Real-time monocular depth estimation on embedded devices: challenges and performances in terrestrial and underwater scenarios

Lorenzo Papa, Paolo Russo, Irene Amerini
Sapienza University of Rome, Italy
Emails: [papa, paolo.russo, amerini]@diag.uniroma1.it

***Abstract*—The knowledge of the environmental depth is essential in multiple robotics and computer vision tasks for both terrestrial and underwater scenarios. Recent works aim at enabling depth perception using single RGB images on deep architectures, such as convolutional neural networks and vision transformers, which are generally unsuitable for real-time inference on low-power embedded hardwares. Moreover, such architectures are trained to estimate depth maps mainly on terrestrial scenarios, due to the scarcity of underwater depth data. Purposely, we present two lightweight architectures based on optimized MobileNetV3 encoders an a specifically designed decoder to achieve fast inferences and accurate estimations over embedded devices, and a feasibility study to predict depth maps over underwater scenarios. Precisely, we propose the MobileNetV3$_{S75}$ configuration to infer on the 32-bit ARM CPU and the MobileNetV3$_{LMin}$ for the 8-bit Edge TPU hardwares. In underwater settings, the proposed design achieves comparable estimations with fast inference performances compared to state of the art methods. The proposed architectures would be considered a promising approach for real-time monocular depth estimation with the aim of improving the environment perception for underwater drones, lightweight robots and internet-of-things.**

***Index Terms*—depth estimation, embedded devices, real-time, underwater**

I. Introduction

In recent computer vision and deep learning (DL) trends, researchers focus their attention on achieving the highest estimation accuracy without taking into account the computational effort required to run developed models in real-world vision applications as small robots, aerial and underwater drones. In the monocular depth estimation (MDE) task, where the scene depth is estimated through a single RGB image, this tendency can be noticed in recent proposed works such as [1]–[4]. Those algorithms typically infer in the cloud or on dedicated servers without considering possible low-resource hardware constraints.

On the other side, some recent DL studies as [5], [6] are going against this paradigm, analyzing the Edge TPUs capabilities in deep learning tasks. Those embedded hardwares are characterized by low power consumption and limited memory capacity, which act as a performance bottleneck for the DL-based techniques. Regarding monocular depth estimation, only few works propose a solution for porting such complex task on low-resource platforms. There are two main approaches: [7]–[9] that focus on MDE on microcontroller and ARM-powered devices without taking into account the inference frequency, and [10]–[12] which analyse the inference performances of MDE on low-power GPUs.

Moreover, previously reported MDE methods are trained on terrestrial datasets such as [13], [14] using supervised learning strategies; due to the lack of data, some works as [15]–[18] propose a qualitative underwater depth estimations by relying on colour restoration and the correspondence between depth and visual style levels.

This work investigate the previously mentioned issues, proposing two lightweight MDE deep models that are able to achieve accurate estimations and fast inference frequencies on the benchmark embedded devices. Secondly, it conducts a feasibility study of such architectures in underwater scenes.

The chosen hardware components i.e. the ARM CPU and Edge TPU are widely employed in lightweight robots, such as aerial and underwater drones, and AI-accelerator platforms (e.g. Coral Dev Board). Due to the constraints introduced by the selected embedded hardwares, we propose two models for different precision data types, i.e. 32-bit floating points to infer on the ARM CPU and 8-bit integers for the Edge TPU.

The rest of the paper is organized as follows: in Section II we describe in details our proposed architectures, while in Section IV we show the obtained results both on terrestrial and underwater scenarios. In the last Section V we conclude our study with some considerations on the underwater setting and on possible future works.

II. Proposed Method

This section describes the proposed MDE architectures for achieving the best trade-off between real-time frequency performance and estimation accuracy across multiple scenarios. We design those models in compliance with two essential concepts: the *speed* to maximize the inference frequency on embedded devices, and the *robustness* to maximize the estimation accuracy across two datasets [13], [19]. Our solution exploits an encoder-decoder model, similarly to previous related works as [10], [11]. More in detail, we perform an in-depth study on lightweight encoders pre-trained on ImageNet [20] to improve their generalization capabilities. A graphical overview of the proposed architectures is reported in Figure 1. We choose MobileNetV3 [21] as the baseline encoder for further

