scientific reports

Check for updates

OPEN

Efficient attention vision transformers for monocular depth estimation on resource-limited hardware

Claudio Schiavella[1]✉, Lorenzo Cirillo[1], Lorenzo Papa[1,2], Paolo Russo[1] & Irene Amerini[1]

Vision Transformers show important results in the current Deep Learning technological landscape, being able to approach complex and dense tasks, for instance, Monocular Depth Estimation. However, in the transformer architecture, the attention module introduces a quadratic cost concerning the processed tokens. In dense Monocular Depth Estimation tasks, the inherently high computational complexity results in slow inference and poses significant challenges, particularly in resource-constrained onboard applications. To mitigate this issue, efficient attention modules have been developed. In this paper, we leverage these techniques to reduce the computational cost of networks designed for Monocular Depth Estimation, to reach an optimal trade-off between the quality of the results and inference speed. More specifically, optimization has been applied not only to the entire network but also independently to the encoder and decoder to assess the model's sensitivity to these modifications. Additionally, this paper introduces the use of the Pareto Frontier as an analytic method to get the optimal trade-off between the two objectives of quality and inference time. The results indicate that various optimised networks achieve performance comparable to, and in some cases surpass, their respective baselines, while significantly enhancing inference speed.

Keywords Computer vision, Edge devices, Efficient vision transformer, Monocular depth estimation

In Computer Vision, most approaches rely on RGB images[1,2]. However, several studies have demonstrated that incorporating a fourth channel, the depth channel, yields significant improvements[3–5] by integrating and making explicit additional information that was previously implicit in the captured image. Utilizing depth information, however, requires access to depth maps, which can be obtained through specialized hardware using an active sensing approach. Sensors such as LiDAR or certain types of cameras acquire depth by emitting a physical signal into the environment and measuring the time taken for its return. Depth plays an essential role in several practical applications. In robotics, it enhances autonomous navigation and obstacle avoidance[6], while in autonomous driving, it provides a precise understanding of the surrounding environment[7]. Additionally, in virtual reality, depth data contributes to more realistic and immersive experiences, improving user engagement[8]. Despite its advantages, actively acquiring depth through sensors presents several challenges. First, depth maps generated by these devices are often sparse, providing depth values only at specific points rather than offering a dense and comprehensive representation of the scene. Moreover, integrating dedicated hardware imposes additional constraints on the system, increasing weight and power consumption, which can be critical for resource-limited applications.

These challenges can be mitigated through passive depth-sensing techniques, which shift the computational complexity from hardware to software. Unlike active sensing, passive approaches estimate scene depth without directly perturbing the environment, eliminating the need for specialized depth sensors. One prominent example of passive depth estimation is Monocular Depth Estimation (MDE)[9], which infers depth information from a single RGB image. This complex, dense task allows for a more global view of the scene's depth and minimises the reliance on specific hardware. In this context, Deep learning models are essential for obtaining accurate and reliable depth estimates. In particular, transformer architectures succeed in approaching this task with impressive results. Due to their ability to capture global features via the attention mechanism[10], the transformer models specific to vision tasks, namely Vision Transformers (ViTs)[11], can predict accurate and precise depth maps[12–14]. A major limitation of the ViT architecture is the computational cost of its attention module, which

[1]Department of Computer, Control and Management Engineering (DIAG), Sapienza University of Rome, Rome 00185, Italy. [2]ϕ-Lab, European Space Agency (ESA), Frascati 00078, Italy. ✉email: schiavella@diag.uniroma1.it

scales quadratically with the number of tokens being processed. While this module is essential for achieving high performance, it significantly impacts processing speed and memory usage, making dense prediction tasks such as MDE highly resource-intensive. This challenge is particularly critical in onboard or real-time applications, where the device's processor is also responsible for tasks beyond network inference. In this context, the device is in a dynamic environment and it should generalize real and new data[15–18] in a reliable and fast way. This fact, combined with the constrained computational resources of certain hardware platforms, creates a challenging environment for deploying ViT-based MDE models efficiently. To address this issue, our research focuses on integrating efficient attention mechanisms into state-of-the-art ViT architectures for MDE, aiming to achieve an optimal trade-off between performance and inference speed.

This paper provides a comprehensive analysis of the impact of efficient attention modules on various ViT architectures for MDE that achieve state-of-the-art performance. Specifically, optimizations are applied not only to the entire network but also independently to the encoder and decoder, enabling a more granular study of how attention modifications influence different network components. This approach helps identify the most effective optimizations, ultimately guiding the development of a model that balances inference speed with fidelity to baseline performance. The architectures under investigation include PixelFormer (PXF)[12] and Neural Window Fully-connected CRFs (NeWCRFs)[13], both of which are computationally intensive models that incorporate attention mechanisms in both the encoder and decoder. Additionally, we analyse METER[14], a lightweight hybrid architecture that combines transformer-based processing with convolutional layers. By comparing the effects of optimizations on these models, we aim to understand how fully attention-based networks, such as PXF and NeWCRFs, perform relative to hybrid architectures like METER. The optimizations explored in this study introduce structural modifications to the attention mechanism. Specifically, MetaFormer (Meta)[19] replaces the attention module with a token mixer of linear complexity, significantly reducing computational overhead. Meanwhile, Pyramid Vision Transformers (Pyra)[20] adjust the computational granularity of the attention module by reducing the size of its processing elements, thereby altering its overall complexity. Through this analysis, we seek to determine the effectiveness of these optimizations in enhancing inference efficiency while maintaining high-quality depth predictions. Finally, the Mixture-of-Head (MoH) method[21] introduces a routing mechanism to dynamically select the most effective attention heads, further enhancing computational efficiency. The choice of these types of efficient attention modules is based on the fact that they have been tested on various computer vision tasks, but not yet to MDE. In all cases considered, these attention modifications have led to promising results, making it interesting to evaluate their performance on a dense and complex task such as MDE. Moreover, each optimisation affects the attention modification differently: Meta simplifies the architecture, Pyra compresses the inputs, while MoH changes the routing of multi-head attention.

Given the structural modifications introduced by these optimizations, it is essential to employ an objective evaluation framework to assess their impact. To achieve this, our research leverages the Pareto Frontier, a systematic and analytical tool for identifying optimal solutions in multi-objective problems. In our context, each candidate model represents a potential solution, and only the Pareto-optimal models, those that are not outperformed across all objective criteria, are selected. A model is considered dominant if no other model surpasses it in both prediction quality and inference speed simultaneously. The resulting Pareto-optimal models define the frontier, representing the best trade-offs between these competing objectives. This approach provides a rigorous evaluation framework, allowing us to systematically compare optimised models against their respective baselines. While Pareto analysis has been applied in deep learning evaluations[22] and in multi-modal object tracking[23], it has not yet been explored specifically for MDE.

Our experiments are conducted on two benchmark datasets widely used for MDE: NYU Depth V2[24], which consists of indoor scenes, and KITTI[25], which focuses on outdoor driving scenarios. By evaluating optimised networks across these datasets, we aim to assess their performance in diverse real-world conditions, ensuring the models generalize well to practical applications.

To provide a clear overview, our work can be summarized as follows:

- We conduct an in-depth study on the impact of efficient attention modules on the qualitative and temporal performance of ViT models for the MDE task.
- We analyse the impact of the efficient attention modules separately on the full network, encoder, and decoder, obtaining a descriptive evaluation of the optimisation's contribution.
- We introduce the Pareto Frontier as a systematic method to identify the optimised models that achieve the optimal trade-off between result quality and inference speed.

The rest of the article is organized as follows: In "Related works" section presents an overview of MDE, describing ViT architectures for this task and suitable optimizations for their attention modules. In "Methods" section exposes networks and efficient attention modules used in the experiments. "Results" section presents a detailed analysis of the experiments carried out. Finally, "Conclusions" section indicates conclusions and potential future research directions.

Related works

Monocular depth estimation

Depth estimation, in its most general form, concerns determining the distance between an object and the imaging sensor. From an application perspective, this process is crucial in various real-world domains, including robotics[6], autonomous driving[7] and augmented reality[8]. The depth measurement can be performed actively, by sensing the depth through dedicated hardware. That approach perceives the scene's depth by perturbing the environment with a physical medium, such as LiDAR or Time-of-Flight. Another possible solution is to retrieve the depth passively, not by measuring but by estimating the value through software solutions. Our application

context focuses on MDE, which is categorised as a passive method. This approach estimates image depth from a single RGB image on a per-pixel basis[9]. Among software-based solutions related to this task, several Deep Learning methods leverage neural networks to predict depth maps[12–14]. These techniques often achieve accurate results that are in part attributed to[10] attention mechanisms, which enable the extraction of global information from the input image. However, despite their potential, these models are often constrained by a slow inference speed, primarily due to the weight of the network and the computational cost of some of their modules. This situation leads to challenges in scenarios where both precision and real-time performance are necessary.

Our research aims to identify solutions that can mitigate the complexity of MDE models while maintaining performance and improving inference speed.

Vision transformers for monocular depth estimation

Deep Learning-based solutions are widely used in the MDE task. Early approaches involved using Convolutional Neural Networks (CNN)[26] to estimate the depth from a single input image. A representative work that introduces this type of model is the study by Eigen et al.[27]. Their approach captures the image context and then reconstructs the depth map by refining the local details of the obtained representation.

With the advent of Transformers and the attention mechanism[10], these architectures have been extended to the field of imaging, through the introduction of ViTs[11]. These models process input images by tokenizing them into smaller patches. Due to their ability to capture long-range dependencies and model global spatial relationships, they have been widely adopted in Computer Vision[28] tasks, including MDE. Several networks have exploited the ViT architecture with specific techniques to improve the quality of the predicted depth map. An example is the PixelFormer network[12], which after extracting the image representation with a Swin Transformer backbone[29], improves the quality of the predicted depth map by merging encoder and decoder features with an attention-based approach. Similarly, the NeWCRFs network[13] exploits a decoder module that introduces Conditional Random Fields (CRF) integrated with attention mechanisms. In this framework, each CRF captures observable image features, such as textures and object positions, to define an energy function, which is then optimised to retrieve the depth estimation. In contrast to these networks, some recent solutions integrate CNNs with the capabilities of ViTs, developing hybrid approaches. For instance, METER[14] is a lightweight model, combining a transformer-based encoder with a fully convolutional decoder. This architecture is suitable for experiments were conducted on devices with limited hardware resources and achieves a reasonable balance between efficiency and performance.

Despite the strong performance of the presented networks for the MDE task, they share common problems with ViT architectures. Transformer models are often characterized by a high number of parameters and floating point operations per second (FLOPs), resulting in memory-intensive networks and often slow inference. This issue is mitigated in lightweight transformer architectures, which achieve reasonable performance while maintaining lower computational demands. However, another challenge that affects all Transformers is the quadratic problem of the attention module[10]. This module has a quadratic computational cost concerning the input tokens, leading to a significant speed issue in MDE, as high-resolution inputs require processing a substantial number of tokens to predict a dense depth map. Recent research on MDE has introduced efficient attention mechanisms in networks addressing this task[30,31]; however, the trade-off between performance and speed remains insufficiently analysed.

This work addresses the optimisation of networks for MDE, which perform well but are limited by the computational cost of the attention module. Furthermore, this work focuses on achieving a trade-off between performance and speed through a structured and quantitative evaluation of the modified networks compared with their original models.

Efficient attention modules

Although in different ways, all the architectures presented use the attention mechanism. This module, in most cases, determines the model's performance. However, attention presents a computational problem due to its quadratic complexity relative to the input tokens[10]. In ViT-based networks, where images are divided into a large number of patches[11], this results in significant computational overhead. This situation poses challenges in time-critical applications such as autonomous driving or rescue robotics[9].

To address this issue, several studies[32] have explored techniques to optimise the attention modules. In particular, these methods aim to reduce the quadratic complexity of the mechanism, proposing new approaches to speed up the calculation and maintain performance. These methods are specifically designed to modify attention, in contrast to the promising optimisation present in the current literature[33,34], or more general solutions such as quantisation[35], pruning[36] and knowledge distillation[37] which reduce computational overhead by optimising the entire network. Regardless of the applied optimisation, modified models will experience a trade-off between performance and inference speed[32,38]. There are cases, however, where performance may even improve once the optimisation is introduced. A consistent example is MoH[21], which modifies the entire multi-head attention mechanism. Since not all attention heads contribute equally to the final prediction[39], the proposed optimisation employs a specialised routing mechanism to efficiently select the most relevant heads. This method has been used in computer vision tasks such as classification or image generation, but its application in MDE has not yet been proven.

Unlike MoH, some optimisations, such as Meta[19] and Pyra[20] focus more specifically on token processing within the attention module. Meta advances the idea that in ViT architectures, the quality of results depends primarily on the overall structure of the network, rather than on the attention module. This part of the network is intended as a token mixer and, given the initial assumption, is replaced with a much simpler module which performs the token mixing operation in a computationally lighter manner. Pyra optimisation, in particular, introduces spatially reduced attention, progressively decreasing the input size of the attention mechanism.

This method allows for the better application of transformation architectures to dense tasks, dealing with less expensive computational elements. Both methods introduced have already been tested in the MDE task[38], bringing encouraging results from the point of view of maintaining the performance and speed of the optimised models. However, a precise analytical definition of the trade-off between prediction quality and speed of inference remains unclear. The Efficient Error Rate introduces a possible approach in[32], but this metric is based on values primarily influenced by network-wide optimisations, which substantially alter the number of parameters and the model's memory weight. Another attempt to measure the trade-off between different objectives of a neural network was introduced by the work[22], in which through the Pareto Frontier a systematic and precise way is proposed that comes directly from optimisation theory. To the best of our knowledge, the application of this method is limited in the context of Deep Learning and unexplored in the analysis of optimised models for MDE.

Building on these concepts, we aim to optimise ViT architectures for the MDE task. More specifically, techniques that modify the complexity of the attention modules have been adopted. The impact of these modifications will be evaluated in a general and systematic manner, independent of specific network values, focusing solely trade-off between quality and speed objectives.

