Optimize Vision Transformer architecture via efficient attention modules: a study on the monocular depth estimation task

Claudio Schiavella[0009−0008−8897−3910], Lorenzo Cirillo[0009−0008−5266−8957], Lorenzo Papa[0000−0002−9393−5248], Paolo Russo[0000−0002−1886−3491], and Irene Amerini[0000−0002−6461−1391]

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

Abstract. IoT and edge devices, capable of capturing data from their surroundings, are becoming increasingly popular. However, the onboard analysis of the acquired data is usually limited by their computational capabilities. Consequently, the most recent and accurate deep learning technologies, such as Vision Transformers (ViT) and their hybrid (hViT) versions, are typically too cumbersome to be exploited for onboard inferences. Therefore, the purpose of this work is to analyze and investigate the impact of efficient ViT methodologies applied to the monocular depth estimation (MDE) task, which computes the depth map from an RGB image. This task is a critical feature for autonomous and robotic systems in order to perceive the surrounding environment. More in detail, this work leverages innovative solutions designed to reduce the computational cost of self-attention, the fundamental element on which ViTs are based, applying this modification to METER architecture, a lightweight model designed to tackle the MDE task which can be further enhanced. The proposed efficient variants, namely Meta-METER and Pyra-METER, are capable of achieving an average speed boost of 41.4% and 34.4% respectively, over a variety of edge devices when compared with the original model, while keeping a limited degradation of the estimation capabilities when tested on the indoor NYU dataset.

Keywords: Computer vision · Edge device · Efficient vision transformer · Monocular depth estimation.

1 Introduction

The last years have seen the wide exploitation of edge devices for various applications that involve IoT and data analytics, such as robotics, autonomous systems, and augmented reality.

These devices can collect real-time data from their surroundings through connections with sensors and detectors and are able to computer onboard analysis thanks to the integrated specific hardware. One of the main application scenarios involves computer vision (CV) paradigms, which are adopted to understand

and interact with the surrounding environments. This process always involves the solution of different tasks based on the considered use case. In this work, we focus on the monocular depth estimation (MDE) task, where a dense depth map is estimated given a single input RGB image. When applied in real-world contexts, this task should often be performed in real-time. [24]. Indeed, the perception of the surroundings has to be fast but at the same time accurate and precise. These constraints are particularly true in applications like augmented reality and robotics, where the lack of these features would certainly lead to unsatisfactory and unsuitable results in terms of quality. In terms of estimation quality, ViTs and Hybrid ViTs (hViT) are able to achieve high accuracy performances at the cost of slow inference speed, which makes them challenging to utilize on edge devices [4].

However, in the current research literature, a number of methods have been proposed that aim to optimize the speed performance of vision transformers [11, 13, 20]. In addition, in the MDE task, recent works have been developed in order to design ad hoc models specifically designed to infer on embedded devices, such as [11, 13, 20]. More in detail, Papa et al. [11] propose METER, an hViT model particularly successful in achieving accurate estimation, taking advantage of the ViT global processing, while keeping fast inference performances. However, as commonly happen in ViT architectures, the self-attention [18] blocks (such as the ones used in METER architecture) exhibit a quadratic cost with respect to the input features, requiring a large amount of computation that is difficult to achieve on edge devices, which are typically low-powered, computationally constrained hardware.

Therefore, in this paper, we propose two techniques to reduce METER attention module computational cost, analyzing their application in order to define the best trade-off between inference performance and estimation capabilities. More in detail, inspired by Metaformer [22] technique and Pyramid ViT [19] technique, we introduce Meta-METER and Pyra-METER, two METER architecture modifications that are able to reduce its computational time by acting on its attention module.

