IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 11, 2022, accepted April 18, 2022, date of publication April 25, 2022, date of current version May 2, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3170425

SPEED: Separable Pyramidal Pooling EncodEr-Decoder for Real-Time Monocular Depth Estimation on Low-Resource Settings

LORENZO PAPA, EDOARDO ALATI, PAOLO RUSSO, AND IRENE AMERINI, (Member, IEEE)

ALCOR Laboratory, Department of Computer, Management and Control Engineering (DIAG), Sapienza University of Rome, 00185 Rome, Italy

Corresponding author: Lorenzo Papa (papa@diag.uniroma1.it)

This work was supported in part by Sapienza University project 2022–2024 "A Novel Vision-Based Detection System for the Control of the Ectoparasitic Mite Varroa Destructor in Honey Bee Colonies."

ABSTRACT The monocular depth estimation (MDE) is the task of estimating depth from a single frame. This information is an essential knowledge in many computer vision tasks such as scene understanding and visual odometry, which are key components in autonomous and robotic systems. Approaches based on the state of the art vision transformer architectures are extremely deep and complex not suitable for real-time inference operations on edge and autonomous systems equipped with low resources (i.e. robot indoor navigation and surveillance). This paper presents SPEED, a Separable Pyramidal pooling EncodEr-Decoder architecture designed to achieve real-time frequency performances on multiple hardware platforms. The proposed model is a fast-throughput deep architecture for MDE able to obtain depth estimations with high accuracy from low resolution images using minimum hardware resources (i.e. edge devices). Our encoder-decoder model exploits two depthwise separable pyramidal pooling layers, which allow to increase the inference frequency while reducing the overall computational complexity. The proposed method performs better than other fast-throughput architectures in terms of both accuracy and frame rates, achieving real-time performances over cloud CPU, TPU and the NVIDIA Jetson TX1 on two indoor benchmarks: the NYU Depth v2 and the DIML Kinect v2 datasets.

INDEX TERMS Computer vision, monocular depth estimation, fast-throughput, edge devices.

I. INTRODUCTION

The estimation and extraction of depth information from images and videos is one of the basic and essential tasks in computer vision. Depth information can be successfully integrated with RGB data to obtain notable improvements in other challenging tasks, like face and object detection, semantic segmentation, visual SLAM, etc. [1]. Several applications, as autonomous systems, robotics manipulators and augmented reality algorithms, usually rely on stereo or depth cameras to achieve their tasks. However, in some settings the presence of a single RGB camera requires the estimation of the monocular depth; this is especially true for indoor or hostile environments where the use of small robots and drones introduce some constraints. Those indoor scenarios are usually characterized by a limited depth range (i.e. usually lower than 10 meters), and the lighting conditions can significantly change.

Depth estimation techniques can be divided into three major groups: geometry-based methods, which work on couples or sequences of images, as the structure from motion [2], the sensor-based methods [3], which exploit laser-based and RGBD devices, and the deep learning (DL) based techniques [1]. The latter methodologies have remarkable abilities to estimate accurate dense depth maps from a single image in an end-to-end fashion, without the need of complex pre-processing or additional assumptions. This approach leads the research community to further investigate and develop many novel architectures for depth estimation through a single camera.

However, recent studies [1] have demonstrated that MDE algorithms have limited capabilities w.r.t. real-time performances, as they focus on the estimation accuracy at the expense of the inference frequency. Indeed, depth estimation

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenhua Guo.

algorithms based on vision transformers [4] are computational heavy and not suitable to perform real-time inference in low-resource settings.

This paper proposes SPEED,[1] a *Separable Pyramidal pooling EncodEr-Decoder* architecture that aims at achieving real-time performances over both cloud and edge devices. The proposed method exploits an optimized version of the MobileNet [5] architecture: the depthwise convolutional blocks have been replaced by a novel depthwise separable pyramidal pooling (DSPP) layer, which led to an improvement in both the latency and the accuracy of the network. We experimented with several available encoder and decoder models to design SPEED, which is the optimal trade-off between accuracy and inference frequency. SPEED has been tested on systems with different resources as cloud CPU, TPU and the edge NVIDIA Jetson TX1.[2] Moreover, to have a complete overview of the behaviour on edge devices, we also test the accuracy and inference performances on the Google Coral Dev Board[3] Edge TPU.

SPEED achieves real-time (30 fps) predictions both on regular CPU and TPU workstations, and on low-power edge GPU; furthermore, it is able to obtain more accurate estimations on very low resolution images (up to 48 $\times$ 64) w.r.t. other fast-throughput MDE methods, such as [6], [7]. Figure 1 shows some examples of SPEED depth estimations. The robustness and the speed of our algorithm could be a valuable baseline for edge computing methods using edge computational components (e.g. TPUs); this will further allow to use DL techniques on several mobile and low-cost indoor systems, which are often characterized by low computational power.

The main contributions of the paper are summarized as follows:

- We propose an improved encoder architecture, based on the MobileNet [5] model, and a specifically designed decoder; both the structures take advantage of a novel layer named DSPP, which is a fast variant of the spatial pyramidal pooling layer.
- We show the effectiveness of SPEED in the supervised fast-throughput MDE task on two indoor datasets, the NYU Depth v2 [8] and the DIML Kinect v2 [39].
- We performed a detailed study on cloud and edge devices, reporting real-time frequency performances over CPU, TPU and low-resource NVIDIA Jetson TX1 GPU.
- We analyze the effect of alternative architectures and images resolutions through ablation studies, measuring the performances both on the accuracy estimation and on the inference frequency.

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 hyperparameters are discussed in Section IV, while Section V reports the results and a quantitative analysis of SPEED w.r.t. other significant works. Some final considerations and future applications are provided in Section VI.

FIGURE 1. SPEED depth map predictions.

II. RELATED WORKS

In this section we report state of the art related works on monocular depth estimation, as well as the pyramidal pooling layer used extensively in our method.