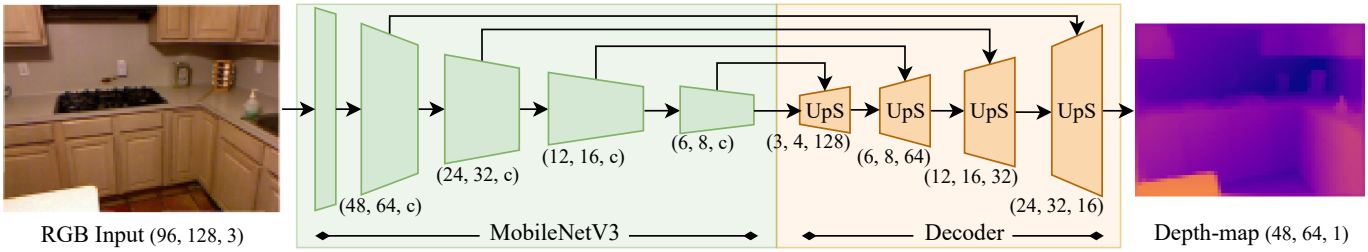


Fig. 1: Graphical overview with respective input shapes of the proposed encoder-decoder structure. The number of channels (c) are respectively $[16, 16, 72, 96]$ for the MobileNetV3$_{S75}$ and $[16, 64, 72, 240]$ for the MobileNetV3$_{LMin}$.

refinement. This fully convolutional network is built on a sequence of downsampling *inverted residuals blocks*, each one characterized by a Squeeze-and-Excite [22] block followed by a pointwise and a depthwise convolution. This procedure keeps a limited number of multiply-accumulate (MAC) operations, thus leading to significant speed improvements without introducing any substantial estimation error compared to other lightweight networks [23]–[26].

To tackle the MDE task under low resource constraints, we propose two configurations of the chosen baseline encoder, respectively named MobileNetV3$_{S75}$ and MobileNetV3$_{LMin}$.

The MobileNetV3$_{S75}$ model is the *Small (S)* configuration of the MobileNetV3 [21] with a reduction of 25% of the original convolutional filters while the MobileNetV3$_{LMin}$ is based on the deeper *Large (L)* but *Minimalistic (Min)* configuration of the MobileNetV3. It is a variant of the original *Large* architecture where the Squeeze-and-Excite blocks are removed, Hard-Swish activations are replaced with ReLUs, and 5×5 convolutions are replaced with 3×3 ones. The MobileNetV3$_{S75}$ model with its shallower design is designed to run on the 32-bit ARM CPU, while the deeper MobileNetV3$_{LMin}$ architecture can be exploited on 8-bit Edge TPU device. As will be shown in Section IV the two encoders, based on different configurations of the same baseline architecture, are both able to achieve almost real-time performances and a notable estimation accuracy in their respective embedded hardware when compared with other lightweight backbones [21], [23], [25], [26].

Furthermore, to maximise the estimation accuracy and the reconstruction capabilities of the network, we design from scratch a fully convolutional decoder that generalises over the developed encoders. It is composed of a sequence of four cascaded upsampling blocks (UpS) that increase the spatial resolution while merging the encoded features through a skip-connection to reconstruct the output depth map (see Figure 1). Each of those UpS, named TDSConv2D, is composed of two 3×3 convolutional operations, a transposed and a depthwise separable one interleaved by the skip-connection used to retrieve the image details from the encoder feature maps. This design allows us to limit the number of computed operations, restricting the computational complexity while improving the overall inference frequency, as will be shown in Section IV-B.

III. Implementation Details

We train the reported architectures with the same input-output resolution (respectively 96×128 and 48×64) on the NYU Depth V2 [13] dataset with the official train-test split. We evaluate the generalization performances in underwater scenarios w.r.t. comparable state of the art related models over the 57 stereo samples of the underwater SQUID [19] dataset. This study has been performed using TensorFlow 2[1] deep learning high level API. Each compared model is initialized on ImageNet [20] pretrained weights. ADAM [27] optimizer has 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 32 and trained the models for a total of 30 epochs on the chosen dataset. Once the training phase is completed, each model is converted and evaluated in TensorFlow Lite[2], an open-source framework specifically designed to infer on embedded devices. Finally, to evaluate and compare the performance of proposed models, we considered the most commonly used metrics in MDE: the root-mean-square error (RMSE), the relative error (REL), and the accuracy value δ_1. Their formal definition are reported in Equations 1, 2 and 3, where p_i and g_i are respectively the predicted depth map and its ground truth while P denotes the total number of the evaluated 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.