The proposed solutions will take advantage of the optimization technique previously introduced while making METER network more accessible by resource-constrained devices. In this work, a special emphasis will be placed on how the inclusion of efficient variants of the attention module affects the non-optimized model. In particular, the effects of this change on both speed and accuracy will be analyzed. Summarizing, the contributions of our work are following reported:

- We propose two ways to reduce the computation time of METER architecture, namely Meta-METER and Pyra-METER.
- We perform some extensive experiments on the NYU Depth v2 dataset [17] and the inference time on the Intel Xeon CPU, Coral DevBoard [1], and NVIDIA Jetson TX1 [2].

[1] https://coral.ai/products/dev-board/
[2] https://developer.nvidia.com/embedded/jetson-tx1

The rest of the paper is organized as follows: Section 2 reports the most related works regarding our method. Section 3 describes our proposed techniques in detail. Section 4 analyzes the obtained results with the used metrics and finally Section 5 reports our conclusions and reasonings on the possible choices and the corresponding trade-offs.

2 Related Works

In this section, we report state-of-the-art related works. In particular, Section 2.1 regards depth estimation, reporting how to approach that task as well as some real applications. We first discuss the general concept of depth estimation, on which monocular depth estimation is based. In Section 2.2, we focus on the methods that have been proposed to enhance the efficiency of Vision Transformers. We also discuss the specific techniques that we use in this paper, to achieve faster inferences with METER [11]. In particular, we will focus on those novel implementations that strongly affect the attention module performances.

2.1 Depth Estimation

Depth estimation involves calculating the distance of an object from the center of the camera. Measuring the depth of a scene is a very important task with relevant practical applications, such as in robotics, augmented reality, biometrics and many others field [2, 9, 12]. Depth estimation techniques can be grouped into two main categories:

- **Active depth sensing**: depth is obtained by perturbing the environment with suitable devices (e.g. lidar).
- **Passive depth sensing**: depth is estimated from one or more RGB images of the target scene.

In our application context, the focus is on monocular depth estimation [24]. This approach is classified as a passive depth-sensing technique which, due to the single input image, cannot make use of triangulation algorithms. Recently some ViT architectures [15]-[7] tackled the monocular depth estimation task reaching high accuracy, but the computational requirement of such networks makes them unsuitable for devices with hardware constraints. On the contrary, convolutional neural network architectures [20]-[13] are able to reach some good results on embedded devices regarding the monocular depth estimation, but they don't have the same low estimation error of ViTs.

Some lightweight architectures have been introduced in order to solve the monocular depth estimation task on hardware-constrained devices. A first architecture is described in [14] called PyD-Net, with a pyramidal structure to be able to get the features of the input image by using different levels of the pyramid. Another fully convolutional network called CReaM has been introduced in [16], while the encoder-decoder FastDepth [20] can also obtain very good results in

this context. Finally, a network with high performances on embedded devices has been proposed in [23].

Another model with good performance on embedded devices is METER, a lightweight vision transformer architecture proposed by Papa et al. [11]. Its structure consists of an encoder-decoder architecture that can achieve state-of-the-art estimation and low-latency inference performance on the same embedded hardware taken in consideration in our work: NVIDIA Jetson TX1 and NVIDIA Jetson Nano. It has been proposed in three variants (S, XS, XXS) that will be the subject of our tests.

2.2 Efficient Vision Transformers

Vision Transformers are the equivalent of the transformer technology [18] (originally designed to work with text input) applied on images. The attention [18] module of a Vision Transformer is often very heavy in terms of computational time due to the quadratic complexities with respect to the input features. Therefore, some efficient ViTs have been introduced in the last years in order to reduce the computational time and also the number of operations of the attention [18] module.

