International Journal of Computer Vision (2026) 134:295
https://doi.org/10.1007/s11263-026-02873-5

Beyond MACs: Hardware Efficient Architecture Design for Vision Backbones

Moritz Nottebaum[1] · **Matteo Dunnhofer**[1,2] · **Christian Micheloni**[1]

Received: 25 February 2025 / Accepted: 27 April 2026

Abstract

Vision backbone networks play a central role in modern computer vision. Enhancing their efficiency directly benefits a wide range of downstream applications. To measure efficiency, many publications rely on MACs (Multiply Accumulate operations) as a predictor of execution time. In this paper, we experimentally demonstrate the shortcomings of such a metric, especially in the context of edge devices. By contrasting the MAC count and execution time of common architectural design elements, we identify key factors for efficient execution and provide insights to optimize backbone design. Based on these insights, we present LowFormer, a novel vision backbone family. LowFormer features a streamlined macro and micro design that includes Lowtention, a lightweight alternative to Multi-Head Self-Attention. Lowtention not only proves more efficient, but also enables superior results on ImageNet. Additionally, we present an edge GPU version of LowFormer, that can further improve upon its baseline's speed on edge GPU and desktop GPU. We demonstrate the LowFormer's wide applicability by evaluating it on smaller image classification datasets, as well as adapting it to several downstream tasks, such as object detection, semantic segmentation, image retrieval, and visual object tracking. LowFormer models consistently achieve remarkable speed-ups across various hardware platforms compared to recent state-of-the-art backbones. Code and models are publicly available at https://github.com/altair199797/LowFormer.

Keywords Vision backbones · Efficient attention · Hardware efficiency · Edge devices

1 Introduction

In many computer vision applications, it is critical to achieve accurate predictions in the shortest time possible. This is important in real-time domains such as robotics (Wen et al., 2024; Dunnhofer et al., 2021; Matthies et al., 2007), autonomous driving (Gopalkrishnan et al., 2024; Jiang et al., 2023), video surveillance (Suzuki & Aoki, 2024; Upmanyu et al., 2009), and user assistance (Mahendran et al., 2021; Tan et al., 2023; Leo et al., 2017), and especially when these systems have to be deployed on mobile and edge devices (Vasu et al., 2023b; Ganesh et al., 2022).

Nowadays, vision backbone networks are critical components of such computer vision systems. They are used to generate representations that support a wide range of high-level tasks for image (Carion et al., 2020; Kirillov et al., 2023; Liu et al., 2025; Ma et al., 2021) and video understanding (Yan et al., 2021; Dunnhofer et al., 2023; Kong & Fu, 2022). Improving the efficiency of vision backbones is, therefore, a crucial step towards enhancing the running time of many computer vision pipelines.

Since the introduction of deep convolutional networks (LeCun et al., 1989), vision backbones have evolved to balance accuracy and efficiency. Recent architectures (Vasu et al., 2023a; Ma et al., 2024; Zhu et al., 2023; Chen et al., 2022a) combine convolutional (Sandler et al., 2018; He et al., 2016) and attention layers (Bolya et al., 2022; Vaswani et al., 2017; Wang et al., 2020): convolutions extract local image features, while attention captures global relationships by aggregating information across the image. To develop com-

Communicated by Jiri Matas.

✉ Moritz Nottebaum
nottebaum.moritz@spes.uniud.it

Matteo Dunnhofer
matteo.dunnhofer@uniud.it

Christian Micheloni
christian.micheloni@uniud.it

[1] Machine Learning and Perception Lab, University of Udine, Via delle Scienze 206, Udine 33100, UD, Italy

[2] Centre for Vision Research, York University, 4700 Keele St, Toronto M3J 1P3, ON, Canada

Published online: 29 May 2026

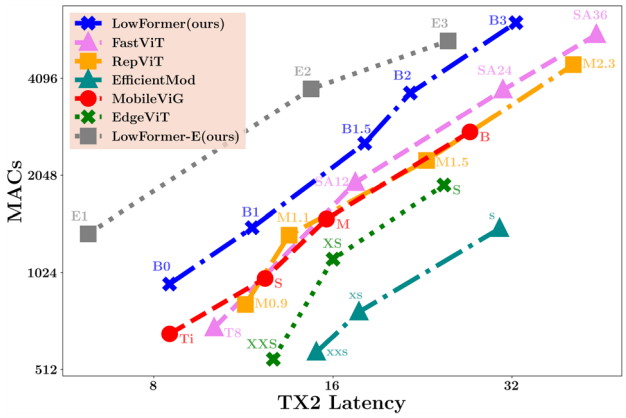

Fig. 1 Comparison of hardware efficiency of different vision backbone architecture families on the Nvidia Jetson TX2. Models in the top-left offer the best hardware efficiency on the Jetson TX2. Both axes are in logarithmic scale. LowFormer base models (B0-B3) outperform all architectures in hardware efficiency, with edge variants (E1-E3) further enhancing efficiency.

putationally efficient deep learning models, including vision backbones, researchers commonly count and minimize Multiply and Accumulate operations (MACs) (Zhu et al., 2023). Simply put, this metric counts the number of multiplications and additions performed by a neural network to compute the output from input data. In other words, MACs can be viewed as a measure of the "tasks" a model must perform to produce an output. Generally, the fewer tasks required, the faster and more efficient the model will be (Tan, 2019a). But research has also shown a strong correlation between a model's increased MAC count and its accuracy, with networks having higher MAC counts achieving better prediction accuracy (Tan, 2019a; Zhai et al., 2022). In light of this evidence, the research community has increasingly focused on developing backbone architectures that maximize accuracy and minimize the number of MACs (Zhu et al., 2023; Chen et al., 2022a; Tan, 2019a). By doing so, it is claimed that the accuracy-speed trade-off is optimized.

We argue that MAC count is not always the best measure of backbone efficiency, and the reduction of MACs does not necessarily translate to a backbone that achieves reduced execution time. This disproportion stems from factors like memory access costs and the degree of parallelism (Dehghani et al., 2021; Liu et al., 2023; Vasu et al., 2023b; Ma et al., 2018a). The former identifies the delay caused by an operation that must wait for the retrieval of its operands from the memory, leading to idle periods that hinder overall execution time (Chen et al., 2023). The latter refers to the parallel execution of core operations like multiplications and additions on modern hardware. In these settings, the number of MACs still counts all the tasks required to produce the output, but fails to account for time saved by performing multiple operations simultaneously.

As a result, models vary in their hardware efficiency, leading to differences in execution time for the same amount of MAC operations. In Figure 1, we analyze the hardware efficiency of different vision backbone families, by relating MAC count and actual execution time – referred to as latency – on an Nvidia Jetson TX2. The figure depicts significant differences in hardware efficiency between the compared vision backbone architecture families. The MAC count of models within an architecture family (e.g. LowFormer) correlates well with execution time (TX2 latency), meaning they feature a similar hardware efficiency. However, the hardware efficiency differs significantly between the architecture families. The EfficientMod (Ma et al., 2024) models (xxs, xs and s) for example have a similar latency as LowFormer-B1.5, B2 and B3, but a fraction of their MAC operations. This emphasizes the importance of a sound architecture design, as a measure to improve hardware efficiency.

Determining the extent to which a model's architecture influences hardware efficiency, due to memory access cost and level of parallelism, is complex and heavily dependent on hardware-specific implementation details (Qin et al., 2025). We believe that latency is a more effective metric for evaluating vision backbone efficiency, as model efficiency cannot be determined by MAC count alone but must be tested across various execution devices.

Many recent publications have contributed to improve the understanding of the relationship between MACs and execution time on real devices (Vasu et al., 2023b; Chen et al., 2023; Ma et al., 2018b; Wang et al., 2024), highlighting architectural configurations that deteriorate model speed. In this paper, we add to this knowledge by pointing out additional factors, like: the operating resolution for convolutions; the shortcomings of the frequently applied mobile inverted bottleneck block (Sandler et al., 2018); and the considerable inefficiency of Multi-Head Self-Attention (MHSA) (Vaswani et al., 2017). All of these factors can slow down model execution, however in what magnitude depends on the hardware device, wherefore we exploit a variety of different devices for a new hardware efficiency study.

We rely on the outcomes of such a study to develop LowFormer, a new class of vision backbone networks that mitigates the impact of the aforementioned factors. By minimizing execution time instead of MAC count, LowFormer is able to achieve a new position in the state-of-the-art speed-accuracy trade-off (see Figure 2), exceeding previous approaches on a variety of hardware devices. A key component in LowFormer's architecture is Lowtention, a new and lightweight adaptation of MHSA (Vaswani et al., 2017). LowFormer features a simple micro and macro design, enabling scalability from low model size (LowFormer-B0) to higher (LowFormer-B3). The backbone family includes five models (B0, B1, B1.5, B2, B3), achieving top-1 accuracy on the ImageNet-1K (Deng et al., 2009) dataset ranging

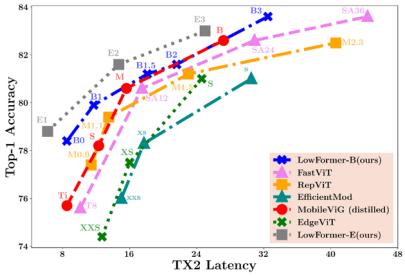

Fig. 2 Comparison of Nvidia Jetson TX2 latency and top-1 accuracy for state-of-the-art vision backbones with LowFormer. Models in the top-left offer the best speed-accuracy trade-off. LowFormer consistently achieves lower latency for similar accuracy. Its edge variants (E1/E2/E3) further enhance this trade-off over the base models (B0-B3).

from 78.4% to 83.64%. We further extend the architecture family by three models, namely LowFormer-E1/E2/E3, which are specifically targeted for edge GPU devices and are derivations of the original LowFormer design. To confirm the applicability of LowFormer backbones in downstream tasks, we evaluated LowFormer's transfer learning capabilities on several image classification datasets. Furthermore, we integrated our backbones in object detection and semantic segmentation frameworks. Additionally, we utilize the embeddings of LowFormer backbones for image retrieval, and present a LowFormer-based visual object tracking architecture. In all of these applications, LowFormer contributes significantly to improve hardware efficiency while maintaining or even increasing accuracy.

The contributions of this paper can be summarized as follows:

- We carry out a new exhaustive hardware efficiency analysis of key design elements in vision backbone architectures. We show how the MAC count of those elements translates to execution time on several devices, and highlight their differences for overall model efficiency.
- We present LowFormer, a new family of vision backbones that features a hardware-efficient macro design and a new lightweight attention operation. These backbone models are faster in terms of latency and throughput compared to models with similar accuracy.
- We further propose three edge GPU variants of LowFormer, that improve upon the base models in terms of edge GPU efficiency.
- We show that LowFormer models generalize well to several downstream tasks, such as image classification, object detection, semantic segmentation, image retrieval, and single object tracking.

2 Related Work

2.1 Hardware-Efficient Model Design

Achieving the highest accuracy at all computational cost has long since ceased to be the only goal in vision backbone networks (Tan, 2019a). An ever increasing share of research focuses on developing the most efficient architecture, subsequently achieving the best accuracy-speed trade-off (Chen et al., 2022b; Munir & Avery, 2023; Howard et al., 2019). Earlier approaches equated speed with the minimal amount of MACs (Tang et al., 2022; Han et al., 2020; Chen et al., 2022a), while more recent research increasingly aims for models that reduce latency or throughput – i.e. number of images processed in a second – on several different types of hardware such as desktop GPU and CPU (Ma et al., 2018b; Yun, 2024; Cai et al., 2023), mobile NPU and GPU (Vasu et al., 2023b), or microcontroller CPU (Paissan et al., 2022). Within these approaches, the cost of memory access and the degree of hardware parallelism have become important factors for efficient model design, as they can have a significant impact on model speed (Chen et al., 2023; Qin et al., 2025). The design of EfficientViT (Liu et al., 2023) demonstrates that MHSA introduces a higher memory access cost compared to the Feed-Forward Network (FFN) within the transformer block. Therefore, this architecture proposes increasing the proportion of FFN operations relative to MHSA, resulting in improved efficiency without compromising accuracy. MobileOne (Vasu et al., 2023b), on the other hand, is based on an analysis of how activation functions and multi-branch architectures impact latency on mobile devices. ShuffleNetV2 (Ma et al., 2018b) and FasterNet (Chen et al., 2023) were proposed on the observation that grouped convolutions are executed inefficiently on GPUs due to their high memory access costs (Chen et al., 2023).

In this paper, we draw inspiration from these insights. However, rather than ungrouping a portion of the convolutions (Ma et al., 2018b) or introducing a new micro design (Chen et al., 2023), we study the impact of fusing depthwise and pointwise convolutions on execution time. Additionally, we examine how the operating resolution of convolutional layers impacts latency, and further explore strategies to effectively mitigate the significant efficiency drop caused by increasing input resolution for MHSA. To the best of our knowledge, we are the first to assemble a diverse set of execution devices to perform these efficiency experiments and analyze how different hardware platforms compare.

This paper extends (Nottebaum & Dunnhofer, 2025), where the LowFormer architecture was initially presented.

In this version, we provide additional contributions, tailored to the domain of edge computing. We expand the execution time analysis to cover several edge devices and examine the efficiency of MHSA for increased input resolution, exploring ways to improve it. We also consistently compare the efficiency of LowFormer on edge devices with the top competing models, across most benchmarks. We further extend the ablation study to provide a stronger rationale for our design choices. Additionally, we propose three edge GPU variants of LowFormer and demonstrate empirically their viability. We evaluate LowFormer on several new downstream tasks, including image classification and image retrieval. Lastly, we present LowFormer-Track, which improves performance of the SMAT (Gopal, 2024) single object tracking (SOT) framework by integrating LowFormer's design principles.