Moreover, we compare the inference performances measured in frame-per-second (fps) using as edge processing unit the 4GB Google Coral Dev Board[3], a cheap and low-power embedded device equipped with a 32-bit ARM Cortex CPU and an 8-bit Edge TPU.

[1]https://www.tensorflow.org/
[2]https://www.tensorflow.org/lite
[3]https://coral.ai/products/dev-board/

IV. RESULTS

This section compares the proposed architectures and the assessed encoders and decoders over different precision data types. The main results are obtained on the terrestrial dataset, while in the feasibility study we analyze the models' performances in underwater settings.

We report in Section IV-A and IV-B the individual contributions of the proposed encoders and the decoder, described in Section II, while in Section IV-C we analyse the accuracy and inference performances changing the input-output image resolution, and in Section IV-D we conduct the feasibility study to estimate depth maps over the underwater scenario.

A. Encoders

The first performed analysis is focused on a comparison of various lightweight pretrained architectures such as [21], [23]–[26] in order to choose the best encoder for our task. The performances of those encoders with the proposed decoder structure are reported in Table I; as can be noticed, the *Small* variant of the MobileNetV3 (Mob.NetV3$_S$) obtains the highest frequency performance (close to 30fps) on the CPU.

TABLE I: Performances comparison of lightweight pretrained encoders (32-bit float) with the TDSConv2D as upsampling block. The best results are in bold and the second best are underlined.

Method	RMSE↓ [dm]	REL↓	δ_1 ↑	CPU↑ [fps]
Eff.NetB0 [25]	6.01	0.179	0.728	4.4
NasNetMob. [26]	8.70	0.276	0.539	6.5
Mob.NetV1 [23]	**5.70**	**0.165**	**0.760**	8.8
Mob.NetV2 [24]	<u>5.72</u>	<u>0.169</u>	<u>0.759</u>	11.3
Mob.NetV3$_S$ [21]	6.77	0.207	0.682	**26.2**
Mob.NetV3$_L$ [21]	6.39	0.195	0.698	<u>13.4</u>

Based on the reported results, with the goal of achieving real-time inference performances, we choose as baseline encoders the *Small (S)* and *Large (L)* configurations of the MobileNetV3 (the last two rows of Table I). Compared with the more accurate MobileNetV1, those models show a boost of 153% and 300% on the inference frequency at the expense of a slight increase of the RMSE equal to 12% and 17% respectively.

As a second step, we focus on the optimization of the two baseline encoders adopting different configurations. A graphical comparison between the frame rates (fps) and the estimation error (RMSE) of the different configurations is reported in Figure 2; we use 30 fps as reference value to identify the target (real-time) inference frequency as in previous works [11], [12].

Precisely, we compare multiple setups of the shallower MobileNetV3$_S$ and deeper MobileNetV3$_L$ configurations with the following modifications: reducing the number of convolutional filters by a specific percentage value, i.e. 50%, 75%, 200% on the encoder and 50% on the decoder, while fixing their size to 3×3 in the *Min* variant, instead of 5×5 used in the original model. We compare all the considered architectures in Figure 2. The names used for the different models are composed as follows: the name of the baseline backbone (Mob.NetV3) followed by the subscript S or L and the type of optimization applied. For example, the MobileNetV3$_{SHD}$ is the MobileNetV3 in the S configuration with a reduction of 50% of the original convolution filters, whereas the MobileNetV3$_{LMin75}$ is the L and minimalistic (Min) MobileNetV3 configuration with a reduction of 75% of the convolution filters. From this analysis, we identify two best models named Mob.NetV3$_{S75}$ (orange-dot in Figure 2a) and Mob.NetV3$_{LMin}$ (light blue-dot in Figure 2b) respectively for the ARM CPU and the Edge TPU.

The selected networks are able to achieve near real-time performances on both hardwares, i.e. 29.6 fps on the CPU and 26.8 fps on the TPU, and an RMSE of 6.86 and 11.54 decimeters, respectively. Moreover, the increased depth estimation error obtained for the MobileNetV3$_{LMin}$ on the Edge TPU is justified by the quantization operation, which compresses the model from 32-bit floating points to 8-bit integers. Finally, a complete overview of the results achieved by the two proposed models is reported in Table II while using the TDSConv2D decoder.

TABLE II: Quantitative evaluation of the proposed models, the 32-bit floating point and the 8-bit integer precision, inferring on the ARM CPU and the Edge TPU.