Two of those novel techniques are the Softmax-free Transformer with Linear Complexity (SOFT) [8], in which the Softmax operation exploited in self-attention modules is substituted by a Gaussian kernel function to perform a similar dot-product function, and SimA [6] where the Softmax layer is replaced by a l_1-norm in order to normalize the query and key matrices. Another technique is Flowformer [21] in which the transformer is linearized by following the flow network theory and taking into consideration the conservation flows. An additional attempt has been performed in the MobileViT [10] work, which introduces a lightweight architecture for mobile devices. A further solution is proposed by Metaformer [22], that presents a revolutionary new baseline in the world of transformers. In fact, as per common thought, the role of the attention [18] module in the context of transformer architecture seems to be very important, if not the most important, for the quality of the result and the efficiency and performance of the model. However, the authors of the paper [22] showed how the latter concept may not actually be true. In a very drastic approach, the entire attention module was removed and replaced with a very simple, almost trivial, token mixer. In this way, the entire conceptual structure of the attention module is dramatically reduced. Particular focus should be given to this token mixer block, as the latter retains a very general nature, being referred to as a simple tool that can mix information, as is similarly the case with the query, values and key fundamental matrices of the transformer block that are in the standard attention module of [18].
Finally, Pyramid Vision Transformers [19] are characterized by the application of a spatial reduction function to the input. In this way, the output of the attention module is smaller and so it can be processed faster by the rest of the architecture. This is due to the so-called spatial-reduction attention layer (SRA) [19] that is substituted to the classical multi-head attention layer [18].

As Metafomers and Pyramid ViTs demonstrate to be two of the most successful method to speed up the inference phase of ViT networks, we have chosen to analyze both of them in order to study their application to a depth estimation architecture.

3 Proposed Method

This section outlines how METER architecture [11] is modified to improve its time performances. In particular, in Section 3.1 and 3.2 we describe the proposed optimizations and the new METER structures, where the attention modules of the transformer blocks will be modified as the methods shown in Section 2.

More in detail, we will focus on the traditional self-attention mechanism described in [18]:

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_{head}}})V \tag{1}$$

where d_{head} is the dimension of each head, Q is the query, K is the key, V is the value and $Softmax$ is the well-known activation function.

The choice to optimize METER's attention module derives mainly from the fact that its computational cost, being a classical attention mechanism, is quadratic with respect to the input. Optimizations proposed in the literature such as [19] and [22], as seen in 2.2, are some of the most promising solutions to deal with this issue.

In Figure 1 we can see how the traditional attention module is changed with Meta-METER and Pyra-METER modules, with a detailed description of the modifications reported in the following sections.

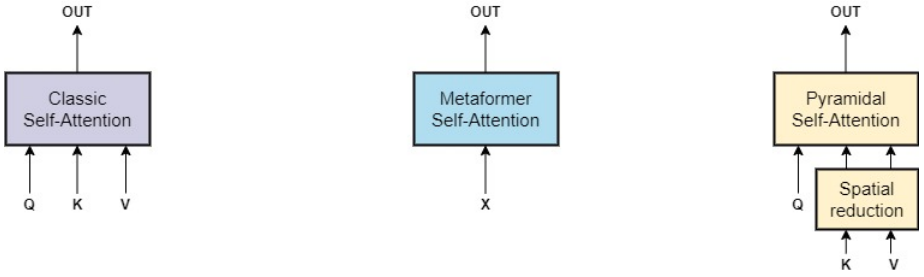

Fig. 1. Differences between attention modules: Classic Self-Attention, Metaformer Self-Attention, Pyramidal Self-Attention with spatial reduction

3.1 Meta-METER

As introduced in Section 2.2, thanks to the work presented in Metaformer [22], we have at our disposal a general technique able to create an attention module

with reduced computational complexity. Initially, in the Metaformer [22], the nature of the token mixer is left general. Subsequently, the use of spatial pooling as a token mixer is proposed in an in-depth discussion of the baseline [22], due to its simplicity. By doing so, we can specify our new attention module as follows

$$Attention(X) = Pooling(Norm(X)) + X \tag{2}$$

Where the terms X indicate the embedded input, $Norm(...)$ refers to the normalization technique chosen. In particular, we use a Layer Normalization technique [1], while *Pooling* is a standard spatial pooling of a suitable size to fit the other METER blocks.

