TakuNet: Energy-Efficient Models for Real-Time Aerial Disaster Response and Monitoring on Edge Devices

Daniel Rossi[a,*], Gianluca Filippini[a,c], Andrea Torlai[d], Guido Borghi[b], Roberto Vezzani[a]

[a]*Department of Engineering "Enzo Ferrari" - University of Modena and Reggio Emilia, Via Vivarelli 10, Modena, 41125, MO, Italy*
[b]*Department of Education and Humanities - University of Modena and Reggio Emilia, Via Timavo 3, Reggio Emilia, 41125, RE, Italy*
[c]*EbV Elektronik, Via alessandro manzoni 44, Cusano Milanino, 20095, MI, Italy*
[d]*System Electronics , via Ghiarola Vecchia 67, Fiorano Modenese, 41042, MO, Italy*

Abstract

In this work, we present TakuNet, a family of ultra-lightweight convolutional neural networks designed for real-time aerial image classification on resource-constrained embedded devices. The proposed TakuNetV2 architecture enhances feature extraction and generalization capabilities through the incorporation of a denser stem coupled with hybrid feature extractor blocks, wherein diverse convolutional operations are synergistically combined to yield richer spatial representations without compromising latency or parameter efficiency. We extensively evaluate the TakuNet family on three public aerial image classification datasets against well-known light-weight and ultra-lightweight architectures, and measured relative performance on five heterogeneous embedded platforms, spanning from CPUs to GPUs, and the Hailo-8 NPU. TakuNet achieves state-of-the-art accuracy and energy efficiency, outperforming competing models in frames per second per additional watt consumed, confirming its suitability for battery-powered edge devices. Additionally, this paper introduces the Astrial platform, highlighting its role in enabling efficient deep learning inference on industrial-grade edge applications. Although current NPU hardware and compiler limitations pose challenges, TakuNet sets a new benchmark for efficient, high-performance embedded artificial intelligence in aerial surveillance and emergency response. Code, models, and weights are publicly available (https://github.com/DanielRossi1/TakuNetV2).

Keywords: Computer Vision, Deep Learning, Embedded Devices, Disaster Management, Real-Time Analysis
PACS: 42.30.Tz, 42.30.Sy
2008 MSC: 68T45

1. Introduction

Since its early days, Artificial Intelligence has made remarkable strides in both the variety of tasks it addresses and the level of accuracy it can achieve. Generally, these results have been reached by increasingly augmenting the dimension and the complexity of neural network models [1], in combination with larger and larger datasets [2] used for their training phase. However, this continuous growth, specifically in model size and parameter count, has progressively degraded inference performance [3], undermining the feasibility of real-time execution without relying on high-end, costly hardware accelerators such as Graphics Processing Units (GPUs).

Consequently, reducing the size and complexity of such models has become an increasingly pressing requirement, enabling effective generalization across a broader range of hardware platforms subject to constraints in computational capacity, memory availability, and energy budget.

Moreover, the interest in deploying Computer Vision (CV)-based models on embedded devices is growing, making the adoption of AI solutions feasible on a variety of tasks. This is the case, for instance, of devices – *e.g.*, drones or Unmanned Aerial Vehicles (UAVs) – for wide-area monitoring. In fact, several application domains benefit from Artificial Intelligence (AI)-powered high-altitude imagery analysis, including precision agriculture [4], environmental and urban monitoring [5], as well as search and rescue operations [6] and risk prevention [7], as depicted in Figure 1. High-altitude image

*Corresponding author

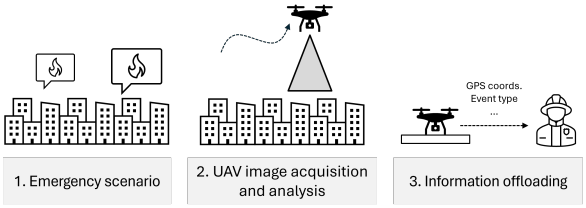

Figure 1: High-altitude imagery can be valuable in various scenarios, including emergencies where a drone acquires and processes images, and subsequently offloads the information to support the monitoring and coordination of rescue operations. In such cases, it is desirable to employ AI systems that can run on embedded boards with limited computational resources and low energy consumption.

acquisition can be performed using various platforms, such as satellites, airplanes or helicopters, airships, stratospheric balloons, drones, and UAVs. Among the others, drones and UAVs have attracted significant interest due to several features that make them highly advantageous for this purpose: they are considerably more cost-effective than alternative solutions, easy to operate, and capable of capturing stabilized, high-resolution imagery. Despite their compact size and low cost, they are capable of operating in remote or hard-to-reach areas, can reach altitudes comparable to commercial airliners, and are able to carry payloads of several tens of kilograms, thus enabling the integration of advanced sensing equipment. It is worth noting that, unfortunately, current wireless telecommunication infrastructures often pose significant constraints on transmitting high-frequency image streams over large distances, making it preferable, if not essential, to execute AI models directly on these embedded devices. The deployment of modern AI and computer vision models on embedded devices remains an open and actively investigated challenge, primarily owing to the stringent resource constraints inherent to such platforms [8]. These limitations encompass severely restricted computational capacity, the frequent absence of dedicated hardware accelerators for neural network inference, tightly constrained memory, and, most critically, a limited energy budget dictated by the onboard battery.

Therefore, in order to run models in real time on these platforms while minimizing energy consumption, it is essential to make efficiency-oriented choices from the very first stages of architecture design [9], while also meeting the requirements of the specific hardware accelerator to be used for inference. Indeed, due to the strict deployment constraints of running neural network models at the edge, inference is often offloaded to the cloud, with the edge device limited to data acquisition and transmission. In practice, the remote device collects and sends images or video streams to dedicated servers via a communication protocol, where the data is then processed. However, as mentioned, this approach is not simple and always feasible, due to several challenges that include the development of a robust communication protocol capable of reliably transferring data over long distances, ensuring connectivity in remote or isolated areas where signal coverage may be lacking, and guaranteeing the reliability of communication channels, which can be severely compromised during natural disasters or other critical events.

Two broad methodological directions may be distinguished for the on-device deployment of such models: the mitigation of model complexity through architectural simplification, and the enhancement of computational throughput maintaining the same energy consumption, *i.e.*, without incurring additional energy overhead.

In the first case, the focus is on reducing the model size while introducing more efficient operations. This refers, for example, to optimized learnable layers or blocks that, at equal or marginally reduced final accuracy of the overall model, are computationally lighter and require fewer parameters. In the second case, the strategy relies on adopting more advanced hardware accelerators, such as improved processors or dedicated neural network accelerators, such as Neural Processing Units (NPUs) [10], at the expense of a generally much higher cost and, sometimes, higher energy consumption.

Following a combination of the two approaches, this paper provides a detailed description of the **TakuNet** family of architectures, which includes ultra-lightweight models and Astrial, an embedded platform equipped with the Hailo-8 NPU. The base architecture, TakuNetV1 was preliminarily introduced [11] and serves as the foundation of the family. Here, we present with more details TakuNetV1 and we introduce the enhanced TakuNetV2 variant, along with additional experiments on new datasets and their deployment on Astrial. The TakuNet family consistently achieves excellent performance in classification tasks, maintaining a minimal parameter count and computational cost. Among all tested models, the TakuNet family stands out for its superior results, with the second version (*i.e.*, TakuNetV2) demonstrating notably higher accuracy and establishing itself as the most effective solution for embedded aerial image analysis.

From an architectural standpoint, the TakuNet family is organized into four distinct stages, at the boundary of each of which a downsampler block reduces the

spatial resolution of the feature maps while enforcing a dense connectivity pattern, a design choice that promotes rapid convergence during training. Furthermore, both model variants are trained at a floating-point precision of 16 bits, facilitating lossless optimization on hardware accelerators commonly employed in embedded systems. Leveraging NVIDIA TensorRT[1], this configuration yields a substantial improvement in inference performance.

The experimental evaluation of model accuracy is conducted on three public datasets: AIDER [12] and AIDERv2 [13], which contain aerial images depicting emergency situations such as floods, traffic accidents, and fires, and CLRS [14], a remote sensing benchmark focused on diverse scene classification across 25 categories, including urban, natural, and infrastructure scenes. This selection reflects both the practical application domains for embedded devices mounted on drones and the broader challenges of aerial image interpretation. Despite having far fewer parameters than typical models reported in the literature, the proposed method achieves accuracy close to or surpassing the state-of-the-art on all three datasets, demonstrating its effectiveness and versatility for embedded vision tasks.

Both TakuNetV1 and TakuNetV2 have been tested on a variety of embedded devices, equipped with different computational resources, in terms of processor, memory and energy consumption. Specifically, in addition to the Astrial Platform, we evaluate the proposed method on the first three versions of the Raspberry Pi, on the NVIDIA Jetson Nano and NVIDIA Jetson Orin Nano. These evaluations demonstrate the feasibility of running the TakuNet family inference on edge devices with extremely low power consumption and remarkable throughput, in terms of frame rates.

