.831**	**16.3**	**12.0**	0.975	**4.603**	**0.126**	**0.829**	**5.9**	**4.8**	2.432	3.3

a sensible lower estimation error, while the lightweight XXS variant exhibits comparable estimation performance and fps with respect to SPEED.

Regarding MAC operations, it is possible to see that SPEED MAC value is on par with METER XS, while FastDepth MAC is sensible higher than all METER architectures.

Furthermore, a qualitative analysis between the proposed variants of METER is reported in Figure 4 over an indoor and outdoor scenarios. The estimated depths and their associated difference (Diff) maps, which are per-pixel differences between the ground truth depth maps (GT Depth) and the predicted (Pred Depth) ones, show how the estimation error is distributed along the frame. Precisely, we notice an error increment fairly distributed over the frame as the model trainable parameters of the model are reduced.

B. Ablation study: the encoder-decoder architecture

In this subsection we compare the performances of the encoder and the decoder components of METER; results are reported in Table III and Table IV, respectively. In particular, the first analysis highlights the contribution of the novel METER block for each configuration (S, XS, and XXS) while keeping METER decoder fixed. The second analysis focuses on the use of alternative decoders with respect to the default METER decoder, such as NNDSConv5, NNConv5 [8] and MDSPP [7] using METER S encoder since it is the encoder that shows the best performances in the evaluated metrics.

Encoder architectures are compared in Table III, reporting a one-to-one comparison between METER encoder and the MobileViT; evaluating the effects of two different activation functions (ReLU, SiLU). From the obtained results, we highlight that METER encoder (in bold) achieves better depth estimation in all the proposed variants, as well as when compared with the same activation function, using fewer trainable parameters and a reduced number of MAC operations. In particular, when compared with the MobileViT, METER achieves an average improvement of 10%, 14% and 6% on RMSE, REL, and δ_1 metrics in the indoor dataset and of 2%, 7% and 2% respectively on the outdoor dataset. Based on those findings, the overall estimation contribution of the proposed encoder over the three configurations is equivalent to 7%, which almost 3% is due to the use of ReLU activation function with respect to SiLU. Moreover, regarding MAC operations we obtain a reduction of 20%, 29%, and 60% with respect to the corresponding MobileViT variants (S, XS, XXS), while the fps improvements are respectively 16% fps, 22% fps, and 32% on the NVIDIA Jetson TX1 and of 16% fps, 15% fps, and 28% fps over the NVIDIA Jetson Nano.

In light of the previous experiments, we can state that all METER variants show good accuracy and frequency performances on the NYU Depth v2, while in the case of KITTI dataset METER XXS variant should be preferred in order to get a reasonable inference speed. Focusing on the timings, the METER XXS variant shows the fastest inference speed, with reasonable results also on high resolution images of KITTI dataset, avoiding the needing of cropping or downscaling the original images.

Decoders architectures are reported in Table IV, comparing METER decoder and those of other lightweight models; we used the METER S encoder as baseline. METER decoder achieves an RMSE improvement of 16% and 19% on NYU Depth v2 dataset and of 6% and 11% on KITTI dataset with respect to NNConv5 and MDSPP models. Furthermore, we compare METER decoder with the NNDSConv5 [8], a variant