Methods

Optimization framework

In this section of the article, the general approach proposed to modify and analyse the networks considered for the study is presented. As highlighted in the current literature, the issue associated with Transformer models, related to the quadratic cost of their attention module, has emerged[10,38]. Consequently, the need to optimize this part of the network arises, applying targeted modifications to mitigate the complexity of attention. However, to have a clear comparison between the modified networks and the baselines, a standardized process that can be applied to the networks and optimizations under study is essential.

This need leads to setting the proposed analysis through a specific framework. This approach takes advantage of the architectural similarities of the elements involved, allowing the various network-optimization combinations to be easily applied and immediately capture the results relevant to the research. The idea is shown in Fig. 1. The approach considers each component as a container, capable of integrating the different elements. In this way, a generic framework is obtained, with different instances depending on the dataset, network and optimization used.

More specifically, the framework includes a module related to the network architectures to be tested. This component itself contains two sub-structures: one for the encoder and the other for the decoder. These two elements are fundamental in the framework because they will integrate the different types of efficient attention modules and the classical one for the baseline. Thus, by instantiating these containers, it is possible to obtain the network configurations in a clear and precise manner.

This approach realises a consistent experimental framework that facilitates the reproducibility and the evaluation of the different modified models. Each network and optimization that realizes a configuration has

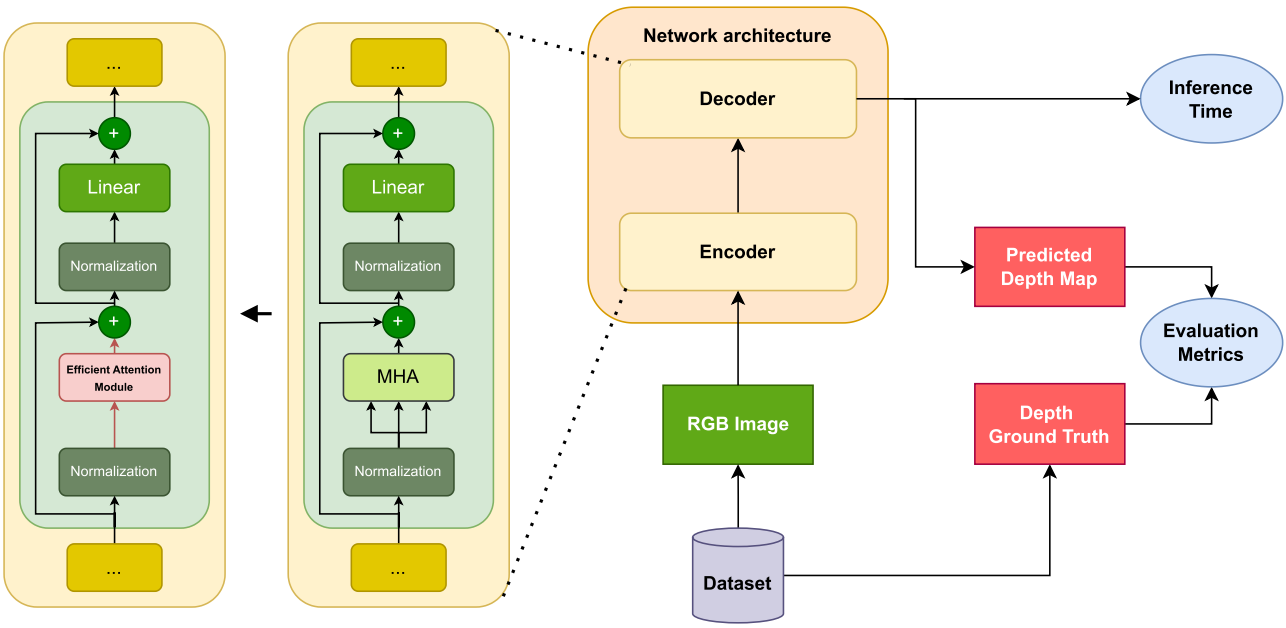

Fig. 1. The general framework used to compare the different network-optimization configurations. The module *Network architecture* instantiates the network to be analysed, while the submodule *Efficient Attention Module* integrates the attention mechanism used by the encoder, decoder, or the entire network. On the left side a comparison is shown to illustrate how each optimization modifies the original module. The left module depicts the optimised attention, while the right one shows its standard version.

its characteristics that determine the behaviour of the modified network. Details on these different elements are discussed in the following sections.

Network architectures

METER

METER[14] is a hybrid and lightweight ViT network. It achieves strong performance, even in scenarios with limited hardware resources.

This network uses an encoder that extracts useful features for depth prediction from the input image. This module comprises different layers, including convolutions with small kernels to maintain fast data processing and transformer blocks. In particular, this is the only part of the network that has attention modules. This module is available in three sizes: XXS, XS, and S. Each increment increases the parameters and the network capabilities. The METER decoder is purely convolutional and is responsible for reconstructing the predicted depth map from the features extracted by the encoder.

Beyond the architecture, this work introduces a composite loss specific to the MDE task. This approach integrates different values tailored for depth estimation, considering the pixel-per-pixel difference, cosine similarity between prediction and ground truth, and penalty factors to enhance detail reconstruction. In addition, a dedicated data augmentation pipeline enhances the robustness of the network. The latter proposes a change in the colour and brightness of the input RGB image, called C Shift, to simulate different data acquisition conditions. Furthermore, virtual data augmentation is also applied to the depth maps, with the D Shift, which randomly shifts the depth map corresponding to the ground truth to reduce the risk of overfitting on specific depth values.

PixelFormer

PixelFormer[12] approaches the MDE task in a particular form. For each image, the network adaptively predicts multiple intervals that divide the continuous depth value into a discrete range. Each pixel of the depth map to be estimated will be associated with a vector of weights, with several elements equal to the number of intervals estimated. The final depth of each pixel is calculated as a linear combination of the centres of the intervals predicted for the image, weighted by the values of the probability vector associated with that pixel.

The encoder, a Swin Transformer[29], was presented in the original paper as a general backbone for computer vision tasks. Its feature extraction function is based on a hierarchical organisation to model information at different resolutions and on non-overlapping windows that, in their context, apply attention computation to the local context of an input patch. As in the previous model, the network is available in three configurations: tiny, base and large, each with increasing parameters. PixelFormer's decoder, on the other hand, introduces the SAM mechanism that merges features from the encoder with those of the decoder through an attention mechanism to maximise information extraction from the encoder's general and decoder-specific features.

NeWCRFs

The NeWCRFs[13] network approaches the MDE task based on so-called CRFs. These elements are probabilistic models that capture spatial relationships in structured data. In MDE, these elements allow dependencies between pixels to be modelled so that the estimated depth is consistent with that of its neighbours. This approach applies CRFs on local windows to capture global information while limiting the computational impact to limited connections between neighbouring pixels.

The network applies this methodology through its decoder, supported by an attention mechanism to precisely and accurately model the connections between the different local windows related to the CRFs. The encoder module consists of a Swin Transformer[29] that processes the input images, extracting global and hierarchical features while capturing long-range dependencies between pixels.

Efficient attention module

Meta-optimised modules

As shown, MetaFormer generalises the concept of ViTs by maintaining the overall architecture while allowing the use of different token mixers to replace the attention module. In particular, the work demonstrates that using a simple operator, such as pooling, instead of attention supports the hypothesis that the general ViT architecture is a key factor in model performance.

Taking the original formula of the attention, the difference between this and the one proposed in[19] is very marked. Considering the classical formulation of the attention

$$\mathrm{Attention}(Q,K,V)=\mathrm{Softmax}\left(\frac{QK^{\top}}{\sqrt{d_h}}\right)V \tag{1}$$

where Q are the queries, K the keys, V the values, d_h the size of the attention head and $\mathrm{Softmax}$ an activation function[10], the Meta approach proposes the following modification

$$\mathrm{MetaAttention}(x)=\mathrm{Pooling}_k\left(x\right) \tag{2}$$

where, in this case, x is directly the input given to the transformer, and the operator *Pooling* operates as suggested by its name, exploiting a kernel of dimension k that aggregates the elements of the input by averaging them among those that fall in its receptive field.

Using pooling as a token mixer allows for a module with linear computational complexity concerning the number of input tokens. In addition, it does not introduce trainable parameters that could burden the network's structure.

Pyra-optimised modules

The Pyramid PVT approach is one of the proposed variants designed to mitigate the drawback associated with the quadratic complexity of the attention module, particularly in transformer-based networks for dense tasks, such as MDE. In this case, the number of tokens present is exceptionally high, exposing the network to the adverse effects of having a quadratic complexity dependent on this factor.

The concept presented in[20] combines the advantages of convolutional neural networks[26] and transformers[10]. It introduces a pyramid approach that progressively reduces the input's resolution, maintaining a high-resolution representation in the first levels for greater accuracy in dense predictions. Although the attention equation closely resembles the original version shown in Eq. 1, a critical detail represents the core idea of the proposed method.

$$\text{PyraAttention}(Q, K, V) = \text{Softmax}\left(\frac{QSR(K)^{\top}}{\sqrt{d_h}}\right) SR(V) \tag{3}$$

The spatial reduction operator *SR* manipulates both *K* and *Q* keys, defined as follows

$$\text{SR}(x) = \text{Norm}\left(\text{Reshape}\left(x, R\right) W\right) \tag{4}$$

This operation is applied to the input *x*, which, in our case, are the keys or values. The reduction ratio *R* applies a reshape operation to the input. The output is then subjected to a linear projection and normalisation. These steps reduce the computational cost of the attention operation by as much as R^2 compared to its original form, allowing even considerable inputs to be better handled.

MoH-optimised modules

The concept presented by MoH does not modify the attention module itself but rather optimises the routing mechanism among multiple heads, as is depicted in Fig. 2. Attention remains so at the equation shown in Eq. 1. In its original form, multi-head attention is formulated as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(H^1, H^2, \ldots, H^h\right) W_O \tag{5}$$

$$\text{H}^i = \text{Attention}(Q^i, K^i, V^i) \tag{6}$$

It represents a linear concatenation of the various attention heads, with the matrix W_O as the final step. This form can be seen from another perspective.

$$\text{MultiHead}(Q, K, V) = \sum_{i=1}^{h} H^i W_O^i \tag{7}$$

This is expressed by decomposing the linear projection matrix W_O row by row and describing the multi-head attention formula as a sum. This point serves as the foundation of the MoH technique. The multi-head attention set as in[21] is defined as follows.

$$\text{MoH}(Q, K, V) = \sum_{i=1}^{h} g_i H^i W_O^i \tag{8}$$

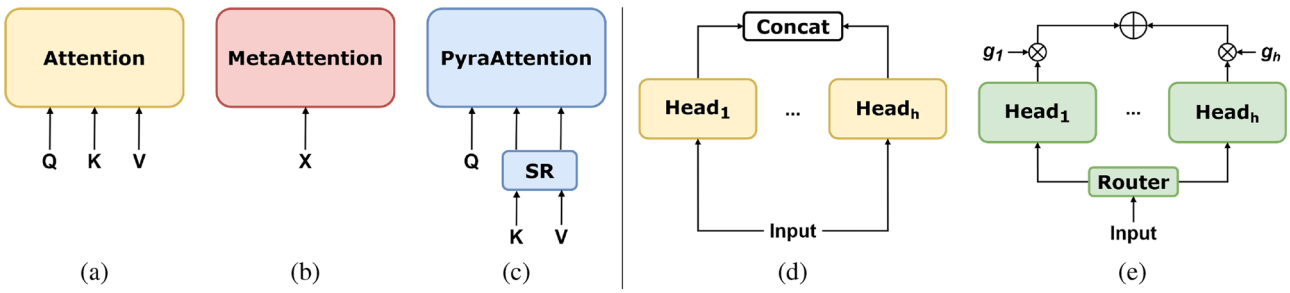

Fig. 2. On one side, the classical attention module (**a**) is compared with the token mixer module of Meta (**b**) implemented by the pooling operation, and with the Pyra (**c**) approach in which key and values are spatially reduced. On the other side, MoH (**e**) approach modifies the routing system in the multi-head attention (**d**), addressing only some specific heads, and then aggregating them using a weighted sum.

In this configuration, the routing score g_i acts as a router to select the best k heads of attention. This g_i value is non-zero if and only if the i_{th} attention head should be activated.

This attention mechanism categorises two types of heads: shared heads, which are always active and capture useful standard information in different contexts, and routed heads, which may or may not be active and do not share information between various contexts. The number of each type of head is a design decision to be made by those implementing the MoH mechanism.

Given this concept, the two-stage routing of the MoH approach is realised by the routing score g_i, which is as follows.

$$g_i = \begin{cases} \alpha_1 \,\mathrm{Softmax}\left(W_s x\right)_i, & \text{if } 1 \le i \le h_s, \\ \alpha_2 \,\mathrm{Softmax}\left(W_r x\right)_i, & \text{if } \left(W_r x_t\right)_i \in \text{Top-K}\left(\left\{\left(W_r x\right)_i \mid h_s + 1 \le i \le h\right\}\right), \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

$$[\alpha_1, \alpha_2] = \mathrm{Softmax}\left(W_h x_t\right) \tag{10}$$

x indicates the input token and h_s the number of shared heads. W_s and W_r are the projection matrices that combine the score of each head type, thus implementing the routing mechanism. The coefficients α_1 and α_2 are learned through the trainable matrix W_h to balance the overall contribution of the two kinds of heads. Learnable elements of the presented routing system are taken into account during training via an additional loss term specific to the task specific for the task that the network is performing[21]. This approach enables the dynamic selection of the most appropriate attention heads for each token, enhancing inference efficiency and accuracy.

Implementation details

Benchmark datasets

The datasets used in the subsequent experiments to analyse the impact of the chosen architectures are NYU Depth V2[24] and KITTI[25]. These two datasets are benchmarks for the MDE task.

The NYU dataset contains a collection of indoor scenes, with RGB images having a resolution of 640×480, and their associated depth maps, which have a maximum depth of 10 metres. There are 120K training samples, while there are 654 samples for the test phase.