In summary, the contributions of this work are as follows:

1. We propose two versions of the TakuNet model, named TakuNetV1 [11] and TakuNetV2 respectively. Both of them are ultra-lightweight architectures specifically designed to improve classification accuracy while preserving high efficiency and low parameter count. These architectures are conceived for deployment across a heterogeneous range of embedded platforms with limited computational power and energy budgets, such as those found in drones and UAV systems.
2. An extensive experimental evaluation is conducted on three public datasets for event recognition from aerial images, which constitute use cases for artificial vision applications deployed on drones and UAV platforms. This evaluation highlights both classification accuracy, in terms of F1-score, and theoretical hardware impact, in terms of FLoating-point Operations Per Second (FLOPS).
3. We introduce and describe the Astrial platform, a single board computer which integrates the HAILO co-processor and enables AI-powered solutions in different applications, including embedded and UAV systems in Emergency Response Scenarios.
4. Both versions of TakuNet models are evaluated on the Astrial board as well as five real-world embedded platforms, including Jetson Nano, Jetson Orin Nano, and three versions of the Raspberry Pi board. Notably, TakuNetV2 achieves more than real-time performance and sets the benchmark for energy efficiency, reaching 21.20 FPS/δ-Power on Raspberry Pi 5, 101.87 FPS/δ-Power on Jetson Orin Nano, and 761.70 FPS/δ-Power on the Astrial platform. In light of these results and the public availability of both code and models, TakuNet stands out as an efficient and deployable solution for embedded computer vision scenarios.

2. Related Work

2.1. Evolution of Neural Network Architectures for Vision Tasks

Over the past two decades, computer vision has undergone a profound methodological evolution, propelled by the progressive sophistication and diversification of neural network architectures. This trajectory reflects a continuous negotiation between representational capacity, computational efficiency, and optimization stability, with each generation of models addressing fundamental limitations of its predecessors while introducing novel architectural principles and design innovations.

Pioneering contributions to convolutional neural networks (CNNs) established the basic idea of local receptive fields, weight sharing, and spatial sub-sampling. Early models like LeNet-5 [15] showed that hierarchical feature learning could work for optical character recognition. LeNet-5 used a structured layout of alternating convolutions and average pooling, finishing with fully connected layers. However, its ability was limited by the small computational resources and datasets at that time, which restricted both the network's depth and its

[1] https://developer.nvidia.com/tensorrt

broader use. The arrival of AlexNet [16] marked a significant milestone by greatly expanding the CNN architecture. This helped achieve impressive results on the ImageNet dataset [17]. AlexNet's success came from important technical innovations, not just its depth. It replaced saturating activation functions with Rectified Linear Units (ReLU), which sped up convergence and reduced the vanishing gradient issue. Additionally, it introduced dropout regularization in the fully connected layers to prevent overfitting, used Local Response Normalization (LRN) to improve generalization, and applied overlapping max-pooling to decrease translation invariance errors. Importantly, its design was specifically made for hardware. It used a new model-parallel architecture spread across two GPUs to get past the memory limits of the time.

Subsequent improvements examined design choices to improve deep feature extraction. VGG [18] fully standardized the convolutional layer design. It established the exclusive use of small 3×3 convolutional kernels. The main technical insight of VGG was mathematical: a deep stack of two or three 3×3 convolutions has the same effective receptive field as a single 5×5 or 7×7 convolution. However, it includes more non-linear rectification layers and significantly reduces the total number of trainable parameters. Concurrently, GoogLeNet (Inception-v1) [19] addressed computational inefficiency with structural innovation. It introduced the Inception module, a parallel multiscale architecture that integrates 1×1, 3×3, and 5×5 convolutions along with max-pooling within a single layer. A key detail of this architecture was the use of 1×1 convolutions as dimensionality reduction bottlenecks prior to the more expensive larger convolutions, keeping the parameter budget manageable. GoogLeNet also replaced heavy fully connected layers with global average pooling to reduce overfitting and added auxiliary classifiers into intermediate layers to pass gradients effectively through the deep network.

While previous architectures focused on the depth of neural networks, scaling them revealed a degradation problem. As model depth increases, accuracy levels off and then declines quickly, which is different from the vanishing gradient issue. ResNet [20] addressed this by introducing the Residual Learning framework. Instead of relying on a stack of layers to fit a desired mapping $H(x)$, ResNet allows these layers to fit a residual mapping $F(x) = H(x) - x$. The formulation $y = F(x, \{W_i\}) + x$ is achieved using identity shortcut connections, which perform skip-layer summation. This ensures that the gradient can flow directly through the identity mapping. As a result, it reduces the optimization challenges of deep networks and allows the training of models with over 1,000 layers. For deeper versions (ResNet-50/101/152), the authors introduced *bottleneck* blocks. They used 1×1 convolutions to reduce and then restore dimensions, which significantly lowers computational costs without losing representational power. DenseNet [21] developed the skip-connection idea further by switching from summation to concatenation. In a Dense Block, each layer gets as input the feature maps from all the previous layers: $x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$. This setting wants to maximize information flow and encourages heavy feature reuse. It allows the model to achieve state-of-the-art results with significantly fewer parameters than ResNet. A key hyperparameter here is the *growth rate* (k), which determines how much new information each layer adds to the network's knowledge. To control memory and spatial dimensions, DenseNet uses Transition Layers that feature 1×1 convolutions and pooling between blocks to reduce the number of feature maps. Subsequent refinements of [20] sought to optimize the trade-off between width, depth, and cardinality. ResNeXt [22] introduced the concept of cardinality—the size of the set of transformations—as a more effective dimension than depth or width. It utilizes a *split-transform-merge* strategy through grouped convolutions, showing that increasing cardinality is a more efficient way to improve accuracy than simply deepening the network. In parallel, Wide ResNet [23] argued that the obsessive focus on depth led to diminishing returns and slow training. By increasing the number of channels (width) and decreasing depth, they achieved superior performance with better hardware utilization. Res2Net [24] further refined this by implementing hierarchical residual-like connections within a single residual block, enabling the extraction of multi-scale features at a granular level, which is particularly beneficial for object detection and segmentation tasks.

Parallel to the evolution of backbone topologies, substantial progress has been made in attention mechanisms designed to dynamically re-calibrate feature representations. Squeeze-and-Excitation (SE) networks [25] pioneered channel-wise attention by modeling inter-dependencies between channels. The "Squeeze" operation employs global average pooling to aggregate spatial information into a channel descriptor $\mathbf{z} \in \mathbb{R}^C$, where $z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j)$. This is followed by the "Excitation" operation, a gating mechanism using two fully connected (FC) layers with a bottleneck reduction ratio r and a sigmoid activation: $\mathbf{s} = \sigma(W_2 \delta(W_1 \mathbf{z}))$. The final output is obtained by rescaling the original feature maps by these adaptive

weights. While SE blocks significantly improve discriminative power with a marginal increase in FLOPS, they do introduce additional parameters through the FC layers, and the dimensionality reduction in the bottleneck can sometimes lead to a loss of channel-wise correlation complexity. To address these limitations, Efficient Channel Attention (ECA) [26] proposed a "local" cross-channel interaction strategy. Instead of global dimensionality reduction via FC layers, ECA captures local dependencies by performing a 1D convolution with kernel-size k across the channel dimension, where k is adaptively determined by the channel capacity C. This ensures that every channel interacts with its k neighbors, maintaining performance while drastically reducing the parameter count compared to SE blocks. Concurrently, the Convolutional Block Attention Module (CBAM) [27] argued that spatial attention is as critical as channel attention. CBAM applies a dual-attention approach: a channel module that leverages both average-pooling and max-pooling to gather richer statistics, followed by a spatial attention module. The spatial module compresses the channel dimension via pooling and applies a 7×7 convolution to generate a spatial mask, effectively telling the network *where* to look after the channel module has decided *what* to look for. This sequential refinement has proven particularly effective in dense prediction tasks like object detection.

A fundamental paradigm shift occurred with the adaptation of the Transformer architecture [28] to visual recognition, which systematically removed the architectural constraints of convolution. The Vision Transformer (ViT) [29] reinterprets an image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ as a sequence of $N = HW/P^2$ non-overlapping patches. These patches are flattened into vectors $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$ and mapped via a trainable linear projection $\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$ to a latent space. A unique aspect of ViT is the prepending of a learnable [class] token $\mathbf{x}_{class}$ to the sequence, such that the input to the encoder is $\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1\mathbf{E}; \ldots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}$, where $\mathbf{E}_{pos}$ represents 1D learnable positional encodings. The Multi-Head Self-Attention (MSA) mechanism then captures global dependencies by projecting the input into Queries (Q), Keys (K), and Values (V) across k heads, allowing the model to attend to disparate image regions simultaneously. To overcome the "data-hungry" nature of pure Transformers, DeiT [30] introduced a distillation token that interacts with the class token and patch tokens through the self-attention layers. This token is specifically tasked with mimicking the hard-label predictions of a teacher CNN, effectively injecting inductive biases back into the Transformer through supervision rather than architecture. Furthermore, Masked Autoencoders (MAE) [31] revolutionized self-supervised pre-training by employing an asymmetric encoder-decoder. By masking a high ratio (75%) of patches and only processing the visible ones through the encoder, MAE forces the model to learn holistic semantic structures to reconstruct pixels, proving that Transformers can learn highly transferable representations without manual labels. Despite their success, pure Transformers face significant challenges regarding computational scaling and lack of spatial priors. The MSA mechanism incurs a quadratic computational complexity $O(N^2 \cdot D)$ because every token must compute an attention score with every other token, rendering high-resolution processing inefficient. The Swin-Transformer [32] addressed this by introducing a hierarchical design and Window-based MSA (W-MSA). Unlike ViT, Swin partitions the image into local windows and restricts attention within them, reducing complexity to $O(N \cdot M^2)$, where M is the window size. To ensure information flow across these isolated windows, Swin-T employs a Shifted Window (SW-MSA) approach, where the window partition is displaced by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ in successive layers. This ensures that the boundaries of one layer's windows become the centers of the next. Additionally, Swin-T uses patch merging layers to reduce the number of tokens as the network goes deeper, mimicking the increasing receptive field of CNNs. In parallel, hybrid models like CvT [33] and CoaT [34] modernized the Transformer block by replacing the standard linear projections for Q, K, V with depthwise separable convolutions. This convolutional projection introduces translation equivariance and local context modeling directly into the attention mechanism, allowing for the removal of explicit positional encodings and achieving superior performance on tasks requiring high-resolution localization, such as segmentation and detection.