2.2 Convolutions, Attention and MLP in Architecture Design

Most modern backbone architectures consist entirely or partially of three main building blocks: convolutions, attention mechanisms, and multi-layer perceptrons (MLPs) (Yun, 2024; Vasu et al., 2023a). Some approaches, called hybrid models, combine all three in their design (Liu et al., 2023; Qin et al., 2025; Yun, 2024; Vasu et al., 2023a), some relieve of the MLP (Cai et al., 2023), while others solely rely on convolutions (Wang et al., 2024; Vasu et al., 2023b). The latter focus towards efficient mobile execution, where convolutions have been shown to achieve superior latency results (Qin et al., 2025). The work of (Brock et al., 2021) demonstrates that purely convolutional backbones can be on par with hybrid models in terms of accuracy, however (Dai et al., 2021) experimentally show that the attention operation provides higher model capacity if incorporated in an architecture. On the other side, convolutions exhibit an improved generalization ability compared to attention modules, wherefore a combination of both on a macro design level is beneficial (Dai et al., 2021). Other works went further by joining convolutions and attention operations on a micro design level (Wu et al., 2021), relieving of the need for hand-crafted positional encoding as a result.

In the design of the LowFormer architecture, we integrate all three building blocks (attention, convolution, MLP) in a straightforward manner, resulting in a robust and versatile architecture. Unlike other approaches that depend on neural architecture search (Qin et al., 2025; Tan, 2019a) resulting in irregular macro designs, enforce a fixed micro design (Liu et al., 2023), or entirely exclude MLPs and/or attention mechanisms to improve performance (Vasu et al., 2023b; Wang et al., 2024; Cai et al., 2023), LowFormer takes a more flexible approach. Its architecture allows the removal of any building block without compromising its core design principles. In Section 4.4, we present a variant of the original LowFormer architecture, where we remove a portion of said components to further boost efficiency on edge GPU devices.

2.3 Efficient Attention

The landscape of attention mechanisms is vast, with many alternatives proposed to replace MHSA (Vaswani et al., 2017). A lot of effort has been spent (Bolya et al., 2022; Cai et al., 2023; Tu et al., 2022) to reduce its quadratic complexity by variations of linear attention (Wang et al., 2020). On the other side, the work (Yu et al., 2022) shows that attention can be replaced by a simple pooling operation. Subsequent research (Li et al., 2022) takes that idea further and uses the efficient pooling operation for the first three backbone stages and the traditional attention for the last two stages (Li et al., 2022). Other works (Wu et al., 2021; Wang et al., 2021; Fan et al., 2021) downsample the key and value vectors before the attention operation, either with convolutions or pooling. Pooling is also used to downsample all query, key, and value vectors in order to make attention completely operate on a lower resolution (Si et al., 2022).

In contrast to previous works, we harness the learning capability of convolutions to downscale the resolution of all input vectors for the Scaled Dot-Product Attention (SDA), effectively serving as conditional position embeddings (Chu et al., 2021). Unlike others, we further reduce the channel dimension before the SDA. Both reductions – of resolution and channel dimension – have a significant effect on efficiency but a minimal effect on accuracy.

3 Optimizing Vision Backbone Design by Contrasting MACs and Latency

To create efficient vision backbone architectures, it is not sufficient to assess them only by the amount of their MAC operations (Vasu et al., 2023b; Chen et al., 2023). Hardware efficiency needs to be taken into account, as it is a crucial factor for the execution time of a model (Qin et al., 2025). For example, a model can be considered more hardware efficient than a compared model, when it executes the same amount of MAC operations in less time.

Modern backbone architectures widely utilize convolutions as a core component (Vasu et al., 2023a; Tan, 2019a; Cai et al., 2023; Zhu et al., 2023), yet their impact on hardware efficiency remains largely unexamined. Therefore, we will investigate the hardware efficiency of convolutions in different scenarios. We will show that grouping (e.g. depthwise convolutions) and operating resolution of convolutions can have a substantial effect on the hardware efficiency. We will further analyze under which configurations (resolution and channel dimension) it is beneficial to replace the mobile inverted bottleneck (MBConv) block (Sandler et al., 2018),

Table 1 Contrasting MAC count and execution time of depthwise convolutions and ungrouped convolutions

Model	Channel $[C_0, C_1, C_2, C_3, C_4]$	Depthwise	MACs (M)	GPU Throughput ↑ (images/s)	Latency(ms) ↓ TX2	GPU	Mobile	ARM CPU
#1	[15, 30, 60, 120, 240]	✗	463	**12,722**	**1.83**	0.78	0.63	**12.58**
#2	[30, 60, 120, 240, 480]	✓	42	10,526	3.55	**0.24**	**0.53**	17.71
#3	[30, 50, 100, 160, 160]	✗	956	**7,142**	**2.90**	0.78	**0.64**	**24.29**
#4	[60,120,240,480,480]	✓	82	5,422	6.16	**0.39**	0.67	36.59
#5	[30, 60, 150, 240, 240]	✗	1710	**5,350**	**4.27**	0.99	1.00	48.19
#6	[60, 180, 360, 720, 720]	✓	104	4,244	8.03	**0.43**	**0.86**	**47.29**

The six toy models (#1-6) are divided into three groups by similar throughput. Each model differs in the number channels (see second column), if its convolutions are depthwise or not (see third column) and the number of layers in each stage. The table shows, that ungrouped convolutions that have a higher MAC count (more than 10× the amount) can still be similarly fast or faster as depthwise convolutions. Bold entries refer to the best value in each group and column

which is a popular component of many backbone architectures (Tan, 2019a; Cai et al., 2023), with the fused MBConv (Gupta & Tan, 2019). At last, we analyze the efficiency of different adaptations of the original MHSA (Vaswani et al., 2017) under different input resolutions.

To quantify hardware efficiency, we measured latency and image throughput on a diverse set of computing devices, including: a server machine with an Nvidia A40 GPU for GPU throughput; a desktop machine with a Nvidia Titan RTX GPU for GPU latency; a GPU-accelerated embedding device Nvidia Jetson TX2; an iPhone 13 smartphone with an Apple A15 Bionic; and a Raspberry Pi5 with an ARM CPU. Following previous practices (Yun, 2024; Cai et al., 2023), we run a model with a batch size of 200 for GPU throughput, while to measure latency we use a batch size of 1. We always use the median time of all iterations for latency and also to calculate throughput. As input images, we generate random tensors beforehand.

3.1 Depthwise Convolutions have low Hardware Efficiency

Previous research demonstrated that when optimizing backbone architectures for mobile-friendly design and efficiency, depthwise convolutions serve as a prominent alternative to standard convolutions (Howard et al., 2019; Tan, 2019a; Qin et al., 2025).

Grouped Convolutions Depthwise convolutions group their channels for computation. They divide them into as many groups as input channels. As a result, each feature map (channel) is processed independently by the depthwise convolution. Therefore each output feature map is only computed by processing its corresponding input feature map. In contrast, standard convolutions (Ma et al., 2018b) are ungrouped, meaning all input channels contribute to every output channel (groups = 1) during the convolutional operation.

Motivation Evaluating models solely based on their MACs often encourages incorporating as many depthwise convolutions as possible into the architecture, without considering their actual execution time speed-up. The efficiency of depthwise convolutions in terms of MACs does not always translate to equivalent gains in execution time. To illustrate the discrepancy between theoretical MAC efficiency and practical execution efficiency, we conduct an experiment using a simplified toy architecture.

Setting In Table 1, we examine the effect of using depthwise convolutions instead of ungrouped ones on throughput and latency. We created three models with only depthwise convolutions (#2,#4,#6) and three with only ungrouped ones (#1,#3,#5). Each model features five stages, mimicking common backbone architecture design. Consequently, with an input of resolution of 224×224, the output is of resolution 7×7. For the depthwise models, we doubled the amount of layers for each stage and increased the channel dimension to close up on the ungrouped models in terms of MACs. The depthwise models still have no more than a 10th of the MACs of the ungrouped convolutional models. Matching MACs – e.g. by repeating the depthwise convolutions 20 additional times – would result in an unusually high number of layers, deviating significantly from typical architectural designs.

Results Even though model #1 and #2 in Table 1 have a similar GPU throughput, their latency differs greatly. On the Jetson TX2 and ARM CPU the ungrouped convolutions are considerably faster in most scenarios, while on desktop GPU the opposite is true. This is due to higher memory bandwidth on the desktop GPU, which is particularly important for efficient execution of depthwise convolutions (Qin et al., 2025). This relationships also mostly holds true for the other models (#3-#6). Regarding mobile execution, the difference in latency is not significant. Even in the best-case scenario, depthwise convolutions require at least three times the processing time for the same number of MAC operations as their ungrouped counterparts. In the worst scenario (#5 and #6),

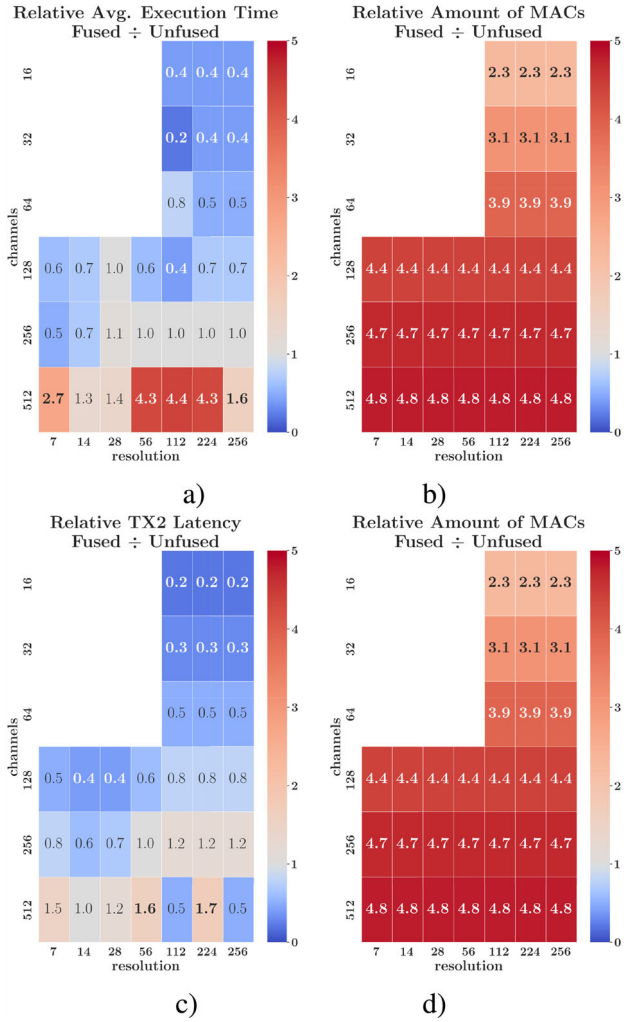

Fig. 3 The left figures a) and c) depict the average execution time (a) and latency (c) of the fused mobile inverted bottleneck (MBConv) relative to the unfused one, while the right figures b) and d) depict the relative amount of MACs. The blue areas in the left figures a) and c) correspond to configurations (number of channels and resolution) where fused MBConv is faster, while red corresponds to the opposite. For the figures b) and d), the red areas correspond to configurations where the fused MBConv has a higher amount of MACs. Bold numbers refer to entries with a particularly unequal ratio. Even though the fused MBConv always has more MACs, it is faster for many configurations.

the ungrouped convolutions execute MACs 34× faster on the Jetson TX2, highlighting the substantial hardware inefficiency of depthwise convolutions.

Conclusion In summary, ungrouped convolutions are more hardware efficient than depthwise convolutions, processing the same number of MACs in less time. However, the extent of this difference varies depending on the device.

3.2 Fusing the MBConv Block can speed up Models

The mobile inverted bottleneck block (MBConv) (Sandler et al., 2018) has a successful history in efficient backbones (Tan, 2019a; Sandler et al., 2018; Howard et al., 2019) and is still used within many new architectures (Cai et al., 2023; Dai et al., 2021; Tan, 2021b). It consists of two pointwise convolutions (PWConv) and a depthwise (DWConv) in between (see Figure 4 for a depiction of it). The PWConvs increase and decrease the channel dimension by the attributed expansion factor. An alternative to the MBConv block is the fused MBConv block (Gupta & Tan, 2019). The latter merges the first PWConv with the DWConv into an ungrouped convolution, thus removing any depthwise convolution. Due to the depthwise convolution, the original MBConv block usually has a lower amount of MACs than the fused one, although it features one layer more.

Motivation In Section 3.1, we showed that depthwise convolutions are hardware inefficient. Here, we extend that experiment to obtain insights into efficient backbone macro-design. We measure under which operating resolution and channel dimension the fused MBConv block (Conv+PWConv) is faster than the original MBConv (PWConv+DWConv +PWConv), allowing us to determine the optimal choice for different parts of the architecture.

Setting For this experiment, we measure latency on the Nvidia Jetson TX2 and average execution time on a Nvidia A40 (see Figure 3). To calculate the average execution time, we run 100 iterations with a batch size of 200, take the median execution time across these iterations, and divide it by 200. This is the inverse of the throughput metric and ensures consistency in the presentation of the sub-figures in Figure 3.