As far as the KITTI dataset is concerned, it is conceptually opposed to the previous one; in fact, it is a collection of outdoor images with a resolution of 1241×376, and their respective depth maps, which, in this case, reaches up to 80 metres. The split of the dataset is such that there are 23K training samples and 697 test samples.

Evaluation metrics

The metrics used to evaluate the experimental results are standard qualitative measures used in MDE[27]: Root Mean Square Error (RMSE, in metres $[m]$), Absolute Relative error (Abs_{Rel}), and the accuracy measures δ_1, δ_2, and δ_3. Those metrics are presented in the following equations.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{11}$$

$$Abs_{Rel} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{y_i} \tag{12}$$

$$\delta_j = \frac{1}{N}\sum_{i=1}^{N}\max\left(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}\right) < 1.25^j \quad \text{for } j = 1, 2, 3 \tag{13}$$

In this expressions, y_i is the depth ground truth for the i_{th} pixel, $\hat{y}_i$ is the estimated depth for the $i - th$ pixel, and N is the total number of pixels of the image.

For the inference speed test, inference time is measured as the duration in seconds $[s]$ required for the network to perform the forward pass over the entire test set.

Experimental setup

The training configurations of the models follow the specifications reported in their respective papers[12–14], with training conducted on multiple NVIDIA RTX5000 GPUs. In particular, PixelFormer and NeWCRFs were trained for 20 epochs, using Adam[40] optimiser with parameters $\beta = (0.9, 0.999)$ and weight decay of 10^{-2}, a batch size of 8 samples, and an initial learning rate of 4×10^{-5} reduced linearly to 4×10^{-6}. METER was trained for 60 epochs under different configurations, introducing the AdamW optimiser[41] with parameters $\beta = (0.9, 0.999)$ and weight decay of 10^{-1}, a batch size of 128 samples, and an initial learning rate of 10^{-3} reduced by a factor of 0.1 every 20 epochs. All the models were implemented through the PyTorch framework[42]. Unlike the other networks in this research, METER operates on smaller image sizes compared to the original samples of both datasets under consideration. However, depth maps predicted by METER can be compared by upscaling, with the downside of a lower resolution. Furthermore, the METER article indicates that the KITTI version should be trained using samples with filled-in depth maps. To ensure a fair comparison between the models in our study, METER was trained on the KITTI raw dataset, which contains sparse depth maps.

With regard to the application of the optimisations, an attempt was made to keep the parameters unchanged from the respective references[19–21]. However, for Meta and Pyra, some modifications, obtained through a trial-and-error process, were necessary to adapt the pooling and the spatial reduction operator to the embedding sizes of the respective networks.

The experiments in our work were performed on three different CPUs: Intel Core i9-7920X (I9X79), Intel Core i9-10900X (I9X10), and Intel Xeon Gold 6338 (XG38). Testing the inference on a CPU aims to isolate the impact of optimisations, to prevent parallelization from mitigating their contribution. Moreover, this scenario reflects real-world applications where networks operate directly on embedded devices. In this context, a hardware accelerator, such as a GPU, is not always available. In such cases, the processor must manage both network calculation and other device functionalities. For this reason, all tests were performed on a CPU limited to a single core. The experiments evaluate the models using standard metrics (RMSE, Abs_{Rel}, δ_1, δ_2, δ_3) and measure the inference speed for both baseline and optimised networks. The reported metrics describe the mean performance, and inference speed is defined as the time each network takes to perform a forward pass, one sample at a time, over the entire test set.

Results

Experiments

The experiments were conducted by applying optimisations to the network's attention modules. In particular, each technique was observed on different parts of the models, analysing the impact on evaluation metrics and inference speed when the modification is applied to the whole network, to the encoder only, or to the decoder only. This approach enables the identification of the network parts most affected by the optimisation, highlighting which region yields the greatest improvements or experiences the most significant performance degradation. From this perspective, this analysis can identify models that achieve a meaningful trade-off between result quality and inference speed. To formalise this approach, the notation follows the network structure, where the encoder always precedes the decoder. More precisely:

- *Optimisation Network*: the optimisation was applied to all the attention modules in the architecture;
- *Optimisation-Base Network*: the optimisation was applied only to the attention modules present in the encoder of the architecture, while the decoder remained as it was in the non-optimised baseline model;
- *Base-Optimisation Network*: the optimisation was only applied to the attention modules in the decoder of the architecture, while the encoder remained as it was in the non-optimised baseline model;

The notation partially applies to METER: it only presents attention to its encoder, and the optimisation will be referred to as if applied to the entire network. Based on that, Tables 1 and 2 present the result of baseline models and their optimised versions, grouped by dataset to evaluate their performance on indoor and outdoor samples. Each test will have three tables, one for each network size: tiny, base and large. Each of these formats increases the number of parameters of the encoder, resulting in a more computationally demanding network.

Some details become evident from the tests on the NYU Depth V2 dataset. For METER, we observe how the optimisations, except MoH, have a minimal impact on degrading qualitative metrics while also reducing inference times. Analysing the different model sizes, we observe that, in general, the Meta and Pyra optimisations enhance network performance in some cases, with Meta consistently achieving the best inference speed. However, in the large version of METER, this trend is observed only in inference speed, where the baseline outperforms the optimised versions. For the deeper architectures, PixelFormer and NeWCRFs, decoder-level optimisations yield performance comparable to, and in some cases better than, their respective baselines. Other optimisation applications tend to degrade performance metrics, although they result in reduced inference times. In particular, it can be seen that, in all dimensions of these networks, Meta is always the fastest when applied to the entire architecture. This behaviour is attributed to the simplicity of pooling, which replaces the computationally intensive attention. In indoor scenarios, optimisations generally improve network speed, occasionally at the cost of performance. However, in some cases, such as with optimised decoders, they maintain competitive evaluation metrics.

When analysing the results obtained on the outdoor samples of the KITTI dataset, METER exhibits difficulties in generalising depth maps different from those it was originally developed to work with. Regarding performance metrics, the optimisations do not replicate the improvements observed with indoor samples. They tend to improve the delta metrics for the tiny and base sizes while improving the RMSE and Abs_{Rel} errors in larger versions. However, the application of efficient attention also increases the network's speed, resulting in better times in all the case studies. In deep architectures, we find similar behaviours in the optimisations, with Meta always presenting the best inference times and the decoder optimisations managing to keep performance equivalent to the baselines. Again, PixelFormer favours Meta and Pyra, while NeWCRFs work better with MoH. More specifically, as the size increases, the latter optimisation improves the performance of NeWCRFs.

Overall, the behaviour of the optimisations across the analysed networks follows identifiable trends. For instance, Meta primarily aims to accelerate inference, but often compromises prediction quality. Pyra is a good compromise between accuracy and speed. Conversely, MoH tends to enhance performance while yielding a comparatively smaller gain in speed. Each network, however, prefers certain optimisations over others regarding quality and speed. METER often works better with Pyra, PixelFormer performs well with the decoder modified with Pyra and NeWCRFs, and MoH is preferred over the decoder.

For the large versions, comparing the depth maps predicted by the baseline and the best-optimised model is insightful. As shown in Fig. 3, the predictions of the optimised models closely match those of the baseline across the analysed networks. In particular, the best-optimised versions of PixelFormer and NeWCRFs present predictions similar to those of their respective baselines. This is noticeable both in qualitative terms, observing

Model	RMSE [m] ↓	Abs_{Rel} ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	I9X79 [s] ↓	I9X10 [s] ↓	XG38 [s] ↓
(a)								
METER	0.544	0.175	0.778	0.946	0.983	15.91	10.96	16.47
Meta METER	*0.553*	***0.159***	***0.787***	*0.948*	*0.984*	***12.95***	***9.54***	***12.96***
Pyra METER	***0.533***	*0.162*	*0.773*	***0.949***	***0.986***	*14.86*	10.15	*14.34*
MoH METER	1.109	0.297	0.342	0.679	0.874	16.08	11.11	17.20
PXF	**0.392**	**0.114**	0.880	**0.982**	**0.996**	370.92	171.90	112.41
Meta PXF	*0.846*	*0.274*	*0.543*	*0.830*	*0.944*	***276.73***	***132.06***	***88.88***
Meta-Base PXF	*0.840*	*0.266*	*0.547*	*0.833*	*0.947*	*294.19*	*140.01*	*0.18*
Base-Meta PXF	*0.395*	*0.116*	*0.879*	*0.981*	*0.995*	*0.19*	*165.79*	*103.53*
Pyra PXF	0.628	0.202	0.697	0.921	0.978	337.07	171.20	115.64
Pyra-Base PXF	0.622	0.198	0.700	0.920	0.977	332.02	167.13	112.21
Base-Pyra PXF	***0.392***	***0.114***	***0.882***	***0.982***	***0.996***	*341.28*	*167.54*	*110.18*
MoH PXF	0.697	0.219	0.637	0.893	0.970	359.53	190.96	124.26
MoH-Base PXF	0.682	0.218	0.647	0.899	0.972	358.67	187.83	121.90
Base-MoH PXF	0.677	0.216	0.648	0.900	0.972	356.17	187.62	115.34
NeWCRFs	0.388	0.112	0.885	0.980	**0.995**	576.44	323.96	170.78
Meta NeWCRFs	1.061	0.349	0.430	0.729	0.890	**335.38**	**161.54**	**109.64**
Meta-Base NeWCRFs	0.844	0.274	0.541	0.833	0.946	510.30	294.72	150.82
Base-Meta NeWCRFs	1.027	0.353	0.448	0.745	0.899	380.64	192.91	126.68
Pyra NeWCRFs	0.964	0.324	0.477	0.778	0.921	425.37	214.19	142.44
Pyra-Base NeWCRFs	0.627	0.194	0.696	0.921	0.979	554.73	327.57	169.94
Base-Pyra NeWCRFs	0.947	0.322	0.489	0.786	0.922	434.61	217.36	148.73
MoH NeWCRFs	0.402	0.116	0.878	0.979	0.994	480.90	242.25	159.40
MoH-Base NeWCRFs	0.397	0.115	0.878	0.980	0.995	588.50	332.93	177.68
Base-MoH NeWCRFs	**0.382**	**0.111**	**0.886**	**0.981**	**0.995**	467.82	233.77	152.54
(b)								
METER	0.497	0.149	0.811	0.951	0.987	28.66	16.84	17.10
Meta METER	*0.499*	*0.146*	*0.811*	*0.955*	*0.988*	*22.52*	***15.77***	***16.52***
Pyra METER	***0.483***	***0.140***	***0.823***	***0.960***	***0.990***	***21.20***	*16.28*	*18.12*
MoH METER	0.921	0.248	0.471	0.791	0.938	25.64	15.33	18.86
PXF	0.338	0.096	0.918	0.988	0.997	542.79	305.30	181.09
Meta PXF	*0.817*	*0.271*	*0.569*	*0.843*	*0.948*	***379.05***	***213.43***	***133.69***
Meta-Base PXF	*0.806*	*0.266*	*0.571*	*0.847*	*0.951*	*393.54*	*220.83*	*141.52*
Base-Meta PXF	*0.340*	*0.097*	*0.918*	***0.989***	*0.997*	*511.37*	*296.31*	*180.42*
Pyra PXF	0.651	0.199	0.679	0.910	0.975	535.42	299.90	189.79
Pyra-Base PXF	0.621	0.192	0.711	0.922	0.977	522.03	295.70	183.70
Base-Pyra PXF	***0.338***	***0.095***	***0.919***	***0.989***	***0.998***	*528.77*	*306.64*	*192.41*
MoH PXF	0.675	0.213	0.655	0.900	0.973	581.83	349.55	211.02
MoH-Base PXF	0.666	0.209	0.658	0.904	0.973	577.37	345.43	208.24
Base-MoH PXF	0.667	0.210	0.666	0.904	0.973	543.79	324.31	193.23
NeWCRFs	0.337	**0.095**	0.918	**0.989**	**0.998**	755.28	458.38	241.93
Meta NeWCRFs	1.055	0.359	0.431	0.732	0.891	**442.80**	**241.67**	**151.82**
Meta-Base NeWCRFs	0.791	0.253	0.569	0.851	0.954	613.03	368.69	191.87
Base-Meta NeWCRFs	1.035	0.345	0.441	0.744	0.901	618.69	322.12	194.90
Pyra NeWCRFs	0.968	0.328	0.479	0.777	0.919	695.89	342.69	219.52
Pyra-Base NeWCRFs	0.632	0.197	0.699	0.919	0.976	792.81	448.06	246.80
Base-Pyra NeWCRFs	0.959	0.321	0.475	0.781	0.923	633.84	346.81	217.15
MoH NeWCRFs	0.349	0.098	0.912	0.987	0.997	711.95	380.77	243.29
MoH-Base NeWCRFs	0.346	0.099	0.913	0.987	0.997	803.02	470.81	261.67
Base-MoH NeWCRFs	**0.336**	0.096	**0.919**	0.988	0.997	650.92	358.43	219.34
(c)								
METER	***0.460***	**0.133**	**0.834**	**0.966**	**0.992**	22.02	18.78	21.52
Meta METER	0.485	0.142	0.816	0.961	0.989	**19.92**	**17.26**	**16.94**
Pyra METER	*0.477*	0.139	0.824	0.962	0.989	22.07	17.47	23.15
MoH METER	0.835	0.228	0.501	0.847	0.966	24.57	20.85	18.06
PXF	0.324	0.091	0.928	**0.991**	**0.998**	893.15	520.45	294.74
Continued								