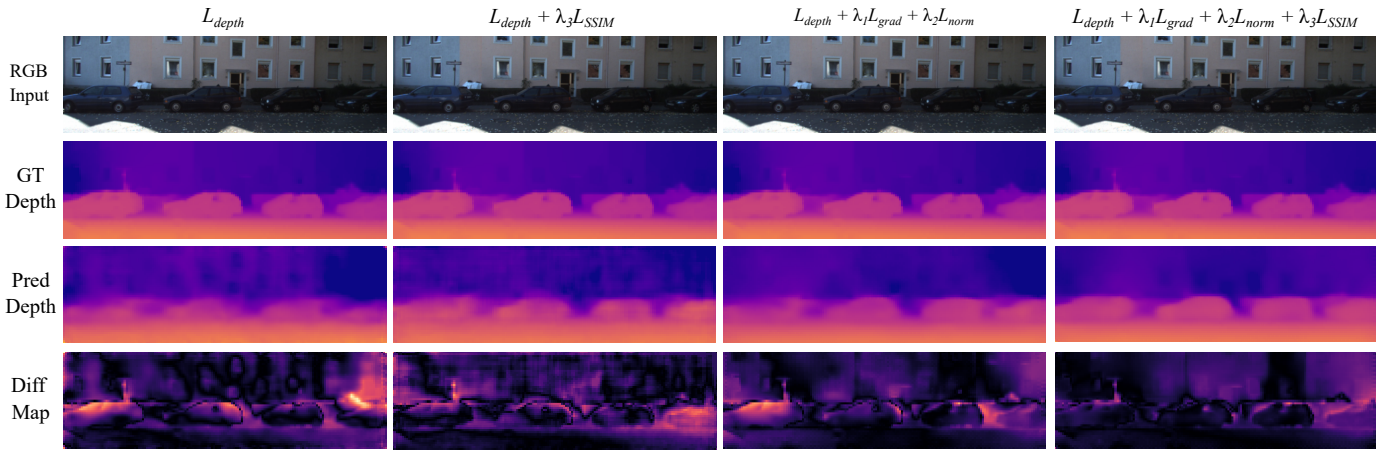

Fig. 5: Qualitative comparison of a predicted frame taking into account different loss components. For a better visualization, we apply to the depth images and to the difference maps uniform colormaps with the same depth range. Precisely, in the ground truth (GT) and predicted depth maps (Pred) a lower color intensity corresponds to further distances, while in the difference map (Diff $= |\text{GT} - \text{Pred}|$) a lower color intensity corresponds to a smaller error.

TABLE V: The effect of each *balanced loss function* components on the METER S over the considered metrics. The best scores are in bold.

Loss Components	NYU			KITTI		
	RMSE↓ [m]	REL↓	δ_1 ↑	RMSE↓ [m]	REL↓	δ_1 ↑
L_{depth}	0.582	0.185	0.736	5.637	0.183	0.741
L_{depth} + $\lambda_3 L_{SSIM}$	0.544	0.161	0.774	5.526	0.198	0.731
L_{depth} + $\lambda_1 L_{grad}$ + $\lambda_2 L_{norm}$	0.522	0.153	0.792	5.285	0.166	0.744
L_{depth} + $\lambda_1 L_{grad}$ + $\lambda_2 L_{norm}$ + $\lambda_3 L_{SSIM}$ **(BLF)**	**0.471**	**0.134**	**0.831**	**4.603**	**0.126**	**0.829**

of the NNConv5 that takes advantage of depthwise separable convolution to reduce the computational cost. Our encoder-decoder architecture is able to achieve higher speed and a significant improvement in all the estimation metrics with comparable MAC operations with respect to NNDSConv5. Finally, when compared with the NNConv5 decoder, ranked second in our analysis, the proposed structure is able to achieve an overall improvement equal to 12% over the two scenarios. Moreover, it can be noticed that the decoder has little influence on the inference frequency; however, METER decoder still shows the best fps on the two hardwares (e.g. 11% of METER S compared to NNConv5 on the TX1 hardware and NYU Depth v2 dataset). The overall MAC operations decrement with respect to NNConv5 and MDSPP decoders is equal to 15% on the same configuration as before, suggesting that the optimized METER decoder is able to produce more accurate estimations while using less operations.

C. Ablation study: loss function

In this subsection we analyze the impact of the different components of the proposed *balanced loss function* introduced in Section III-B. METER S architecture is used as a baseline model. The quantitative and qualitatively comparisons are provided in Table V and Figure 5 respectively, while Figure 6 shows the converging trends of each introduced component, referring to L_{depth} (blue), L_{grad} (orange), L_{norm} (green) and L_{SSIM} (red).

The curves shape show that the initial loss contribution is mostly attributed to the L_{depth} and L_{grad}, while the contributions of the L_{SSIM} and L_{norm} penalize from start to finish structural and high-level details prediction errors.

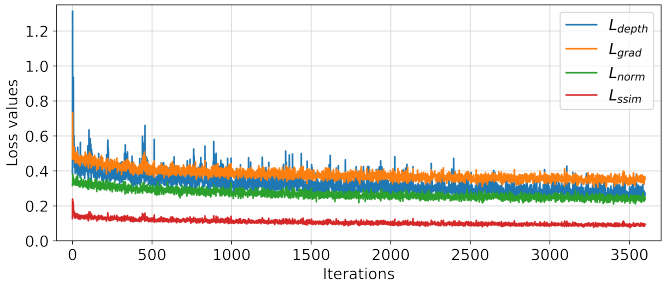

Fig. 6: Plot of the individual loss components composing the *balanced loss function* in the first ten epochs, i.e. almost 3600 iterations.