A. MONOCULAR DEPTH ESTIMATION

Deep architectures used to estimate depth maps from single images are usually trained on large-scale datasets [8], [9]. Plenty of deep neural networks have demonstrated their effectiveness to address the MDE problem mainly focusing on improving the estimation accuracy: from the most traditional convolution neural network (CNN) to the recent state of the art ones, as for example the methods based on vision transformers (ViT) [4]. Those architectures are usually based on an encoder-decoder structure [10], that provides sensible improvements on scene reconstruction [11] and image segmentation [12] tasks.

Among CNN-based approaches, Wofk *et al.* [6] propose an architecture that achieves fast-throughput using MobileNet [5] as a backbone, while Alhashim and Wonka [13] exploit DenseNet [14] model to obtain high-resolution depth estimation. Moreover, Eigen *et al.* [34] propose a multi-scale two steam architecture for depth estimation, while Zhang *et al.* [16] explores temporal information from monocular videos to capture temporal correlations among frames. Chen *et al.* [17] propose a specific architecture and a new dataset called Depth in the Wild for single image depth estimation in unconstrained settings; similarly

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

[2] https://developer.nvidia.com/embedded/jetson-tx1

[3] https://coral.ai/products/dev-board/

to Kim *et al.* [39] with the introduction of the DIML Kinect v2 dataset for indoor scenarios.

ViT networks have recently gained great visibility, since their accurate estimation capabilities related to the self-attention mechanism [18] can simultaneously extract the necessary information from both local and global pixel inter-relation, as described in Dosovitskiy *et al.* [4].

Some MDE works exploit ViT architectures to estimate depth by combining self-attention Transformers layers and classical CNN. AdaBins [19] makes use of the self-attention module as network head, transforming the depth regression task into a classification problem as proposed by Fu *et al.* [21]. In Ranftl *et al.* [20], the authors propose a novel architecture called Dense Prediction Transformer, which uses a transformer encoder combined with a traditional decoder to perform the estimation. Moreover, in [22] Ranftl *et al.* propose to use a zero-shot cross-dataset transfer technique to improve the generalization capability over different datasets while Gur and Wolf [41] propose to learn depth focus cues.

It is worth saying that the state of the art methods are mainly focused on attaining higher accuracy performances, at the expense of an increasing complexity and inference frequency; with SPEED we propose instead a fast-throughput CNN architecture with a balanced trade-off between accuracy and inference frequency. We define as *fast-throughput* architecture, a model that is characterized by an high inference frame rate i.e. the average frequency needed to the model to perform a decision in a low-resource setting while keeping consistent accuracy scores.

Up to now, few works have been proposed to address this problem in the indoor scenario. Most of them [6], [24] are designed to achieve real-time frequencies on a NVIDIA Jetson TX2 GPU[4] while being also widely employed in mobile applications [23]. Differently, we are interested in testing the those methods on less investigated devices; for this reason we chose as benchmark hardware the TPU-v2 and an Intel CPU provided by Google Cloud Platform. Furthermore, we compare the state of the art lightweight models on two edge devices: the NVIDIA Jetson TX1 and the Google Edge TPU. Furthermore, other relevant MDE methods have been developed for the automotive tasks tested on the outdoor KITTI [9] dataset. Godard *et al.* [25] propose an architecture to estimate high-quality depth maps while Abarghouei and Breckon [27] a CNN-based approach combined with domain adaptation techniques. Liu *et al.* [26] propose a semi-unsupervised recurrent module to accomplish accuracy scores competitive to deeper networks while maintaining real-time performances. Poggi *et al.* [7] propose a novel approach based on a pyramidal structure to obtain a good inference frequency over a standard CPU. Guizilini *et al.* [28] propose a 3D solution based on packing and unpacking 3D convolutions blocks used to preserve the representation of the details. From the best of our knowledge, our work is the first to profile an architecture designed to run at real-time speed on both cloud and edge devices. The obtained results are remarkable also w.r.t. the very low-resource Edge TPU, on which the model maintains an excellent final estimation accuracy and a faster frame rate w.r.t. other available models.

[4]https://developer.nvidia.com/embedded/buy/

B. PYRAMIDAL POOLING LAYER

The spatial pyramidal pooling (SPP) layer has been introduced for the first time by He *et al.* [29] for visual recognition tasks. Such a method avoids to repeatedly compute the same convolutional features, demonstrating to be faster than recurrent-CNN [30] with a similar accuracy. Many vision-related works for object detection and classification exploit pyramidal pooling networks (or some of its variations) as a complement to standard CNN architectures. Zhang *et al.* [31] proposed two pyramidal structures to capture the global context information in object detection task through different region-based context aggregations. Differently, Masci *et al.* [32] developed a multi-scale pyramidal pooling layer applied to the detection and classification of steel defects.

Inspired from these previous works, in this paper we propose an improved SPP layer, called DSPP, composed of multiple depthwise separable convolutions which are able to reduce the architecture computational cost while producing an accurate depth estimation.

III. PROPOSED METHOD

In this section, a sketch of SPEED is provided, with a detailed architecture analysis; an overview of the model is reported in Figure 2. In particular, the encoder and the decoder modules will be described in two separated sections (III-A and III-B) to highlight the reasons which led to each design choice. The consequent impact on the final performances will be discussed in Section V.

A. THE SPEED ENCODER

The encoder has the objective to progressively downsample the input image to extract features at multiple different scales, giving as output a compact set of high-level features. Popular backbone architectures in the MDE field, as previous mentioned, are the DenseNet [14] and the MobileNet [5].

The encoder architecture proposed in this work is a variation of the second one that is widely employed for multiple tasks ranging from MDE [6] to semantic segmentation [33]. More in detail, the MobileNet depthwise blocks perform two operations: a 3×3 depthwise convolution and a point-wise convolution.