We can assume that the pooling operator is the most appropriate one for the attention mechanism simplification, due to how the pooling context is closely linked to tasks related to convolutional neural networks, so computer vision and image processing in general. Moreover, as emphasized by the authors of [22], the choice of pooling avoids the introduction of further learnable parameters, reducing not only the computational complexity but also the chance of overfitting. This reinforces the choice of this particular token mixer, as it is a further approach to lightening the network, completely in tune with one of the intentions stated in Section 1. The modification applied to the transformer blocks of METER [11] is completed by the addition of a small convolutional network, as described in Section 2.2, composed of two convolutional layers followed by ReLU activation functions. This last step slightly increases the number of learnable parameters, but this increase is balanced by the nature of the new attention module and is necessary to avoid a reduction in accuracy performances.

Finally, the two attention blocks employed by METER architecture are equipped with a residual connection.

3.2 Pyra-METER

In the pyramid ViT paper [19] the authors propose to substitute the multi-head attention layer (MHA) [18] with the so-called spatial-reduction attention layer (SRA) [19]. The SRA, as the MHA, takes as input a query Q, a key K, and a value V and returns as output a refined feature. The difference between SRA and MHA is that SRA reduces the spatial scale of K and V before the attention [18] operation. This reduces the computational operation of the network.

Pyramid ViTs can be successfully employed also on high-resolution images by splitting and processing the data in patches, as suggested in [19]. As a consequence, we can use them for dense prediction tasks. Furthermore, they are characterized by a shrinking pyramid that allows us to cut the computational time of feature maps. The spatial scale of K and V have done as follows [19]:

$$SpatialReduction(X) = Norm(Reshape(X, R)W) \tag{3}$$

where X is the input sequence, R is the reduction ratio of the attention layers, C is the number of channels of the output, $Reshape(X, R)$ is an operation of

reshaping X to size $\frac{HW}{R^2} \times (R^2C)$, $W \in \mathbb{R}^{(R^2C)\times C}$ is a linear projection that reduces the dimension of X to C and $Norm(...)$ refers to Layer Normalization. The attention operation is computed as reported in (1).

By looking at these formulas (3)-(1), we can notice that the computational cost of the pyramid ViT attention operation is reduced by R^2 times with respect to the classical multi-head attention operation [18].

To build the Pyra-METER architecture, we have substituted METER attention module with the pyramid ViT attention module. In particular, we reshaped the input image only if $R > 1$, while the reshaping of K and V is performed even if $R < 1$ by dividing the number of channels by the number of heads. In the attention layers, the kernel has a shape equal to $R \times R$. The greater that ratio is, the more the image is reduced with the spatial reduction. After that, we put a simple normalization layer and we applied also the dropout mechanism to reduce overfitting. Finally, the projection mentioned in (3) is performed before the last layer which is the classical Softmax operation layer described in [18].

4 Results

In this section, we report the results obtained by Meta-METER and Pyra-METER structures described in Section 3. The rest of the section is organized as follows: Section 4.1 describes the implementation details and the evaluation metrics used. Section 4.2 analyzes the obtained results by comparing proposed models with the original METER architecture when tested over the NYU dataset. Finally, Section 4.3 highlights how the optimization strategy applied to the self-attention and respective transformer blocks impacts the inference performances.

4.1 Experimental setup

All the algorithms have been implemented in Python by leveraging the PyTorch deep learning framework. We use the same training procedure, loss function, and data augmentation strategy described in the reference METER paper with ADAM [5] optimizer and a batch size of 64. Moreover, we evaluate the performance of compared models by exploiting some common metrics used in monocular depth estimation tasks: the root-mean-square error (RMSE), the mean absolute error (MAE), respectively measured in meters [m], the relative error (REL), and the accuracy values δ_1, δ_2 and δ_3 [3]. Furthermore, we also evaluate the inference time, namely, the time measured in milliseconds [ms] that a model takes to perform a single inference over different devices: the Intel Xeon CPU 2.20GHz, which is representative of the x86 architecture, a Coral DevBoard equipped with an ARM Cortex-A53 and the NVIDIA Jetson TX1 equipped with a Cortex-A57. In the next Section they will be reported as CPU, DEV, and TX1 respectively. In particular, we also specifically evaluate the inference timings of the three transformer blocks of METER architecture, in order to highlight the impact of the proposed architectural changes on the transformer modules speed. Finally, we tested the proposed attention modifications on three METER configurations,