Model	RMSE [m] ↓	Abs_{Rel} ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	I9X79 [s] ↓	I9X10 [s] ↓	XG38 [s] ↓
Meta PXF	*0.720*	*0.231*	*0.624*	*0.882*	*0.965*	***629.44***	***355.11***	***204.61***
Meta-Base PXF	*0.693*	*0.218*	*0.648*	*0.896*	*0.971*	*638.91*	*359.58*	*215.09*
Base-Meta PXF	***0.321***	***0.090***	***0.930***	***0.991***	***0.998***	*873.93*	*505.98*	*282.41*
Pyra PXF	0.680	0.208	0.657	0.900	0.976	922.30	532.98	312.46
Pyra-Base PXF	0.638	0.198	0.692	0.916	0.976	924.81	541.79	317.44
Base-Pyra PXF	0.322	**0.090**	0.929	**0.991**	**0.998**	892.15	514.00	299.51
MoH PXF	0.678	0.212	0.650	0.900	0.973	972.18	576.50	334.21
MoH-Base PXF	0.668	0.208	0.661	0.901	0.973	961.29	570.71	321.89
Base-MoH PXF	0.651	0.201	0.680	0.912	0.975	925.16	545.08	302.05
NeWCRFs	**0.322**	**0.091**	**0.929**	**0.992**	**0.998**	1345.95	658.96	339.68
Meta NeWCRFs	1.043	0.357	0.434	0.737	0.894	**999.90**	**376.78**	**225.67**
Meta-Base NeWCRFs	0.692	0.220	0.648	0.896	0.971	1157.19	518.98	271.43
Base-Meta NeWCRFs	1.019	0.344	0.451	0.752	0.905	1026.18	537.24	303.27
Pyra NeWCRFs	1.038	0.379	0.444	0.739	0.892	1120.98	593.90	335.55
Pyra-Base NeWCRFs	0.966	0.336	0.459	0.773	0.917	1238.04	682.72	350.57
Base-Pyra NeWCRFs	0.934	0.309	0.497	0.795	0.930	1071.33	562.12	318.50
MoH NeWCRFs	0.334	0.094	0.923	0.990	0.997	1142.63	605.58	356.84
MoH-Base NeWCRFs	0.332	0.094	0.923	0.989	0.998	1269.16	694.72	366.10
Base-MoH NeWCRFs	0.325	0.092	0.925	0.990	0.998	1109.97	585.42	329.66

Table 1. Experimental results on the tiny (a), base (b), and large (c) models with the NYU Depth V2 dataset. Values in bold represent the best results between the baseline and the optimisations applied for that model. The lines highlighted in italic show the best trade-offs between RMSE and inference time considering all models. These points are those values that compose the Pareto frontier.

the disparity maps between the two predictions, and in quantitative terms, with the RMSE between these two predictions. This behaviour is also understandable because of the similar evaluation metrics between the baseline and the best-optimised version. METER, however, is more affected by the optimisations, exhibiting a more pronounced difference between baseline and optimised predictions.

Trade-off performance-inference speed

The experimental setup yielded a comprehensive set of results, extensively illustrating how each type of optimisation impacts the network and its different modules. However, identifying the optimal trade-off between the obtained models solely from the tabular results is challenging. For this reason, the results were organised using the Pareto Frontier. This tool represents the set of optimised solutions where improving one objective would worsen at least one other. More specifically, all solutions are initially assumed to be optimal. Subsequently, each of them is compared with the others. Suppose one of the two chosen values is less than or equal to the other, with at least one of the two inequalities being strict. In that case, the first solution dominates the other, which is eliminated from the optimal solutions describing the best trade-offs. After this comparison, the remaining solutions will form the frontier.

In our case study, the two objectives are the RMSE, chosen as a quantitative index of model performance, and the average of the three CPU times, reporting the best trade-offs for these measures only. The choice of the RMSE as the indicator of network performance is given by the fact that this metric plays a key role in assessing the quality of the predicted depth, as it provides an indication of the overall consistency between prediction and ground truth[9].

From these concepts, Pareto Frontiers were constructed by grouping the results by dataset and size of the networks under analysis, as illustrated in Figs. 4 and 5. Analytically, this approach identifies the optimal trade-offs between the two selected objectives. Graphically, this corresponds to a frontier where models closest to it are valuable compromises, while those farther away are sub-optimal solutions. The elements of each Pareto frontier, optimal solutions between quality and speed, are highlighted in Tables 1 and 2 with a italic line.

The Pareto Frontier provides a summary of the model distributions across different sizes and datasets. Each point represents a model. A point further to the left indicates a better RMSE, while a point lower on the graph indicates a faster inference time. As can be seen, in addition to the best models, there are several elements on the Pareto frontier which, from an analysis of the tables alone, could have turned out to be sub-optimal solutions. For example, in the case of the PixelFormer tiny architectures on NYU, the best optimisation turned out to be Meta on the decoder. However, employing the analytical method of the Pareto frontier, we find that all Meta and Pyra optimisations on the decoder turn out to be the optimal solutions with the best trade-off between performance and speed of inference. Looking at the graphs, we can see specific trends that the optimisations tend to present in each case study. Considering Meta optimisation, the models where it is applied generally exhibit reduced performance but improved inference speed, except when applied only to the decoder, where it sacrifices speed but achieves better performance.

Model	RMSE [m] ↓	Abs_{Rel} ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	I9X79 [s] ↓	I9X10 [s] ↓	XG38 [s] ↓
(a)								
METER	***5.945***	***7.408***	*0.287*	*0.485*	*0.604*	*26.16*	*18.04*	*8.13*
Meta METER	*6.065*	*8.268*	***0.312***	***0.506***	***0.617***	***20.87***	***17.95***	***15.38***
Pyra METER	*6.048*	*7.615*	**0.312**	*0.502*	*0.612*	*23.61*	*17.87*	*17.91*
MoH METER	6.173	9.508	0.320	**0.506**	0.615	31.97	20.06	18.75
PXF	*2.324*	**0.060**	**0.966**	**0.996**	*0.999*	*468.91*	*256.89*	*160.90*
Meta PXF	*4.231*	*0.125*	*0.837*	*0.956*	*0.987*	***394.75***	***193.13***	***126.90***
Meta-Base PXF	*4.312*	*0.126*	*0.833*	*0.953*	*0.987*	*420.84*	*209.51*	*136.83*
Base-Meta PXF	*2.335*	***0.060***	***0.964***	*0.995*	***0.999***	*456.77*	*236.17*	*145.98*
Pyra PXF	3.150	0.087	0.915	0.984	0.996	482.70	243.56	171.71
Pyra-Base PXF	3.140	0.084	0.916	0.985	0.996	482.27	241.43	158.92
Base-Pyra PXF	**2.310**	**0.060**	0.964	**0.996**	**0.999**	499.32	259.68	161.13
MoH PXF	3.516	0.095	0.895	0.976	0.994	533.67	275.82	171.57
MoH-Base PXF	3.600	0.097	0.892	0.977	0.994	525.27	278.92	168.56
Base-MoH PXF	3.546	0.096	0.894	0.977	0.994	516.21	261.16	161.84
NeWCRFs	2.373	**0.059**	**0.965**	**0.995**	**0.999**	847.32	463.72	234.65
Meta NeWCRFs	6.929	0.284	0.559	0.822	0.932	**505.81**	**235.54**	**156.89**
Meta-Base NeWCRFs	4.201	0.122	0.844	0.959	0.988	780.12	427.10	207.85
Base-Meta NeWCRFs	6.069	0.219	0.662	0.879	0.958	611.31	280.70	180.89
Pyra NeWCRFs	5.140	0.177	0.738	0.920	0.976	667.40	306.87	205.90
Pyra-Base NeWCRFs	3.193	0.088	0.914	0.983	0.996	801.98	457.53	231.76
Base-Pyra NeWCRFs	4.750	0.171	0.756	0.934	0.981	697.67	312.78	200.27
MoH NeWCRFs	2.483	0.062	0.958	0.994	0.999	704.72	355.31	223.71
MoH-Base NeWCRFs	2.432	0.060	0.962	0.995	0.999	821.65	468.97	245.97
Base-MoH NeWCRFs	***2.361***	*0.060*	*0.963*	***0.995***	***0.999***	*687.85*	*347.26*	*214.89*
(b)								
METER	***5.794***	***6.625***	*0.302*	*0.504*	*0.618*	*42.51*	*29.66*	*22.16*
Meta METER	*5.920*	*7.797*	***0.329***	***0.516***	***0.622***	*44.45*	***24.12***	***18.22***
Pyra METER	6.052	7.010	0.319	0.498	0.605	**36.50**	26.22	26.65
MoH METER	5.958	8.033	**0.329**	0.514	0.621	44.69	31.88	25.17
PXF	2.205	0.055	**0.972**	**0.997**	**0.999**	781.45	437.31	262.77
Meta PXF	*4.161*	*0.120*	*0.845*	*0.956*	*0.987*	***567.21***	***300.64***	***194.17***
Meta-Base PXF	4.250	0.119	0.843	0.953	0.986	585.94	317.15	200.21
Base-Meta PXF	*2.195*	***0.054***	***0.972***	***0.997***	***0.999***	*754.56*	*428.33*	*258.40*
Pyra PXF	3.243	0.086	0.914	0.983	0.996	781.64	439.35	275.37
Pyra-Base PXF	3.254	0.087	0.915	0.983	0.995	780.42	437.67	263.11
Base-Pyra PXF	***2.192***	*0.055*	***0.972***	***0.997***	***0.999***	*792.97*	*444.23*	*267.44*
MoH PXF	3.561	0.094	0.894	0.975	0.993	867.64	514.36	290.09
MoH-Base PXF	3.508	0.093	0.901	0.978	0.994	870.25	502.47	295.32
Base-MoH PXF	3.562	0.093	0.894	0.976	0.993	837.29	474.92	273.32
NeWCRFs	***2.185***	***0.054***	***0.972***	***0.999***	***0.997***	*1152.80*	*718.32*	*356.12*
Meta NeWCRFs	7.302	0.300	0.522	0.802	0.922	**679.21**	**372.90**	**221.74**
Meta-Base NeWCRFs	4.322	0.122	0.838	0.953	0.986	876.38	530.25	270.77
Base-Meta NeWCRFs	6.221	0.233	0.647	0.872	0.954	877.22	483.00	279.96
Pyra NeWCRFs	5.311	0.186	0.725	0.914	0.973	1001.02	508.04	303.65
Pyra-Base NeWCRFs	3.250	0.087	0.914	0.984	0.996	1093.31	644.32	339.26
Base-Pyra NeWCRFs	5.062	0.177	0.744	0.926	0.977	951.53	505.20	309.29
MoH NeWCRFs	2.272	0.057	0.969	0.996	**0.999**	1041.32	557.87	342.92
MoH-Base NeWCRFs	2.219	0.056	0.970	0.996	**0.999**	1115.38	672.96	352.22
Base-MoH NeWCRFs	2.203	0.055	0.971	0.996	**0.999**	955.48	526.33	316.13
(c)								
METER	5.726	7.299	**0.332**	**0.524**	**0.630**	44.05	30.68	34.50
Meta METER	***5.711***	*7.469*	*0.270*	*.493*	*0.625*	***41.78***	*29.97*	***26.27***
Pyra METER	5.744	**7.122**	0.288	0.501	0.621	51.37	**29.30**	38.01
MoH METER	5.714	7.470	0.275	0.498	0.628	63.78	34.45	35.24
PXF	2.123	**0.052**	0.975	**0.997**	**0.999**	1498.72	738.41	426.68
Continued								

Model	RMSE [m] ↓	Abs_{Rel} ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	I9X79 [s] ↓	I9X10 [s] ↓	XG38 [s] ↓
Meta PXF	*3.426*	*0.094*	*0.892*	*0.977*	*0.995*	***959.37***	***504.63***	**302.80**
Meta-Base PXF	3.454	0.094	0.896	0.978	0.995	984.05	510.99	304.09
Base-Meta PXF	***2.108***	***0.052***	***0.976***	***0.997***	***0.999***	*1314.52*	*729.29*	*388.99*
Pyra PXF	3.273	0.087	0.913	0.983	0.996	1359.07	802.24	430.41
Pyra-Base PXF	3.202	0.086	0.913	0.983	0.996	1383.07	759.66	437.13
Base-Pyra PXF	2.111	**0.052**	0.975	**0.997**	**0.999**	1575.81	748.99	412.63
MoH PXF	3.427	0.089	0.906	0.980	0.995	1781.64	830.96	462.96
MoH-Base PXF	3.399	0.089	0.909	0.980	0.995	2491.49	833.11	473.86
Base-MoH PXF	3.311	0.086	0.913	0.982	0.995	2111.08	785.55	429.66
NeWCRFs	***2.072***	***0.052***	***0.975***	***0.997***	***0.999***	*1863.63*	*969.38*	*500.17*
Meta NeWCRFs	6.952	0.288	0.545	0.815	0.930	**1297.96**	**555.62**	**326.76**
Meta-Base NeWCRFs	3.481	0.096	0.895	0.979	0.995	1423.14	725.79	374.50
Base-Meta NeWCRFs	5.626	0.203	0.691	0.898	0.968	1490.97	796.73	433.06
Pyra NeWCRFs	4.830	0.164	0.762	0.932	0.979	1730.30	837.99	477.72
Pyra-Base NeWCRFs	3.302	0.087	0.910	0.982	0.996	1813.54	969.33	503.52
Base-Pyra NeWCRFs	4.529	0.153	0.785	0.944	0.985	1575.83	801.59	460.03
MoH NeWCRFs	2.176	0.053	0.972	**0.997**	**0.999**	1668.14	869.06	503.80
MoH-Base NeWCRFs	2.123	**0.052**	0.974	**0.997**	**0.999**	1709.84	992.06	518.11
Base-MoH NeWCRFs	2.128	**0.052**	0.974	**0.997**	**0.999**	1616.46	855.36	473.46

Table 2. Experimental results on the tiny (**a**), base (**b**), and large (**c**) models with the KITTI dataset. Values in bold represent the best results between the baseline and the optimisations applied for that model. The lines highlighted in italic show the best trade-offs between RMSE and inference time considering all models. These points are those values that compose the Pareto frontier.