In SPEED we supersede the final depthwise decomposition layers of the MobileNet (from the eighth to the eleventh, for a total of eight convolutional layers) with a DSPP layer, a fast-throughput variation of the spatial pyramidal pooling layer obtained replacing the original convolution operation with depthwise separable one. The DSPP is composed of four average pooling and four separable convolution layers. The whole

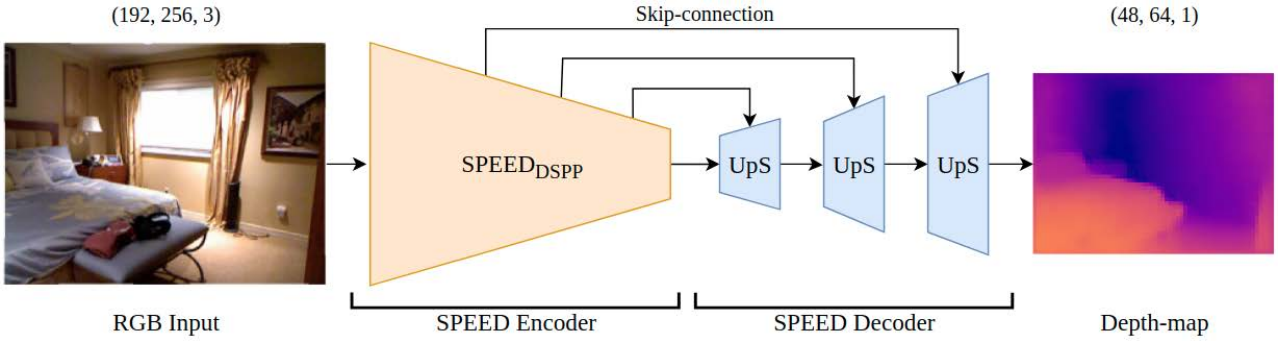


FIGURE 2. Overview of the SPEED encoder-decoder network architecture. It consists in two main components: the SPEED$_{DSPP}$ encoder and three cascaded up-sampling (UpS) blocks constituting the decoder. The input-output spatial dimensions are reported in the (H, W, C) format.

DSPP structure contributes with 281K trainable parameters, w.r.t the original configuration, which uses 1.08M parameters for the same objective.

As will be shown in Section V, this operation improves not only the inference frequency but also the estimation accuracy.

In fact, the depthwise separable convolution achieves the same result of a classical convolution through fewer multiplications, thus reducing the computational complexity and increasing the consequent inference frequency. The complete structure of the proposed DSPP layer is reported in Figure 3(a). The input features are first processed through four average pooling layers with different resolutions and then filtered by a depthwise separable convolution where its output features are a quarter of the input. Finally, the four outputs are resized to the desired shape and then concatenated.

Moreover, inspired by [13], we decrease by a factor of 4 the output features of the encoder, and the input-decoder features consequently, without introducing any substantial accuracy degradation. Please refer to Table 1, at the *Depthwise-block*$_{13lite}$,[5] for further details on output shape and parameters of the SPEED encoder.

The final encoder architecture, reported in Table 1, has 2.1M of parameters w.r.t. the 3.6M of the original MobileNet.

B. THE SPEED DECODER

The decoder objective is to upsample the features learned by the encoder while reconstructing the image details to obtain the desired output depth map (i.e. a per-pixel distance map).

In SPEED, both the encoder and the decoder exploit the just defined DSPP layer. More in detail, for the decoder we propose the mixed depthwise separable pyramidal pooling (MDSPP) layer, which keeps the features pipeline divided into two different flows until their final concatenation. This choice is due to the beneficial effect of having more depthwise separable convolution operations on specific features, which improves the decoder process while keeping a fast computation w.r.t. standard convolution ones. Via this setup, the network further increase the inference frequency by exploiting such MDSPP layer, which is designed over depthwise separable convolutions. Its graphical representation is reported in Figure 3(b). It has to be noticed that the sizes of the four average pooling layers and the flow of the computed operations in the MDSPP layer are the same as the DSPP one; such structure is also evidenced by the coloured arrows reported in Figure 3.

This layer is integrated in an up-sampling (UpS) block; in detail, the MDSPP output feature maps are up-sampled by a factor of 2 with a transposed convolution layer, and finally concatenated through the skip connection to recover the image details from the encoder feature maps. Finally, the merged output is processed via a depthwise separable convolution layer; to further diminish the decoder dimension, after the concatenation operation, we trim its number of parameters to the nearest power of two after reducing them by a factor of 4 (e.g. from 272 to 68, and finally rounded to 64). A graphical representation of the up-sampling block is reported in Figure 3(c). Finally, as depicted in Figure 2, the proposed decoder is composed of a sequence of three cascaded up-sampling blocks that progressively increase the feature resolution to the desired output. Via this set-up, the SPEED decoder architecture

TABLE 1. Structure of the SPEED encoder architecture with respective output shapes and number of parameters.

Layers	Output Shape [H, W, C]	Parameters [K]
Input image	(192, 256, 3)	0
Conv2D	(96, 128, 32)	0.99
Depthwise-block$_1$	(96, 128, 64)	2.72
Depthwise-block$_2$	(48, 64, 128)	9.53
Depthwise-block$_3$	(48, 64, 128)	18.56
Depthwise-block$_4$	(24, 32, 256)	35.45
Depthwise-block$_5$	(24, 32, 256)	69.88
Depthwise-block$_6$	(12, 16, 512)	136.44
Depthwise-block$_7$	(12, 16, 512)	270.84
DSPP layer	(12, 16, 1024)	281.08
Depthwise-block$_{12}$	(6, 8, 1024)	1065.98
Depthwise-block$_{13lite}$	(6, 8, 256)	276.48
Total Sum		2167.95

[5]The architecture blocks names are given following the original layers nomenclature of the MobileNet model.

counts a total of $\sim 500k$ trainable parameters while keeping the overall model fast and accurate as will be shown in Section V-D.