In summary, the historical trajectory of neural network evolution reflects a widening bifurcation between theoretical representational capacity and operational feasibility [35, 36]. Recent research has largely prioritized the development of high-capacity, over-parameterized models—such as the *Huge* variants of Vision Transformers and extensively deep residual frameworks—engineered specifically to saturate performance metrics on large-scale benchmarks [37, 38, 39] While these architectures are indispensable for probing the upper bounds of hierarchical feature extraction and optimization stability, they function primarily as theoretical probes. Their design often elides the physical constraints of deployment, resulting in models that, while achieving record-breaking accuracy on high-performance compute clusters, remain computationally

prohibitive for edge-tier silicon [36, 40]. This pursuit of state-of-the-art (SOTA) metrics typically ignores the non-linear relationship between marginal accuracy gains and the exponential increase in computational cost, leading to a state of architectural over-engineering relative to practical application needs [41, 35]. This emphasis on benchmark-centric excellence has inadvertently exacerbated a deployment gap, where the scaling of model complexity outpaces the hardware evolution of mobile and embedded systems [40, 42]. Paradoxically, the mechanisms that ensure global context and high precision—such as dense multi-head self-attention or multi-stage feature concatenation—introduce significant memory bandwidth bottlenecks and stochastic latency that are incompatible with real-time requirements, particularly on resource-constrained devices [42, 43]. Consequently, these models serve as an essential theoretical foundation, advancing our knowledge of how to construct high-quality vision representations, yet they lack the structural pragmatism required for field-ready systems. This disconnect underscores a critical need for a hardware-aware architectural paradigm: one where computational efficiency and inference-time constraints are treated as primary objective functions rather than post-hoc optimizations [44, 45]. This necessity drives the exploration of the specialized, ultra-lightweight frameworks discussed in the following section.

2.2. *Towards more efficient convolutional architectures*

In parallel with the development of increasingly deep and expressive neural networks, a robust and expanding line of research has been devoted to the design of architectures explicitly optimized for computational efficiency and reduced resource requirements, while attempting to maintain competitive levels of accuracy. The momentum for this field originates from the pressing need to enable deployment of deep learning models on power and memory constrained hardware platforms, such as smartphones, edge devices, and embedded systems.

Early landmark contributions, such as SqueezeNet [46], introduced the Fire module as a primary building block. This module consists of a *squeeze* layer—utilizing 1×1 convolutions with s_1 filters—followed by an *expand* layer that concatenates 1×1 and 3×3 convolutions with e_1 and e_3 filters, respectively. The technical innovation lies in the squeeze ratio $SR = s_1/(e_1 + e_3)$, which serves as a bottleneck to aggressively reduce input channel dimensionality. This configuration minimizes the total number of parameters, defined as $P = (C_{in} \cdot s_1) + (s_1 \cdot e_1) + (9 \cdot s_1 \cdot e_3)$, where the $9\times$ factor for 3×3 kernels is applied only to the reduced s_1 dimension. This enabled a $50\times$ reduction in parameters compared to AlexNet [17] while maintaining equivalent Top-1 accuracy. Building on these principles, ShuffleNet [47] addressed the high computational cost of 1×1 convolutions, which can account for over 90% of FLOPS in compact models. It introduced Pointwise Group Convolutions and a novel Channel Shuffle operation. Standard group convolutions suffer from a lack of cross-talk, where output channels only relate to a small fraction of input channels. ShuffleNet resolves this by reshaping the output of a group convolution from (g, n) to (n, g) and flattening it, effectively permuting the channels to ensure that the next layer receives information from all preceding groups. This design allows for a much wider channel count within a fixed computational budget, which is crucial for maintaining accuracy in extremely small networks.

The transition toward specialized mobile architectures was catalyzed by the adoption of Depthwise Separable Convolutions (DSC). While standard convolutions perform spatial filtering and channel-wise integration simultaneously, DSC—originally popularized as a structural alternative to Inception modules in Xception [48]—factorizes the operation into two distinct stages: a depthwise convolution (spatial filtering) and a pointwise 1×1 convolution (channel mixing). MobileNetV1 [49] standardized this as an efficiency primitive, reducing the computational overhead from $O(D_K^2 \cdot M \cdot N \cdot D_F^2)$ to $O((D_K^2 + N) \cdot M \cdot D_F^2)$, where D_K is the kernel size, M and N represent input/output channels, and D_F is the feature map resolution. For a standard 3×3 kernel, the reduction ratio is approximately $\frac{1}{N} + \frac{1}{D_K^2}$, yielding an $8\times$ to $9\times$ reduction in computational cost. To allow for flexible deployment, the authors introduced two global hyperparameters: the width multiplier α, which thins the number of channels at each layer, and the resolution multiplier ρ, which scales the input image. These parameters allow the computational cost to be tuned by approximately α^2 and ρ^2, providing a controlled mechanism for accuracy-latency trade-offs. MobileNetV2 [43] introduced Inverted Residuals and Linear Bottlenecks, representing a significant topological departure from the classical residual blocks used in ResNet. The design is predicated on the Manifold of Interest hypothesis: the assumption that the manifold of information can be embedded in a low-dimensional subspace. While standard residual blocks follow a *wide-narrow-wide* pattern (compressing dimensions to save compute), MobileNetV2 employs a narrow-wide-narrow structure. A

1×1 expansion layer first projects the low-dimensional input into a high-dimensional space (expansion factor $t = 6$), where the 3×3 depthwise convolution can extract features without collapsing the manifold. Crucially, the final 1×1 projection layer utilizes a linear activation. The authors demonstrated that non-linear transformations like ReLU perform well in high-dimensional spaces but tend to destroy information when applied to low-dimensional bottlenecks. This structural innovation allowed MobileNetV2 to achieve higher accuracy with fewer parameters by preserving feature diversity across the shortcut connections. MobileNetV3 [50] evolved this lineage by moving from heuristic design to automated discovery through Hardware-Aware Neural Architecture Search (NAS) combined with the NetAdapt [51] algorithm. This allowed for a heterogeneous architecture where each layer is individually optimized for the target hardware's latency profile. MobileNetV3 redesigned several expensive layers identified by the search: it reduced the number of filters in the initial layer and simplified the final stage by moving the feature-rich 1×1 convolution after the global average pooling layer, significantly reducing latency with no accuracy loss. Furthermore, it introduced the Hard-Swish activation function:

$$\text{h-swish}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6}$$

This serves as a piecewise linear approximation of the Swish nonlinearity ($x \cdot \text{sigmoid}(x)$), providing the same optimization benefits (smooth gradients and non-monotonicity) while avoiding the high transcendental computation cost of the sigmoid on mobile hardware. The integration of Squeeze-and-Excitation (SE) modules, optimized with a reduction ratio of 4, further enhanced the model's ability to perform cross-channel feature recalibration at a granular, per-block level.

Other structural innovations addressed specific operational bottlenecks. ESPNet [52] utilized the Efficient Spatial Pyramid (ESP) module to replace standard high-parameter convolutions. By decomposing a convolution into a pointwise stage and a spatial pyramid of dilated convolutions (dilation rates 2^k), the module captures multiscale context with a computational footprint similar to DSC. Similarly, GhostNet [53] targeted the empirical observation of feature map redundancy in deep networks. The Ghost Module generates a limited set of intrinsic feature maps and uses inexpensive linear transformations—specifically 3×3 depthwise convolutions—to produce ghost maps. This approach mathematically exploits the high correlation between feature channels to achieve a near 2× reduction in FLOPS while maintaining the same expressive capacity as standard MobileNet-style layers.

Automated architecture search has fundamentally shifted the focus from manual heuristic tuning to algorithmic optimization of network topologies. NASNet [54] pioneered this transition by defining a search space composed of *Normal* and *Reduction* cells. The controller, typically a Reinforcement Learning (RL) agent, identifies an optimal cell structure on a smaller dataset (e.g., CIFAR-10) which is then stacked to form a larger architecture. However, early NAS approaches were computationally prohibitive and relied on proxy metrics (like FLOPS) that often do not correlate with actual hardware latency. FBNet [55] and Proxyless-NAS [45] addressed this by employing Differentiable Architecture Search (DNAS). These frameworks utilize a gradient-based approach to navigate the search space and incorporate a latency-aware loss function: $L = CE(w, a) + \lambda \cdot \log(Latency(a))$, where $Latency(a)$ is estimated via a pre-built hardware lookup table. By performing the search directly on the target device and task, these methods eliminate the proxy gap, ensuring that the resulting architecture is optimized for specific NPU or GPU execution patterns. A landmark synthesis of automated search and principled scaling is represented by EfficientNet [44]. While previous methods scaled depth (d), width (w), or resolution (r) independently, the authors identified a fixed relationship between these dimensions. They introduced the Compound Scaling method, which uses a compound coefficient ϕ to scale all three dimensions uniformly: $d = \alpha^\phi$, $w = \beta^\phi$, and $r = \gamma^\phi$, subject to the constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$. This ensures that as the computational budget increases by 2^n, the resources are distributed optimally across the network's volume. Built upon a baseline architecture (MBConv) discovered via multi-objective NAS, EfficientNet-B0 to B7 demonstrated that this balanced scaling provides a superior Pareto frontier compared to any single-dimension scaling approach.