The L_{depth} component showed to be fundamental for the training convergence, thus it is applied on every experiment of Table V. The obtained results demonstrate that each loss component is crucial to get the final METER performance, balancing the reconstruction of the entire image and of edges details. In fact, the loss formulation in the second row focuses only on the overall image, failing at reaching satisfying results. At the same time, the third row shows a typical loss exploited in [38] focusing on edge details but not taking into account the image structure similarity, thus producing an unbalanced loss achieving a worse result with respect to the proposed one, which is able to obtain the lowest estimation error by

TABLE VI: Comparison between different augmentation strategies. The *default* policy comprises the flip, random crop and channel swap while the others represent the different components of the *shifting strategy* described in Section III-C. The reference model is METER S. The best scores are in bold.

Augmentation Components	NYU			KITTI		
	RMSE↓ [m]	REL↓	δ_1 ↑	RMSE↓ [m]	REL↓	δ_1 ↑
default	0.511	0.143	0.813	4.839	0.128	0.826
default + *C* shift	0.506	0.143	0.815	4.897	0.136	0.810
default + *D* shift	0.585	0.144	0.805	4.938	0.141	0.804
default + *C* shift + *D* shift **(our)**	**0.471**	**0.134**	**0.831**	**4.603**	**0.126**	**0.829**

balancing all the components. In detail, the BLF achieves an improvement of 10%, 12%, and 5% for RMSE, REL and δ_1 metrics on NYU dataset, and a boost of 13%, 24%, and 10% over the KITTI dataset compared to [38].

Moreover, to better show the qualitative contribution of each loss component, provided in Figure 5 the estimated depth under the four analyzed configurations given an input sample from KITTI dataset. Based on such example, we can observe a similar behaviour to the one found in Figure 6 and Table V: the L_{depth} component is fundamental for a correct image reconstruction while the weighted addition of specific loss components ($\lambda_1 L_{grad}$, $\lambda_2 L_{norm}$, $\lambda_3 L_{SSIM}$) can quantitatively and qualitatively improve the final estimation. This improvement may also be noticed by observing the predicted frames from left to right, where the object details and the overall estimation increase significantly as difference maps darken.

Therefore, we can conclude that the proposed *balanced loss function* can successfully enhance the training process, while each component can effectively contribute to more accurate estimations, hence enhancing the entire framework. Precisely, the overall quantitative contribution of the *balanced loss function* over the two scenarios is equal to 25% when compared with L_{depth}, and 13% with respect to the loss formulation used in [38].

D. Ablation study: data augmentation

In this ablation study, we evaluate the performances of the proposed data augmentation strategy in comparison with standard MDE data augmentation. We report in Table VI the quantitative results of *shifting strategy* (*C* shift, *D* shift) and the *default* DA (flip, random crop and channel swap) and the combinations of the two. The proposed *shifting strategy* (last row) achieves, on METER S architecture, an improvement of 8%, 6%, and 2% over the RMSE, REL and δ_1 on the NYU Depth v2 dataset, and of 6%, 2%, and 1% over the KITTI dataset. On the other hand, the single use of the *C* shift or *D* shift with the *default* augmentation does not lead to an improvement in the final estimation, resulting in equivalent or slightly worst final prediction. Then, the overall improvement of the *shifting strategy* over the two scenarios is equal to 4% with respect to the *default* data augmentation policy.

E. Real-case scenario

One of the main objectives of exploring lightweight deep learning solutions is to close the gap between computer vision and practical applications, where the proposed models may be integrated as perception systems, such as robotic systems, thus taking into account possible hardware limitations. Therefore, in this subsection, we present an example of a real-case application in which METER is used to estimate the depth scene obtained from a generic camera image. We used a Kinetic V2 to measure the reference depth of the scene. The extracted acquisition is reported in Figure 7.

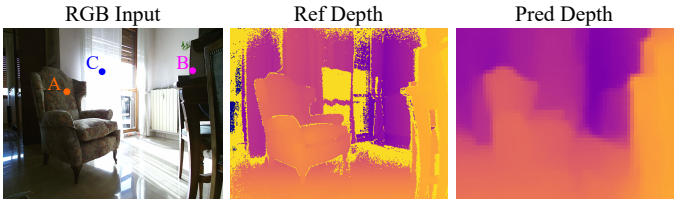

Fig. 7: METER application in a real-case scenario. Missing depth measurements of reference (Ref) depth are shown as yellow pixels. A uniform colormap, with the same depth range, has been applied to the depth maps. Points A (armchair), B (box), and C (curtain) on the RGB frame indicates object used for quantitative comparison.

Qualitatively comparing the reference depth and the estimated one, we can notice a less sharp prediction, which can be mainly attributed to the lower working resolution that ensures high frame rates on edge devices. However, the object shapes are still adequately defined, and the overall estimation is visually comparable with the reference frame.