We apply an expansion factor of 4 for the fused and original MBConv in Figure 3. For this, we created toy models that just consist out of the same layer repeated after another.

Relative execution time In Figure 3, each metric is a relative metric, meaning it is always the value of the fused MBConv divided by the unfused one. For example in a) of Figure 3, we depict the relative average execution time, meaning the average execution time of the fused MBConv divided by the unfused one. In b) and d), we depict their relative MAC count (fused divided by unfused) and in c) their relative latency on the TX2. We apply these metrics for various resolutions and input channel dimensions. We omit the scenarios where channel dimension range from 16 to 64 and resolutions from 7 to 28, as GPU utilization is too low for the results to have significance.

Results It can be noticed that resolution and channel dimension have a big influence on the relative latency c) of Figure 3 and average execution time a). Even though the fused MBConv always has more MACs –values over one in right part b) and d) –, it is faster in many scenarios – i.e., values smaller than one in a) and c) – and is always more hardware efficient, i.e. the relative latency/average execution time is smaller than relative MAC count. Only for an high number of channels (>=256) and an high operating resolution (>=56), the fused MBConv presents slower execution. For channel dimension 512, and resolutions 112 and 256, the fused MBConv is faster than the unfused one. However, for resolution 224, it is the other way around. This inconsistency occurs because the computational load and memory requirements for these configurations exceed the Jetson TX2's capacity, leading to unpredictable behavior.

Conclusion The efficiency of the fused MBConv heavily depends on the channel dimension and to a lesser extent on the operating resolution. Consequently, it is advantageous to apply the fused MBConv in the early stages of an architecture, where the channel dimension is typically low (<256). In later stages that feature a higher channel dimension, the original MBConv proves more efficient.

3.3 High Resolution vs. High Channel

Motivation A key aspect of architecture design is determining the distribution of layers across different stages of the model. In backbone architectures, early stages typically have a high operating resolution and a low channel dimension, whereas later stages the opposite trend (Tan, 2019a; Cai et al., 2023; Qin et al., 2025). Therefore, understanding whether layers with a high channel dimension and low operating resolution are more efficient than those with a low channel dimension and high operating resolution is crucial for efficient architecture design. This insight helps optimize the allocation of layers across stages. In the following, we will analyze this factor for convolutional layers.

Setting For this experiment, we create toy models that consist of 20 times the same convolution stacked after another, where each toy model has a different configuration regarding operating resolution and channel dimension. In Table 2, we put at test seven scenarios, each contrasting two convolutions with the same amount of MACs, but differing in channel dimension and operating resolution. The models only contain standard convolutions (ungrouped). The upper row in each scenario is always the model operating on a higher resolution, while the lower row features a higher channel dimension. For improved readability we state relative throughput, latency and MAC count in Table 2. In each scenario, the metrics (throughput, latency, MAC count) of the low resolution model are expressed relative to those of the high resolution model and vice versa for the high resolution model.

Results In scenario #1 the first model has a third of the throughput of the second one and almost twice the latency. It operates on eight times the resolution, however features less channels and its MACs equal the second layer. The same effect also occurs with a smaller resolution difference, as scenario #2 shows, where the first model runs on twice the resolution, but still fails to execute its MACs as fast as the second one in terms of throughput and latency. On the other side in scenario #7 the model with a higher operating resolution, has a slightly lower latency. Regarding GPU throughput however, the lower resolution models always have a considerably higher throughput, ranging from a factor of 1.8 to 3.3. In scenario #5 the lower resolution model achieves lower latencies for the TX2 and ARM CPU, while for mobile the higher resolution model prevails, showing the impact that different hardware can have on model execution. Nevertheless a clear trend is visible. Models with a high operating resolution tend to be slower in most scenarios in terms of latency and throughput, than their MAC count might suggest. In (Nottebaum & Dunnhofer, 2025) this is also shown for GPU latency.

Conclusion We can conclude that in most cases it proves more hardware efficient to apply more layers in later stages of the backbone architecture and have only few layers in the high resolution stages. Based on these experiments, we emphasize that model scaling (increasing width, depth and input resolution (Zhai et al., 2022)) should be guided by actual measured execution time (Tan, 2019a), as MACs can be misleading, when scaling models by higher input resolution (Yun, 2024; Cai et al., 2023), as our results indicate.

3.4 Optimizing MHSA Efficiency for Higher Input Resolutions

The MHSA introduced by (Vaswani et al., 2017) is a crucial building block in many recent computer vision frameworks (Kirillov et al., 2023; Yan et al., 2021; Carion et al., 2020; Dunnhofer & Simonato, 2022; Khan & Micheloni, 2024; Bansal et al., 2022). The actual attention operation takes place within the SDA module of MHSA.

Motivation A key limitation of MHSA is that the computational complexity of SDA scales quadratically with spatial dimensions and linearly with the number of channels. To address this issue, we propose adaptations that reduce the input dimensions for SDA, enhancing efficiency without sacrificing performance. We evaluate the efficiency of both the original MHSA and our adapted versions across input resolutions ranging from 8×8 to 64×64. This is motivated from the fact, that many frameworks incorporating MHSA typically feature a high input resolution ($>512\times512$) (Kirillov

Table 2 Experiment on the impact of resolution on hardware efficiency

Scenario	Resolution (pixel)	Channels (#)	Relative MACs	Relative Throughput ↑ GPU (images/s)	Relative Latency (ms) ↓ TX2	Mobile	ARM CPU
#1	224	24	1.0	0.3	1.71	1.26	1.42
	28	196	1.0	**3.3**	**0.58**	**0.79**	**0.70**
#2	224	48	1.0	0.5	1.60	1.22	1.16
	112	96	1.0	**1.9**	**0.63**	**0.82**	**0.86**
#3	224	96	1.0	0.5	1.06	1.02	1.09
	56	384	1.0	**1.9**	**0.94**	**0.98**	**0.92**
#4	224	48	1.0	0.5	1.16	1.17	1.02
	56	196	1.0	**2.0**	**0.86**	**0.85**	**0.98**
#5	112	24	1.0	0.3	1.62	**0.89**	1.12
	14	196	1.0	**3.3**	**0.62**	1.12	**0.89**
#6	56	96	1.0	0.5	**1.0**	**0.78**	**0.74**
	14	384	1.0	**2.2**	**1.0**	1.28	1.35
#7	112	96	1.0	0.6	**0.88**	**0.85**	**0.99**
	28	384	1.0	**1.8**	1.14	1.18	1.01

Each scenario contains two configurations of convolutions (first row and second row in each scenario), that approximately feature the same amount of MACs (see 4th column), but strongly differ in operating resolution and number of channels. We set their throughput and latency in relation to each other (see 5th-8th column). The table shows that, for the same number of MACs, using convolutions with lower operating resolution and higher channel dimension (highlighted in gray) is more hardware efficient than using a higher resolution with lower channel dimension. Bold values refer to the best value for each scenario and column

Table 3 Efficiency comparison between original MHSA and three adaptations of it

	Attention Resolution	8×8	16×16	32×32	64×64	avg. diff.
	MHSA	0.59	2.79	19.70	260.97	
TX2 Latency	chcompr.	0.38 **(-37%)**	1.63 **(-41%)**	10.29 **(-47%)**	130.73 **(-50%)**	**−43%**
(ms)	conv+low	0.53 **(-10%)**	1.39 **(-50%)**	4.87 **(-75%)**	28.09 **(-89%)**	**−56%**
	conv+low+chcompr.	0.39 **(-35%)**	1.21 **(-56%)**	3.62 **(-81%)**	18.56 **(-92%)**	**−66%**
	MHSA	0.20	1.60	5.48	179.75	
Mobile Latency	chcompr.	0.15 **(-25%)**	0.79 **(-50%)**	2.62 **(-52%)**	65.10 **(-63%)**	**−47%**
(ms)	conv+low	0.24 **(+20%)**	0.45 **(-71%)**	1.34 **(-75%)**	5.60 **(-97%)**	**−56%**
	conv+low+chcompr.	0.21 **(+5%)**	0.33 **(-79%)**	0.71 **(-87%)**	2.71 **(-98%)**	v**65%**
	MHSA	0.81	6.16	95.32	1808.16	
ARM CPU Latency	chcompr.	0.57 **(-30%)**	2.42 **(-60%)**	46.73 **(-50%)**	915.41 **(-49%)**	**−47%**
(ms)	conv+low	0.85 **(+5%)**	2.21 **(-64%)**	10.97 **(-88%)**	126.48 **(-93%)**	**−60%**
	conv+low+chcompr.	0.75 **(-7%)**	1.88 **(-69%)**	7.23 **(-92%)**	75.79 **(-95%)**	**−66%**
	MHSA	0.49	0.49	1.24	13.85	
GPU Latency	chcompr.	0.49 **(-0%)**	0.50 **(+2%)**	0.85 **(-31%)**	7.76 **(-44%)**	**−18%**
(ms)	conv+low	0.62 **(+26%)**	0.63 **(+28%)**	0.63 **(-49%)**	1.42 **(-90%)**	**−22%**
	conv+low+chcompr.	0.63 **(+28%)**	0.64 **(+30%)**	0.64 **(-48%)**	1.02 **(-92%)**	**−20%**

"conv+low" refers to encapsulating MHSA with convolutions that reduce the resolution of the attention by a factor of 2, "chcompr." refers to a reduced channel dimension for SDA and "conv+low+chcompr." refers to the combination of the latter two. We evaluate execution time under different input resolutions. The highest efficiency is achieved by applying the "conv+low+chcompr." adaptation on MHSA. Results for original MHSA are highlighted in gray

et al., 2023; Zheng et al., 2023; Zhang et al., 2021; Carion et al., 2020), consequently leading to SDA being executed at resolutions above 16×16. This is particularly true when the backbone itself includes MHSA. For instance, in the ViT-B/16 vision backbone (Dosovitskiy et al., 2020), an input resolution of 1024×1024 (Kirillov et al., 2023) leads to SDA being computed at resolution 64×64.

Setting To conduct this experiment, we employ toy models that consist of 4 times the same attention layer after another. We feature an input and output channel dimension of 128 to prevent overloading the edge devices, as MHSA at increased input resolutions imposes a high computational burden. We measure latency on the Jetson TX2, the iPhone 13, the ARM CPU of the Raspberry Pi5 and the Nvidia TITAN RTX GPU.

Efficient Adaptations In total, we feature two modifications of MHSA: "chcompr." (channel compression) stands for halving the channel dimension before the SDA by the input projection and restoring the input channel dimension by the output projection of the MHSA. "conv+low" means we add a convolution before the input projection and after the output projection, which downsample and upsample the resolution of the feature maps. In Table 3, we compare how these two modifications, taken together and separately, influence the execution time under different input resolution (from 8×8 to 64×64).

Results In Table 3 we compare MHSA with the two adaptations we propose. Table 3 illustrates the impact of the quadratic explosion of MHSA. On edge devices, latency is at least 442 times higher for a resolution of 64×64 compared to 8×8, and even 2232 times higher on an ARM CPU. In contrast, GPU execution shows only a 28-fold increase in latency, highlighting the immense parallelization capabilities of GPUs compared to edge devices. Regarding our adaptations, we can see that "chcompr." leads to a reduction in latency of between 30% and 63% on edge devices, while "conv+low" leads to an even higher efficiency gain. The performance improvement for both adaptations increases significantly with higher input resolution. On GPU with input resolutions 8×8 and 16×16, the efficiency remains similar to MHSA due to the potential of parallelization a desktop GPU has or slightly declines, because of the layers added by "conv+low" adaptation. Nevertheless for input resolutions above 16×16, both adaptations lead to a considerable reduction in latency by up to 90% compared to MHSA. Combining both methods ("conv+low+chcompr") maximizes efficiency. For input resolution 64×64 its latency reduction ranges between 92% and 98% across all devices.

Conclusion While "chcompr." can lead to a reduction of up to half of the latency, "conv+low" can reduce latency to a smaller fraction of it. Overall, the combination of the two optimizations ("chcompr." and "conv+low") provides significant benefits for performance, particularly for edge devices. At high input resolutions, the adaptations also lead to substantial performance gains for GPUs. Since many downstream tasks rely on high input resolution, these optimizations are highly relevant (Kirillov et al., 2023; Carion et al., 2020; Liu et al., 2025). In Section 5.2, we will further demonstrate that the combination of both optimizations has a positive impact on ImageNet (Deng et al., 2009) accuracy.

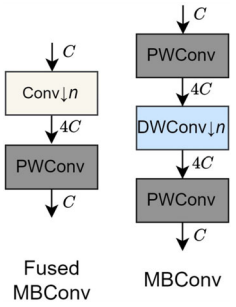

Fig. 4 Structure of the fused and unfused MBConv block. *C* refers to the channel dimension. Both have an expansion factor of 4

4 LowFormer

The aim of the LowFormer architecture design is to improve hardware efficiency, allowing vision backbones to execute faster while maintaining a high model size that enables reaching high accuracy. To achieve this goal, we follow the insights of the analysis presented in Section 3. In this section, we will first introduce Lowtention, a lightweight adaptation of the original MHSA (Vaswani et al., 2017) (Section 4.1), and then outline the key principles of LowFormer's macro design and provide both an overview (Section 4.2) and detailed description of the overall architecture (Section 4.3). Finally, we will propose adaptations to the main LowFormer architecture aimed specifically at further enhancing performance on edge devices (Section 4.4).