IV. EXPERIMENTAL SETUP

In this section we give a detailed description on the experimental setup; in particular, the dataset, the training hyper-parameters and the evaluation metrics are described to provide a complete overview of our method highlighting its strengths.

A. DATASETS

The datasets used to show the performance of SPEED are the NYU Depth v2 [8] and the DIML Kinect v2 [39], two popular datasets to benchmark MDE in an indoor setting. The NYU Depth v2 dataset provides RGB indoor images and ground truths depth maps, with a maximum distance of $10m$ and a resolution of 480×640 pixels. It contains $120K$ training samples and 654 testing samples [34]. We train our method on a $50K$ subset as previously done in [13] and [19]. The DIML Kinect v2 dataset is composed of more than $480K$ indoor images at a resolution of 792×1408 in a depth range between $0.5m$ to $7m$, taken with a Kinect v2 device. The indoor subset is composed by frames taken in several room categories, e.g. offices, dormitories, classrooms and restaurants. The dataset is split in a $150K$ training set and a $70K$ test set, as reported in [39].

The images of both datasets are resized to the optimal resolution of 192×256 due to the trade-off between inference frequency and accuracy score, discussed in the following in Section V-E.

B. EVALUATION METRICS

To evaluate and compare the performance of SPEED, we considered the most commonly used metrics in depth estimation tasks: root-mean-square error (RMSE), relative error (REL), and the accuracy value δ_1. Please refer to equations (1, 2, 3) for their formulation, where p_i and g_i are respectively the predicted depth map and its ground truth while P denotes the total number of pixels.

$$RMSE = \sqrt{\frac{1}{|P|}\sum_{i \in P} ||p_i - g_i||^2} \tag{1}$$

$$REL = \frac{1}{|P|}\sum_{i \in P} \frac{|p_i - g_i|}{g_i} \tag{2}$$

$$\delta_1 = \frac{1}{|P|}\sum_{i \in P} \max\left(\frac{p_i}{g_i}, \frac{g_i}{p_i}\right) < thr \tag{3}$$

In the last equation thr is a threshold commonly set to 1.25. Due to the importance of the inference frequency in our work, we propose a comparison w.r.t. the frame rate (fps) obtained over the different benchmark hardware of the analyzed methods.

C. TRAINING SETUP

We implemented SPEED using TensorFlow 2[6] deep learning high level API. We randomly initialized the convolutional kernels of both the encoder and the decoder (as described in [35]). ADAM [36] optimizer and its AMSGrad [37] variant have been used in all the experiments with the following setup: learning rate 0.0001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We have set a batch size of 16 and trained the model for a total of 25 epochs on the chosen dataset.

The loss function (4) is the Accurate Object Boundaries Loss [38], where the error e_i is computed as the absolute difference between the predicted image p_i and the ground truth g_i.

$$L = l_{depth} + l_{grad} + l_{norm} \tag{4}$$

This loss function is computed as combination of the point-wise depth loss l_{depth} (5), the Sobel gradient loss l_{grad} (6), where ∇ is the spatial derivative of e_i w.r.t. the specific axis, and the normal loss l_{norm} (7), where $\langle n_{p_i}, n_{g_i} \rangle$ is the inner product of the surface normal vectors n_{g_i} and n_{p_i} computed for each depth map.

$$l_{depth} = \frac{1}{n}\sum_{i=1}^{n} e_i \tag{5}$$

$$l_{grad} = \frac{1}{n}\sum_{i=1}^{n} (\nabla_x(e_i) + \nabla_y(e_i)) \tag{6}$$

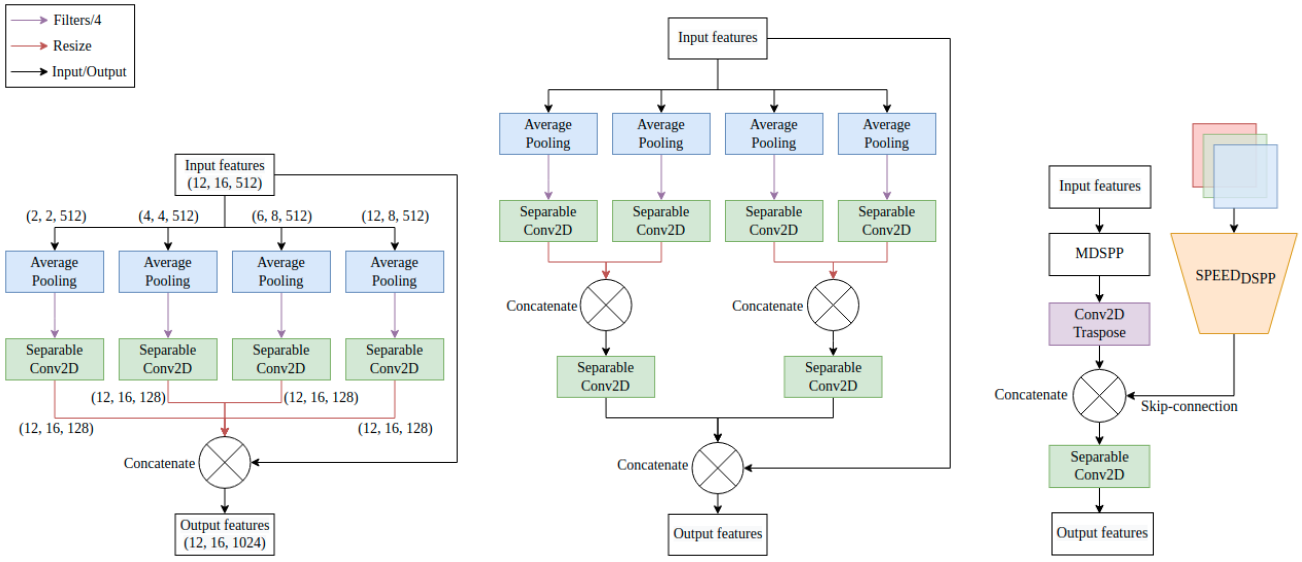
$$l_{norm} = \frac{1}{n}\sum_{i=1}^{n} \left(1 - \frac{\langle n_{p_i}, n_{g_i} \rangle}{\sqrt{\langle n_{p_i}, n_{p_i} \rangle}\sqrt{\langle n_{g_i}, n_{g_i} \rangle}}\right) \tag{7}$$