Moreover, in order to perform a quantitative analysis, we compute the average error of three salient objects that appear in the input frame (RGB Input), which are point A for the armchair, point B for the box and point C for the curtain. The estimation error for the first two points (A and B) is almost equal to 0.79m, respectively. The obtained value is related to the fact that we are working in a challenging open-set scenario with different statistics with respect to the training set. On the other hand, by analyzing point C, we can identify one of the main drawbacks of active depth sensing, i.e. missing or incorrect depth measurements under particular lighting conditions. In this scenario, although the estimated depth error is unknown, most likely due to the intense light source directed towards the camera sensor, our model can still correctly identify and estimate the area as a single surface.

VI. Conclusions

In this work, we propose METER, a MDE architecture characterized by a novel lightweight vision transformer model,

a multi-component loss function and a specific data augmentation policy. Our method exploits a lightweight encoder-decoder architecture characterized by a transformer METER block, which is able to improve the final depth estimation with a small number of computed operations, and a fast upsampling block employed in the decoder. METER achieves high inference speed over low-resource embedded hardwares such as the NVIDIA Jetson TX1 and the NVIDIA Jetson Nano. Moreover, METER architecture in its three configurations is able to outperform previous state of the art lightweight related works. Thanks to the obtained performances on inference frequency and accuracy in the estimation, such proposed architectures can be good candidate to work on multiple MDE scenarios and real-world embedded applications. Precisely, METER S outperforms the accuracy of state of the art lightweight methods over the two datasets, METER XS represents the best trade-off between inference speed and estimation error, and METER XXS reaches a high inference frequency, up to 25.8 fps, on the two hardwares at the cost of a small increment in the estimation error.

The obtained results and the limited MAC operations of the proposed network demonstrate that our framework could be valuable in a variety of resource-constrained applications, such as autonomous systems, drones, and IoT. Moreover, we also test METER in a real-case scenario with a frame captured by a generic camera achieving a reasonable estimation error.

Finally, METER architecture could be a valuable starting point for future studies, in order to get real-time inference frequency on high resolution images, as well as building transformer architectures to take advantage of the attention mechanism both in encoder and decoder structures.

ACKNOWLEDGMENTS

This work was partially supported by the Sapienza University of Rome project 2022-2024 “A Novel Vision-based detection system for the control of the ectoparasitic mite Varroa destructor in honey bee colonies”.

REFERENCES

[1] Y. Ming, X. Meng, C. Fan, and H. Yu, “Deep learning for monocular depth estimation: A review,” *Neurocomputing*, vol. 438, pp. 14–33, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220320014

[2] R. Xiaogang, Y. Wenjing, H. Jing, G. Peiyuan, and G. Wei, “Monocular depth estimation based on deep learning:a survey,” in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 2436–2440.

[3] Z. Liu, Y. Tan, Q. He, and Y. Xiao, “Swinnet: Swin transformer drives edge-aware rgb-d and rgb-t salient object detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 7, pp. 4486–4497, 2022.

[4] R. Ranftl, A. Bochkovskiy, and V. Koltun, “Vision transformers for dense prediction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 179–12 188.

[5] S. Farooq Bhat, I. Alhashim, and P. Wonka, “Adabins: Depth estimation using adaptive bins,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 4008–4017.

[6] Z. Li, X. Wang, X. Liu, and J. Jiang, “Binsformer: Revisiting adaptive bins for monocular depth estimation,” 2022. [Online]. Available: https://arxiv.org/abs/2204.00987

[7] 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. 44 881–44 890, 2022.

[8] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, “Fastdepth: Fast monocular depth estimation on embedded systems,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6101–6108.

[9] S. Mehta and M. Rastegari, “Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer,” 2021. [Online]. Available: https://arxiv.org/abs/2110.02178

[10] Y. Chen, X. Dai, D. Chen, M. Liu, X. Dong, L. Yuan, and Z. Liu, “Mobile-former: Bridging mobilenet and transformer,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5270–5279.

[11] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in *ECCV*, 2012.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

[13] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS’14. Cambridge, MA, USA: MIT Press, 2014, p. 2366–2374.

[14] Y. Cao, Z. Wu, and C. Shen, “Estimating depth from monocular images as classification using deep fully convolutional residual networks,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3174–3182, 2018.

[15] Y. Cao, T. Zhao, K. Xian, C. Shen, Z. Cao, and S. Xu, “Monocular depth estimation with augmented ordinal depth relationships,” *IEEE Transactions on Image Processing*, pp. 1–1, 2018.