In addition to architectural design, model compression techniques—specifically pruning and quantization—have become indispensable for deploying high-performance models on edge hardware. Unstructured pruning [56] targets individual weights based on a magnitude threshold, achieving high sparsity ratios. However, because most off-the-shelf hardware (e.g., standard CPUs/GPUs) is not optimized for sparse matrix multiplication, this rarely translates into real-world speedups. In contrast, structured pruning [57, 58] removes entire filters or channels, yielding a direct reduction in the feature map dimensions and achieving predictable acceleration. To recover accuracy lost

during pruning, Knowledge Distillation (KD) is frequently employed, where a compact student model is trained to minimize the Kullback-Leibler (KL) divergence between its soft-target predictions and those of a high-capacity teacher model. This process allows the student to inherit the complex inter-class correlations learned by the teacher, mitigating the representational collapse that often follows aggressive compression. Complementary to pruning, quantization addresses the memory bandwidth bottleneck by reducing the bit-precision of weights and activations. Standard models utilize 32-bit floating point (FP32), which is memory-intensive. Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) map these values to 8-bit integers (INT8) or lower. Mathematically, this involves a scaling factor S and a zero-point Z: $q = \text{clip}(\text{round}(x/S + Z), q_{min}, q_{max})$. While quantization-aware training allows the network to adapt to the rounding errors introduced during inference, aggressive 4-bit or 1-bit (Binarized) quantization often leads to significant accuracy degradation in tasks requiring high-precision spatial localization, such as those encountered in aerial imagery.

Despite these considerable advances, several persistent limitations remain that hinder universal real-time deployment. First, the metric misalignment problem persists: many models optimized for low FLOPS exhibit high latency due to memory access patterns or poorly supported operations (e.g., depthwise convolutions on older hardware). Second, the NAS overhead remains significant; searching for an optimized model can require thousands of GPU hours, making it inaccessible for small-scale domain-specific tasks. Third, pruning and quantization often lead to a brittleness in generalization, where a compressed model performs well on the training distribution but fails under the atmospheric perturbations and viewpoint variations common in aerial scenes. These challenges necessitate a shift toward hardware-aware co-design, where the neural architecture and the underlying hardware accelerator are optimized in tandem to achieve a sustainable balance between inference speed, energy consumption, and predictive robustness.

2.3. Challenges in Aerial Image Recognition Tasks

High-altitude imagery, acquired from satellites, manned aircraft, or unmanned aerial vehicles (UAVs), constitutes a critical modality for large-scale Earth observation, environmental monitoring, and geospatial intelligence. However, such imagery presents unique and substantial challenges for automated image recognition that differ fundamentally from those encountered in conventional ground-level computer vision tasks.

A distinguishing characteristic of high-altitude remote sensing data is the frequent incorporation of multispectral and hyperspectral bands extending beyond the visible spectrum, including near-infrared (NIR), shortwave infrared (SWIR), and thermal infrared (TIR) wavelengths [59, 60]. These extended spectral channels encode information about vegetation health, soil composition, surface temperature, and material properties that are inaccessible to standard RGB sensors, yet their effective utilization requires dedicated feature extraction, spectral fusion, and domain-specific preprocessing strategies [61]. The acquisition of such data is typically contingent upon specialized and costly spaceborne or airborne sensors, resulting in infrequent temporal coverage and operational constraints. Satellite platforms, for instance, are constrained by low revisit frequencies—ranging from several hours to multiple days depending on orbital configuration and latitude—which limits their applicability for time-sensitive applications such as disaster response, agricultural monitoring, and real-time traffic analysis [60]. Conversely, UAV and aircraft-based platforms, while generally restricted to RGB or limited multispectral configurations, afford greater operational flexibility, lower latency, and the ability to perform targeted, high-resolution acquisitions over specific regions of interest [12, 13].

A central challenge in aerial image recognition stems from the inherently limited spatial resolution relative to the scale of objects of interest. In satellite imagery, ground sampling distances (GSDs) often range from sub-meter to tens of meters per pixel, causing small or densely distributed objects, such as individual vehicles, buildings, or agricultural plots, to occupy only a few pixels or appear as unresolved features [59]. This constraint requires architectures capable of extracting multiscale discriminative features, leveraging hierarchical representations, and employing spatial attention mechanisms or feature pyramid networks to preserve fine-grained spatial details across varying scales [61]. Additional complications arise from significant intra and inter-scene variability: objects exhibit wide variations in scale, orientation, and appearance due to differing altitudes, sensor perspectives, and imaging geometries. Atmospheric conditions—including haze, clouds, and aerosol scattering—further degrade image quality and introduce radiometric inconsistencies that complicate feature learning and model generalization [60, 61]. Moreover, illumination variations due to solar angle, seasonal changes, and shadowing effects contribute to temporal and spatial heterogeneity in the spectral signa-

tures of surface materials.

Beyond resolution and spectral complexity, the top-down viewing perspective introduces a further layer of difficulty that fundamentally distinguishes aerial imagery from standard ground-level benchmarks such as ImageNet [17] or CIFAR [62]. Objects observed from nadir or oblique angles present unfamiliar aspect ratios and silhouettes — a car viewed from directly above, for instance, loses the profile cues that ground-level detectors rely upon — while scene context, which is a powerful discriminative signal in terrestrial imagery, changes dramatically with altitude and geographic region. This perspective shift has been shown to significantly reduce the transferability of features learned on standard ground-level datasets [63], motivating the development of domain-specific training strategies and architectures.

The class distribution in aerial disaster datasets further exacerbates these challenges. Emergency events such as floods, fires, and collapsed structures are inherently rare relative to background "normal" scenes, resulting in severe class imbalance that can bias classifiers toward majority classes [64]. Moreover, intra-class visual variability is high: a flood may manifest as submerged urban streets, overflowing riverbanks, or coastal inundation, all sharing the same label but exhibiting very different spectral and spatial signatures. Conversely, inter-class similarity between, for example, fire smoke and industrial haze can introduce significant ambiguity. These properties demand models with strong discriminative capacity and robustness to distributional shift, which purely parameter-efficient architectures optimized for clean, balanced benchmarks may fail to provide.

These challenges impose stringent requirements on model design and training strategies. Algorithms must exhibit robustness to low spatial resolution and possess the capacity to capture hierarchical, multiscale representations to reliably detect and classify small, sparse, or partially occluded objects. Invariance or adaptability to scale, rotation, illumination, and atmospheric perturbations is essential to ensure generalization across diverse acquisition conditions and geographic regions. Furthermore, irregular and sparse temporal sampling inherent in high-altitude data streams, particularly from satellite sources, complicates tasks that require temporal consistency, such as change detection, land cover dynamics, and predictive modeling. Models must therefore be designed to handle incomplete temporal sequences, take advantage of temporal context when available, and maintain robust inference capabilities despite gaps or irregularities in observation cadence [61].

General-purpose architectures, despite their impressive performance on standard benchmarks, often fall short in this domain for several compounding reasons. First, models such as VGG [18] and ResNet [20], pre-trained on ImageNet, encode strong biases toward ground-level object appearances and textures that do not transfer effectively to the top-down, multi-scale nature of aerial scenes [63]. Second, their large parameter counts and high FLOPS make real-time deployment on resource-constrained UAV hardware infeasible without significant compression or approximation, which in turn risks accuracy degradation in safety-critical applications such as disaster response [64]. Third, Transformer-based architectures [29], while capable of modeling long-range spatial dependencies that are beneficial for wide-area scene understanding, require large-scale pre-training datasets and substantial memory to operate effectively, making them poorly suited for the data-scarce, latency-sensitive conditions characteristic of UAV deployment.

Collectively, these factors underscore the need for aerial-aware architectural design: models whose inductive biases, receptive field characteristics, and computational footprint are explicitly co-designed with the constraints of high-altitude imagery and embedded deployment in mind. This includes the adoption of multiscale feature extraction mechanisms — such as dilated convolutions [64] or feature pyramid structures — to simultaneously capture fine-grained local detail and broad spatial context, hardware-aware design choices that minimize latency and energy consumption without sacrificing discriminative power, and training strategies robust to class imbalance and domain shift. Without these considerations, even highly accurate general-purpose models risk becoming brittle, inefficient, or entirely impractical when deployed in the demanding real-world conditions of aerial disaster monitoring.

2.4. Methods for aerial image recognition

In response to the challenges outlined in the preceding section, a substantial line of research has emerged focusing on the development of algorithms and architectures tailored for aerial and UAV-based image recognition. These methods can be broadly categorized according to two orthogonal axes: their computational paradigm — offline versus real-time processing — and their architectural foundation — spanning traditional feature-based approaches, deep convolutional networks, attention-based models, and hybrid architectures explicitly optimized for embedded deployment.