Once the training phase is completed, the model is converted to TensorFlow Lite[7] (we call this version of the model SPEED$_{lite}$ thereafter). We use as benchmark hardware the Google Cloud TPU-v2[8] equipped with 8 cores, 8GB of memory per-core, and the Google server Intel Xeon CPU[9] @2.20GHz (single core). Moreover, two edge processing units are compared: the 4GB NVIDIA Jetson TX1, an ARM-powered device equipped with a 256-core NVIDIA Maxwell GPU,[10] and the 4GB Google Coral Dev Board, an ARM-powered device with an on-board Edge TPU[11] coprocessor. Finally, the inference frequencies reported for each hardware in the Chapter V are computed as average frame-per-seconds (fps) of the model inference frequency.

V. RESULTS

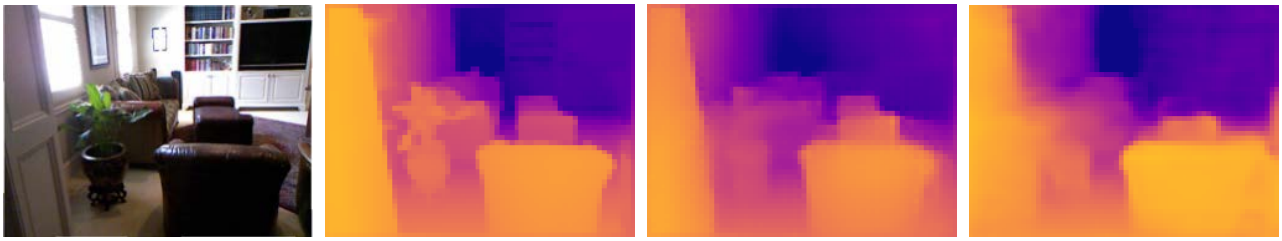
In this section, we report the SPEED results on two benchmark datasets, the NYU Depth v2 and the DIML Kinect v2, described in Section IV. We provide in Section V-A

[6] https://www.tensorflow.org/
[7] https://www.tensorflow.org/lite
[8] https://cloud.google.com/tpu/docs/system-architecture-tpu-vm
[9] https://www.intel.it/content/www/it/it/products/details/processors.html
[10] https://developer.nvidia.com/maxwell-compute-architecture
[11] https://cloud.google.com/edge-tpu



(a) **DSPP layer** (b) **MDSPP layer** (c) **up-sampling block**

FIGURE 3. **A representation of the depthwise separable pyramidal pooling (DSPP) layer (3(a)) with respective shapes (H, W, C), the mixed depthwise separable pyramidal pooling (MDSPP) layer (3(b)), and of the up-sampling (UpS) block (3(c)). The values of the input-output shapes for (3(b)) and (3(c)) change every time the MDSPP layer and the UpS block are applied in the SPEED decoder. Please refer to the legend to understand their general behaviour.**



(a) **RGB Input** (b) **Ground truth** (c) **SPEED prediction** (d) **FastDepth [6] prediction**

FIGURE 4. **Qualitative comparison between SPEED and FastDepth [6]. The depth maps are all resized to the same image resolution and converted in RGB format with a perceptually uniform colormap (Plasma-reversed) extracted from the ground truth, for a better view.**

a comparison with state of the art algorithms in terms of the metrics described in Section IV-B. In Section V-B is shown a detailed analysis of SPEED over different frameworks and edge devices. Finally, in the ablation studies (Sections V-C, V-D, V-E) we analyze the individual contribution of the encoder and the decoder model on the overall architecture performances together with the performance behaviour at different resolutions.

A. COMPARISON WITH STATE OF THE ART METHODS

In this section, SPEED is compared with different methods working in real-time and low-resource settings as [6], [24], and with deeper but slower methodologies, as [16], [19], [41]. Table 2 shows the results on the NYU Depth v2 dataset while in Table 3 a comparison is given with respect to the DIML Kinect v2 dataset. In each Table, the reported values for the evaluation metrics are extracted from the respective papers, with the exception of FastDepth values in Table 3 since this method has not been originally tested on the DIML Kinect v2 dataset, thus requiring a training from scratch. It can be noticed that SPEED achieves state of the art results w.r.t. fast-throughput related works, such as [6] and [24] but also remarkable results in the error estimation w.r.t. more complex models as [41].

TABLE 2. **Comparison with prior works on the NYU Depth v2 dataset. The best scores are in bold and second best are underlined, - when the value is not reported in the original paper.**

Method	RMSE↓ [m]	REL↓	δ_1 ↑
AdaBins [19]	**0.465**	**0.126**	**0.849**
DeepLabV3 (F10) [41]	0.575	0.162	0.772
ST-CLSTM [16]	1.884	0.239	0.159
CReaM [24]	0.687	0.190	0.704
FastDepth [6]	0.599	-	0.775
SPEED	<u>0.566</u>	<u>0.158</u>	<u>0.783</u>

TABLE 3. Comparison with prior works on the DIML Kinect v2.

Method	RMSE↓ [m]	REL↓	δ_1 ↑
Y. Kim et al. [39]	0.551	0.226	-
FastDepth [6]	0.376	0.142	0.766
SPEED	**0.368**	**0.123**	**0.818**

For example, if compared against FastDepth with the best RMSE, SPEED shows a decrement in the RMSE of 5.8% (see Table 2); furthermore, as can be seen from Figure 4, the SPEED prediction (4(c)) is visually closer to the ground truth (4(b)) w.r.t. FastDepth (4(d)) estimation. Differently, AdaBins [19] specifically designed to optimize the estimation accuracy obtains state of the art results at the expense of the inference frequency (less than 1 fps as evidenced in Table 4). The spatial-temporal convolutional long short-term memory (ST-CLSTM) structure proposed by [16] shows unsatisfied results on the single image depth estimation task.