4.1 Micro Design - Lowtention

In LowFormer's architecture, we employ a lightweight adaptation of the original MHSA (Vaswani et al., 2017), which we call Lowtention (see Figure 5). Lowtention encapsulates the SDA by two depthwise convolutions and two pointwise convolutions. The latter perform the input and output projections of the queries (Q), keys (K) and values (V), just as in the original MHSA.

Channel compression However, in Lowtention the input projection (before the SDA) reduces the channel dimension of Q, K and V by half (see Figure 5). The output projection (following the SDA) then restores the original channel dimension, which is required for the residual connection after the Lowtention (see Figure 5). In Section 3.4, we demonstrated that compressing the channel dimension

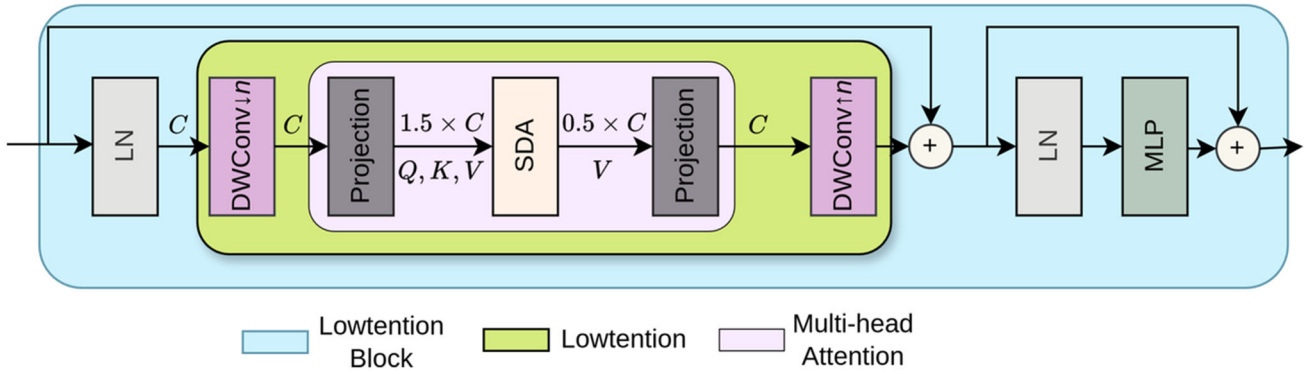

Fig. 5 Lowtention block design. LN refers to layer normalization. In contrast to the traditional MHSA, we encapsulate the SDA with two depthwise convolutions (the second is a transposed depthwise convolution). The projections for MHSA are realized with pointwise convolutions. The $DW\downarrow_n$ means that the resolution is downscaled by the factor n and $DW\uparrow_n$ that it is upscaled by n

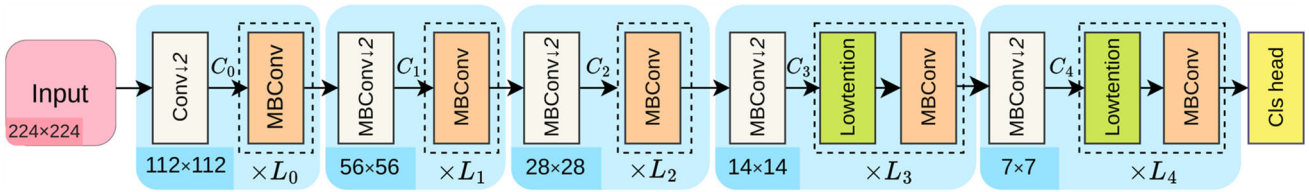

Fig. 6 Architecture of LowFormer. The resolutions refer to a 224x224 sized input. Lowtention block can be seen in Figure 5. "Conv" refers to convolution and "Cls head" refers to the image classification head (see Figure 7). Specification of $C_0 - C_4$ and $L_0 - L_4$ depend of the version of LowFormer and can be found in Table 4.

can significantly improve latency on a variety of devices, especially at high input resolutions.

Lower resolution As depicted in Figure 5 we down- and upsample the resolution of the feature maps in Lowtention around the SDA, such that the SDA is executed on half the resolution. This motivates from the experiments in Section 3.4, where we have shown that this can improve latency significantly, similar to the channel compression in Lowtention.

In Section 5.2 we further show that the combination of both strategies (channel compression and lower resolution) improves top-1 accuracy on ImageNet (Deng et al., 2009).

MLP following Attention Following (Vaswani et al., 2017), we append layer normalization and a multi-layer perceptron (MLP) after the Lowtention. This is motivated by Liu et al. (2023), who pointed out the significance of MLPs for improving accuracy of a backbone.

4.2 Macro Design

The LowFormer architecture features five stages that adapt the architectural macro design of EfficientViT (Cai et al., 2023) and MobileViT (Mehta, 2021a) according to the insights gained from the hardware efficiency analysis presented in Section 3. The whole architecture is depicted in Figure 6.

In total we present five different base versions of LowFormer, namely B0, B1, B1.5, B2, B3. The versions differ in the number of layers and channel dimension employed in each stage (see Table 4). We chose to feature five base versions of LowFormer to demonstrate that its design principles are adaptable to various model sizes and accuracy levels. LowFormer-B0 represents the model with the lowest model size, while LowFormer-B3 has the highest. Consequently a base version with a higher model size also achieves superior results compared to a lower size variant, as shown in Section 5.

Lowtention We include Lowtention in the last two stages and keep the first three stages purely convolutional. Additionally, we only downsample the feature maps of Lowtention (as mentioned in Section 4.1) in the forelast stage.

Fusing Depthwise and Pointwise Convolutions In Section 3.1 we showed that depthwise convolutions are not as hardware-efficient as standard convolution and in Section 3.2 we came to the conclusion that the fused MBConv (see Figure 4) can be faster than the unfused one, even though it usually has a higher MAC count. This effect diminishes however with increasing number of channels. We therefore

Table 4 Specification of LowFormer architecture versions B0-B3

Model	$\{L_0, L_1, L_2, L_3, L_4\}$	$\{C_0, C_1, C_2, C_3, C_4\}$
LowFormer-B0	{0, 0, 0, 3, 4}	{16, 32, 64, 128, 256}
LowFormer-B1	{0, 0, 0, 5, 5}	{16, 32, 64, 128, 256}
LowFormer-B1.5	{0, 0, 0, 6, 6}	{20, 40, 80, 160, 320}
LowFormer-B2	{0, 0, 0, 6, 6}	{24, 48, 96, 192, 384}
LowFormer-B3	{1, 1, 2, 6, 6}	{32, 64, 128, 256, 512}

The number of layers ($L_0 - L_4$) and channels ($C_0 - C_4$) relates to Figure 6

fused the MBConv in our architecture, wherever the number of input channels reach at most 256, except for the strided MBConv blocks at the beginning of the last two stages (see Figure 6). We additionally fuse the depthwise and pointwise convolutions after the SDA in the Lowtention (see Figure 5), as their input channel dimension does not exceed 256 for any LowFormer model, due to the channel compression. We confirm the effect of this approach by reverting the fusion of the MBConv block for LowFormer-B1 in the ablation in Section 5.2.

Less layers in the first stages From the insights in Section 3.3 we conclude that a minimal amount of layers in the first stages is more hardware-efficient (see Table 4). It proved optimal to apply the reduction for the first three stages. Most computation is therefore concentrated in the last two stages, where for an input size of 224×224, the operating resolutions are 14×14 and 7×7.

4.3 Additional Architectural Details

The details of LowFormer's architecture mainly follow design principles of previous publications (Cai et al., 2023; Vasu et al., 2023a).

As activation function, we utilize HardSwish (Howard et al., 2019), except for the MLP, where we use GeLU (Hendrycks, 2016) in between its two linear layers (Vaswani et al., 2017).

MBConv Micro Design Details LowFormer's micro design for MBConv blocks, regarding the combination of batch normaliation (bn) (Ioffe, 2015) and activation functions (act), differs between fused and original (unfused) MBConv. For the fused MBConv, each block follows the design of "conv,bn,act,pwconv,bn", while the original MBConvs design is "pwconv,act,dwconv,act,pwconv,norm".

MBConv Expansion Factor The expansion factor in MBConvs controls by which factor the channel dimension is increased by the first convolution and decreased by the last convolution (see Figure 4). We set it to 6 for all MBConv blocks (fused and unfused) that reduce resolution and 4 otherwise.

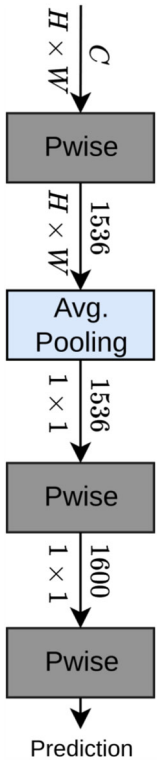

Fig. 7 Design of classification head that is used for all variants of LowFormer.

Residual Connections Besides the residual connections in the Lowtention block (see Figure 5), every MBConv block that does not reduce resolution, has a residual connection. We only reduce resolution by strided MBConv blocks at the beginning of each stage.

Classification Head In Figure 7 the design of our classification head is depicted, which we apply for all versions of LowFormer and only differs in the input channel dimension, given by the last layer of the final stage. It is based on EfficientViTs (Cai et al., 2023) classification head.

4.4 LowFormer for Edge GPU

We further present three edge GPU variants of LowFormer, namely LowFormer-E1/E2/E3. They are derivations of the so far described architecture. The changes only focus on the last two stages, as most of the computational load is concentrated there (see Table 4). We feature three changes to the original LowFormer Architecture:

Table 5 Architecture changes for LowFormer edge variant, versions E1/E2/E3

Model	Changes	Base model
LowFormer-E1	a), b), c)	LowFormer-B1.5
LowFormer-E2	a), b), c)	LowFormer-B3
LowFormer-E3	b)	LowFormer-B3

The changes a), b) and c) refer to the enumeration in Section 4.4

a) Reduction of the depth by 2
b) Removal of the MLP
c) Removal of the Lowtention

All changes follow the intuition that Edge GPUs (like the Jetson TX2) usually consist of many cores[1] and focus on high parallelization, similar to GPUs[2]. This is in contrast to CPUs for example, who usually feature much less cores, like the ARM CPU in the Raspberry Pi5, who operates on 4 cores[3].

Change a) therefore originates from the intuition, that depth is a crucial factor in optimizing efficiency on edge GPU devices, as the compulsory sequential execution of high amount of layers, prohibits parallelization, thus it would be beneficial to rather have fewer layers, but with high amount of computation, that is parallelizable.

Change b) motivates from the observation of (Qin et al., 2025), that MLP executions are usually memory bound and therefore are not efficient on hardware with high compute ability (e.g. GPUs and edge GPUs). Additionally MLPs account for a significant portion of the whole models computational load (MACs). For LowFormer-B1 for example, the MLPs make up 17% of the total MACs.

Change c) on the other side motivates from our observation in Section 3.4, where the attention mechanisms show an immense computational burden, especially with increased input resolution. Furthermore, (Liu et al., 2023) demonstrated that MHSA operations are heavily memory bound, even more so than the MLP. They recommend allocating a lower portion of the model to MHSA to yield the best speed accuracy trade-off.

We combine the three changes a), b) and c) into the models LowFormer-E1/E2/E3 as depicted in Table 5. In Section 5.3.1, we substantiate our reasoning by comparing how each mentioned change impacts accuracy and efficiency for edge GPU and GPU.

[1] https://developer.nvidia.com/embedded/jetson-tx2

[2] https://www.nvidia.com/content/dam/en-zz/Solutions/titan/documents/titan-rtx-for-creators-us-nvidia-1011126-r6-web.pdf

[3] https://www.raspberrypi.com/products/raspberry-pi-5/

5 Experiments

In this section, we present and the discuss the experimental results achieved by the LowFormer family of vision backbones (LowFormer-B0/-/B3), including the edge variants (LowFormer-E1/E2/E3). For the experiments, we examine model efficiency by measuring GPU throughput, GPU latency and edge device latency. Additionally measuring throughput is motivated by its ability to assess how efficiently big quantities of data can be processed. A high throughput is particularly beneficial for tasks such as image retrieval and batched offline processing of video data by detection and segmentation algorithms. It is also crucial to reduce training time (Tan, 2021b).

Protocols for Measuring Execution Time In Table 6 specifics of the measuring protocols are listed , that we apply throughout the paper. We always take the median time per input instance for latency and throughput measurements (Vasu et al., 2023a; Yun, 2024). For latency measurements we always use a batch size of 1, while for GPU throughput we feature a batch size of 200. As depicted in Table 6, we measure latency on three edge devices, namely the Nvidia Jetson TX2 8GB developer kit, the iPhone 13 and the ARM CPU of the Raspberry Pi5 8GB (Arm Cortex A76 processor @ 2.4GHz). The amount of iterations differ, because some devices require a higher amount of iterations to retrieve stable results. However we always feature 5 warm-up iterations. Regarding the iPhone 13 we utilize the CoreML[4] performance tool to retrieve latency results, wherefore we do not have specifics about the amount of iterations.