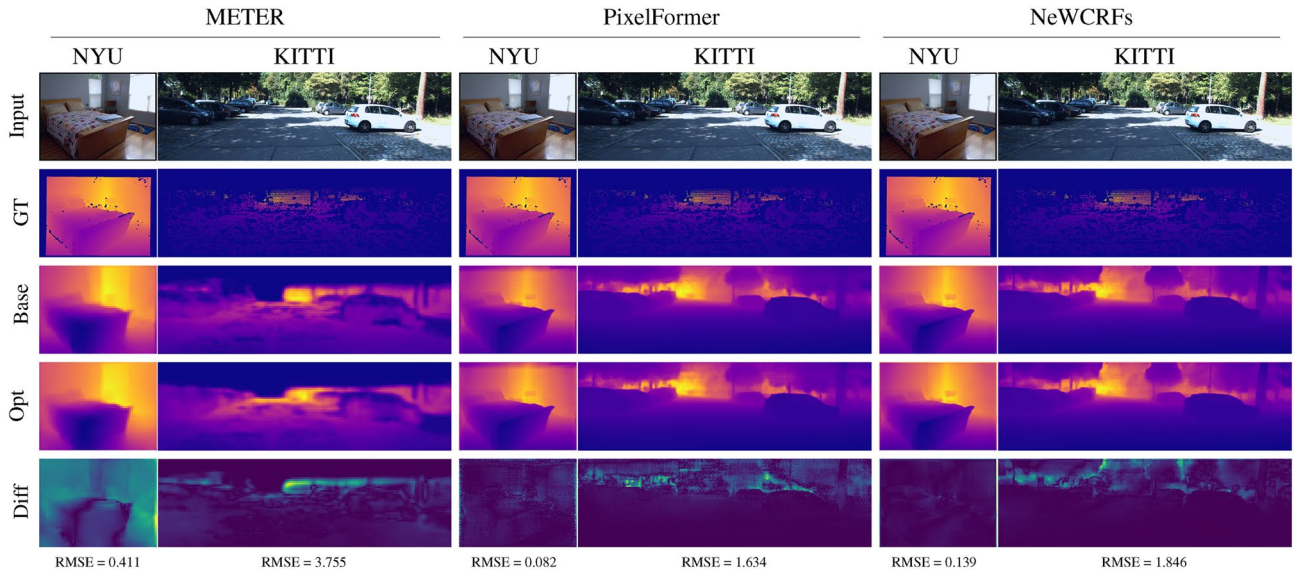

Fig. 3. Visual results for the large versions of the networks. For each model and dataset, we show the input RGB image (Input), the depth ground truth (GT), the prediction of the baseline model (Base), the prediction of the best-performing optimised model (Opt) and the qualitative and quantitative difference, indicated by the RMSE, between these two predictions (Diff).

In most cases, Pyra often finds itself in better performance zones than Meta, but in worse zones in terms of time. This phenomenon is amplified with MoH, which seems to prevail in performance over inference time. Meta is the optimisation most frequently appearing on the Pareto frontier, while Pyra and MoH occasionally approach or reach it. Sometimes, we also find baselines within this set.

So far, all results discussed have been based on the RMSE metric, as indicated at the beginning of this subsection. The RMSE provides an overall view of prediction quality, but has certain limitations. In particular, it tends to heavily penalise larger errors, without accurately representing the relative consistency between predicted depths, as it only measures absolute stability with respect to ground truth[9].

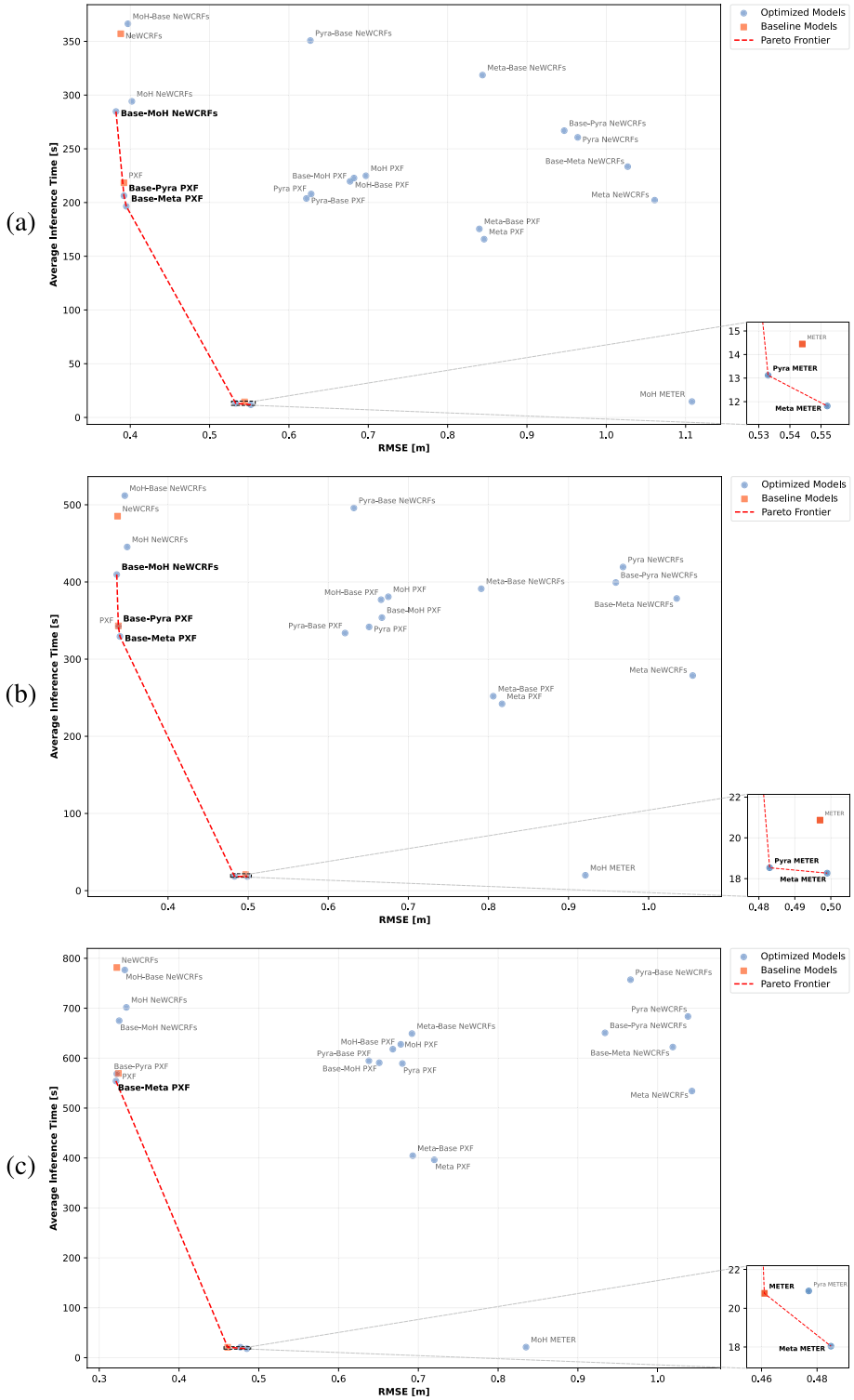

Fig. 4. Pareto Frontiers built considering the RMSE and the mean of the three inference times of the models. The networks in these plots refer to the NYU dataset and are grouped by dataset and size: tiny (**a**), base (**b**), and large (**c**) models. The models in bold represent the optimal trade-offs, lying on the Pareto Frontier.

To address this limitation, we have also represented the Pareto Frontier plots using the Abs_{rel} metric as a quality indicator. This index normalises the error with respect to the true depth, making it particularly suitable in scenarios with varying scales. Furthermore, Abs_{rel} proves to be more robust for assessing relative depth estimation quality[9].

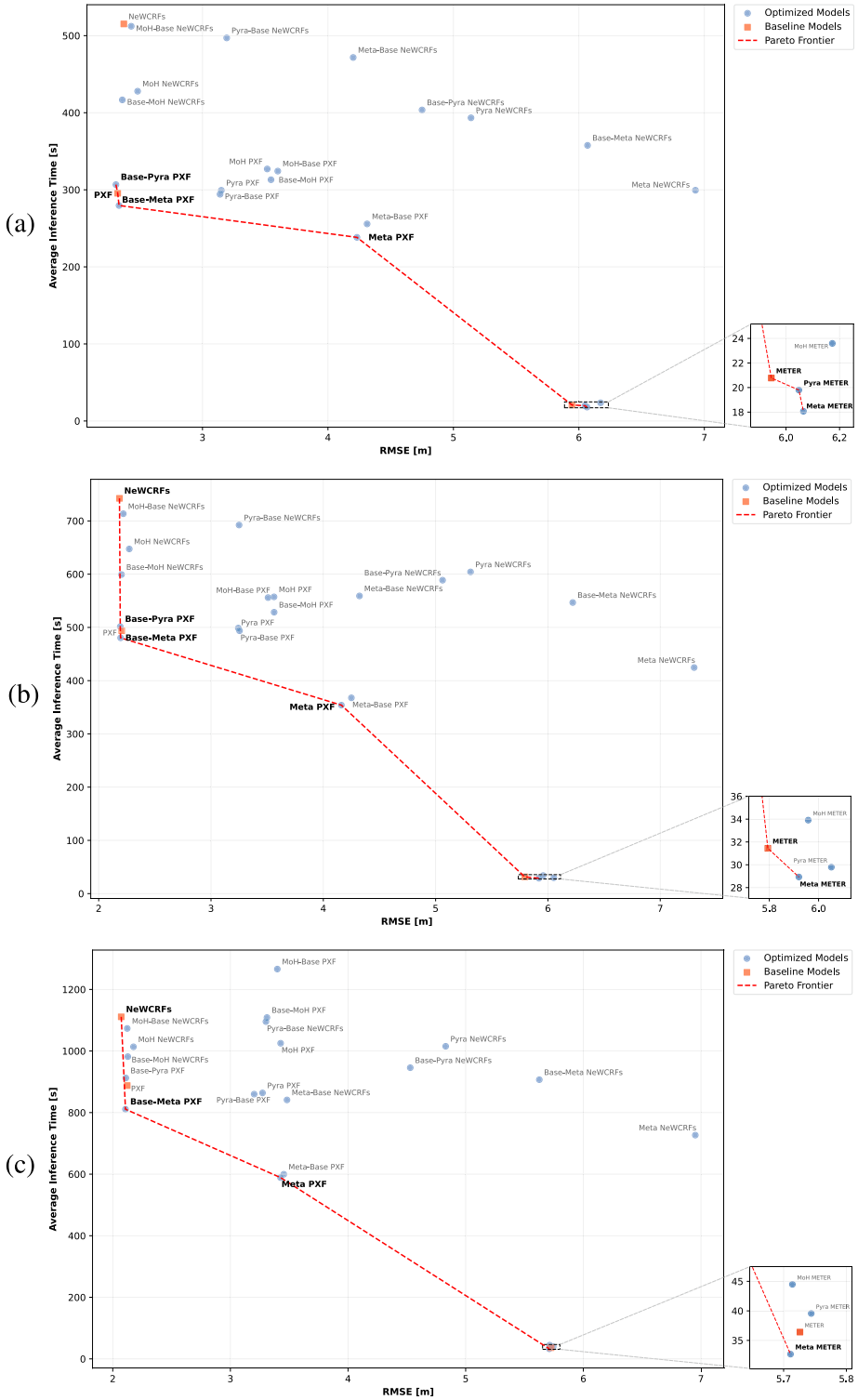

Fig. 5. Pareto Frontiers built considering the RMSE and the mean of the three inference times of the models. The networks in these plots refer to the KITTI dataset and are grouped by dataset and size: tiny (**a**), base (**b**), and large (**c**) models. The models in bold represent the optimal trade-offs, lying on the Pareto Frontier.

Analysing the impact of this change on the Pareto Frontier plots in Fig. 6, it can be seen that the overall situation remains substantially unchanged. Again, the most effective optimisations in terms of trade-off between quality and computational costs are those applied exclusively to the decoder. Among these, the Meta optimisation is frequently found on the optimal frontier, reinforcing what has emerged in previous experiments. It represents a particularly good choice for achieving a good balance between performance and speed of inference.