In the case of DIML Kinect v2 dataset we compare SPEED with the fast-throughput architecture which better performed on NYU Depth v2, FastDepth, and with the method proposed by the dataset authors [39]. As shown in Table 3 the error reduction of SPEED with respect to Y. Kim *et al.* [39] is around 45% on the RMSE and 83% on REL, underlying the power of our approach on a different set of data. At the same time an evident improvement is noticeable also for FastDepth, with an error reduction of 2% (RMSE), 15% (REL) and 7% (δ_1).

As previously said, most of the techniques developed for MDE are mainly focused on optimizing the estimation accuracy without considering other important aspects as the inference frequency. In Table 4 we compare the frame rate and the number of parameters of SPEED w.r.t. FastDepth [6] and two computationally intensive methods: a deep CNN [16] and a ViT [19]. We compare the chosen pretrained models running on the low-resource benchmark hardware. The accuracy performances of ViT [19] comes at the expense of a low frame rate, with a value lower than 0.1 fps on the cloud CPU and almost equal to 1 fps on the cloud TPU. Our method is instead able to improve the inference frequency by almost 540 times on the CPU and more than 30 times on the TPU w.r.t. [19] (i.e. achieving 0.05 fps and $\sim$ 1 fps respectively), with an increase of 0.1 meters on the RMSE. Furthermore, SPEED produces an improvement on the two benchmark hardware w.r.t. FastDepth in terms of fps equal to 385% on the CPU and 334% on the TPU.

Regarding the number of trainable parameters, SPEED and FastDepth are one order of magnitude smaller than the other methods, which are indeed sensibly slower.

B. FRAMEWORKS AND EDGE DEVICES

In this section, we first report an overview of the performance degradation introduced when a trained model is converted in TensorFlow Lite format (keeping the same resolution data type). This operation is mandatory for some edge devices, as for example Edge TPUs. Secondly, we report the inference frequency on the proposed edge devices.

TABLE 4. Inference frequency and number of parameters comparison with related works. The best result is reported in bold.

Method	ST-CLSTM [16]	AdaBins [19]	FastDepth [6]	**SPEED**
CPU↑ [fps]	< 1	< 1	7	**27**
TPU↑ [fps]	$\sim$ 1	$\sim$ 1	9	**30**
Param. [M]	15.1	77.8	3.9	2.6

Table 5, in the last two columns shows the comparison between the frame rate obtained by our model, prior and after the conversion to TensorFlow Lite (SPEED$_{lite}$), on both cloud CPU and TPU. Results show also that the accuracy degradation introduced by Tensorflow Lite conversion is considerably small (2% only on the RMSE), while the frame rate is boosted by 11% on the CPU and 17% on the TPU.

TABLE 5. SPEED performances over different frameworks.

Method	RMSE↓ [m]	REL↓	δ_1 ↑	CPU↑ [fps]	TPU↑ [fps]
SPEED	0.566	0.158	0.783	27	30
SPEED$_{lite}$	0.578	0.158	0.783	30	35

Subsequently, in Table 6 we report a comparison of the frame rates over the two considered edge devices: the NVIDIA Jetson TX1 and the Google Dev Board Edge TPU; as can be noticed, SPEED is able to perform real-time frequency performances also on the edge NVIDIA Jetson TX1. The second edge device tested is the Google Coral Dev Board; to run the proposed architecture on low-resource Edge TPU, we need to perform computations with 8-bit network weights and activations, which means undergoing a quantization process. Such operation will notably reduce the precision data type of the architecture from 32-bit floating point to 8-bit integer. Nevertheless, the estimation accuracy of the model is still acceptable, obtaining an increase in the RMSE, REL and δ_1 less than 15%, 32%, and 18% respectively w.r.t. SPEED$_{lite}$ without quantization.

However, looking at the performance on the edge devices of FastDepth, the obtained results are respectively 28 fps and 2.1 fps for the TX1 GPU and the Edge TPU. We can therefore note that SPEED is able to infer three times faster on the Edge TPU, while producing similar performances on the low-power GPU (i.e. with an improvement equal to 11%). Finally, as additional experiment, we test these models on the NVIDIA Jetson TX1 CPU, i.e. excluding the GPU. The obtained inference frequencies for SPEED and FastDepth are respectively 0.4 fps and 0.2 fps; although the frequency values are not acceptable, the proposed model proves to be still two times faster.

TABLE 6. Comparison of SPEED over edge GPU and TPU equipped devices.

Device	Hardware	Quantization	Frame rate↑ [fps]
NVIDIA Jetson TX1	GPU	No	31
Google Dev Board	TPU	Yes	6

TABLE 7. Comparison of the different encoders. The best results are in bold, and the second-best are underlined. N° indicates the number of layers of each architecture.

Encoder	RMSE↓ [m]	REL↓	δ_1 ↑	CPU↑ [fps]	TPU↑ [fps]	N°
M.Net [5]	0.829	0.242	0.589	22	24	86
M.NetV2 [43]	0.807	0.234	0.611	20	23	154
M.NetV3 [44]	1.196	0.329	0.393	**59**	**65**	243
E.NetB0 [40]	0.852	0.251	0.584	9	11	237
SPEED$_{Plite}$	**0.557**	**0.154**	**0.783**	24	30	75
SPEED$_{lite}$	0.578	0.158	0.784	30	35	75

C. ABLATION STUDY: ENCODERS ARCHITECTURES

In the light of previously mentioned results in Table 5, due to the trade-off between estimation accuracy and inference frequency, we focus on the SPEED$_{lite}$ architecture in order to perform an ablation analysis using different spatial pyramidal pooling layers and encoders.