Prior to the widespread adoption of deep learning, aerial image classification relied primarily on handcrafted feature engineering and classical machine learn-

ing pipelines, which, while effective in narrow domains, lacked the generalization capacity required for heterogeneous real-world aerial scenes. Early contributions to aerial image classification leveraged hybrid methodologies that combine spatial and spectral information. Yang *Et al.* [65] proposed a fusion-based classification technique that integrates multispectral characteristics and spatial context to achieve high precision in identifying damaged rice lodging areas from UAV-acquired imagery, demonstrating the effectiveness of domain-specific feature engineering for precision agriculture. Similarly, Mafanya *Et al.* [66] evaluated the performance of multiple image classifiers, including support vector machines, random forests, and maximum likelihood estimators, to map the invasive plant species *Harrisia pomanensis* using UAV imagery, concluding that ensemble methods and texture-based features significantly improve classification robustness in challenging ecological monitoring scenarios.

With the advent of deep learning, convolutional neural networks have become the dominant paradigm for aerial image analysis. Among recent architectures, APDC-Net, proposed by Bi *Et al.* [67], introduced an attention-based pooling mechanism combined with dense connection structures to preserve hierarchical multilevel features and enhance local semantic representation, achieving state-of-the-art accuracy on aerial scene classification benchmarks. To address the specific challenges of object detection in UAV imagery—such as small object size, high density, and arbitrary orientation, Hendria *Et al.* [68] proposed a combined framework integrating Transformer-based models [32] with convolutional detectors [69], employing ensemble and hybrid inference strategies to improve detection recall and precision on various object scales and environmental conditions.

Despite their impressive accuracy, the aforementioned methods are predominantly designed for offline processing on high-performance computing infrastructure, thus making them unsuitable for real-time deployment on resource-constrained UAV platforms. The latency and power consumption associated with deep architectures — such as VGG16 (14.8M parameters, 17.62G FLOPS) or ResNet50 (23.5M parameters, 4.83G FLOPS) — pose severe constraints for onboard embedded systems with limited computational, memory, and energy budgets, where models must typically operate within a power envelope of 4–15W while maintaining real-time throughput.

To bridge this gap, recent efforts have focused on developing lightweight architectures explicitly optimized for real-time inference on UAV hardware. Kyrkou *Et al.* introduced EmergencyNet [64], a compact convolutional model that employs atrous (dilated) convolutions to expand the receptive field without increasing the parameter count, followed by multiresolution feature fusion to integrate contextual information across scales. EmergencyNet was specifically designed for disaster response scenarios, achieving real-time performance on embedded GPUs while maintaining competitive accuracy for scene classification and anomaly detection in aerial imagery. Building upon this foundation, Mogaka *Et al.* proposed TinyEmergencyNet [70], which applies iterative magnitude-based pruning to remove up to 60% of the least significant weights, further reducing model size and inference latency with minimal accuracy degradation. This approach underscores the viability of structured pruning strategies for adapting existing architectures to embedded constraints.

Notwithstanding these advances, several critical gaps remain in the design of efficient neural architectures for embedded aerial image recognition. First, many existing lightweight models are optimized primarily for parameter reduction and FLOPS minimization, without explicit consideration of platform-specific latency characteristics or memory access patterns, which are often the dominant bottlenecks on embedded hardware. Second, fine-grained architectural components—such as activation functions, normalization layers, and pooling strategies, are frequently overlooked despite their substantial impact on inference latency and energy consumption, particularly on hardware accelerators with limited support for certain operations. Third, the majority of model compression techniques, including quantization and pruning, are applied post-training, potentially introducing accuracy degradation and necessitating retraining or fine-tuning, which may not be feasible in resource-limited scenarios.

Furthermore, contemporary neural network accelerators—such as edge Tensor Processing Units (TPUs), NPUs, and Digital Signal Processor (DSP)-based inference engines—offer highly optimized execution paths for specific operations (e.g., depthwise convolutions, low-bit integer arithmetic), yet these hardware capabilities are often underutilized due to a fundamental disconnect between architecture design and hardware-aware optimization objectives. Existing neural architecture search (NAS) frameworks, while effective in discovering efficient models, typically employ FLOPS or parameter count as proxy metrics, which may not accurately reflect real-world latency or energy consumption on target hardware [45, 44].

Building upon the core principles of the original TakuNetV1, which proved that the synergy of depth-

wise convolutions, early downsampling, dense connectivity, and hardware-aware design enables highly efficient real-time inference on embedded UAV platforms [11], this work presents an enhanced architecture, TakuNetV2. While TakuNetV1 demonstrated that minimizing parameter count and inference latency need not come at the expense of accuracy, TakuNetV2 systematically extends this foundation by introducing architectural refinements — namely a denser stem and hybrid depthwise feature extraction blocks — that push the accuracy–efficiency frontier further, particularly on larger and more complex datasets.

3. TakuNet

We present two versions of the same TakuNet family, namely TakuNetV1 [11] and TakuNetV2, both designed to achieve high classification accuracy while maintaining strict computational efficiency on resource-constrained embedded systems. To this end, both architectures integrate several well-established paradigms from the efficient neural network literature, including depthwise separable convolutions (*DWConv*) [49], pointwise convolutions (*Conv 1×1*), grouped pointwise convolutions (*GConv 1×1*) [71], residual and dense connections [20, 21], and channel shuffling [72]. This combination enables a favorable trade-off between representational power and computational cost, supporting real-time inference even on severely resource-constrained hardware.

While TakuNetV1 establishes an extremely compact and efficient baseline, its design deliberately sacrifices inter-channel feature interaction within its processing blocks in favor of minimal parameter count. TakuNetV2 is specifically designed to overcome this limitation, introducing targeted architectural changes that improve representational capacity — and consequently classification accuracy — with only a modest increase in computational overhead. In the following subsections, we detail the architecture of both versions and motivate each design choice with respect to its predecessor.

3.1. TakuNetV1 Architecture

The architecture of TakuNetV1 is illustrated in Figure 2 and comprises five macro-components: a stem block, four sequential processing stages each terminated by a Downsampler block, and a final linear classifier.

3.1.1. Stem.

The stem is responsible for the initial spatial compression of the input and consists of two stacked convolutional layers:

$$\begin{aligned} x_1 &= \mathrm{ReLU6}(\mathrm{BN}(\mathrm{Conv}_{3\times3}(x_0))), \\ x_2 &= \mathrm{ReLU6}(\mathrm{BN}(\mathrm{DWConv}_{3\times3}(x_1))), \end{aligned} \tag{1}$$

where ReLU6 denotes the Rectified Linear Unit clamped to $[0, 6]$ [49], BN denotes Batch Normalization [73], and DWConv refers to a depthwise convolution. The initial 3×3 full convolution uses a stride of 2, a padding of 2, a dilation of 2, and 40 output filters. A subsequent depthwise convolution with stride 2 and padding 2 further reduces the spatial resolution, yielding feature maps of size $\frac{H}{4} \times \frac{W}{4} \times 40$. This aggressive early downsampling substantially reduces the computational cost of all subsequent layers by shrinking the spatial dimensions on which they operate — a strategy also adopted in efficient architectures such as MobileNet [49] and EfficientNet [44].

3.1.2. Processing Stages and TakuBlockV1.

The four processing stages each consist of a stack of TakuBlocksV1 followed by a Downsampler. Each TakuBlockV1 is defined as:

$$x_1 = \mathrm{ReLU6}(\mathrm{BN}(\mathrm{DWConv}_{3\times3}(x_0))) + x_0, \tag{2}$$

comprising a residual depthwise convolution with stride 1, dilation 1, and padding 1. The four stages have depths of $(5, 5, 5, 4)$ TakuBlocksV1, respectively, with these values determined empirically to maximize the accuracy-efficiency trade-off. The exclusive use of depthwise convolutions within TakuBlockV1 keeps per-block parameter count and FLOPS extremely low, as each filter operates on a single channel independently [49]. However, as a direct consequence, *no inter-channel interaction occurs within the stages themselves*: the network's ability to mix information across feature channels is entirely delegated to the Downsampler blocks. This design decision is the primary limiting factor of TakuNetV1 in tasks that require richer feature representations, such as fine-grained discrimination among a larger number of classes.