namely S, XS, and XXS, which represent the same model structure but with a different number of layers and trainable parameters, in order to create S, XS, and XXS variants of Meta-METER and Pyra-METER.

4.2 Performance analysis

The overall estimation performances of introduced models, tested over the three configurations (S, XS, and XXS) are compared in this section. The experiments were carried out on the Xeon CPU 2.20GHz, exploiting a single core, and then on the Coral DevBoard and NVIDIA Jetson TX1 boards, all while avoiding task parallelization. Ee report in Table 1 the obtained results over the evaluation metrics introduced in Section 4.1.

Table 1. Error metrics, accuracy values and total inference time of S, XS, XXS METER, Meta-METER and Pyra-METER measured on Intel Xeon CPU (T_{CPU}), Coral DevBoard (T_{DEV}) and NVIDIA Jetson TX1 (T_{TX1}).

Model	RMSE [cm]	MAE [cm]	Abs_{Rel}	δ_1	δ_2	δ_3	T_{CPU} [ms]	T_{DEV} [ms]	T_{TX1} [ms]
METER S	**47.035**	**35.151**	**0.137**	0.816	**0.962**	**0.990**	68.576	154.880	105.837
Meta-METER S	49.129	36.650	0.144	0.811	0.959	0.988	**40.164**	**147.608**	**102.167**
Pyra-METER S	48.424	36.350	0.142	**0.817**	0.960	0.989	46.586	155.584	112.521
METER XS	49.565	37.301	0.151	0.811	0.953	0.987	49.616	128.613	80.824
Meta-METER XS	51.499	38.710	0.154	0.801	0.951	0.986	**31.373**	**120.406**	**77.077**
Pyra-METER XS	**49.389**	**36.843**	**0.145**	**0.812**	**0.956**	**0.989**	36.246	124.221	81.743
METER XXS	54.665	41.505	0.172	0.770	0.944	0.983	29.734	68.906	45.343
Meta-METER XXS	57.111	43.610	0.182	0.754	0.936	0.981	**15.927**	**61.084**	**31.183**
Pyra-METER XXS	**53.684**	**40.832**	**0.162**	**0.775**	**0.946**	**0.984**	16.567	64.744	48.675

Based on the reported values, we can notice that on the S-configuration, the non-optimized model is slightly more accurate, while for the XS and XXS variants, the Pyra-METER is able to achieve the most accurate estimations. In contrast, by looking at the inference timings, Meta-METER always achieves the fastest inference performances (+20.7%) with a limited average degradation of −4.3%, and −1.3% for the RMSE e δ_1 accuracy respectively, when compared with the original METER model.

More in detail, when tested over different devices, i.e., CPU, DEV, and TX1, Meta-METER is able to achieve an average performance boost of +41.5%, +7.5%, and +13.1% respectively, while, Pyra-METER improves the non-optimized version by +35.1% and +3.0% respectively for the CPU and DEV. In contrast, on TX1 it obtains a decrement with respect to the original METER inference time of −4.9%; this fact could be due to the usage of the GPU by the TX1 which performs in parallel some processes that improve the performances with respect to the CPU. Moreover, by comparing the two introduced models, we can observe that the averaged timing boost equal to +10.6% of Meta-METER with respect to Pyra-METER comes at the expense of worse (−2.9%) estimation performances.

The latter results are due to the fact that Meta-METER exploits only one single pooling layer that is able to considerably speed up the inference time, at the expense of some details loss, while Pyra-METER still employs a lot of computational time on its attention module, because it gives its main contribution with large resolution features, decreasing it as the features get smaller.