We tested different lightweight backbones as EfficientNet B0 [40], the MobileNet [5] and its later versions [43], [44]. We also report a comparison between SPEED with spatial pyramidal pooling layer (SPEED$_{Plite}$) with the proposed DSPP (SPEED$_{lite}$). Some considerations can be inferred from the results reported in Table 7:

- Despite being shallower than all the other models, SPEED$_{lite}$, in both the setups, with its custom backbones provides the lower estimation error.
- Comparing SPEED$_{lite}$ with the original MobileNet encoder architecture, we see a decrease of RMSE equal to 30% and an increase of inference frame rates equal to 27% on the CPU and 31% on the TPU. This is due to the use of DSPP layer instead of the original Depthwise-blocks.
- EfficientNet B0, used as encoder, provides similar accuracy w.r.t. the MobileNet model, while reporting a sensible reduction of the inference frame rate.
- MobileNet v3 produces the fastest frame rate. However, the performances on the other metrics are not good enough to consider the architecture reliable for depth estimations.

Finally, from Table 7 we observed that the two pyramidal pooling layers produce a similar error in the considered metrics; however, DSPP inference frequency is higher w.r.t. the non separable variant, with a boost of 20%. Furthermore, the accuracy degradation introduced by the DSPP layer is almost neglectable, less than 4%; these results emphasize that the DSPP is the optimal choice for the proposed task.

TABLE 8. Comparison between decoder architectures keeping the SPEED encoder fixed.

Decoder Type	RMSE↓ [m]	REL↓	δ_1 ↑	CPU↑ [fps]	TPU↑ [fps]
U-Net [42]	**0.569**	**0.156**	**0.782**	27	33
NNConv5 [6]	0.779	0.225	0.632	28	34
SPEED$_{lite}$	0.578	0.158	0.783	**30**	**35**

TABLE 9. Comparison SPEED$_{lite}$ with different input-output resolutions.

Input Size	Output Size	RMSE↓ [m]	REL↓	δ_1 ↑	TPU↑ [fps]
480×640	240×320	0.588	0.163	0.770	4
192×256	98×128	**0.561**	**0.150**	**0.799**	21
192 × 256	**48 × 64**	0.578	0.158	0.783	35
96×128	96×128	0.636	0.167	0.763	49
96×128	48×64	0.635	0.171	0.748	**63**

D. ABLATION STUDY: DECODERS ARCHITECTURES

In this subsection we test the performances of our proposed decoder w.r.t. the well known U-Net [42] architecture, and the NNConv5 decoder proposed by [6] while keeping fixed the SPEED$_{lite}$ encoder.

Table 8 shows that while U-Net and NNConv5 decoders almost reach real-time frequencies, SPEED$_{lite}$ with its custom decoder is able to improve the frame rate w.r.t. the U-Net of respectively 11% on the cloud CPU and 6% on the cloud TPU. Specifically, SPEED is able to produce stable real-time frequencies with an RMSE increment of only 2%.

Moreover, the NNConv5 model, despite being specifically developed as a MobileNet decoder, is not able to achieve a good final estimation, producing an increase of the RMSE equal to 35% w.r.t. our proposed architecture.

E. ABLATION STUDY: INPUT-OUTPUT RESOLUTIONS

Determining the optimal input-output resolution is a non trivial task since both accuracy and inference frequency are affected by the image input-output size.

For this reason, we report a detailed study over the different possible input-output resolution pairs, to determine the optimal trade-off. As can be noticed by Table 9, the input size has the greater impact on the performances, while the results are almost invariant to the output resolution. More in detail, if we consider the same output shape, a sensible increase of RMSE (12.5%) is obtained when changing the input shape from 192×256 to 96×128. Vice versa, the error is relatively unchanged when switching the output resolution from 98×128 to 48×64. For this reason, a input-output resolution of 192×256 and 48×64 respectively is chosen for our model, as it makes SPEED$_{lite}$ fast and lightweight as possible assuring good accuracy performances and stable real-time inference frequencies.

VI. CONCLUSION

In this work we propose SPEED, a model able to tackle the fast-throughput MDE task while improving accuracy

performances with respect to the state of the art related works. Our model achieves high frame rates on different resource hardwares by exploiting an efficient lightweight encoder and a specifically designed decoder, using two custom pyramidal pooling layers. With only 2.6M trainable parameters, SPEED is able to improve the prediction estimation of fast-throughput MDE while guaranteeing real-time performances on the chosen benchmark hardware such as the Cloud TPU-v2 and the Intel CPU but also on edge devices such as the NVIDIA Jetson TX1.

The obtained results demonstrate that our proposed method could be useful not only in many different computer vision tasks that requires the combination of RGB images with the depth information but also in autonomous system applications. In fact in those tasks, as robot indoor navigation, indoor surveillance and IoT systems, the computational power of the embedded devices is a de facto bottleneck and depth information is a crucial feature to perceive the surrounding environment and to estimate the system own state. Finally, the real-time performances obtained by SPEED could be a valuable starting point for future studies: analyzing the impact of the proposed layers on different architectures and embedded devices, testing different indoor and outdoor datasets with higher depth range and investigating less explored research areas, i.e. transferability and security.

AKNOWLEDGMENT

The authors would like to thank Fabiana Di Ciaccio, University La Parthenope, Naples, Italy, for the extensive editing performed on the text. They would also like to thank ALCOR Laboratory for having made available the workstations for the tests.

REFERENCES

[1] R. Xiaogang, Y. Wenjing, H. Jing, G. Peiyuan, and G. Wei, "Monocular depth estimation based on deep learning: A survey," in *Proc. Chin. Automat. Congr. (CAC)*, Nov. 2020, pp. 2436–2440, doi: 10.1109/CAC51589.2020.9327548.

[2] S. Ullman, *The Interpretation of Structure From Motion*. Cambridge, MA, USA: MIT Press, 1979, pp. 133–175.