3.1.3. Downsampler.

At the end of each stage, the Downsampler simultaneously reduces the spatial resolution of the feature maps and expands the channel dimensionality. Inspired by DenseNet [21] and ShuffleNet [47], the feature maps produced by the last TakuBlockV1 of the stage are concatenated with the stage's input features, forming a dense connection that encourages feature reuse across processing depth. A grouped pointwise convolution is then applied to the concatenated representation to

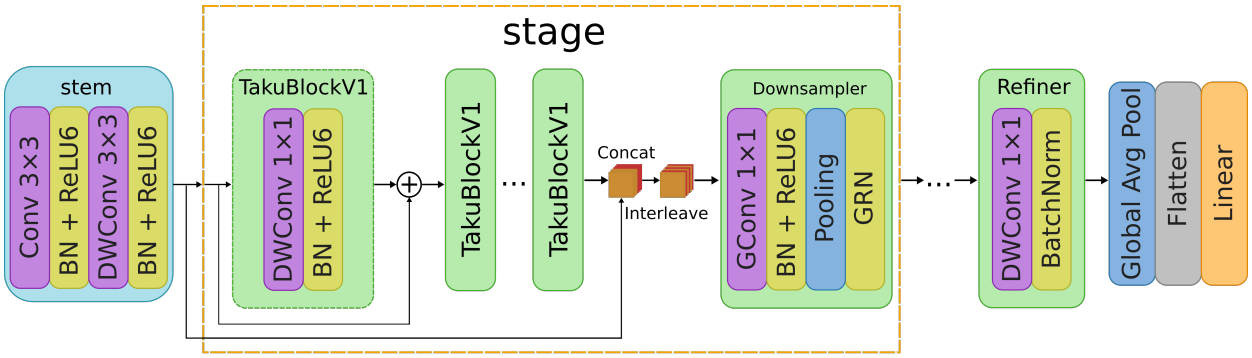

Figure 2: Overview of the TakuNetV1 architecture. The input image is first processed by a convolutional stem block. Four subsequent stages progressively extract spatial features, whose final output is later integrated along the channel axis with dense connection feature maps coming from the stage's input. At the end of each stage, the Downsampler block reduces the spatial size, while expanding the channel dimension. Finally, the Refiner block balances spatial features before the linear classification layer.

project it into the desired output dimensionality while limiting parameter growth; the number of groups is set to $\lfloor {}^{(\text{in\_channels}+\text{out\_channels})}/_{8} \rfloor$. This is followed by Batch Normalization and ReLU6. Spatial downsampling is achieved via max-pooling in the first three stages and average pooling in the last stage, each followed by a Global Response Normalization (GRN) layer [74], which suppresses redundant feature activations and improves generalization. The grouped pointwise convolution uses $\lfloor {}^{(\text{in\_channels}+\text{out\_channels})}/_{4} \rfloor$ groups, a ratio that prioritizes parameter reduction over channel mixing capacity. After spatial downsampling, a Global Response Normalization (GRN) layer [74] is applied unconditionally to the output of every Downsampler, suppressing redundant feature activations and sharpening inter-channel contrast. The Downsampler thus serves as the sole site of cross-channel interaction in TakuNetV1, a design bottleneck that TakuNetV2 systematically addresses.

3.2. TakuNetV2 Architecture

TakuNetV2 is motivated by a systematic analysis of the representational limitations of TakuNetV1. Two primary bottlenecks are identified: (i) the absence of inter-channel feature mixing within the processing blocks, and (ii) the limited expressiveness of the single-layer depthwise stem. Both issues are addressed through targeted architectural modifications, resulting in a model that achieves markedly higher classification accuracy — particularly in datasets with a larger number of semantic categories — at a carefully controlled increase in computational cost. The architecture of TakuNetV2 is depicted in Figure 3.

3.2.1. Dense convolution Stem.

In TakuNetV1, the stem performs one full convolution followed by a depthwise convolution. While this limits the parameter count, the depthwise layer processes each channel independently, precluding inter-channel interaction during the critical initial feature extraction stage. To address this, we redesign the stem of TakuNetV2 as two cascaded full convolutions:

$$\begin{aligned} x_1 &= \text{LReLU}(\text{BN}(\text{Conv}_{3\times3}(x_0))), \\ x_2 &= \text{LReLU}(\text{BN}(\text{Conv}_{3\times3}(x_1))), \end{aligned} \tag{3}$$

where LReLU denotes the LeakyReLU activation function (discussed below). The first convolution projects the three input RGB channels into an intermediate representation of half the stem's output channel count, while the second expands this to the full output channel dimensionality. Both convolutions use a stride of 2, achieving the same 4× spatial reduction as in TakuNetV1.

Crucially, replacing the second depthwise stem layer with a full convolution expands the channel context available to the first TakuBlockV2. This modification is directly complementary to the grouped (1×1) convolution used for channel compression at the entrance of the block. Although the grouped convolution mixes only a subset of its immediate input channels, the preceding full convolution allows each of these input channels to encode information from the complete intermediate stem representation.

To make this interaction explicit, let (u) denote the output of the first stem convolution, (s) the output of the second stem operation, and (q) the compressed representation produced by the grouped pointwise convolu-

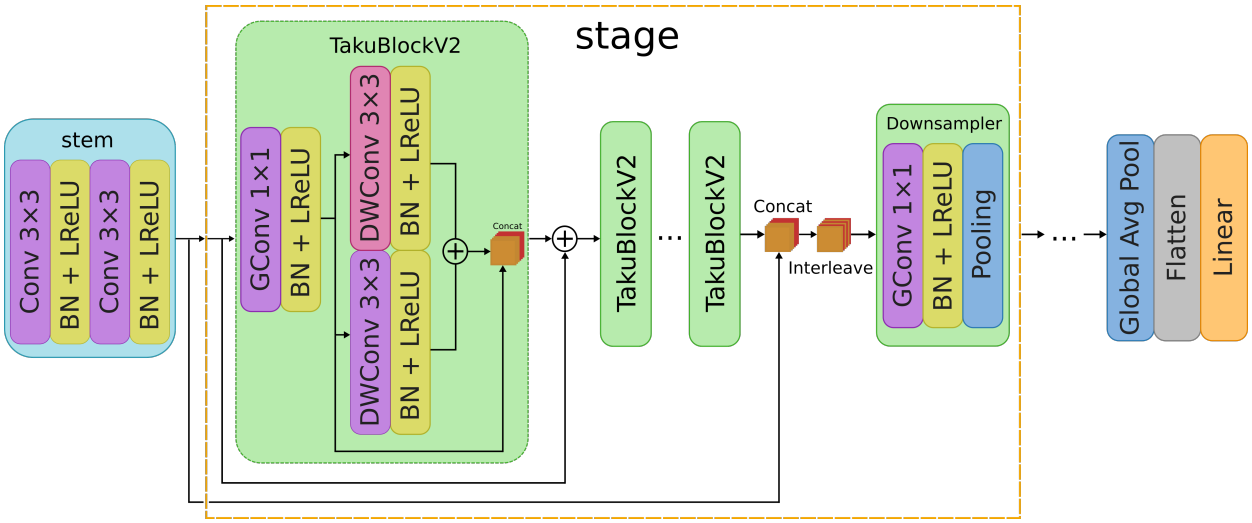

Figure 3: Overview of TakuNetV2's archtecture. Differently from the original TakuNetV1, the new version includes a stem composed of two densely connected convolutional layers, and introduces the updated TakuBlocksV2. These blocks are based on ghost feature generation through two depthwise convolutions, one of which is dilated (bordeaux color).

tion at the entrance of TakuBlockV2. Batch normalization and activation functions are omitted for clarity, as they do not alter the channel-dependency structure.

When TakuBlockV2 is preceded by the original depthwise stem operator, the second stem transformation can be written as

$$s_j^{\mathrm{DW}} = D_j * u_j, \tag{4}$$

where (D_j) denotes the depthwise kernel associated with channel (j). Since each output channel (s_j^{DW}) depends only on the corresponding input channel (u_j), the subsequent grouped pointwise convolution produces

$$q_r^{\mathrm{DW}} = \sum_{j \in \mathcal{G}_r} P_{rj} s_j^{\mathrm{DW}} = \sum_{j \in \mathcal{G}_r} P_{rj} \left(D_j * u_j \right), \tag{5}$$

where $(\mathcal{G}_r)$ denotes the group of input channels connected to output channel (r). Therefore, (q_r^{DW}) depends only on the subset of intermediate stem channels contained in $(\mathcal{G}_r)$. The channel context available to the grouped convolution remains limited by its group partition.

In TakuNetV2, the second stem operation is instead a full convolution:

$$s_j^{\mathrm{Full}} = \sum_{k=1}^{M} K_{jk} * u_k, \tag{6}$$

where (M) is the number of intermediate stem channels and (K_{jk}) denotes the convolution kernel connecting input channel (k) to output channel (j). Each channel (s_j^{Full}) can therefore combine information from all (M) intermediate stem channels. The grouped pointwise convolution in TakuBlockV2 then produces

$$\begin{aligned} q_r^{\mathrm{Full}} &= \sum_{j \in \mathcal{G}_r} P_{rj} s_j^{\mathrm{Full}} \\ &= \sum_{j \in \mathcal{G}_r} P_{rj} \left(\sum_{k=1}^{M} K_{jk} * u_k \right) \\ &= \sum_{k=1}^{M} \left(\sum_{j \in \mathcal{G}_r} P_{rj} K_{jk} \right) * u_k \end{aligned} \tag{7}$$

Consequently, although the pointwise convolution inside TakuBlockV2 remains grouped, each compressed output channel (q_r^{Full}) can depend indirectly on all intermediate stem channels. The revised stem therefore expands the effective channel-dependency set before the low-cost grouped compression stage: the dense stem performs global mixing along the channel dimension, while the grouped pointwise convolution retains its computational advantage by operating only within predefined groups. The ablation study in Table 7 confirms that this architectural modification independently improves the final classification accuracy.

We note that the introduction of a denser stem translates in higher FLOPS, considering that both full-convolutions process the highest-resolution feature

maps. However, given the small number of channels involved at this stage ($3 \rightarrow C/2 \rightarrow C$, where C is modest in our ultra-compact setting), the absolute cost remains low. Furthermore, this architectural design aligns with the prevailing trajectory of hardware development. Modern dedicated AI accelerators are increasingly optimized for high arithmetic intensity and dense matrix operations. In these settings, full convolutions often achieve superior hardware utilization compared to fragmented or sparse operations, such as depthwise convolutions, which frequently suffer from memory-bandwidth bottlenecks and lower execution efficiency on specialized silicon. Consequently, the transition to a denser stem represents a strategic trade-off that prioritizes real-world latency and throughput over theoretical FLOPS minimization.