5.1 ImageNet-1K Classification

Settings We conduct image classification experiments on ImageNet-1K (Deng et al., 2009), which includes 1.28M training and 50K validation images for 1000 categories. All models were trained from scratch using a similar setting as (Cai et al., 2023) and featuring an input resolution of 224. We also trained for a total of 320 epochs using AdamW (Loshchilov & Hutter, 2017) optimizer and a learning rate of 10^{-3}, however we use a batch size of 512. As learning rate scheduler we use cosine decay (Loshchilov, 2016) and 20 warm-up epochs with a linear schedule. We also feature the multi-scale learning from (Cai et al., 2023). We trained LowFormer-B3 with a batch size of 2400 and a base learning rate of 3×10^{-3}. For LowFormer-B2 we had a batch size of 850 and a base learning rate of 8.3×10^{-4}.

Results In Table 7, we evaluate the speed accuracy trade-off of the compared models. For this we measure ImageNet (Deng et al., 2009) top-1 accuracy and assess efficiency based

[4] https://apple.github.io/coremltools/docs-guides/

Table 6 Specifics of efficiency measurements. Further details are given in the supplementary document.

Device	Metric	Iterations	Framework
Nvidia A40 GPU	throughput	100	PyTorch
Nvidia TITAN RTX GPU	latency	4000	TorchScript[5]
ARM Cortex A76 CPU	latency	400	ONNX[6]
iPhone13	latency	-	CoreML
Nvidia Jetson TX2 8GB	latency	200	ONNX[6] +TensorRT[7]

[5] https://pytorch.org/docs/stable/generated/torch.jit.optimize_for_inference.html
[6] https://github.com/microsoft/onnxruntime
[7] https://docs.nvidia.com/deeplearning/tensorrt/

on GPU throughput, Jetson TX2 latency and ARM CPU latency. The base models of LowFormer (LowFormer-B0 to B3) achieve a superior speed accuracy trade-off, outperforming most compared models in all three efficiency metrics. LowFormer-B0, has a slightly higher top-1 accuracy than EfficientMod-xs (Ma et al., 2024), twice the throughput, half of its TX2 latency and executes 38 % faster on ARM CPU. Although BiFormer-T (Zhu et al., 2023) has 40% less MACs than LowFormer-B2$_{r224}$, our LowFormer has a 305% higher throughput and only a third of its TX2 and ARM CPU latency. At the same time it slightly outperforms BiFormer-T by 0.2% top-1 accuracy. The largest base model of LowFormer, LowFormer-B3, surpasses FastViT-SA36 (Vasu et al., 2023a) in efficiency, achieving nearly three times the GPU throughput and running 36% faster on the TX2, and 46% faster on an ARM CPU. Both models share the same top-1 accuracy. The hardware efficient design of LowFormer base models not only execute MACs more efficiently but also leads to higher top-1 accuracies, while achieving improved efficiency.

Resolution Scaling A model's efficiency at increased input resolution is critical, as many downstream tasks run the backbone on high-resolution inputs (Carion et al., 2020; Kirillov et al., 2023). As shown in Table 2, model efficiency can vary in response to higher input resolutions, depending on the architecture. Therefore, we examine in Figure 8 how increasing input resolution affects latency on the Jetson TX2 for LowFormer-B1 and other approaches. LowFormer-B1 outperforms depicted models in top-1 accuracy on ImageNet-1K and at the same time remains considerably faster, independent of the input resolution. The model "Stride 1 Attention" refers to an ablation of LowFormer-B1, discussed in Section 5.2.

5.2 Ablation Study of LowFormer Base Models

In Table 8 we ablate our model design decisions. We revert a singular design decision of LowFormer-B1 to demonstrate the impact of that change on accuracy, GPU throughput and TX2 latency. The featured ablations are the following:

- We replace all fused MBConv blocks with the unfused version.
- We replace our Lowtention with ReLU linear attention from (Cai et al., 2023) in order to compare our attention approach with other recent adaptations.
- We omit the downscaling of the feature maps for the Lowtention (high-res attention).
- We remove the channel compression done during the projection in Lowtention.
- We replace Lowtention with the original MHSA proposed by (Vaswani et al., 2017).

5.2.1 Ablation Results

Unfused MBConv Replacing the fused MBConv with the unfused one results in a 16% lower GPU throughput and 10% higher TX2 latency (see Table 8). On the other side ARM CPU latency improves by 6%, however top-1 accuracy drops significantly by 0.8% As we can see, next to a mostly improved execution time, fusing the MBConv can increase performance significantly.

ReLU Linear Attention Applying ReLU linear attention from (Cai et al., 2023) on the other side results in 0.3% reduced top-1 accuracy, as well as $\tilde{8}$% higher latency and 20% lower GPU throughput, showing the benefit of Lowtention.
Downsampling in Lowtention When we remove the downsampling for the Lowtention, top-1 accuracy stays the same, but GPU throughput and latency worsen significantly. In Figure 8 we can see that for higher input resolutions the latency difference multiplies. For input resolution 1024×1024 for example, latency is increased by 220% on the Jetson TX2.
Channel Compression Removing the channel compression of Lowtention during the projection phase does not improve accuracy, even though the SDA operates on a lower channel dimension with compression. However throughput and latency worsen significantly without channel compression.

MHSA When removing channel compression and convolutions from Lowtention (no downsampling), we revert to the original MHSA. As shown in Table 8, this not only reduces top-1 accuracy, but also increases model latency significantly.

Table 7 Performance on ImageNet-1K validation set

Model	Venue	Params (M)	MACs (M)	GPU Throughput ↑ (images/s)	TX2 Latency ↓ (ms)	ARM CPU ↓ (ms)	Resolution (pixel)	Top-1 (%)
MobileViG-Ti* (Munir & Avery, 2023)	CVPRW 2023	5.3	661	2500	**8.5**	48.2	224	75.7
FastViT-T8 (Vasu et al., 2023a)	CVPR 2023	3.6	690	1694	10.1	65.9	256	75.6
EfficientMod-xxs (Ma et al., 2024)	ICLR 2024	4.7	579	2857	15.0	47.5	224	76.0
RepViT-M0.9 (Wang et al., 2024)	CVPR 2024	5.1	816	2512	11.4	40.8	224	77.4
MobileOne-S2 (Vasu et al., 2023b)	CVPR 2023	7.8	1298	2967	9.1	53.8	224	77.4
EdgeViT-XS (Chen et al., 2022b)	ECCV 2022	6.8	1127	2127	16.0	58.9	224	77.5
MobileOne-S3 (Vasu et al., 2023b)	CVPR 2023	10.1	1895	2433	11.8	74.2	224	78.1
MobileViG-S* (Munir & Avery, 2023)	CVPRW 2023	7.3	983	1724	12.3	73.9	224	78.2
EfficientMod-xs (Ma et al., 2024)	ICLR 2024	6.6	773	2352	17.7	53.8	224	78.3
LowFormer-B0 (ours)		14.1	944	**5988**	**8.5**	**39.1**	224	**78.4**
FastViT-T12 (Vasu et al., 2023a)	CVPR 2023	6.8	1400	2054	14.5	110.4	256	79.1
RepViT-M1.1 (Wang et al., 2024)	CVPR 2024	8.2	1338	1941	13.5	63.5	224	79.4
MobileOne-S4 (Vasu et al., 2023b)	CVPR 2023	14.8	2978	1550	18.6	122.9	224	79.4
LowFormer-B1 (ours)		17.9	1410	**4237**	**11.7**	**59.1**	224	**79.9**
EfficientFormerV2-S2 (Li et al., 2023)	ICCV 2023	12.6	1250	468	19.9	102.3	224	80.4
FastViT-SA12 (Vasu et al., 2023a)	CVPR 2023	10.9	1943	1075	**17.5**	136.4	256	80.6
EdgeViT-S (Chen et al., 2022b)	ECCV 2022	11.1	1910	1449	24.6	**99.2**	224	81.0
EfficientMod-s (Ma et al., 2024)	ICLR 2024	12.9	1402	1381	30.5	105.6	224	81.0
RepViT-M1.5 (Wang et al., 2024)	CVPR 2024	14.0	2276	1146	23.0	113.2	224	**81.2**
LowFormer-B1.5 (ours)		33.9	2573	**2739**	18.1	111.6	224	**81.2**
FFNet-1 Yun and Lee (2024)	arXiv 2024	13.8	3000	1090	30.4	242.1	256	81.3
BiFormer-T Zhu et al. (2023)	CVPR2023	13.1	2200	729	61.3	523.9	224	81.4
LowFormer-B2 (ours)		45.0	3689	**2227**	**21.6**	**144.2**	224	**81.6**
SMT-T (Lin et al., 2023)	ICCV2023	11.5	2400	770	50.3	**195.6**	224	82.2
RepViT-M2.3 (Wang et al., 2024)	CVPR 2024	22.9	4520	642	40.6	227.0	224	82.5
FastViT-SA24 (Vasu et al., 2023a)	CVPR 2023	20.6	3769	606	30.9	273.7	256	82.6
LowFormer-B3$_{r192}$ (ours)		57.1	4479	**1562**	**30.0**	198.6	192	**82.7**
iFormer-S (Si et al., 2022)	NeurIPS 2022	19.9	4825	555	51.2	270.6	224	83.4
FastViT-SA36 (Vasu et al., 2023a)	CVPR 2023	30.4	5595	429	44.4	399.8	256	83.6
SMT-S (Lin et al., 2023)	ICCV2023	20.5	4700	418	96.4	397.8	224	83.7
BiFormer-S (Zhu et al., 2023)	CVPR2023	26.0	4500	348	130.2	1134.1	224	**83.8**
LowFormer-B3 (ours)		57.1	6098	**1162**	**32.5**	**273.8**	224	83.6

The table is divided into different groups, determined by similar top-1 accuracy (bold horizontal lines separate groups). Values in bold are the best results for each group and column, while underlined results refer to the second best. For models marked with *, only distilled model results are publicly available. LowFormer models (highlighted in gray) achieve superior speed accuracy trade-offs in terms of GPU throughput, TX2 latency and ARM CPU latency.

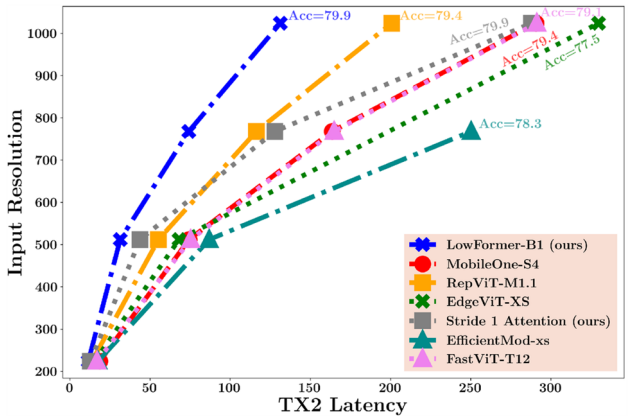

Fig. 8 Impact of input resolution on Jetson TX2 latency for LowFormer-B1 (ours), LowFormer-B1 without downsampling in the Lowtention layers (see Section 5.2), MobileOne-S4 (Vasu et al., 2023b), RepViT-M1.1 (Wang et al., 2024), EdgeViT-XS (Chen et al., 2022b), EfficientMod-xs (Ma et al., 2024) and FastViT-T12 (Vasu et al., 2023a). "Acc" refers to ImageNet top-1 accuracy. LowFormer-B1 demonstrates remarkable efficiency when operating at a higher input resolution, outperforming compared state-of-the-art architectures.

This difference multiplies, when input resolution is increased as shown in Table 3.

Summary The reduced dimensions on which we apply the attention operation (channel compression and resolution reduction) have no effect on top-1 accuracy, but improve the efficiency of the model significantly. The additional convolutions improve accuracy (original MHSA fairs worse in Table 8), but only mildly worsen execution time (see Table 3). In conclusion Lowtention is both more efficient and effective than the original MHSA (Vaswani et al., 2017), with significantly better scalability for increased input resolution.

5.3 Comparison of LowFormer Edge GPU Variants with the Base Models

In Section 4.4 we presented the three derivations LowFormer-E1/E2/E3 from the original LowFormer base models (B1.5 and B3). In the following part we will first experimentally justify why these specific changes were chosen by analyzing the effect of each change on efficiency and accuracy. Then we will put the LowFormer edge variants into perspective with the best competing models of Table 7, based on their speed accuracy trade-off on the Jetson TX2.

5.3.1 Efficiency Analysis of Attention, MLP and Depth

In Section 4.4 we presented three possible changes to the LowFormer architecture, namely removing the MLP, reducing the depth and removing the Lowtention. In Table 9 we show the effect of each of these changes cumulatively. First we remove the MLP ("mlpless"), then we additionally reduce the depth ("shallow") and at last we also remove the Lowtention ("conv"), leaving only convolutions in the architecture. We use LowFormer B1, B1.5 and B3 as baselines for the modifications. All derivations have the same hyperparameter setting during training as their corresponding base version.

Removing Attention From the model B1.5_conv_shallow we can see the enourmous penalty attention can have on latency. It has 42% of the GPU latency and 64% of the TX2 latency of B1.5_mlpless_shallow, but loses less than 1% top-1 accuracy. Both models only differ in that B1.5_conv_shallow does not feature the Lowtention. The difference becomes more apparent if you compare B1.5_conv_shallow to b1_mlpless, which has the same top-1 accuracy, but fares far worst regarding GPU and TX2 latency.

Removing MLP On the other side, omitting the MLP can also be beneficial, although not as pivotally. B1_mlpless has a 0.4% higher top-1 accuracy than LowFormer-B0, while being slightly faster in terms of latency and throughput.

Reducing Model Depth Reduction of the model depth especially improves latency, while its impact on throughput is less pronounced. For example B1.5_mlpless_shallow achieves a 0.9% higher top-1 accuracy compared to B1_mlpless, while having an improved GPU latency and a slightly worse throughput.