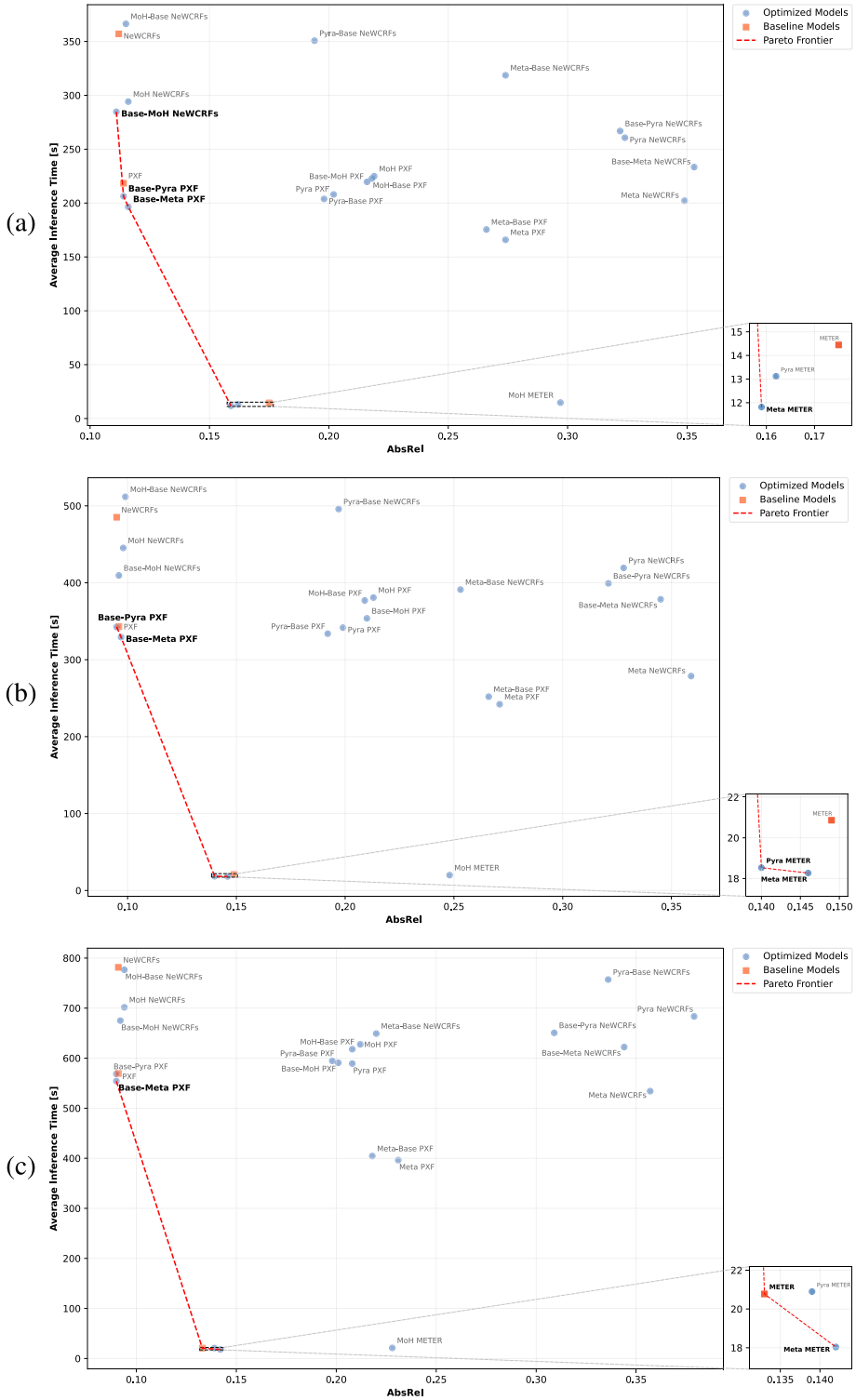

Fig. 6. Pareto Frontiers built considering the Abs_{rel} and the mean of the three inference times of the models. The networks in these plots refer to the NYU dataset and are grouped by dataset and size: tiny (**a**), base (**b**), and large (**c**) models. The models in bold represent the optimal trade-offs, lying on the Pareto Frontier.

A further observation concerns the size of the frontiers. Compared to the graphs based on the RMSE metric, the frontiers obtained with Abs_{rel} tend to include a smaller number of models, thus being more contained. This suggests that the new metric applies a stricter criterion, favouring models that maintain better proportional accuracy across varying depth ranges.

Embedding entropy analysis

The various experiments conducted showed promising results for the optimised architectures, confirming the validity of the structural changes made to the attention modules, especially from an application perspective.

However, focusing on the PixelFormer and NeWCRFs models, the only ones to have undergone optimisations on both the encoder and decoder sides, an interesting result emerges. Among all the combinations tested, the architectures in which only the decoder has been optimised tend to perform best, particularly in terms of accuracy and also with respect to the trade-off between prediction quality and inference speed. This phenomenon is clearly observable in the results reported in Tables 1 and 2. In both datasets, models with an optimised decoder often exceed the baseline, or at least equal it, while maintaining acceptable computational efficiency.

Although not always the fastest in terms of inference time, these models show a more stable and robust balance compared to the fully optimised ones. This behaviour is further confirmed by the analysis of the Pareto Frontier, constructed using both the RMSE and the Abs_{rel} metric, as most of the models that lie on the optimal frontier are precisely those with decoder optimisation.

At this point, in the light of such evident results, it is relevant to question the causes that justify the advantage obtained by the architectures with optimised decoders, and at the same time to understand the reasons why performance worsens when optimisations are applied to the encoder module, especially in the PixelFormer and NeWCRFs models.

The analysis of the results in Tables 1 and 2 lead to the idea that optimised encoders tend to generate embeddings that are less effective in guiding the depth map reconstruction process. Indeed, the quality metrics show a greater difficulty in producing consistent predictions faithful to ground truth. This leads us to formulate the hypothesis that such encoders, once modified, generate degraded latent representations.

Dispersed or unstructured embedding distribution in latent space makes the decoder reconstruction task more complex, as it struggles to predict accurate depth maps. And this is exactly what we observe in our experiments. Optimised decoders seem to work effectively when they receive sufficiently rich and coherent embeddings as input, which does not always seem to be the case when the encoder is modified.

In order to quantify how dispersed the embedding distribution actually is, it is necessary to use a precise metric established in the literature. The objective is to find a measure that indicates how much the embeddings produced by the encoder are representative of a compact and well-defined distribution.

Based on this idea, we leverage the theoretical framework of Variational Autoencoders (VAEs), because in its loss the regularisation term quantifies the trade-off between embedding compactness and informativeness[43]. This term corresponds to the Kullback-Leibler divergence between the distribution learned by the encoder and a standard Gaussian distribution (H). It includes, among other components, the differential entropy of the learned distribution, which quantifies the dispersion of the latent encoding, a value that is useful to gather information needed to verify our hypothesis.

Differential entropy measures, in nats (units of information based on the natural logarithm[44]), the continuous volume occupied by the embeddings. A lower, i.e. more negative, value indicates that the embeddings are more concentrated, so the latent distribution is more compact and less uncertain. While lower, intended as more negative, differential entropy does not necessarily imply higher informativeness with respect to the input, it does ensure a less dispersed embedding distribution, which generally makes the decoder's reconstruction task easier[43].

This metric assumes that the distribution that the distribution of embeddings follows a standard Gaussian. This concept may be too restrictive, since embeddings may present multimodal structures or heavy tails, not captured by a simple covariance matrix. For this reason, it is also useful to analyse information dispersion through non-parametric methods, such as the Kozachenko-Leonenko entropy estimator (H_KL), due to its robustness to non-standard distributions[45].

Again, a lower, more negative, value of the entropy estimated with the Kozachenko-Leonenko estimator, indicates more compact embeddings with less dispersion, while higher values reflect a greater dispersion of the point distribution and a possible loss of structure.

To perform the analysis with the tools described above, the embeddings produced as output by the encoder were considered. This approach was applied to all versions of the PixelFormer and NeWCRFs models, on NYU Depth V2 and KITTI datasets. The METER network was not included in the analysis, as it does not present the problem that is the subject of this study. In fact, the optimisations were only applied to the encoder, which is the only one containing attention modules[14].

It is important to emphasise that the embeddings analysed come from all the architectural combinations considered (baseline, encoder-optimised, decoder-optimised, full-optimised). It might appear that by optimising only the decoder, the encoder remains unchanged compared to the non-optimised version. However, during training, the parameters of each module are updated differently according to the chosen optimisation. Consequently, the encoder of a decoder-only version never exactly coincides with that of the baseline, but also incorporates the changes resulting from the overall optimisation.

On the theoretical basis set out above, the implementation of the tools discussed supports the hypothesis formulated. In all the cases analysed, as shown in Figs. 7 and 8, any modification made to the encoder structure leads to a deterioration in the compactness of the embeddings it produces. A greater dispersion of the embeddings may compromises the decoder's ability to effectively reconstruct the input, with a negative impact on the final quality metrics. This behaviour is consistent for both entropy indices considered, with stronger evidence in the case of the Kozachenko-Leonenko entropy estimator, whose non-parametric nature makes it more robust in non-standard contexts.

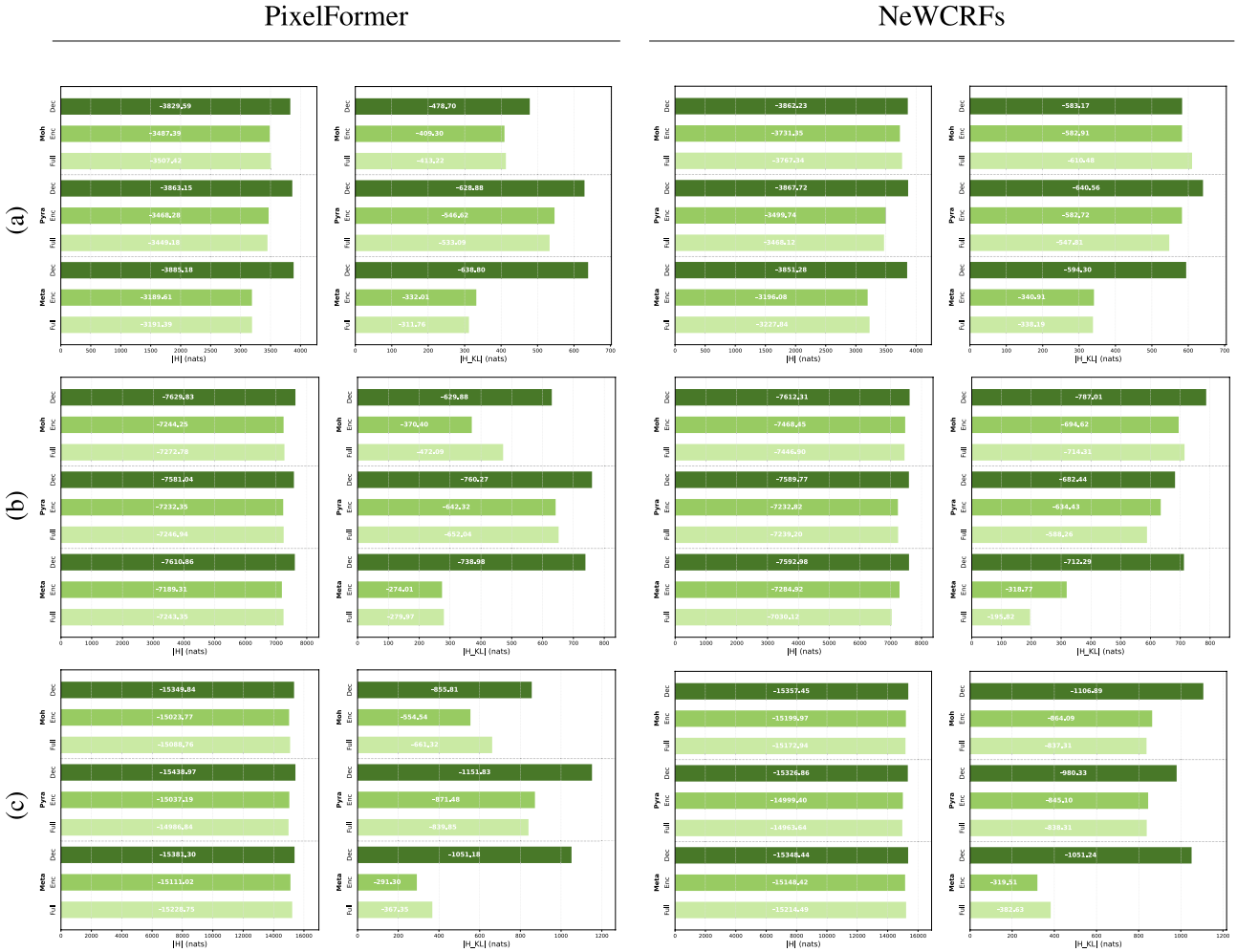

Fig. 7. Embedding dispersion is measured using entropy values, specifically differential Gaussian entropy (left plot) and Kozachenko-Leonenko entropy (right plot), where a lower value indicates a better encoder embedding distribution compactness. Results for each network on the NYU Depth V2 dataset are grouped by model size: tiny (**a**), base (**b**), and large (**c**).

General optimisation comparison

Following the results obtained, it can be observed that optimisations aimed exclusively at the attention module confirm their effectiveness[19,32]. However, at this point in the analysis, it is interesting to compare these optimisations with more generic and widely used techniques. For this reason, the three neural networks analysed in this work were subjected to unstructured pruning and dynamic quantization techniques, both applied in post-training contexts.

The choice fell on these approaches because they were the most suitable for an unbiased comparison, given the fact that no modification to the network is required. to the network is required to fit them. These techniques allow a significant reduction in occupied memory and, in specific contexts, may improve inference times without requiring model retraining. Both techniques were implemented using the native functionality offered by the PyTorch[42] framework.

Focusing on the general optimisations applied, a global unstructured approach was chosen for pruning, based on the L1 norm, i.e. the absolute value of the weights[36]. In this application, 70% of the smallest weights were pruned, selected globally from all convolutional and linear layers in the model. The choice of such a high pruning percentage was motivated by the aim to clearly observe the impact of this optimisation on the behaviour of the model.

As for the quantisation method, a dynamic approach was chosen, limited to the linear layers only. In this case, the weights were converted from 32-bit floating-point representation to 8-bit integer[35].

From the results shown in Table 3, it is clear tha the application of these techniques leads to a worsening of the metrics analysed. In particular, a decrease in prediction quality metrics was to be expected, since the optimisations considered have a strong structural impact on the model, modifying its topology[35,36]. In contrast, the optimisations applied to the attention module alone keep the structure of the architecture almost unchanged[19–21].