3.2.2. *TakuBlockV2: Compressed Ghost Feature Generation with Multi-Scale Depthwise Fusion.*

The core innovation of TakuNetV2 is the replacement of TakuBlockV1 with the newly proposed TakuBlockV2, which directly addresses the channel-isolation limitation of its predecessor. TakuBlockV2 is structured around a *GhostModule* that generates enriched feature maps through three sequential operations: grouped channel compression, dual-scale depthwise spatial processing, and multi-scale feature fusion via summation and concatenation. The full block is defined as:

$$\begin{aligned} x_1 &= \mathrm{LReLU}(\mathrm{BN}(\mathrm{GConv}_{1\times1}(x_0))), \\ x_2 &= \mathrm{LReLU}\Big(\mathrm{BN}\Big(\mathrm{DWConv}_{3\times3}(x_1)\Big)\Big), \\ x_3 &= \mathrm{LReLU}\Big(\mathrm{BN}\Big(\mathrm{DWConv}^{(d=2)}_{3\times3}(x_1)\Big)\Big), \\ x_4 &= \mathrm{Concat}[x_1,\ x_2 + x_3] + x_0, \end{aligned} \tag{8}$$

where $\mathrm{GConv}_{1\times1}$ denotes a grouped pointwise convolution, $\mathrm{DWConv}^{(d=2)}_{3\times3}$ a dilated depthwise convolution with dilation rate $d = 2$, and Concat the channel-wise concatenation operator.

Grouped channel compression. The first operation is a grouped 1×1 convolution that maps the C input channels to $\lceil C/2 \rceil$ intermediate channels, where the number of groups is set to $C/4$. This simultaneously achieves two goals. First, it introduces explicit cross-channel interaction that was entirely absent from TakuBlockV1: features from different input channels are mixed within each group before spatial processing. Second, it halves the channel count at which all subsequent spatial operations are applied, so the depthwise convolutions in the next step operate on a compressed representation, directly reducing the computational cost of the spatial processing stage. This compression-before-spatial-processing strategy inverts the inverted-bottleneck paradigm of MobileNetV2/V3 [43, 50], which *expands* channels before depthwise operations, incurring higher intermediate memory and bandwidth costs. Here, compression reduces cost while the cross-channel mixing of the grouped convolution compensates for the reduction in redundancy.

Dual-scale depthwise spatial processing. The compressed features x_1 are simultaneously processed by two independent depthwise convolutions operating at different spatial scales. The standard 3×3 depthwise convolution captures high-frequency local structure — edges, textures, and fine object boundaries — within a 3×3 receptive field. The dilated 3×3 depthwise convolution with dilation $d = 2$ expands the effective receptive field to 5×5 pixels while using an identical number of parameters, since dilation modifies only the sampling stride and not the kernel size [75]. The two kernels $\mathbf{K}_s$ and $\mathbf{K}_d$ are learned *independently without parameter sharing*, allowing each to specialize:

$$x_2[h, w, c] = \sum_{i,j} \mathbf{K}_s[i, j, c] \cdot x_1[h + i,\ w + j,\ c], \tag{9}$$

$$x_3[h, w, c] = \sum_{i,j} \mathbf{K}_d[i, j, c] \cdot x_1[h + 2i,\ w + 2j,\ c]. \tag{10}$$

This dual-path design distinguishes TakuBlockV2 from all prior lightweight blocks. ShuffleNetV2 [72] employs a single 3×3 depthwise convolution per branch, with no multi-scale capability. GhostNet [53] augments primary features with a *single* cheap linear transformation at one scale. MobileNetV3 [50] applies one depthwise convolution per block, optionally augmented by squeeze-and-excitation attention, but without any spatial multi-scale processing within the block. In TakuNetV2, two spatial scales are computed simultaneously and fused, providing a representational richness that none of these prior designs offer within a single block.

Multi-scale fusion and restoration. The outputs of the two depthwise paths are fused by element-wise summation: x_2+x_3. This summation, rather than concatenation, is a deliberate design choice. Concatenation would double the channel count, requiring a subsequent projection and increasing cost. Summation instead produces a *multi-scale spatial response* in which each of the $\lceil C/2 \rceil$

channels encodes information from both the local 3×3 neighbourhood and the wider 5×5 context simultaneously, at no additional parameter cost. The fused representation is then concatenated with the channel-mixed features x_1 from grouped channel compression, restoring the full channel count C. A residual connection $+x_0$ spans the entire block, preserving gradient flow [20] and ensuring that the original input features are never discarded.

The resulting output $x_4 \in \mathbb{R}^{H \times W \times C}$ therefore integrates three sources of information: (i) cross-channel relationships learned by the grouped convolution; (ii) fine-grained local spatial structure from the standard depthwise path; and (iii) broader contextual structure from the dilated path. This combination is particularly suited to aerial disaster imagery, where semantically relevant patterns span vastly different spatial scales: local features such as the structural edges of a collapsed building and wider contextual patterns such as the spatial extent of a flood or fire zone are simultaneously necessary for accurate scene classification [64].

When comparing parameter cost, a TakuBlockV1 with C channels contains $C \cdot 9$ parameters (one 3×3 depthwise kernel). A TakuBlockV2 with the same C channels contains $\frac{C}{4} \cdot \lceil \frac{C}{2} \rceil$ parameters for the grouped conv, plus $\lceil \frac{C}{2} \rceil \cdot 9$ for each of the two depthwise kernels — a total of roughly $\frac{C^2}{8} + 9C$ parameters. For the channel widths used in TakuNetV2 ($C \in \{40, 80, 160\}$), this represents a moderate per-block overhead that is offset by the reduction in stage depth (from $(5, 5, 5, 4)$ to $(3, 3, 5, 4)$ blocks), yielding the final model with only ~8K additional parameters over TakuNetV1.

3.2.3. Revised Downsampler with Interleaved Dense Connection.

The Downsampler in TakuNetV2 is revised along four dimensions relative to TakuNetV1. At the end of each stage, the Downsampler receives two inputs: the output of the last TakuBlockV2 in the stage ($x_{\text{out}} \in \mathbb{R}^{H \times W \times C_{\text{in}}}$) and the stage's input features ($x_{\text{in}} \in \mathbb{R}^{H \times W \times C_{\text{dense}}}$), forming a dense skip connection inspired by DenseNet [21].

Interleaved channel fusion. The two feature tensors are not simply concatenated along the channel dimension. Instead, they are interleaved at the sub-channel level: x_{out} is reshaped to expose groups of $C_{\text{in}}/C_{\text{dense}}$ channels per spatial position, and x_{in} is inserted between these groups before the tensor is flattened back to $\mathbb{R}^{H \times W \times (C_{\text{in}} + C_{\text{dense}})}$. This channel interleaving ensures that the subsequent grouped pointwise convolution acts on spatially interleaved features from both the beginning and end of the stage, improving local mixing across the dense connection — an operation conceptually related to the channel shuffle of ShuffleNetV2 [72] but applied to the dense connection rather than within a single branch.

Relaxed grouping. The grouped pointwise convolution that projects the combined representation to C_{out} channels uses $\lfloor (C_{\text{in}} + C_{\text{dense}})/8 \rfloor$ groups, relaxed from the $\lfloor (C_{\text{in}} + C_{\text{dense}})/4 \rfloor$ used in TakuNetV1. Fewer groups mean more cross-channel interaction within the Downsampler projection, consistent with the overall design philosophy of TakuNetV2.

GRN removal. The Global Response Normalization (GRN) layer applied unconditionally after each Downsampler in TakuNetV1 is removed in TakuNetV2. The ablation study (Table 7, row 2) shows that GRN contributes positively in TakuNetV1, where it compensates for the limited feature diversity of purely depthwise TakuBlocksV1 by sharpening inter-channel contrast. In TakuNetV2, the richer multi-scale representations produced by TakuBlocksV2 already exhibit sufficient diversity, rendering GRN redundant; its removal frees parameters reallocated to the block design.

Adaptive pooling in the final stage. The last Downsampler uses AdaptiveAvgPool2d rather than fixed-kernel average pooling, collapsing the spatial dimensions to 1×1 regardless of input resolution. This allows TakuNetV2 to process images of varying resolutions without architectural modification — a practically important property across datasets where input sizes differ (cf. Tables 4–6, where input sizes range from 224 to 256 pixels).

3.2.4. LeakyReLU Activation.

Both the stem and TakuBlockV2 adopt LeakyReLU in place of the ReLU6 used in TakuNetV1. The hard clamping of ReLU6 can cause dead neurons when activations are driven into the saturated region, impairing gradient flow and limiting the effective learning capacity of deep networks — an effect that is exacerbated in the compact models considered here, where each neuron contributes proportionally more to the overall representation. LeakyReLU mitigates this by preserving a small, non-zero gradient for negative inputs [76], enabling stable backpropagation without the representational cost of clipping. Although more expressive activations such as GeLU [77] are available, they incur a non-trivial computational overhead on embedded hardware. As shown in Table 11, LeakyReLU achieves a favorable trade-off

between non-linearity and inference efficiency, making it the preferred choice for our deployment target.