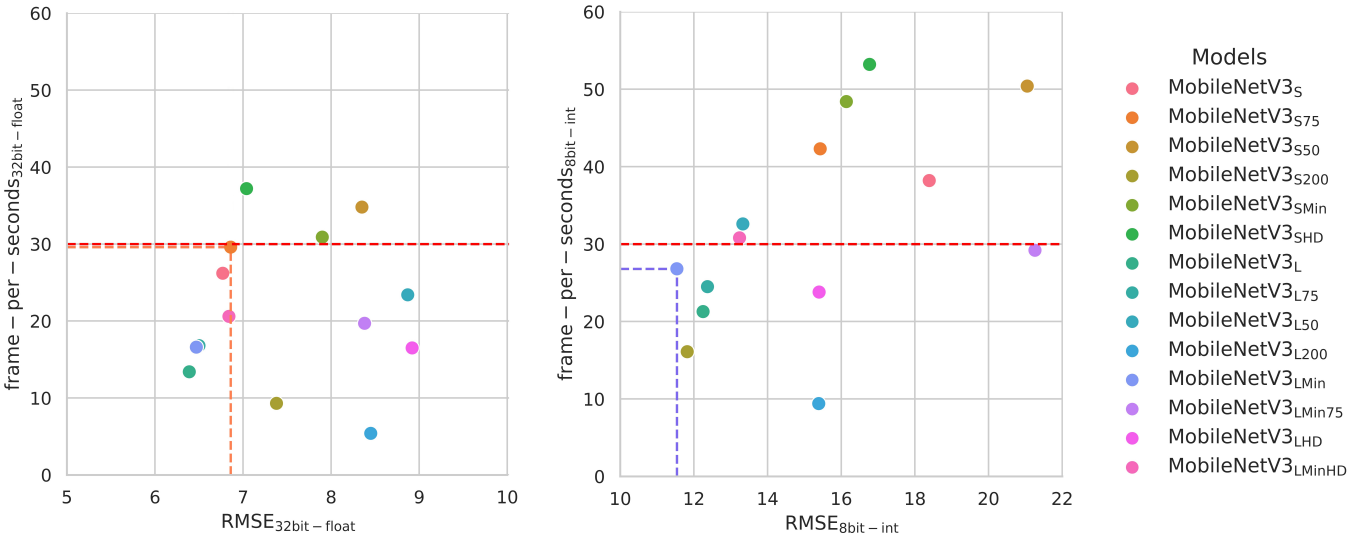
Method	Type	RMSE↓ [dm]	REL↓	δ_1 ↑
Mob.NetV3$_{S75}$	32-bit	6.86	0.209	0.666
Mob.NetV3$_{LMin}$	8-bit	11.54	0.429	0.412

B. Decoder

Once the encoder configuration has been determined, we compare five different UpS blocks. The upsampling operation has the fundamental role of progressively increasing the features resolution learned by the encoder while reconstructing the image details to obtain the final output depth map.
The detailed characteristics of each UpS are below reported:

- UpConv2D: a 2×2 upsampling layer followed by a 3×3 convolution.
- UpDSConv2D: a 2×2 upsampling layer followed by a 3×3 depthwise separable convolution.
- NNConv5 [11]: a 5×5 convolution followed by a 2×2 upsampling with nearest-neighbor interpolation.
- TConv2D: a 3×3 transposed convolution followed by a 3×3 convolution.
- TDSConv2D: a 3×3 transposed convolution followed by a 3×3 depthwise separable convolution.

The quantitative comparison of the evaluated UpS over the respective encoder architecture and precision data type is reported in Table III. From the obtained results, we determine the TDSConv2D as the best upsampling block since it is able to guarantee almost real-time frequency performances over both 32-bit floating point and 8-bit integer architectures while



(a) The evaluated models for ARM CPU, 32-bit floating point precision. In orange the best configuration, named MobileNetV3$_{S75}$ (orange-dot).

(b) The evaluated models for Edge TPU, 8-bit integer precision. In light blue the best configuration, named MobileNetV3$_{LMin}$ (light blue-dot).

Fig. 2: Graphical comparison between different MobileNetV3 configurations. In each graph the RMSE, in different ranges due to the respective error distribution, and the frame-per-seconds (fps) are reported. The red dotted line is used to represent the real-time frame rate i.e. 30 fps, while the colored dotted segments, respectively in orange and light blue, are used to identify the best models.

TABLE III: Comparison of different decoders with the proposed encoders.