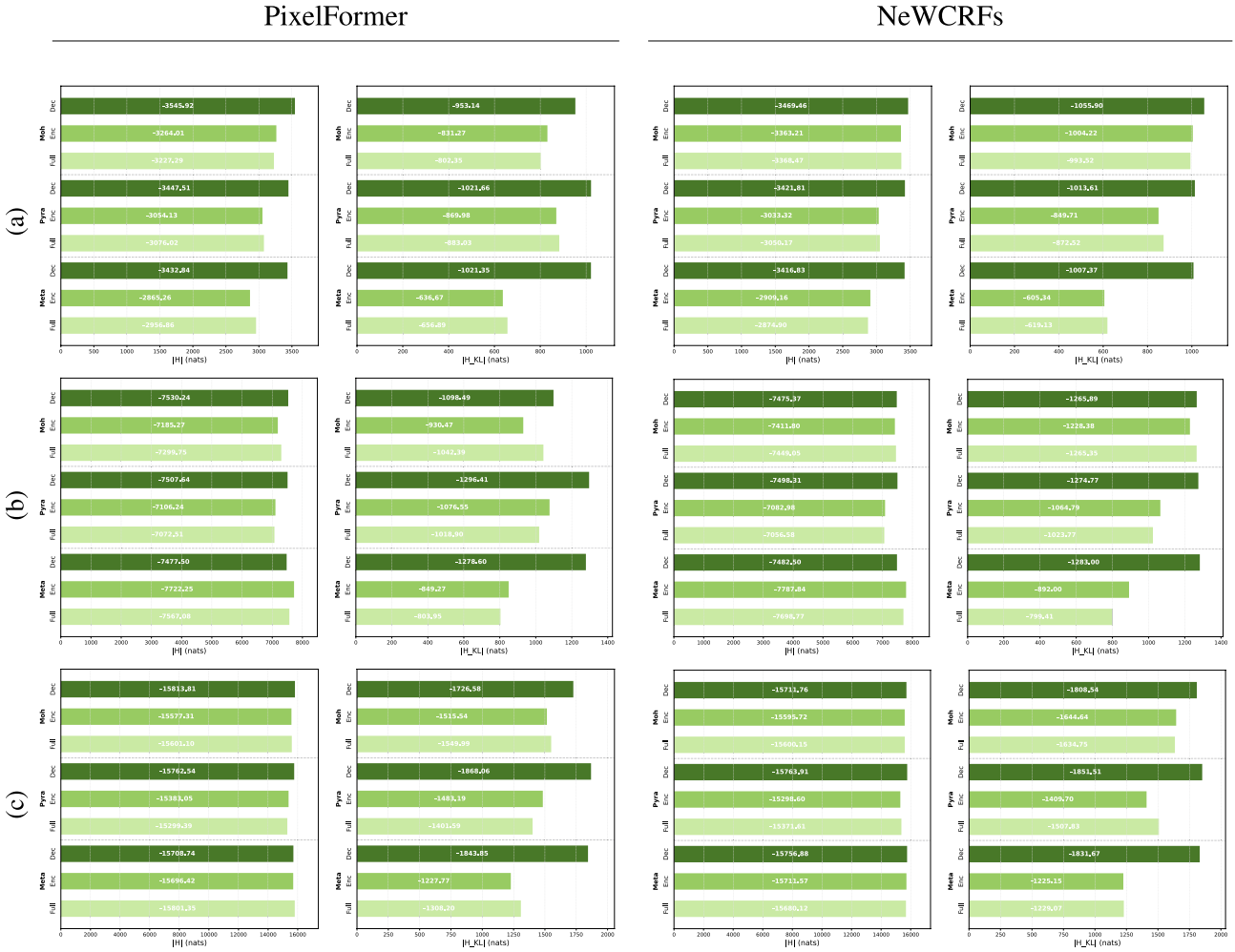

Fig. 8. Embedding dispersion is measured using entropy values, specifically differential Gaussian entropy (left plot) and Kozachenko-Leonenko entropy (right plot), where a lower value indicates a better encoder embedding distribution compactness. Results for each network on the KITTI dataset are grouped by model size: tiny (**a**), base (**b**), and large (**c**).

The negative effects on the speed of inference can be explained in the way these optimisation are applied. In the first case, the unstructured pruning introduces sparse matrices to remove the network connections. Even if that approach reduces the number of effective active weights, those matrices may not be well supported on generic hardware, leading to the possibility of a less efficient computation[36]. For the dynamic quantization, the concept is similar. The post training quantization may not be completely supported or optimised for a general-purpose hardware, providing no guarantee of computational speed-up[35].

Optimisations memory footprint

In addition to assessing the impact of optimisations on the performance and speed of neural networks, it is equally crucial to analyse how the structural characteristics of the models change as a result of a modification to the attention modules. In the context of efficient models, metrics such as the number of parameters and memory occupancy are commonly used to describe the memory footprint of the networks[32].

Although the modifications discussed in this research focus exclusively on lowering the computation complexity of the attention modules and don't primarily focus on memory reduction, observing how they affect the overall network structure can provide useful insights useful insights into the applicability of the modified models in real-world usage scenarios.

Significant observations emerge from Table 4, where the information about the different networks is collected. In particular, it can be seen that Meta optimisation consistently shows the best results in terms of parameters and memory weight among all the techniques analysed. This behaviour is consistent across all the application networks and combinations considered. The advantage of Meta is most likely linked to the way it is implemented[19]. In fact, by entirely replacing the structure of the attention module, including all the linear projection layers, with simple pooling, leads to a substantial reduction in memory and parameters associated with the network.

Model	RMSE [m] ↓	Abs_{Rel} ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	XG38 [s] ↓	Model	RMSE [m] ↓	Abs_{Rel} ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	XG38 [s] ↓
NYU Depth V2							KITTI						
(a)													
METER	**0.544**	**0.175**	**0.778**	**0.946**	**0.983**	**16.47**	METER	**5.945**	**7.408**	**0.287**	**0.485**	**0.604**	**18.13**
METER P	2.789	0.938	0.001	0.001	0.001	15.17	METER P	10.960	1.630	0.023	0.047	0.071	20.26
METER Q	1.334	0.346	0.278	0.560	0.793	17.86	METER Q	6.985	7.390	0.183	0.362	0.499	17.40
PXF	**0.392**	**0.114**	**0.880**	**0.982**	**0.996**	**112.41**	PXF	**2.324**	**0.060**	**0.966**	**0.996**	**0.999**	**160.90**
PXF P	1.159	0.524	0.368	0.651	0.829	136.60	PXF P	17.172	1.609	0.112	0.225	0.337	188.94
PXF Q	1.594	0.746	0.265	0.504	0.706	136.03	PXF Q	11.379	0.502	0.262	0.502	0.725	185.37
NeWCRFs	**0.388**	**0.112**	**0.885**	**0.980**	**0.995**	**170.78**	NeWCRFs	**2.373**	**0.059**	**0.965**	**0.995**	**0.999**	**234.65**
NeWCRF P	2.355	1.177	0.126	0.303	0.509	187.33	NeWCRF P	20.433	2.037	0.076	0.158	0.261	255.80
NeWCRFs Q	1.820	0.845	0.235	0.465	0.662	211.43	NeWCRFs Q	20.951	2.086	0.077	0.160	0.261	303.66
(b)													
METER	**0.497**	**0.149**	**0.811**	**0.951**	**0.987**	**17.10**	METER	**5.794**	**6.625**	**0.302**	**0.504**	**0.618**	**22.16**
METER P	2.777	0.930	0.019	0.019	0.019	19.72	METER P	11.206	1.134	0.191	0.199	0.208	29.89
METER Q	1.487	0.663	0.296	0.555	0.752	19.63	METER Q	6.046	8.031	0.231	0.459	0.607	23.59
PXF	**0.338**	**0.096**	**0.918**	**0.988**	**0.997**	**181.09**	PXF	**2.205**	**0.055**	**0.972**	**0.997**	**0.999**	**262.77**
PXF P	2.573	1.287	0.109	0.262	0.463	234.45	PXF P	26.822	2.704	0.055	0.116	0.188	343.31
PXF Q	1.487	0.692	0.281	0.550	0.738	238.05	PXF Q	13.297	0.433	0.255	0.427	0.571	334.93
NeWCRFs	**0.337**	**0.095**	**0.918**	**0.989**	**0.998**	**241.93**	NeWCRFs	**2.185**	**0.054**	**0.972**	**0.999**	**0.997**	**356.12**
NeWCRF P	2.566	1.283	0.109	0.266	0.464	304.09	NeWCRFs P	26.995	2.722	0.054	0.115	0.187	368.20
NeWCRFs Q	1.870	0.908	0.202	0.428	0.635	358.69	NeWCRFs Q	18.496	1.808	0.087	0.186	0.308	488.34
(c)													
METER	**0.460**	**0.133**	**0.834**	**0.966**	**0.992**	**21.52**	METER	**5.726**	**7.299**	**0.332**	**0.524**	**0.630**	**34.50**
METER P	2.207	0.649	0.039	0.093	0.182	31.30	METER P	10.732	2.291	0.034	0.068	0.103	51.86
METER Q	1.728	0.812	0.223	0.473	0.683	26.41	METER Q	10.955	4.002	0.078	0.134	0.172	35.69
PXF	**0.324**	**0.091**	**0.928**	**0.991**	**0.998**	**294.74**	PXF	**2.123**	**0.052**	**0.975**	**0.997**	**0.999**	**426.68**
PXF P	2.616	1.308	0.105	0.255	0.457	437.33	PXF P	27.298	2.752	0.053	0.113	0.184	683.33
PXF Q	1.387	0.632	0.306	0.570	0.761	430.85	PXF Q	10.530	0.516	0.273	0.544	0.766	729.44
NeWCRFs	**0.322**	**0.091**	**0.929**	**0.992**	**0.998**	**339.68**	NeWCRFs	**2.072**	**0.052**	**0.975**	**0.997**	**0.999**	**500.17**
NeWCRF P	2.656	1.328	0.105	0.248	0.445	477.79	NeWCRF P	27.389	2.761	0.053	0.113	0.183	733.47
NeWCRFs Q	2.327	1.144	0.145	0.329	0.531	565.98	NeWCRFs Q	21.612	2.164	0.070	0.146	0.244	899.43

Table 3. Performance and inference speed of the pruned (P) and quantized (Q) models with respect to the baselines. Results are reported for the tiny (**a**), base (**b**), and large (**c**) variants on the NYU Depth V2 (left) and KITTI (right) datasets.

In contrast, the other optimisations considered show a different pattern regarding their impact on memory footprint. In particular, Pyra optimisation is the technique that leads to the largest increase in the analysed values. This is due to the spatial reduction modules that introduce new layers[20] to consider in the overall network structure. For what concern MoH optimisation, however, the modified multi-head attention routing system[21] does not introduce more overhead, maintaining almost unchanged the network parameters and memory weight.

These results, however, do not significantly impact in the applicability of the modified models. The number of parameters and memory fluctuations remain bounded in a value near the baseline values. At the same time, often this increase is accompanied by an increment in quality of predictions. For this reason, optimisations aimed at the attention modules represent an effective strategy, offering a favourable trade-off between portability, efficiency and prediction quality, even if at the cost of a slight increase in certain computational metrics.

Resource-constrained device experiments

After analysing various aspects of modified models, in order to go into the applicative details of this study, experiments were conducted using limited hardware resources devices. In particular, the Jetson Orin Nano board was considered, a device frequently used in real-world scenarios for tasks involving Deep Learning models. Its low computational resources, compared to an ordinary device, make it an ideal context for evaluating the behaviour of optimised models under conditions closer to real-world conditions.

The experiments conducted on this hardware focused on analysing inference times, with the aim of highlighting the effectiveness of the proposed optimisations. The tests were carried out maintaining the same configuration used in the previous evaluations. In fact, inference was performed sample by sample on the entire test set of datasets used in this research. In this context, to adapt to the applicative scenario, we used only the tiny versions of the networks, which are more suitable for this application.

A detailed analysis of the Table 5 shows how the optimised models perform in the context considered. In particular, it is observed that Meta optimisation systematically leads to faster models than the baseline,

Model	Parameters [M] ↓	Memory [MB] ↓
(a)		
METER	0.71	2.72
Meta METER	**0.68**	**2.60**
Pyra METER	1.08	4.11
MoH METER	0.71	2.73
PXF	88.91	339.15
Meta PXF	**76.08**	**290.22**
Meta-Base PXF	80.26	306.18
Base-Meta PXF	84.72	323.19
Pyra PXF	103.12	393.38
Pyra-Base PXF	97.55	372.11
Base-Pyra PXF	94.48	360.42
MoH PXF	89.03	339.61
MoH-Base PXF	88.98	339.44
Base-MoH PXF	88.95	339.33
NeWCRFs	88.46	337.43
Meta NeWCRFs	**74.23**	**283.18**
Meta-Base NeWCRFs	79.81	304.46
Base-Meta NeWCRFs	82.88	316.15
Pyra NeWCRFs	111.04	423.57
Pyra-Base NeWCRFs	97.09	370.39
Base-Pyra NeWCRFs	102.40	390.61
MoH NeWCRFs	91.41	348.71
MoH-Base NeWCRFs	88.53	337.72
Base-MoH NeWCRFs	91.34	348.43
(b)		
METER	1.45	5.53
Meta METER	**1.40**	**5.35**
Pyra METER	2.29	8.74
MoH METER	1.45	5.54
PXF	140.43	535.71
Meta PXF	**108.28**	**413.06**
Meta-Base PXF	112.47	429.02
Base-Meta PXF	136.25	519.75
Pyra PXF	173.96	663.62
Pyra-Base PXF	168.39	642.34
Base-Pyra PXF	146.01	556.98
MoH PXF	140.72	536.80
MoH-Base PXF	140.67	536.62
Base-MoH PXF	140.48	535.88
NeWCRFs	139.98	533.99
Meta NeWCRFs	**106.44**	**406.02**
Meta-Base NeWCRFs	112.01	427.30
Base-Meta NeWCRFs	134.40	512.71
Pyra NeWCRFs	181.88	693.81
Pyra-Base NeWCRFs	167.94	640.62
Base-Pyra NeWCRFs	153.92	587.17
MoH NeWCRFs	143.10	545.90
MoH-Base NeWCRFs	140.22	534.90
Base-MoH NeWCRFs	142.86	544.98
(c)		
METER	3.30	12.57
Meta METER	**3.22**	**12.29**
Pyra METER	5.53	21.08
MoH METER	3.30	12.58
PXF	270.90	1033.39
Continued		

Model	Parameters [M] ↓	Memory [MB] ↓
Meta PXF	**203.82**	**777.53**
Meta-Base PXF	208.01	793.49
Base-Meta PXF	266.71	1017.43
Pyra PXF	339.34	1294.49
Pyra-Base PXF	333.77	1273.22
Base-Pyra PXF	276.47	1054.66
MoH PXF	271.47	1035.56
MoH-Base PXF	271.42	1035.39
Base-MoH PXF	270.94	1033.56
NeWCRFs	270.44	1031.67
Meta NeWCRFs	**201.98**	**770.49**
Meta-Base NeWCRFs	207.56	791.76
Base-Meta NeWCRFs	264.87	1010.39
Pyra NeWCRFs	347.26	1324.68
Pyra-Base NeWCRFs	333.32	1271.50
Base-Pyra NeWCRFs	284.39	1084.85
MoH NeWCRFs	273.85	1044.66
MoH-Base NeWCRFs	270.97	1033.67
Base-MoH NeWCRFs	273.33	1042.66

Table 4. Number of network parameters and memory footprint relative to the tiny (a), base (b), and large (c) size. Values in bold represent the best values between the baseline and the optimisations applied for that model.