[3] J. Furmonas, J. Liobe, and V. Barzdenas, "Analytical review of event-based camera depth estimation methods and systems," *Sensors*, vol. 22, no. 3, p. 1201, Feb. 2022, doi: 10.3390/s22031201.

[4] 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 16 × 16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[5] 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, *arXiv:1704.04861*.

[6] 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. Automat. (ICRA)*, May 2019, pp. 6101–6108, doi: 10.1109/ICRA.2019.8794182.

[7] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "Towards real-time unsupervised monocular depth estimation on CPU," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 5848–5854, doi: 10.1109/IROS.2018.8593814.

[8] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Computer Vision—ECCV* (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.

[9] 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.

[10] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/j.neunet.2014.09.003.

[11] A. Palla, D. Moloney, and L. Fanucci, "Fully convolutional denoising autoencoder for 3D scene reconstruction from a single depth image," in *Proc. 4th Int. Conf. Syst. Informat. (ICSAI)*, Nov. 2017, pp. 566–575, doi: 10.1109/ICSAI.2017.8248355.

[12] L. Hoyer, D. Dai, Y. Chen, A. Köring, S. Saha, and L. Van Gool, "Three ways to improve semantic segmentation with self-supervised depth estimation," 2020, *arXiv:2012.10782*.

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

[14] Y. Zhu and S. Newsam, "DenseNet for dense flow," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 790–794, doi: 10.1109/ICIP.2017.8296389.

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

[16] H. Zhang, Y. Li, Y. Cao, Y. Liu, C. Shen, and Y. Yan, "Exploiting temporal consistency for real-time video depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1725–1734, doi: 10.1109/ICCV.2019.00181.

[17] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," 2016, *arXiv:1604.03901*.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.

[19] S. F. Bhat, I. Alhashim, and P. Wonka, "AdaBins: Depth estimation using adaptive bins," 2020, *arXiv:2011.14141*.

[20] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," 2021, *arXiv:2103.13413*.

[21] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2002–2011, doi: 10.1109/CVPR.2018.00214.

[22] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp. 1623–1637, Mar. 2022, doi: 10.1109/TPAMI.2020.3019967.

[23] I. Andrey, M. Grigory, P. David, S. Samarth, T. Radu, Z. Ziyu, W. Yicheng, H. Zilong, L. Guozhong, Y. Gang, F. Bin, W. Yiran, L. Xingyi, S. Min, X. Ke, C. Zhi-Guo, D. Jin-Hua, W. Pei-Lin, G. Chao, and B. Fausto, "Fast and accurate single-image depth estimation on mobile devices, mobile AI 2021 challenge: Report," Tech. Rep., 2021, pp. 2547–2557, doi: 10.1109/CVPRW53098.2021.00288.

[24] A. Spek, T. Dharmasiri, and T. Drummond, "CReaM: Condensed real-time models for depth prediction using convolutional neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 540–547, doi: 10.1109/IROS.2018.8594243.

[25] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," 2018, *arXiv:1806.01260*.

[26] J. Liu, Q. Li, R. Cao, W. Tang, and G. Qiu, "MiniNet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation," 2020, *arXiv:2006.15350*.

[27] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2800–2810, doi: 10.1109/CVPR.2018.00296.

[28] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3D packing for self-supervised monocular depth estimation," 2019, *arXiv:1905.02693*.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015, doi: 10.1109/TPAMI.2015.2389824.

[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.

[31] M. Zhang, Z. Wang, T. Sun, and X. Li, "Salient object detection by pyramid networks with gating," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2019, pp. 1791–1796, doi: 10.1109/ROBIO49542.2019.8961768.

[32] J. Masci, U. Meier, G. Fricout, and J. Schmidhuber, "Multi-scale pyramidal pooling network for generic steel defect classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2013, pp. 1–8, doi: 10.1109/IJCNN.2013.6706920.
[33] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, M. Jagersand, and H. Zhang, "A comparative study of real-time semantic segmentation for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 587–597, doi: 10.1109/CVPRW.2018.00101.
[34] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," 2014, *arXiv:1406.2283*.
[35] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, 2010.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[37] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," 2019, *arXiv:1904.09237*.
[38] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, "Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries," 2018, *arXiv:1803.08673*.
[39] Y. Kim, H. Jung, D. Min, and K. Sohn, "Deep monocular depth estimation via integration of global and local predictions," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 4131–4144, Aug. 2018, doi: 10.1109/TIP.2018.2836318.
[40] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.
[41] S. Gur and L. Wolf, "Single image depth estimation trained via depth from defocus cues," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7675–7684, doi: 10.1109/CVPR.2019.00787.
[42] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," 2015, *arXiv:1505.04597*.
[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.
[44] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," 2019, *arXiv:1905.02244*.

LORENZO PAPA received the B.S. degree in computer and system engineering and the M.S. degree in artificial intelligence and robotics from the Sapienza University of Rome, Italy, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree in computer science engineering that collaborates with the ALCOR Laboratory, DIAG Department. His main research interests include deep learning, computer vision, and cyber security.

EDOARDO ALATI received the M.Sc. degree in AI and robotics engineering from the Sapienza University of Rome, Italy, where he is currently pursuing the Ph.D. degree in computer science and AI engineering. He worked for three years in the European Project Second Hands (Horizon 2020), while applying his knowledge in ML and DL to several parallel projects, such as vertical farming, beekeeping, and COVID detection.

PAOLO RUSSO received the B.S. degree in telecommunication engineering from the Università 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 has been a Researcher at the Italian Institute of Technology (IIT), Turin, Italy. He is currently an Assistant Researcher at the ALCOR Laboratory, 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 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 was a Visiting Scholar with Binghamton University, Binghamton, NY, USA, in 2010, and a Visiting Research Fellow with Charles Sturt University, Australia, in 2018, with a fellowship offered by the Australian Government Department of Education and Training, through the Endeavour Scholarship and Fellowship Program. She is currently an Assistant 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, multimedia content security technologies, secure media, 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. She 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*.