Upsampling block	MobileNetV3$_{S75}$ (32-bit float)				MobileNetV3$_{LMin}$ (8-bit int)			
	RMSE↓ [dm]	REL↓	δ_1 ↑	CPU↑ [fps]	RMSE↓ [dm]	REL↓	δ_1 ↑	TPU↑ [fps]
UpConv2D	6.95	0.211	0.664	15.2	12.27	**0.360**	0.376	16.9
UpDSConv2D	7.02	0.212	0.660	**31.1**	13.55	0.399	0.283	**28.3**
NNConv5 [11]	7.69	0.236	0.608	6.8	21.86	0.654	0.030	9.5
TConv2D	6.88	**0.204**	**0.669**	18.1	17.63	0.836	0.218	17.5
TDSConv2D	**6.86**	0.209	0.666	29.6	**11.54**	0.429	**0.412**	26.8

TABLE IV: Comparison of the proposed encoder-decoder models with different input-output resolutions.

Resolutions		MobileNetV3$_{S75}$ (32-bit float)				MobileNetV3$_{LMin}$ (8-bit int)			
Input	Output	RMSE↓ [dm]	REL↓	δ_1 ↑	CPU↑ [fps]	RMSE↓ [dm]	REL↓	δ_1 ↑	TPU↑ [fps]
192 × 256	192 × 256	**6.38**	**0.194**	**0.687**	5.8	20.19	0.961	0.051	4.5
192 × 256	96 × 128	7.09	0.215	0.652	9.6	12.85	0.363	0.342	6.6
192 × 256	48 × 64	7.01	0.211	0.654	13.9	12.32	**0.314**	0.349	7.9
96 × 128	96 × 128	8.07	0.244	0.571	20.3	11.56	0.413	0.407	17.3
96 × 128	48 × 64	6.86	0.209	0.666	**29.6**	**11.54**	0.429	**0.412**	**26.8**

showing the lowest RMSE. Moreover, two important aspects can be inferred from Table III:

- The use of depthwise separable convolutions guarantees a boost of the frame rate in a range of 150% up to 200% w.r.t. classical convolutions.
- The transposed convolution layers achieve comparable speed w.r.t. upsampling operations (with a difference less than 2 fps) with an average RMSE improvement of almost 3%.

C. Input-Output resolution

In this subsection, we analyse the behaviour of the network while providing different input-output resolution pairs. We report in Table IV the quantitative results where is evidenced that both input and output dimensions remarkably affect the inference frequency on embedded devices. First of all, the input resolution has a stronger effect w.r.t. the output. This aspect can be recognized by observing, for example, the inference frequency boost (212% on the CPU and 339% on the TPU) while reducing the input resolution from 192 × 256

pixels to 96×128 pixels keeping fixed the output resolution to 48×64. Vice versa, the resolution reduction in relation to the output depth map leads to a less noticeable inference frequency boost, of 165% on average on the CPU and 146% on the TPU. Moreover, the final RMSE is not remarkably affected by the input-output image resolutions, with a maximum difference equal to 1.7 decimeters on the CPU and 8.6 on the TPU. Finally, from the reported analysis, we can conclude that the 96×128 input resolution and the 48×64 output resolution are the best trade-off between speed and estimation error and thus are chosen for the proposed architectures.

D. Feasibility study in underwater settings

In this section, we perform a feasibility study to understand the behaviour of such models in the estimation of underwater depth maps. Due to the lack of underwater labelled depth data publicly available for research, we compare state of the art MDE methods (working at 32-bit precision) pretrained on the terrestrial NYU Depth V2 dataset while evaluating their generalization performances by testing on the underwater dataset [19]. Other depth estimation methods designed for underwater environments as [15]–[18] are not taken into account since their main task is the colour restoration process without providing quantitative measurements of depth estimation error. We apply [28] as a pre-processing operation to the SQUID images in order to fix colour imbalance and recover their contrasts. Moreover, due to the different input-output image resolutions among the compared methods, we resize their predicted depth maps at the same 48×64 output size. The obtained results are reported in Table V. As can be seen, despite the high estimation error achieved by all the methods w.r.t. the terrestrial dataset, the proposed MobileNetV3$_{S75}$ with only $1.1M$ of training parameters outperforms both [11] and [4] while obtaining the same RMSE and an higher δ_1 to [10]. In addition, the proposed model achieves a boost on the inference frequency equal to $\times 4.7$ w.r.t. [10] on the same benchmark hardware.

We assume that the underwater performances are caused by a sensible statistical gap between the terrestrial training set and the underwater test set; we expect better results when a dedicated underwater dataset equivalent to the terrestrial one becomes available.