Model	Jetson NYU [s] ↓	Jetson KITTI [s] ↓
METER	328.49	343.79
Meta METER	215.33	221.21
Pyra METER	310.59	336.41
MoH METER	344.43	351.07
PXF	1924.38	2765.59
Meta PXF	1373.43	1970.51
Meta-Base PXF	1543.57	2216.83
Base-Meta PXF	1722.62	2469.82
Pyra PXF	1915.70	2709.43
Pyra-Base PXF	1927.62	2712.86
Base-Pyra PXF	1906.42	2735.54
MoH PXF	2027.07	2876.13
MoH-Base PXF	2000.42	2844.66
Base-MoH PXF	1942.72	2809.53
NeWCRFs	2316.55	3185.19
Meta NeWCRFs	1615.87	2235.18
Meta-Base NeWCRFs	1958.86	2730.05
Base-Meta NeWCRFs	2007.11	2779.84
Pyra NeWCRFs	2380.42	3391.28
Pyra-Base NeWCRFs	2323.66	3241.11
Base-Pyra NeWCRFs	2368.85	3281.75
MoH NeWCRFs	2482.51	3456.72
MoH-Base NeWCRFs	2375.26	3329.97
Base-MoH NeWCRFs	2292.42	3409.21

Table 5. Inference times in seconds from the experiments on Jetson Orin Nano using the tiny versions of the networks.

confirming that its modification to the attention module is often the most efficient choice. Models optimised with Pyra also show an improvement in inference times, particularly when the optimisation is applied to only one module of the network, although performance is generally lower than in the Meta case. MoH optimisation, on the other hand, is the least suitable in this context, probably due to the overhead introduced by the routing mechanism, which may not fit well with resource-limited devices.

This behaviour is also confirmed in the case of the METER model, where once again MoH optimisation is the one with the worst performance, whereas Meta and Pyra allow significantly shorter inference times[46,47].

Conclusions

This work analyses the impact of optimisations on ViT-based architectures for MDE by conducting experiments on indoor and outdoor scenario. In particular, the focus of the research was the application of efficient attention modules. To better understand the impact of these optimisations have been applied at different levels within each network, targeting the entire architecture, the encoder and the decoder. A detailed analysis was conducted to assess how these optimisations, applied to one of the most computationally intensive components of the models, influenced network performance in terms of quality and speed. Given the importance of a precise balance between these two objectives, the Pareto Frontier was important in analysing the trade-offs and analytically obtaining the optimal ones.

Some modifications led to optimised models with improved inference speed but a significant loss in performance, as seen in fully modified NeWCRFs with Meta or the application of MoH on METER. On the other hand, some results showed noteworthy cases where the optimised models have reported promising results, sometimes even surpassing the respective baseline, together with a consistent improvement in inference time. This is the case of the models optimised only on the decoder part, where the Pyra and Meta optimisations for PixelFormer and MoH for NeWCRFs have introduced promising models from the point of view of performance and speed.

These findings suggest new potential research directions based on the provided results. In particular, future works could investigate how broader optimisation techniques, such as quantisation, knowledge distillation, and pruning, behave when applied to the whole network and then to specific components. In addition, it may be worth investigating how the optimisations presented in this work behave on particularly complex dense tasks, such as optical flow estimation, where the balance between performance and speed is crucial in practical applications. In all of these cases, the application of the Pareto Frontier could serve as a valuable tool for objectively analysing trade-offs in Deep Learning tasks.

Data availability

The data used in this research are publicly available. The NYU Depth V2 dataset can be obtained at the following link. Similarly, the KITTI dataset is available at the link

Received: 28 March 2025; Accepted: 6 June 2025
Published online: 05 July 2025

References

1. Voulodimos, A., Doulamis, N., Doulamis, A. & Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* https://doi.org/10.1155/2018/7068349 (2018).
2. Islam, S. et al. A comprehensive survey on applications of transformers for deep learning tasks. *Expert Syst. Appl.* **241**, 122666. https://doi.org/10.1016/j.eswa.2023.122666 (2024).
3. Zhou, T., Fan, D., Cheng, M., Shen, J. & Shao, L. RGB-D salient object detection: A survey. *Comput. Vis. Media* **7**, 37–69. https://doi.org/10.1007/s41095-020-0199-z (2021).
4. Y. Hu, Z. Chen & W. Lin. RGB-D semantic segmentation: A review. In *Proceedings of the IEEE International Conference on Multimedia & Expo Workshops*. https://doi.org/10.1109/ICMEW.2018.8551554(2018).
5. Maiano, L., Papa, L., Vocaj, K. & Amerini, I. DepthFake: A depth-based strategy for detecting deepfake videos. In *Proceedings of the Pattern Recognition, Computer Vision, and Image Processing. ICPR 2022 International Workshops and Challenges*, **13646**, 17–31. https://doi.org/10.1007/978-3-031-37745-7_2(2023).
6. Dong, X., Garratt, M. A., Anavatti, S. G. & Abbass, H. A. Towards real-time monocular depth estimation for robotics: A survey. *IEEE Trans. Intell. Transp. Syst.* **23**, 16940–16961. https://doi.org/10.1109/TITS.2022.3160741 (2022).
7. Wang, Y.et al. Pseudo-LiDAR from visual depth estimation: bridging the gap in 3D object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8437–8445. https://doi.org/10.1109/CVPR.2019.00864(2019).
8. El Jamiy & F. and Marsh, R. Survey on depth perception in head mounted displays: distance estimation in virtual reality, augmented reality, and mixed reality. *IET Image Processing*, **13**, 707–712. https://doi.org/10.1049/iet-ipr.2018.5920(2019).
9. Zhao, C., Sun, Q., Zhang, C., Tang, Y. & Qian, F. Monocular depth estimation based on deep learning: An overview. *Sci. China Technol. Sci.* **63**, 1612–1627. https://doi.org/10.1007/s11431-020-1582-8 (2020).
10. Vaswani, A. et al. Attention is All you Need. In *Proceedings of the International Conference on Neural Information Processing Systems*, 6000–6010. https://doi.org/10.5555/3295222.3295349(2017).
11. Dosovitskiy, A. et al. An image is worth 16x16 words: transformers for image recognition at scale. arXiv:2010.11929v2 (2021).
12. Agarwal, A. & Arora, C. Attention attention everywhere: monocular depth prediction with skip attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 5850–5859. https://doi.org/10.1109/WACV56688.2023.00581(2023).
13. Yuan, W., Gu, X., Dai, Z., Zhu, S. & Tan, P. Neural window fully-connected CRFs for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3906–3915. https://doi.org/10.1109/CVPR52688.2022.00389 (2022).
14. Papa, L., Russo, P. & Amerini, I. METER: A mobile vision transformer architecture for monocular depth estimation. *Trans. Circ. Syst. Video Technol.* **33**, 5882–5893. https://doi.org/10.1109/TCSVT.2023.3260310 (2023).
15. Ye, N. et al. Ood-control: generalizing control in unseen environments. *IEEE Trans. Pattern Anal. Mach. Intell.* **46**, 7421–7433. https://doi.org/10.1109/TPAMI.2024.3395484 (2024).

16. Zhu, L. et al. Vision-language alignment learning under affinity and divergence principles for few-shot out-of-distribution generalization. *Int. J. Comput. Vis.* **132**, 3375–3407. https://doi.org/10.1007/s11263-024-02036-4 (2024).
17. Ye N. et al. Ood-bench: quantifying and understanding two dimensions of out-of-distribution generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3906–3915. https://doi.org/10.1109/CVPR52688.2022.00779(2022).
18. L.,Zhu,Y., Yang, Q.,Gu, X. Wang, C., Zhou, N.Y. CRoFT: Robust fine-tuning with concurrent optimization for ood generalization and open-set OOD detection. arXiv:abs/2405.16417 (2024).
19. Yu, W. et al. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10809–10819. https://doi.org/10.1109/CVPR52688.2022.01055(2022).
20. Wang, W. et al. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, 548–558. https://doi.org/10.1109/ICCV48922.2021.00061(2021).
21. Jin, P., Zhu, B., Yuan, L. & Yan, S. MoH: multi-head attention as mixture-of-head attention. arXiv:2410.11842 (2024).
22. Nia, V.P., Ghaffari, A., Zolnouri, M. & Savaria, Y. Rethinking pareto frontier for performance evaluation of deep neural networks. arXiv:abs/2202.09275 (2022).
23. Peng, C. et al. PNAS-MOT: multi-modal object tracking with pareto neural architecture search. *IEEE Robot. Autom. Lett.* **9**, 4377–4384. https://doi.org/10.1109/LRA.2024.3379865 (2024).
24. Silberman, N., Hoiem, D., Kohli, P. & Fergus, R. Indoor segmentation and support inference from RGBD images. *Comput. Vis. ECCV* **2012**(7576), 746–760. https://doi.org/10.1007/978-3-642-33715-4_54 (2012).
25. Geiger, A., Lenz, P., Stiller, C. & Urtasun, R. Vision meets robotics: the KITTI dataset. *Int. J. Robot. Res.* **32**, 1231–1237. https://doi.org/10.1177/0278364913491297 (2013).
26. O'Shea, K. & Nash, R. An Introduction to Convolutional Neural Networks. arXiv:1511.08458v2 (2015).
27. Eigen, D., Puhrsch, C. & Fergus, R. Depth map prediction from a single image using a multi-scale deep network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, vol. 2, pp. 2366–2374. https://doi.org/10.5555/2969033.2969091(2014).
28. Salman, K. et al. Transformers in vision: A survey. *ACM Comput. Surv.* **54**, 1–41. https://doi.org/10.1145/3505244 (2022).
29. Z. Liu et al. Swin transformer: hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9992–10002. https://doi.org/10.1109/ICCV48922.2021.00986(2021).
30. J. Yan, H. Zhao, P. Bu & Y. Jin. Channel-wise attention-based network for self-supervised monocular depth estimation. In *Proceedings of the International Conference on 3D Vision*, 464–473. https://doi.org/10.1109/3DV53792.2021.00056(2021).
31. Ning, C. & Gan, H. Trap attention: monocular depth estimation with manual traps. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5033–5043. https://doi.org/10.1109/CVPR52729.2023.00487(2023).
32. Papa, L., Russo, P., Amerini, I. & Zhou, L. A survey on efficient vision transformers: Algorithms, techniques, and performance benchmarking. *IEEE Trans. Pattern Anal. Mach. Intell.* **46**, 7682–7700. https://doi.org/10.1109/TPAMI.2024.3392941 (2024).
33. Zhou, Y., Yi, Z. & Yen, G. G. Efficient fine-tuning of vision transformer via path-augmented parameter adaptation. *Inform. Sci.* **703**, 121948. https://doi.org/10.1016/j.ins.2025.121948 (2025).
34. Zhou, Y. et al. Multiobjective evolutionary generative adversarial network compression for image translation. *IEEE Trans. Evolut. Comput.* **28**, 798–809. https://doi.org/10.1109/TEVC.2023.3261135 (2024).
35. Liu, Z., Wang, Y., Han, K., Ma, S. & Gao, W. Post-training quantization for vision transformer. arXiv:abs/2106.14156 (2021).
36. Zhu, M., Tang, Y. & Han, K. Vision transformer pruning. arXiv:abs/2104.08500 (2021).
37. Touvron H. et al. Training data-efficient image transformers and distillation through attention. arXiv:abs/2012.12877 (2021).
38. Schiavella, C., Cirillo, L., Papa, L., Russo, P. & Amerini, I. Optimize vision transformer architecture via efficient attention modules: A study on the monocular depth estimation task. In *Image Analysis and Processing - ICIAP 2023 Workshops*, **14365**, 383–394. https://doi.org/10.1007/978-3-031-51023-6_32(2024).
39. Voita, E., Talbot, D., Moiseev, F., Sennrich, R. & Titov, I. Analyzing multi-head self-attention: specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5797–5808. https://doi.org/10.18653/v1/P19-1580(2019).
40. Kingma, D.P. & Ba, J. Adam: A method for stochastic optimization. arXiv:abs/1412.6980 (2017).
41. Loshchilov, I. & Hutter, F. Decoupled weight decay regularization. arXiv:abs/1711.05101. (2019)
42. Paszke, A. et al. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 8026-8037. https://doi.org/10.5555/3454287.3455008(2019).
43. Kingma, D.P., & Welling, M. Auto-encoding variational bayes. arXiv:abs/1312.6114 (2022).
44. Cover, T. M. & Thomas, J. A. Entropy, relative entropy, and mutual information. *Elements Inf. Theory* **2**, 13–55. https://doi.org/10.1002/047174882X.ch2 (2006).
45. Kozachenko, L. F. & Leonenko, N. N. Sample estimate of the entropy of a random vector. *Problems Inform. Trans.* **23**(2), 95–101 (1987).
46. Hua, Y. & Tian, H. Depth estimation with convolutional conditional random field network. *Neurocomputing* **214**, 546–554. https://doi.org/10.1016/j.neucom.2016.06.029 (2016).
47. Zhang, N., Nex, F., Vosselman, G. & Kerle,N. Lite-mono: A lightweight CNN and transformer architecture for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18537–18546. https://doi.org/10.1109/CVPR52729.2023.01778(2023).

Author contributions
C.S. and L.C. took care of the experimental part. All authors participated in the drafting and reviewing of the manuscript.

Funding
This study has been financially supported by Sapienza University funds (RM123188F4A97C9C).

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to C.S.

Reprints and permissions information is available at www.nature.com/reprints.