4.3 Inference time analysis

Once the general model's performances have been analyzed, we report in this section an in-depth analysis of the inference times of the different self-attention blocks by measuring the inference time of the three METER transformer blocks only. In fact, by isolating the transformer blocks with respect to the other model layers, the impact of attention modifications is better highlighted.

We report the results obtained over the three configurations (S, XS, and XXS) in Table 2.

Table 2. Transformer blocks inference time of S, XS, XXS configurations of METER, Meta-METER and Pyra-METER measured on Intel Xeon CPU, Coral DevBoard and NVIDIA Jetson TX1.

Model	CPU			DEV			TX1		
	T_1 [ms]	T_2 [ms]	T_3 [ms]	T_1 [ms]	T_2 [ms]	T_3 [ms]	T_1 [ms]	T_2 [ms]	T_3 [ms]
METER S	0.011	0.006	0.004	0.028	0.011	0.008	0.016	**0.0075**	**0.005**
Meta-METER S	**0.004**	**0.002**	**0.001**	**0.013**	**0.008**	**0.005**	**0.011**	0.010	0.006
Pyra-METER S	0.005	0.004	0.003	0.022	0.017	0.018	0.017	0.017	0.019
METER XS	0.006	0.003	0.002	0.023	0.008	0.005	0.013	**0.006**	0.004
Meta-METER XS	**0.002**	**0.001**	**0.0006**	**0.008**	**0.005**	**0.0026**	**0.007**	**0.006**	**0.003**
Pyra-METER XS	0.003	0.002	0.002	0.012	0.011	0.009	0.010	0.009	0.009
METER XXS	0.006	0.002	0.001	0.020	0.006	0.004	0.012	0.004	0.003
Meta-METER XXS	**0.002**	**0.001**	**0.0004**	**0.005**	**0.003**	**0.0017**	**0.005**	**0.003**	**0.001**
Pyra-METER XXS	**0.002**	**0.001**	0.001	0.008	0.006	0.005	0.007	0.006	0.005

Based on the reported values, it can be noticed that Meta-METER always achieves the lowest inference timings. Although the latter result could have been inferred from the general results given in Table 1, in this analysis, we report in detail the individual contribution of the optimized method in each transformer block, i.e., T_1, T_2, and T_3. Therefore, when Meta-METER and Pyra-METER are compared with the original structure, the inference times over each transformer block are equal to -58.5%, -32.2%, and -46.6% for the first model and -39.9%, $+14.1\%$, and $+75.2\%$ for the second one. Moreover, when respectively tested over the CPU, DEV, and TX1 edge devices, the overall performance of the Meta-METER and Pyra-METER self-attention over the non-optimized variant are equal to -65.0%, -50.2%, and -22.1% for the first model and -34.8%, $+13.1\%$, and $+71.1\%$ for the second one. From these results, we can derive that

Meta-METER attention is able to effectively guarantee an inference boost over both transformer blocks with various input shapes and devices. In contrast, the spatial-reduction self-attention layer of Pyra-METER better fits scenarios where the resolution of input features is high.

5 Conclusions

In this paper, we propose two optimized versions of METER [11] architecture, Meta-METER and Pyra-METER. The introduced models can achieve comparable estimation performances over the non-optimized model with lower inference timings. The latter results are due to the usage of a pooling layer for the Meta-METER self-attention block and to the spatial reduction for the Pyra-METER self-attention block.

We can assess that the proposed models are promising optimized alternatives of METER architecture which enable us to tackle the MDE task on resource-constrained devices. Based on the reported results, we can conclude that the optimization of self-attention quadratic cost is a valuable starting point to close the gap between high-accurate deep learning models and devices with limited computational power. In particular, Meta-METER is able to sensibly improve both the total and the transformer blocks inference time, even if some reduction in the estimation metrics due to the usage of a single pooling layer that discards some features. On the other side, Pyra-METER improves all the estimation metrics, especially for the XS and XXS sizes, and also the inference times on the CPU, whereas, on the other two boards, is not able to reach excellent inference times on the second and third transformer blocks because the employed structure works well when the reduction in resolution is large, so as one goes towards the small end features of the encoder, its contribution is less.