TABLE V: Generalization capability comparison over the SQUID dataset [19].

Method	RMSE↓ [m]	REL↓	$\delta_1 \uparrow$	CPU↑ [fps]	Parameters [M]
DenseDetpth [4]	5.23	5.275	0.047	< 1	42.6
FastDepth [11]	<u>5.17</u>	5.493	0.055	2.0	3.9
SPEED [10]	**4.49**	**4.732**	<u>0.088</u>	<u>6.2</u>	2.6
Mob.NetV3$_{S75}$	**4.49**	<u>4.956</u>	**0.089**	**29.6**	1.1

V. Conclusions

In this work, we propose two variants of the MobileNetV3 encoder and a specifically designed decoder to tackle the MDE task in order to achieve real-time frequency performances on the embedded ARM CPUs and Edge TPUs. The method performances are tested on the terrestrial NYU Depth V2 and underwater SQUID datasets. The obtained results demonstrate that the MobileNetV3$_{S75}$ and the MobileNetV3$_{LMin}$ models can be effectively considered as a solid candidates for the MDE task on the benchmark hardwares, guaranteeing real-time frequency performances at the cost of a small increase of the estimation error on the terrestrial dataset. On the other side, the results in the underwater scenario show that in this peculiar environment, further studies and data are required in order to get reasonable monocular depth estimations. Despite the high error, our models are still producing better or on-par results w.r.t. the compared works with a sensible increase of inference speed.

Those findings will be a valuable baseline for future studies and advancements in the MDE field across embedded and mobile hardwares. Further research will also be focused on embedded GPUs, architectural and data changes required to get a robust and reliable depth estimation in underwater settings.

Acknowledgments

This work was partially supported by The Social Museum and Smart Tourism, MIUR project and 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] Z. Li, Z. Chen, X. Liu, and J. Jiang, “Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation,” 03 2022.

[2] R. Ranftl, A. Bochkovskiy, and V. Koltun, “Vision transformers for dense prediction,” 2021.

[3] S. F. Bhat, I. Alhashim, and P. Wonka, “Adabins: Depth estimation using adaptive bins,” 2020.

[4] I. Alhashim and P. Wonka, “High quality monocular depth estimation via transfer learning,” 2019.

[5] A. M. Kist, “Deep learning on edge tpus,” 2021.

[6] A. Yazdanbakhsh, K. Seshadri, B. Akin, J. Laudon, and R. Narayanaswami, “An evaluation of edge tpu accelerators for convolutional neural networks,” 2021.

[7] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, and S. Mattoccia, “Enabling energy-efficient unsupervised monocular depth estimation on armv7-based platforms,” pp. 1703–1708, 2019.

[8] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, “Towards real-time unsupervised monocular depth estimation on cpu,” 2018.

[9] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, F. Aleotti, and S. Mattoccia, “Monocular depth perception on microcontrollers for edge applications,” pp. 1–1, 2021.

[10] 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.

[11] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, “Fastdepth: Fast monocular depth estimation on embedded systems,” 2019.

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

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

[14] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[15] X. Ye, Z. Li, B. Sun, Z. Wang, R. Xu, H. Li, and X. Fan, "Deep joint depth estimation and color correction from monocular underwater images based on unsupervised adaptation networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 11, pp. 3995–4008, 2020.

[16] H. Gupta and K. Mitra, "Unsupervised single image underwater depth estimation," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 624–628.

[17] Y.-T. Peng, X. Zhao, and P. C. Cosman, "Single underwater image enhancement using depth estimation based on blurriness," in *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 4952–4956.

[18] P. L. Drews, E. R. Nascimento, S. S. Botelho, and M. F. Montenegro Campos, "Underwater depth estimation and image restoration based on single images," *IEEE Computer Graphics and Applications*, vol. 36, no. 2, pp. 24–35, 2016.

[19] D. Berman, D. Levy, S. Avidan, and T. Treibitz, "Underwater single image color restoration using haze-lines and a new quantitative dataset," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 43, no. 08, pp. 2822–2837, aug 2021.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[21] 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.

[22] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2019.

[23] 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.

[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 06 2018, pp. 4510–4520.

[25] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[26] B. Zoph, V. Vasudevan, J. Shlens, and Q. Le, "Learning transferable architectures for scalable image recognition," 06 2018, pp. 8697–8710.

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

[28] Y.-T. Peng and P. C. Cosman, "Underwater image restoration based on image blurriness and light absorption," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1579–1594, 2017.