[16] I. Alhashim and P. Wonka, “High quality monocular depth estimation via transfer learning,” 2018. [Online]. Available: https://arxiv.org/abs/1812.11941

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[18] S. Gur and L. Wolf, “Single image depth estimation trained via depth from defocus cues,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7683–7692.

[19] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” 02 2018.

[20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[21] M. Song, S. Lim, and W. Kim, “Monocular depth estimation using laplacian pyramid-based depth residuals,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 11, pp. 4381–4393, 2021.

[22] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[24] I. Yun, H.-J. Lee, and C. E. Rhee, “Improving 360 monocular depth estimation via non-local dense prediction transformer and joint supervised and self-supervised learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 3224–3233.

[25] R. Li, P. Ji, Y. Xu, and B. Bhanu, “Monoindoor++:towards better practice of self-supervised monocular depth estimation for indoor environments,” *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2022.

[26] D. Kim, W. Ga, P. Ahn, D. Joo, S. Chun, and J. Kim, “Global-local path networks for monocular depth estimation with vertical cutdepth,” 2022. [Online]. Available: https://arxiv.org/abs/2201.07436

[27] Y. Ishii and T. Yamashita, “Cutdepth: Edge-aware data augmentation in depth estimation,” *ArXiv*, vol. abs/2107.07684, 2021.

[28] Z. Li, Z. Chen, X. Liu, and J. Jiang, “Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation,” 2022. [Online]. Available: https://arxiv.org/abs/2203.14211

[29] M. Véstias, R. Duarte, J. Sousa, and H. Neto, “Moving deep learning to the edge,” *Algorithms*, vol. 13, p. 125, 05 2020.

[30] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Q. AlDujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. AlAmidie, and L. Farhan, “Review of deep learning: concepts, cnn architectures, challenges,

applications, future directions," *Journal of Big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[31] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "Towards real-time unsupervised monocular depth estimation on cpu," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5848–5854.

[32] A. Spek, T. Dharmasiri, and T. Drummond, "Cream: Condensed real-time models for depth prediction using convolutional neural networks," 2018.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[34] M. K. Yücel, V. Dimaridou, A. Drosou, and A. Saà-Garriga, "Real-time monocular depth estimation with sparse supervision on mobile," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2428–2437.

[35] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 726–10 734.

[36] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[37] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[38] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, "Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1043–1051.

[39] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6602–6611.

[40] J. W. Foreman, "Data smart: Using data science to transform information into insight," 2013.

[41] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

LORENZO PAPA is a Ph.D. student in Computer Science Engineering. He collaborates with the AlcorLab in the DIAG department, University of Rome Sapienza, Italy. He received the B.S. degree in Computer and Automation Engineering and the M.S. degree in Artificial Intelligence and Robotics from the University of Rome La Sapienza, Italy, in 2019 and 2021, respectively. His main research interests are Deep Learning, Computer Vision and Cyber Security.

PAOLO RUSSO is an Assistant Researcher at AlcorLab in DIAG department, University of Rome Sapienza, Italy. He received the B.S. degree in Telecommunication Engineering from Università degli studi di Cassino, Italy, in 2008, and the M.S. degree in Artificial Intelligence and Robotics from University of Rome La Sapienza, Italy, in 2016. He received Ph.D. degree in Computer Science from University of Rome La Sapienza in 2020. From 2018 to 2019, he has been a researcher at Italian Institute of Technology (IIT) in Tourin, Italy. His main research interests are Deep Learning, Computer Vision, Generative Adversarial Networks, Reinforcement Learning.

IRENE AMERINI (M'17) received the Laurea degree in computer engineering and the Ph.D. degree in computer engineering, multimedia, and telecommunication from the University of Florence, Italy, in 2006 and 2010, respectively. She is currently an Assistant Professor with the Department of Computer, Control, and Management Engineering A. Ruberti, Sapienza Univeristy of Rome, Italy. She was a Visiting Scholar with Binghamton University, NY, USA, in 2010, and a Visiting Research Fellow of Charles Sturt University, Australia, in 2018, with a fellowship offered by the Australian Government Department of Education and Training, through the Endeavour Scholarship & Fellowship Program. Her main research activities include digital image processing, multimedia content security technologies, secure media, and multimedia forensics. She is a member of the IEEE Information Forensics and Security Technical Committee and the EURASIP TAC Biometrics, Data Forensics, and Security and the IAPR TC6 - Computational Forensics Committee. She has received the Italian Habilitation for an Associate Professor in telecommunications and computer science. She is a Guest Editor of several international journals. She is an Associate Editor of IEEE ACCESS, Journal of Electronic Imaging and Journal of Information Security and Applications.