Table 8 Ablation study of LowFormer-B1, featuring singular changes to the original model

Model version	Params (M)	MACs (M)	GPU Throughput ↑ (images/s)	TX2 Latency ↓ (ms)	ARM CPU Latency ↓ (ms)	Top-1 (%)
unfused MBConv	12.4	716	3558 **(-16%)**	12.8 **(+9%)**	**55.5 (-6%)**	79.1 **(-0.8)**
relu-linear att	14.15	1210	3367 **(-20%)**	12.7 **(+8%)**	64.8 **(+9%)**	79.6 **(-0.3)**
original MHSA	16.8	1460	3590 **(-15%)**	12.5 **(+7%)**	61.2 **(+4%)**	79.8 **(-0.1)**
high-res attention	17.65	1494	3759 **(-11%)**	13.4 **(+14%)**	64.0 **(+8%)**	**79.9 (+0.0)**
no channel compr.	20.68	1650	3921 **(-7%)**	13.4 **(+14%)**	63.1 **(+7%)**	**79.9 (+0.0)**
Baseline (B1)	17.94	1410	**4237**	**11.7**	59.1	**79.9**

Bold entries mark the best in its column. The LowFormer architecture design elements consistently demonstrate superior efficiency while achieving higher ImageNet accuracy

Table 9 Efficiency comparison of LowFormer on several computing devices with modified versions of the original architecture, where the multi-layer-perceptron (MLP) or the Lowtention (Att) is removed, or the depth of the last two stages is reduced

Model	MLP	Original Depth	Att	GPU Throughput (images/s)	GPU Latency (ms)	TX2 Latency (ms)	iPhone 13 (ms)	Top-1 (%)
LowFormer-B0	✓	✓	✓	5988	2.9	8.5	1.5	78.4
LowFormer-B1	✓	✓	✓	4237	4.0	11.7	1.8	79.9
B1_mlpless	✗	✓	✓	6067	2.8	8.5	1.5	78.8
B1_mlpless_shallow	✗	✗	✓	8254	1.8	5.9	1.3	77.2
LowFormer-B1.5	✓	✓	✓	2739	4.8	2.8	111.6	81.2
B1.5_mlpless	✗	✓	✓	4019	3.3	13.2	2.4	80.7
B1.5_mlpless_shallow	✗	✗	✓	5268	2.4	9.7	2.0	79.7
B1.5_conv_shallow (E1)	✗	✗	✗	6337	1.0	6.2	1.6	78.8
LowFormer-B3	✓	✓	✓	1162	5.2	32.5	4.5	83.6
B3_mlpless (E3)	✗	✓	✓	1566	3.6	25.0	3.6	83.0
B3_mlpless_shallow	✗	✗	✓	1848	2.7	19.6	2.8	82.2
B3_conv_shallow (E2)	✗	✗	✗	2070	1.5	14.7	2.5	81.6

The highest efficiency increase can be achieved by removing the attention operation. The connotations E1,E2,E3 in braces refer to the proposed edge GPU variants. LowFormer base models are highlighted in gray.

Summary Removing Lowtention, the MLP and reducing the model depth can significantly improve the speed accuracy trade-off, especially in terms of latency on GPU and edge GPU (Jetson TX2). These three modifications are combined in LowFormer-E1 and LowFormer-E2, yielding a substantial efficiency gain with only a minimal accuracy drop. In LowFormer-E3, only the MLP is removed, allowing it to achieve a higher accuracy of 83.0% without relying on model-scaling strategies such as width scaling, which become less efficient as model capacity increases (Tan, 2019a).

5.3.2 Evaluation of Edge Optimization

To put our LowFormer edge GPU variants in perspective, we compare them in Table 10 to the highest competing models from Table 7 based on their respective Jetson TX2 latency. In the n material, we further analyzed the power consumption of models in Table 10.

GPU Throughput & TX2 Latency LowFormer-E1/E2/E3 consistently achieve a better throughput and TX2 latency than all compared state-of-the-art models with a similar or lower top-1 accuracy, including the LowFormer base models (B0/B1.5/B2/B3). LowFormer-E1 for example has half of the TX2 latency of MobileOne-S3 (Vasu et al., 2023b), while having 0.7% higher accuracy. LowFormer-E2 has a similar latency compared to FastViT-T12 (Vasu et al., 2023a), but scores 2.5% higher in ImageNet top-1 accuracy.

GPU Latency LowFormer-E1/E2 similarly outperform all compared models in GPU latency, however LowFormer-E3 fares considerably worse in that regard. This is mainly to the fact, that it still makes use of the attention operation, which has a negative impact on GPU latency (as mentioned in Section 5.3.1). Nevertheless it is considerably more efficient when compared to LowFormer-B3$_{r192}$. **Mobile Latency** Regarding mobile execution on the iPhone 13, the edge variant do not give a consistent speed-up, when compared with the LowFormer base versions. Moreover, architectures like RepViT and FastViT (Wang et al., 2024; Vasu et al., 2023a) achieve superior performance on iPhone 13. This is in part because the edge variants and base models feature a higher amount of MACs, which the mobile compute hardware cannot effectively parallelize.

Summary LowFormer-E1, E2 and E3 lead the table (Table 10) in terms of GPU throughput, GPU latency and Jetson TX2 latency. However, the optimizations from Section 4.4 do not translate well to mobile execution on the iPhone 13 NPU and GPU. Despite this, the edge GPU variants achieve a speed-up of up to 3×, compared to the LowFormer base models.

5.4 Application to Downstream Tasks

For a fair comparison, we compare models of similar size with each other and select those that achieve the best performance in the respective benchmark.

5.4.1 Image Classification

We assess LowFormer's transfer learning capabilities by evaluating its performance when finetuned on smaller image classification datasets. We feature three datasets, namely Oxford-IIIT-Pets (Parkhi et al., 2012), Stanford Cars (Krause

Table 10 Efficiency comparison between the LowFormer edge GPU variants and the best competing models of Table 7

Model	MACs (M)	GPU Throughput (images/s)	GPU latency (ms)	TX2 Latency (ms)	iPhone 13 (ms)	Top-1 (%)
EfficientMod-xxs (Ma et al., 2024)	579	2857	2.1	15.0	1.71	76.0
EdgeViT-XS (Chen et al., 2022b)	1127	2127	2.7	16.0	1.5	77.5
MobileOne-S3 (Vasu et al., 2023b)	1895	2433	1.0	11.8	**1.2**	78.1
EfficientMod-xs (Ma et al., 2024)	773	2352	2.5	17.7	2.2	78.3
LowFormer-B0 (ours)	944	5988	2.9	8.5	1.5	78.4
LowFormer-E1 (ours)	1350	**6337**	**1.0**	**6.2**	1.7	**78.8**
FastViT-SA12 (Vasu et al., 2023a)	1943	1075	1.7	17.5	1.6	80.6
EfficientMod-s (Ma et al., 2024)	1402	1381	3.8	30.5	2.6	81.0
RepViT-M1.5 (Wang et al., 2024)	2276	1146	4.2	23.0	**1.5**	81.2
LowFormer-B1.5 (ours)	2573	**2739**	4.8	18.1	2.8	81.2
LowFormer-B2 (ours)	3689	2227	4.8	21.6	3.5	**81.6**
LowFormer-E2 (ours)	3800	2070	**1.5**	**14.7**	2.5	**81.6**
FastViT-SA24 (Vasu et al., 2023a)	3769	606	**3.0**	30.9	2.6	82.6
RepViT-M2.3 (Wang et al., 2024)	4520	642	5.5	40.6	**2.4**	82.5
LowFormer-B3$_{r192}$ (ours)	4479	1562	5.5	30.0	4.5	82.7
LowFormer-E3 (ours)	5350	**1566**	3.6	**25.0**	3.6	**83.0**

LowFormer-E1/E2/E3 (highlighted in gray) consistently rank among the most efficient models in terms of GPU throughput, GPU latency, and TX2 latency

Table 11 Evaluation on transfer learning classification datasets

Model	GPU Throughput ↑ (images/sec)	Flowers Top-1 (%)	Cars Top-1 (%)	Pets Top-1 (%)
ViT-L/16 (Dosovitskiy et al., 2020)	36	89.7	-	93.6
ViT-B/16 (Dosovitskiy et al., 2020)	117	89.5	-	93.8
TNT-S (Han et al., 2021)	141	98.8	-	94.7
DeiT-B (Touvron et al., 2021)	114	**98.9**	93.9	-
EfficientNetV2-M (Tan, 2021b)	277	98.5	**94.6**	-
CeiT-S (Yuan et al., 2021)	260	98.6	94.1	94.9
LowFormer-B3 (ours)	**424**	**98.9**	94.4	**95.0**

All models are finetuned and evaluated on resolution 384×384. Best Results for each column are marked bold. LowFormer model is highlighted in gray

et al., 2013) and Oxford-102 Flowers (Nilsback, 2008), following (Tan, 2021b; Dosovitskiy et al., 2020; Han et al., 2021).

Settings For finetuning LowFormer on the classification datasets, we maintained a setup similar to that used for ImageNet training (see Section 5.1). However we increased training and evaluation resolution to 384×384, applied a batch size of 512, a base learning rate of 2.5×10^4 and removed weight decay, following previous procedures for transfer learning datasets (Tan, 2021b; Dosovitskiy et al., 2020).

We train for 360 steps on the train splits of Oxford-IIIT-Pets (Parkhi et al., 2012), 800 steps on Oxford-102 Flowers (Nilsback, 2008), and 3200 steps on Stanford Cars (Krause et al., 2013).

Results In Table 11, we compare the evaluation results of LowFormer-B3 against both convolutional and transformer-based approaches. GPU throughput is measured at a resolution of 384×384. The LowFormer models mostly achieve superior results across all three datasets (Parkhi et al., 2012; Nilsback, 2008; Krause et al., 2013), while maintaining equal or lower GPU throughput. The ViT (Dosovitskiy et al., 2020) models, for instance, fall significantly behind in both efficiency and accuracy, whereas CeiT-S (Yuan et al., 2021) achieves accuracy results closer to LowFormer-B3 but with only half the GPU throughput.

5.4.2 Object Detection

We show the applicability of the LowFormer architecture for object detection. Backbone GPU throughput and Jetson

Table 12 Comparison results on object detection on COCO 2017 (Lin et al., 2014) using RetinaNet (Lin et al., 2017) head

Backbone	GPU Throughput (images/s)	TX2 Latency (ms)	mAP (%)	mAP_{50} (%)	mAP_{75} (%)	mAP_s (%)	mAP_m (%)	mAP_l (%)
MobileNetV3 (Howard et al., 2019)	862	19.7	29.9	49.3	30.8	14.9	33.3	41.1
MobileNetV4-Conv-M (Qin et al., 2025)	517	27.4	32.6	-	-	-	-	-
EfficientViT-M4 (Liu et al., 2023)	**1700**	**17.6**	32.7	52.2	34.1	17.6	35.3	46.0
PVTv2-B0 (Wang et al., 2022)	355	96.0	37.2	57.2	39.5	**23.1**	40.4	49.7
LowFormer-B0 (ours)	1190	22.4	**38.6**	**59.1**	**40.9**	21.8	**41.8**	**51.7**
EdgeViT-XXS (Chen et al., 2022b)	518	51.8	38.7	59.0	41.0	**22.4**	42.0	51.6
LowFormer-B1 (ours)	**840**	**31.6**	**39.4**	**59.8**	**41.7**	**22.4**	**42.9**	**52.4**
FAT-B0 (Fan et al., 2023)	232	94.2	40.4	61.6	42.7	24.0	44.3	53.1
EdgeViT-XS (Chen et al., 2022b)	400	68.1	40.6	61.3	43.3	25.2	43.9	54.6
PVTv2-B1 (Wang et al., 2022)	215	268.8	41.2	61.9	43.9	**25.4**	44.5	54.3
LowFormer-B2 (ours)	**450**	**63.3**	**41.4**	**62.2**	**44.1**	24.5	**45.1**	**55.5**
FAT-B1 (Fan et al., 2023)	174	125.0	42.5	64.0	45.1	26.9	46.0	**56.7**
LowFormer-B3 (ours)	**245**	**109.0**	**43.1**	**64.5**	**45.9**	**27.1**	**47.1**	**56.7**

LowFormer base models (B0, B1, B2, B3) are able to outperform all compared models in speed accuracy trade-off. Backbone GPU throughput and TX2 latency are measured under resolution of 512×512. LowFormer base models are highlighted in gray

TX2 latency measurement in Table 12 are conducted using an input resolution of 512×512 (Yun, 2024; Vasu et al., 2023a; Fan et al., 2023).

Settings We plug the pretrained LowFormer base models (B0/B1/B2/B3) into the RetinaNet framework (Lin et al., 2017) and utilize COCO 2017 (Lin et al., 2014) for training and evaluation. We train the LowFormer base models for 12 epochs (1x schedule) and following (Cai et al., 2023; Fan et al., 2023) regarding all hyperparameters. As evaluation metric we use mean average precision (mAP)[8] (Chen et al., 2022b; Fan et al., 2023).

Results In Table 12, we compare the performance of LowFormer base models in object detection against recent vision backbones. LowFormer-B2 for example outperforms FAT-B0 (Fan et al., 2023) by **+1.0** AP, while having 93% higher backbone throughput on resolution 512×512 and only 67% of its latency. On the other side, LowFormer-B0 with a smaller model capacity is able to achieve an increase in AP of **+1.4** compared to PVTv2-B0 (Wang et al., 2022), while being 4× faster in terms of TX2 latency.