Given the above, if is possible to take some losses on estimation metrics in favor of large gains on inference time, then Meta-METER is a good application choice. On the other hand, Pyra-METER is a more balanced choice because it has a smaller boost on inference time in favor, however, of some gains on estimation metrics as well.

Given the positive results of such approaches on hybrid ViTs architectures, we plan in the future to extend such optimizations to more complex architectures and test over different benchmark datasets. Furthermore, we will investigate other ways to make ViTs efficient by using pruning/knowledge distillation or quantization strategies.

Acknowledgments

This study has been partially supported by SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, Sapienza University of Rome project 2022–2024 “EV2” (003_009_22), and project 2022–2023 “RobFastMDE”.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[2] Xingshuai Dong et al. "Towards real-time monocular depth estimation for robotics: A survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.10 (2022), pp. 16940–16961.

[3] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network". In: *Advances in neural information processing systems* 27 (2014).

[4] Kai Han et al. "A survey on vision transformer". In: *IEEE transactions on pattern analysis and machine intelligence* 45.1 (2022), pp. 87–110.

[5] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[6] Soroush Abbasi Koohpayegani and Hamed Pirsiavash. "Sima: Simple softmax-free attention for vision transformers". In: *arXiv preprint arXiv:2206.08898* (2022).

[7] Zhenyu Li et al. "Binsformer: Revisiting adaptive bins for monocular depth estimation". In: *arXiv preprint arXiv:2204.00987* (2022).

[8] Jiachen Lu et al. "Soft: Softmax-free transformer with linear complexity". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 21297–21309.

[9] Ilya Makarov and Gleb Borisenko. "Depth inpainting via vision transformer". In: *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE. 2021, pp. 286–291.

[10] Sachin Mehta and Mohammad Rastegari. "Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer". In: *arXiv preprint arXiv:2110.02178* (2021).

[11] Lorenzo Papa, Paolo Russo, and Irene Amerini. "METER: a mobile vision transformer architecture for monocular depth estimation". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2023).

[12] Lorenzo Papa et al. "Lightweight and Energy-Aware Monocular Depth Estimation Models for IoT Embedded Devices: Challenges and Performances in Terrestrial and Underwater Scenarios". In: *Sensors* 23.4 (2023), p. 2223.

[13] Lorenzo Papa et al. "Speed: Separable pyramidal pooling encoder-decoder for real-time monocular depth estimation on low-resource settings". In: *IEEE Access* 10 (2022), pp. 44881–44890.

[14] Matteo Poggi et al. "Towards real-time unsupervised monocular depth estimation on cpu". In: *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2018, pp. 5848–5854.

[15] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. "Vision transformers for dense prediction". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 12179–12188.

[16] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[17] Nathan Silberman et al. "Indoor segmentation and support inference from rgbd images". In: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*. Springer. 2012, pp. 746–760.

[18] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[19] Wenhai Wang et al. "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 568–578.

[20] Diana Wofk et al. "Fastdepth: Fast monocular depth estimation on embedded systems". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6101–6108.

[21] Haixu Wu et al. "Flowformer: Linearizing transformers with conservation flows". In: *arXiv preprint arXiv:2202.06258* (2022).

[22] Weihao Yu et al. "Metaformer is actually what you need for vision". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10819–10829.

[23] Mehmet Kerim Yucel et al. "Real-time monocular depth estimation with sparse supervision on mobile". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2428–2437.

[24] Chaoqiang Zhao et al. "Monocular depth estimation based on deep learning: An overview". In: *Science China Technological Sciences* 63.9 (2020), pp. 1612–1627.