In summary, LowFormer base models are able to outperform all compared vision backbones in terms of speed accuracy trade-off, when plugged into the RetinaNet framework (Lin et al., 2017).

[8] Mean average precision (mAP) is commonly abbreviated as AP in many publications that evaluate on COCO. Within the context of COCO evaluation, AP always refers to mAP.

5.4.3 Semantic Segmentation

We further demonstrate LowFormer's applicability to semantic segmentation in a similar fashion as object detection. GPU throughput and TX2 latency is again measured with an input resolution of 512×512 (Yun, 2024; Vasu et al., 2023a; Fan et al., 2023).

Settings We plug the pretrained LowFormer base models (B1,B2,B3) into the Semantic FPN framework (Kirillov et al., 2019) and use the ADE20K dataset (Zhou et al., 2017) for training and evaluation. We train the models for 40K iterations with a batch size of 32, following (Fan et al., 2023; Wang et al., 2024; Vasu et al., 2023a; Ma et al., 2024). We use AdamW optimizer (Loshchilov & Hutter, 2017), cosine annealing for the learning rate (Loshchilov, 2016) with a base learning rate of 2×10^{-3} and 1K warm-up steps with linear increase. As evaluation metric we use mean intersection over union (mIoU) (Vasu et al., 2023a; Wang et al., 2024).

Results In Table 13, we compare the performance of LowFormer models in semantic segmentation against recent vision backbones. LowFormer-B2 for example has 2.4× the throughput and a 25% lower latency than EfficientFormerV2-S2(Li et al., 2023), but achieves **+0.4** mIoU when plugged into Semantic FPN. FastViT-MA36 (Vasu et al., 2023a) achieves a similar mIoU as LowFormer-B3, but has approximately twice the TX2 latency and 35% of its GPU throughput.

In summary, LowFormer models show significant efficiency gains compared to previous approaches, while maintaining a similar or superior mIoU.

Table 13 Results on semantic segmentation, using Semantic FPN (Kirillov et al., 2019)

Backbone	GPU Throug. (images/s)	TX2 Lat. (ms)	mIoU (%)
ResNet50 (He et al., 2016)	271	45.9	36.7
PVTv2-B0 (Wang et al., 2022)	355	96.0	37.2
FastViT-SA12 (Vasu et al., 2023a)	265	62.0	38.0
EdgeViT-XXS (Chen et al., 2022b)	518	51.8	**39.7**
LowFormer-B1 (ours)	**840**	**31.6**	**39.7**
RepViT-M1.1 (Wang et al., 2024)	404	**55.0**	40.6
FastViT-SA24 (Vasu et al., 2023a)	151	109.9	41.0
EdgeViT-XS (Chen et al., 2022b)	400	68.1	41.4
FAT-B0 (Fan et al., 2023)	232	94.2	41.5
EfficientFormerV2-S2 (Li et al., 2023)	182	85.3	42.4
PVTv2-B1 (Wang et al., 2022)	215	268.8	42.5
LowFormer-B2 (ours)	**450**	63.3	**42.8**
FAT-B1 (Fan et al., 2023)	174	125.0	42.9
RepViT-M1.5 (Wang et al., 2024)	238	217.4	43.6
FastViT-MA36 (Vasu et al., 2023a)	86	208.4	**44.6**
LowFormer-B3 (ours)	**245**	**109.0**	**44.6**

LowFormer models achieve superior speed mIoU trade-offs. Backbone GPU throughput and TX2 latency are measured under resolution of 512×512. Results are grouped by mIoU. Bold marks the best results in each group and column, underline refers to the second best. LowFormer models are highlighted in gray

5.4.4 Image Retrieval

In order to evaluate the quality of the image embedding of LowFormer, we compare it on the GPR1200 (General-Purpose Image Retrieval) benchmark (Schall et al., 2022).

Settings The GPR1200 benchmark data is selected from several datasets, namely Google Landmarks V2, ImageNet Sketch, INat, INSTRE, SOP and IMDB Faces. The combination ensures that the data spans a variety of domains. In total GPR1200 features 12k images and 1200 different classes. For evaluation we follow the protocol of (Schall et al., 2022) and measure performance by mean-Average-Precision (mAP). All models are executed on the input image resolution they are executed on for ImageNet evaluation. As image embedding we take the output after the final pooling, that reduces the resolution to 1×1. We compare models by GPU throughput, as this efficiency measure directly reflects a model's ability to process large amounts of data, which is crucial for retrieving image embeddings from large datasets.

Results Results of the evaluation are depicted in Table 14. LowFormer architecture variants are able to clearly outperform all compared models in throughput accuracy trade-off (see Table 14). Compared to the recently published MobileNetv4-Conv-Medium (Qin et al., 2025), LowFormer-B1 achieves a 0.5% higher mAP and processes 54% more images in the same time. Though FastViT-SA36 (Vasu et al., 2023a) for example achieves a similar mAP score as LowFormer-B3, it has less than half of the GPU throughput.

5.4.5 Visual Object Tracking

Besides tasks that process images separately, we applied the LowFormer architecture to the video task of single object tracking (Kristan et al., 2020; Wu & Lim, 2015). We use the SMAT architecture (Gopal, 2024) as a baseline, replacing its backbone (Mehta, 2022b) with LowFormer-B1.5 and substituting the attention layers in its Separable Self-Attention Head with Lowtention layers. We refer to this adapted model as LowFormer-Track.

Settings We train LowFormer-Track and the baseline SMAT on the train splits of LaSOT (Fan et al., 2019), GOT10K (Huang & Zhao, 2019), and COCO (Lin et al., 2014). For the latter dataset, we use data augmentations to generate image pairs from the still images, following (Gopal, 2024). To best compare LowFormer-Track and SMAT, we assimilate their efficiency by using LowFormer-B1.5 as a backbone, as well as train and evaluate it on a slightly lower resolution than SMAT. For search images we resize them to 224×224 instead of 256×256, for template images we resize to 112×112 instead of 128×128. Besides that, we adopt the hyperparameter setting of SMAT for both models. We assess both models performance across six diverse and widely used single object tracking benchmarks: the validation set of GOT10K (Huang & Zhao, 2019), the test set of LaSOT (Fan et al., 2019),

Table 14 Image retrieval results on GPR1200 (Schall et al., 2022) benchmark

Backbone	Resol. (pixel)	GPU Throug. (images/s)	mAP (%)
EfficientViT-M5 (Liu et al., 2023)	224	5681	31.9
SHViT-S4 (Yun, 2024)	256	4255	35.7
FastViT-T8 (Vasu et al., 2023a)	256	1694	42.1
ResNet-101* (He et al., 2016)	224	868	42.8
LowFormer-B0 (ours)	224	**5988**	**44.0**
EfficientViT-B1 (Cai et al., 2023)	224	2739	44.6
MNv4-Conv-M (Qin et al., 2025)	224	2741	45.3
LowFormer-B1 (ours)	224	**4237**	**45.8**
CoaT-Lite Tiny (Xu et al., 2021)	224	1153	46.4
EfficientViT-B2 (Cai et al., 2023)	224	1298	47.3
LowFormer-B1.5 (ours)	224	**2739**	**47.6**
FastViT-SA12 (Vasu et al., 2023a)	256	1075	48.0
CoaT-Lite Mini (Xu et al., 2021)	224	1065	48.3
EfficientViT-L1 (Cai et al., 2023)	224	1020	48.4
FastViT-SA36 (Vasu et al., 2023a)	256	429	**49.0**
EfficientNetV2-S (Tan, 2021b)	300	690	**49.0**
LowFormer-B3 (ours)	224	**1162**	**49.0**

LowFormer models achieve superior speed accuracy trade-offs. Results marked with * are taken from (Schall et al., 2022). Models are grouped by mAP. Entries marked as bold, refer to the best results in the respective group of the table. LowFormer models are highlighted in gray

Table 15 Evaluation of LowFormer-Track with its baseline tracker SMAT (Gopal, 2024)

Model	TX2 fps	GPU fps	GOT10K-val AUC	GOT10K-val P	LaSOT-Test AUC	LaSOT-Test P	TREK-150 AUC	TREK-150 P	NfS30 AUC	NfS30 P	AVisT AUC	AVisT P	UAV123 AUC	UAV123 P
SMAT (Gopal, 2024)	**53**	90	77.0	66.3	60.4	62.8	39.6	**23.2**	62.4	74.0	46.0	41.5	64.1	83.7
LowFormer-Track (ours)	51	**92**	**78.9**	**69.3**	**61.7**	**64.6**	**39.7**	22.4	**63.1**	**75.0**	**47.0**	**42.0**	**65.2**	**85.2**

LowFormer-Track is an adaptation of the SMAT architecture that replaces the backbone with a LowFormer-B1.5 and changes the attention layers in the head to Lowtentionlayers. LowFormer-Track is similarly efficient as SMAT, but consistently outperforms SMAT in terms of AUC and Precision, showing the benefits of the hardware efficient design of LowFormer and Lowtention. Bold values mark the best in each column

the TREK-150 first person vision benchmark (Dunnhofer et al., 2023), the NfS benchmark (Galoogahi et al., 2017) that predominantly contains fast-moving objects, the AVisT (Noman et al., 2022) benchmark featuring diverse scenarios with reduced object visibility, and the UAV123 benchmark (Mueller & Smith, 2016) consisting of sequences captured from an aerial perspective.

Results In Table 15, we compare LowFormer-Track and SMAT on the aforementioned single object tracking benchmarks using the Area-Under-the-Curve (AUC) and Precision (P) metric, following Gopal (2024). Both achieve similar fps (frames per second) on the Jetson TX2 and GPU. However, LowFormer-Track surpasses SMAT in AUC across all benchmarks. In terms of Precision, SMAT achieves a higher score on the TREK-150 benchmark but lags behind by a large margin in all other cases. The superior results achieved by our adaptation of the SMAT framework highlight the significance of efficient backbone design as well as the versatility and effectiveness of our proposed Lowtention for computer vision tasks beyond image understanding.

6 Conclusion

In this paper, we have examined the relationship of MACs and execution time for several architectural design choices in vision backbones, such as depthwise convolutions, operating resolution of layers, fusing the MBConv block, and attention mechanisms. We have particularly shown how the execution time differs between several different devices and that MACs can be an insufficient predictor of it. The analysis further gave us guidance to create a new hardware efficient vision backbone architecture family, named LowFormer, that features Lowtention, a lightweight adaptation of the original MHSA. The LowFormer base models (LowFormer-B0-B3) surpass competing approaches in terms

of speed accuracy trade-off on GPU, the Jetson TX2, and ARM CPU. Additionally, we presented three edge GPU variants of LowFormer (LowFormer-E1/E2/E3) that further enhance efficiency on the Nvidia Jetson TX2 and GPU. We have shown that using LowFormer as a backbone improves efficiency across several downstream computer vision tasks, including various transfer learning image classification datasets, object detection, semantic segmentation, and image retrieval. We also presented LowFormer-Track, an adaptation of a recently published tracking framework, where we apply LowFormer-B1.5 as a backbone and incorporate our proposed Lowtention, clearly outperforming the baseline. Overall, the results achieved demonstrate the wide applicability of our LowFormer architecture for the implementation of efficient computer vision pipelines.

Acknowledgements This research has been funded by the European Union, NextGenerationEU – PNRR M4 C2 I1.1, RS Micheloni. Progetto PRIN 2022 EXTRA-EYE CUP G53D23002920006, PRIN 2022 PNRR TEAM CUP G53D23006680001. MD received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement n. 101151834 PRINNEVOT (CUP G23C24000910006). Open access funding provided by Universitlá degli Studi di Udine within the CRUI-CARE Agreement.

Funding Open access funding provided by Università degli Studi di Udine within the CRUI-CARE Agreement.

Data Availability ImageNet (Deng et al., 2009) dataset is available at https://www.image-net.org/. GOT10K (Huang & Zhao, 2019), LaSOT (Fan et al., 2019), TREK-150 (Dunnhofer et al., 2023), NfS30 (Galoogahi et al., 2017), AVisT (Noman et al., 2022), GPR1200 (Schall et al., 2022), ADE20K (Zhou et al., 2017), COCO 2017 (Lin et al., 2014), Oxford-IIIT-Pets (Parkhi et al., 2012), Oxford-102 Flowers (Nilsback, 2008) and Stanford Cars (Krause et al., 2013) are publicly available and can be found under the references.

References

Bansal, V., Micheloni, C., Foresti, G., et al. (2022). Spatio-temporal attention for cloth-changing reid in videos. In:European Conference on Computer Vision, Springer, pp. 353–368.

Bolya, D., Fu, C. Y., Dai, X., et al. (2022). Hydra attention: Efficient attention with many heads. In:European Conference on Computer Vision, Springer, pp. 35–49.

Brock, A., De, S., Smith, S. L., et al. (2021). High-performance large-scale image recognition without normalization. In:International conference on machine learning, PMLR, pp. 1059–1071.

Cai, H., Li, J., Hu, M., et al. (2023). Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In:Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 17302–17313.

Carion, N., Massa, F., Synnaeve, G., et al. (2020). End-to-end object detection with transformers. In:European conference on computer vision, Springer, pp. 213–229.

Chen, J., Kao, S., He, H., et al. (2023). Run, don't walk: Chasing higher flops for faster neural networks. In:Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12021–12031.

Chen, Y., Dai, X., Chen, D., et al. (2022). Mobile-former: Bridging mobilenet and transformer. In:Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 5270–5279.

Chen, Z., Zhong, F., Luo, Q., et al. (2022). Edgevit: Efficient visual modeling for edge computing. In:International Conference on Wireless Algorithms, Systems, and Applications, Springer, pp. 393–405.

Chu X, Tian Z, Zhang B, et al (2021) Conditional positional encodings for vision transformers. arXiv preprint arXiv:2102.10882

Dai, Z., Liu, H., Le, Q. V., et al. (2021). Coatnet: Marrying convolution and attention for all data sizes. *Advances in neural information processing systems, 34*, 3965–3977.

Dehghani, M., Arnab, A., Beyer, L., et al. (2021). The efficiency misnomer. arXiv preprint arXiv:2110.12894

Deng, J., Dong, W., Socher, R., et al. (2009). (2009) Imagenet: A large-scale hierarchical image database. In:IEEE conference on computer vision and pattern recognition, Ieee, pp. 248–255.

Dosovitskiy, A., Beyer, L, Kolesnikov, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Dunnhofer, M., Martinel, N., & Micheloni, C. (2021). Weakly-supervised domain adaptation of deep regression trackers via reinforced knowledge distillation. *IEEE Robotics and Automation Letters, 6*(3), 5016–5023.

Dunnhofer, M., Simonato, K. & Micheloni, C. (2022). Combining complementary trackers for enhanced long-term visual object tracking. *Image and Vision Computing,122*, Article 104448.

Dunnhofer, M., Furnari, A., Farinella, G. M., et al. (2023). Visual object tracking in first person vision. *International Journal of Computer Vision, 131*(1), 259–283.

Fan, H., Lin, L., Yang, F., et al. (2019). Lasot: A high-quality benchmark for large-scale single object tracking. In:Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 5374–5383.

Fan, H., Xiong, B., Mangalam, K., et al. (2021). Multiscale vision transformers. In:Proceedings of the IEEE/CVF international conference on computer vision, pp. 6824–6835.

Fan, Q., Huang, H., Zhou, X., et al. (2023). Lightweight vision transformer with bidirectional interaction. Advances in Neural Information Processing Systems 36

Galoogahi, HK., Fagg, A., Huang, C., et al. (2017). Need for Speed: A Benchmark for Higher Frame Rate Object Tracking. In: ICCV

Ganesh, P., Chen, Y., Yang, Y., et al. (2022). Yolo-ret: Towards high accuracy real-time object detection on edge gpus. In:Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 3267–3277.

Gopal, G. Y. & Amer, M. A. (2024). Separable self and mixed attention transformers for efficient object tracking. In:Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 6708–6717.

Gopalkrishnan, A., Greer, R., Trivedi, M. (2024). Multi-frame, lightweight & efficient vision-language models for ques-

tion answering in autonomous driving. arXiv preprint arXiv:2403.19838
Gupta, S., Tan, M. (2019). Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl. https://aigoogleblogcom/2019/08/efficientnetedgetpu-creatinghtml
Han, K., Wang, Y., Tian, Q., et al. (2020). Ghostnet: More features from cheap operations. In:Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1580–1589.
Han, K., Xiao, A., Wu, E., et al. (2021). Transformer in transformer. *Advances in neural information processing systems, 34*, 15908–15919.
He, K., Zhang, X., Ren, S., et al. (2016). Deep residual learning for image recognition. In:Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.
Hendrycks, D. & Gimpel, K. (2016). Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415
Howard, A., Sandler, M., Chu, G., et al. (2019). Searching for mobilenetv3. In:Proceedings of the IEEE/CVF international conference on computer vision, pp. 1314–1324.
Huang, L., Zhao, X. & Huang, K. (2019). GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. IEEE TPAMI
Ioffe, S. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167
Jiang, B., Chen, S., Xu, Q., et al. (2023). Vad: Vectorized scene representation for efficient autonomous driving. In:Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8340–8350.
Khan, A. H., Micheloni, C. & Martinel, N. (2024). Idenet: Implicit degradation estimation network for efficient blind super resolution. In:Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6065–6075.
Kirillov, A., Girshick, R., He, K., et al. (2019). Panoptic feature pyramid networks. In:Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 6399–6408.
Kirillov, A., Mintun, E., Ravi, N., et al. (2023). Segment anything. In:Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4015–4026.
Kong, Y., & Fu, Y. (2022). Human action recognition and prediction: A survey. *International Journal of Computer Vision, 130*(5), 1366–1401.
Krause, J., Stark, M., Deng, J., et al. (2013). 3d object representations for fine-grained categorization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops
Kristan, M., Leonardis, A., Matas, J., et al. (2020). The eighth visual object tracking vot2020 challenge results
LeCun, Y., Boser, B., Denker, J. S., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation, 1*(4), 541–551.
Leo, M., Medioni, G., Trivedi, M., et al. (2017). Computer vision for assistive technologies. *Computer Vision and Image Understanding, 154*, 1–15.
Li, Y., Yuan, G., Wen, Y., et al. (2022). Efficientformer: Vision transformers at mobilenet speed. *Advances in Neural Information Processing Systems, 35*, 12934–12949.
Li, Y., Hu, J., Wen, Y., et al. (2023). Rethinking vision transformers for mobilenet size and speed. In:Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16889–16900.
Lin, TY., Maire, M., Belongie, S., et al. (2014). Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, Springer, pp 740–755
Lin, T. Y., Goyal, P., Girshick, R., et al. (2017). Focal loss for dense object detection. In:Proceedings of the IEEE international conference on computer vision, pp. 2980–2988.
Lin, W., Wu, Z., Chen, J., et al. (2023). Scale-aware modulation meet transformer. In:Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6015–6026.
Liu, S., Zeng, Z., Ren, T., et al. (2025). Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In:European Conference on Computer Vision, Springer, pp. 38–55.
Liu, X., Peng, H., Zheng, N., et al. (2023). Efficientvit: Memory efficient vision transformer with cascaded group attention. In:Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14420–14430.
Loshchilov, I. & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983
Loshchilov, I., Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101
Ma, J., Jiang, X., Fan, A., et al. (2021). Image matching from handcrafted to deep features: A survey. In:International Journal of Computer Vision, *129*(1), 23–79.
Ma, N., Zhang, X., Zheng, H. T., et al. (2018a). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In:Proceedings of the European conference on computer vision (ECCV), pp. 116–131.
Ma, N., Zhang, X., Zheng, H. T., et al. (2018b). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In:Proceedings of the European conference on computer vision (ECCV), pp. 116–131.
Ma, X., Dai, X., Yang, J., et al. (2024). Efficient modulation for vision networks. arXiv preprint arXiv:2403.19963
Mahendran, J. K., Barry, D. T., Nivedha, A. K., et al. (2021). Computer vision-based assistance system for the visually impaired using mobile edge artificial intelligence. In:Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2418–2427.
Matthies, L., Maimone, M., Johnson, A., et al. (2007). Computer vision on mars. *International Journal of Computer Vision, 75*, 67–92.
Mehta, S.,& Rastegari, M. (2021). Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. arXiv preprint arXiv:2110.02178
Mehta, S. & Rastegari, M. (2022). Separable self-attention for mobile vision transformers. arXiv preprint arXiv:2206.02680
Mueller, M., Smith, N. & Ghanem, B. (2016). A Benchmark and Simulator for UAV Tracking. In: ECCV
Munir, M., Avery, W. & Marculescu, R. (2023). Mobilevig: Graph-based sparse attention for mobile vision applications. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 2211–2219
Nilsback, ME. & Zisserman, A. (2008). Automated flower classification over a large number of classes. In: Indian Conference on Computer Vision, Graphics and Image Processing
Noman, M., Ghallabi, WA., Najiha, D., et al. (2022). Avist: A benchmark for visual object tracking in adverse visibility. arXiv preprint arXiv:2208.06888
Nottebaum, M., Dunnhofer, M. & Micheloni, C. (2025). Lowformer: Hardware efficient design for convolutional transformer backbones. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision
Paissan, F., Ancilotto, A., & Farella, E. (2022). Phinets: a scalable backbone for low-power ai at the edge. *ACM Transactions on Embedded Computing Systems, 21*(5), 1–18.
Parkhi, OM., Vedaldi, A., Zisserman, A., et al. (2012). Cats and dogs. In: 2012 IEEE conference on computer vision and pattern recognition, IEEE, pp. 3498–3505

Qin, D., Leichner, C., Delakis, M., et al. (2025). Mobilenetv4: Universal models for the mobile ecosystem. *European Conference on Computer Vision, Springer,* pp. 78–96.
Sandler, M., Howard, A., Zhu, M., et al. (2018). Mobilenetv 2: Inverted residuals and linear bottlenecks. In:Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510–4520.
Schall, K., Barthel, K. U., Hezel, N., et al. (2022). Gpr1200: a benchmark for general-purpose content-based image retrieval. In:International Conference on Multimedia Modeling, Springer, pp. 205–216.
Si, C., Yu, W., Zhou, P., et al. (2022). Inception transformer. *Advances in Neural Information Processing Systems, 35*, 23495–23509.
Suzuki, T., & Aoki, Y. (2024). Retinavit: Efficient visual backbone for online video streams. *Sensors, 24*(17), 5457.
Tan, M. & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In:International conference on machine learning, PMLR, pp. 6105–6114.
Tan, M. & Le, Q. (2021). Efficientnetv2: Smaller models and faster training. In:International conference on machine learning, PMLR, pp. 10096–10106.
Tan, S., Nagarajan, T., & Grauman, K. (2023). Egodistill: Egocentric head motion distillation for efficient video understanding. *Advances in Neural Information Processing Systems, 36*, 33485–33498.
Tang, Y., Han, K., Guo, J., et al. (2022). Ghostnetv2: Enhance cheap operation with long-range attention. *Advances in Neural Information Processing Systems, 35*, 9969–9982.
Touvron, H., Cord, M., Douze, M., et al. (2021). Training data-efficient image transformers & distillation through attention. In:International conference on machine learning, PMLR, pp. 10347–10357.
Tu, Z., Talebi, H., Zhang, H., et al. (2022). Maxvit: Multi-axis vision transformer. In:European conference on computer vision, Springer, pp. 459–479.
Upmanyu, M., Namboodiri, A.M., Srinathan, K., et al. (2009). Efficient privacy preserving video surveillance. In: 2009 IEEE 12th international conference on computer vision, IEEE, pp 1639–1646
Vasu, P. K. A., Gabriel, J., Zhu, J., et al. (2023a). Fastvit: A fast hybrid vision transformer using structural reparameterization. In:Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5785–5795.
Vasu, P. K. A., Gabriel, J., Zhu, J., et al. (2023b). Mobileone: An improved one millisecond mobile backbone. In:Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 7907–7917.
Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. Advances in neural information processing systems 30
Wang, A., Chen, H., Lin, Z., et al. (2024). Repvit: Revisiting mobile cnn from vit perspective. In:Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15909–15920.
Wang, S., Li, B.Z., Khabsa, M., et al. (2020). Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768
Wang, W., Xie, E., Li, X., et al. (2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In:Proceedings of the IEEE/CVF international conference on computer vision, pp. 568–578.
Wang, W., Xie, E., Li, X., et al. (2022). Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media, 8*(3), 415–424.
Wen, J., Zhu, Y., Li, J., et al. (2024). Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. arXiv preprint arXiv:2409.12514
Wu, H., Xiao, B., Codella, N., et al. (2021). Cvt: Introducing convolutions to vision transformers. arXiv:2103.15808
Wu, Y., Lim, J. & Yang, M.H. (2015). Object tracking benchmark. IEEE TPAMI
Xu, W., Xu, Y., Chang, T., et al. (2021). Co-scale conv-attentional image transformers. In:Proceedings of the IEEE/CVF international conference on computer vision, pp. 9981–9990.
Yan, B., Peng, H., Fu, J., et al. (2021). Learning spatio-temporal transformer for visual tracking. In:Proceedings of the IEEE/CVF international conference on computer vision, pp. 10448–10457.
Yu, W., Luo, M., Zhou, P., et al. (2022). Metaformer is actually what you need for vision. In:Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10819–10829.
Yuan, K., Guo, S., Liu, Z., et al. (2021). Incorporating convolution designs into visual transformers. In:Proceedings of the IEEE/CVF international conference on computer vision, pp. 579–588.
Yun, S. & Ro, Y. (2024). Shvit: Single-head vision transformer with memory efficient macro design. arXiv preprint arXiv:2401.16456
Yun, S., Lee, D. & Ro, Y. (2024). Metamixer is all you need. arXiv preprint arXiv:2406.02021
Zhai, X., Kolesnikov, A., Houlsby, N., et al. (2022). Scaling vision transformers. In:Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 12104–12113.
Zhang, Z., Lu, X., Cao, G., et al. (2021). Vit-yolo: Transformer-based yolo for object detection. In:Proceedings of the IEEE/CVF international conference on computer vision, pp. 2799–2808.
Zheng, C., Wu, W., Chen, C., et al. (2023). Deep learning-based human pose estimation: A survey. *ACM Computing Surveys, 56*(1), 1–37.
Zhou, B., Zhao, H., Puig, X., et al. (2017). Scene parsing through ade20k dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition
Zhu, L., Wang, X., Ke, Z., et al. (2023). Biformer: Vision transformer with bi-level routing attention. In:Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10323–10333.