3.2.5. Stage Configuration and Classifier Head.

TakuNetV2 comprises four stages with depths of $[3, 3, 5, 4]$ TakuBlockV2 units, respectively. The reduced depth in the early stages ($[3, 3, 5, 4]$ vs. $[5, 5, 5, 4]$ in TakuNetV1) reflects a key consequence of the richer per-block representation. In TakuNetV1, early stages require greater depth to progressively accumulate representational capacity through repeated application of channel-isolated depthwise convolutions. In TakuNetV2, each TakuBlockV2 already performs cross-channel mixing via its grouped pointwise convolution, meaning that fewer sequential blocks are needed to reach an equivalent level of feature expressiveness. Empirical evaluation confirmed that reducing the depth of the two early stages from 5 to 3 blocks does not degrade accuracy, while meaningfully reducing FLOPS in the higher-resolution early feature maps where the computational cost per block is greatest. The Refiner module present in TakuNetV1 has been removed; the parameters freed by this removal are reallocated to the richer TakuBlockV2 units, where they contribute more directly to feature quality. Accordingly, the network concludes with a Global Average Pooling layer, a flatten operation, and a linear classifier — a standard and computationally efficient classification head widely adopted in compact architectures [49, 44].

3.3. Summary of Contributions.

Table 7 provides a comprehensive ablation over the differences between TakuNetV1 and TakuNetV2, validating each design choice individually. The progression from row 1 to row 7 demonstrates that the accuracy improvements are cumulative, with the introduction of TakuBlocksV2 (row 5) and the new stem (row 4) accounting for the largest individual gains. The final model, TakuNetV2, achieves state-of-the-art F1-score among all micro-scale models on the AIDERv2 and CLRS benchmarks (Tables 5 and 6), with an absolute F1 improvement of up to +3.3 percentage points over TakuNetV1 on the more challenging CLRS dataset, at a parameter overhead of only ∼8K additional weights (50K vs. 42K) — a gain-to-cost ratio that compares favorably with all competing architectures.

4. The Astrial platform

As above mentioned, the progressive miniaturization and increasing computational capacity of embedded systems have significantly transformed a lot of applications, including the capabilities of unmanned aerial vehicles. In particular, in this context, the integration of single board computers (SBCs) enable the adoption of computer vision for autonomous behaviors such as visual navigation, object detection, and real-time decision-making, even in GPS-denied environments. These enhancements, however, come at the cost of increased power consumption, thermal management challenges, and more stringent security requirements, especially when deployed in compact airborne platforms where volume, mass, and energy are tightly constrained.

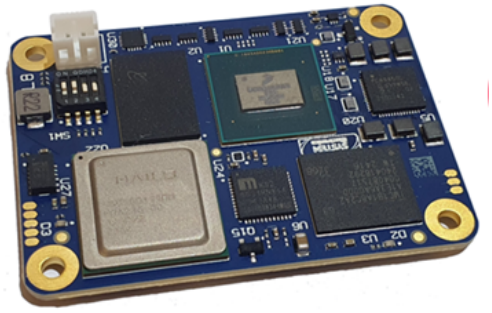

Figure 4: The Astrial single board computer by System Electronics

Astrial[2] is a state-of-the-art, AI-driven embedded platform (see Fig. 4) developed within System Electronics[3]. Astrial is designed to streamline processes in a variety of applications and not only on unmanned vehicles. Industrial automation and smart manufacturing are some examples that can leverage the common interfaces of Astrial (see Fig. 5).

In particular, Astrial provides a dual camera input (MIPI-CSI2 standard) supporting two streams at 1920x1080@30fps, in addition to HDMI and MIPI-DSI for monitors and the standard set of I/O busses for peripherals control. Thus, computer vision algorithms requiring dual-camera setups are supported, such as stereo vision or monocular systems with wide field-of-view optics to allow depth estimation, feature tracking, and simultaneous localization and mapping (SLAM). These systems are particularly valuable in environments where GNSS signals are degraded or unavailable. To support such workloads, Astrial allows to maintain low-latency video pipelines (below 100 ms), with direct memory access (DMA) paths between camera interfaces and the memory hierarchy to prevent bottlenecks.

The software OS is provided as a YOCTO repository[4]. The small form factor is compatible with the

[2] http://www.astrial.ai

[3] https://www.systemelectronics.com

[4] https://github.com/System-Electronics/astrial-howto

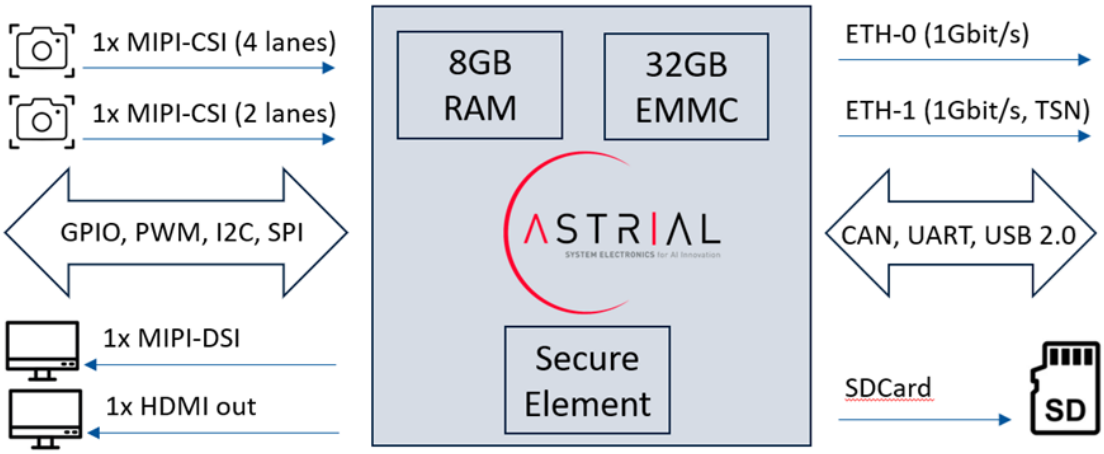

Figure 5: Diagram with the main elements and interfaces of the Astrial board.

specification of a RaspberryPi ComputeModule-4[5]. Beside these common specifications, Astrial integrates two key elements for such a small form factor design (see Fig. 6). The main CPU (*i.e.*, NXP i.mx8m-plus) is connected via PCIe bus to the neural network accelerator HAILO-8 that is capable of 26 TOPS (integer 8 bits) in terms of multiply-and-add operations and an on-chip memory of 32Mbytes. Furthermore, a Secure Element (i.e., NXP SE050) is made available for enhanced security integration, where its inclusion in UAV SBCs would provide a trusted anchor for system integrity. Secure Elements manage cryptographic keys within an isolated environment, preventing exposure to the main processing unit or system memory. These chips perform digital signatures, encryption, decryption, and certificate management internally, thereby reducing vulnerability to software-based attacks.

In hostile environments, drones may be captured or damaged. Secure elements provide tamper-evident mechanisms, such as voltage and temperature monitoring, secure erase on intrusion detection, and non-repudiable event logs. These features are vital in defense, surveillance, and industrial applications where data confidentiality is paramount.

The integration of SBCs with heat-dissipating elements introduces mechanical and dynamic challenges. The placement and mass of the heat sinks can alter the moment of inertia of the UAV. Large or asymmetrically placed components can introduce unwanted torques during rotation, impairing agility and control

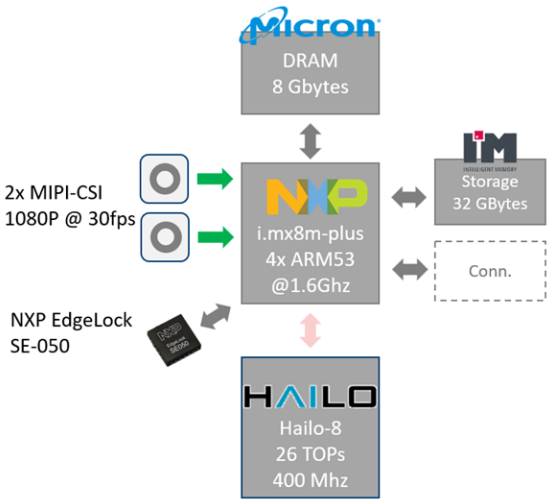

Figure 6: Astrial co-processors

authority. This effect is particularly pronounced during fast yaw or pitch maneuvers, where moment of inertia and dynamic forces may affect the overall performance of the flight.

Exposed heat sinks or cooling ducts may increase the cross-sectional area, thus introducing additional drag. In fixed-wing UAVs, this may reduce airspeed and endurance; in rotary-wing systems, it may influence lift asymmetry during lateral translation. To this aim, Astrial is an SBC qualified for extended range temperature (-40C to 85C) and it is provided with a custom heatsink tested for full system utilization. The aluminum heatsink has a dimension of 50x40x20mm, weighting 64 grams.

[5] https://www.raspberrypi.com/products/compute-module-4/?variant=raspberry-pi-cm4001000

Component	Specification
CPU	NXP i.MX8-m-plus (4 × Cortex-A53 @ 1.6GHz + 1 × Cortex-M7 @ 800MHz)
NPU	internal: Verisilicon VIP8000 (2.3 TOPS), external: HAILO-8 (26 TOPS, 32MB on-chip memory)
RAM	8GB, 32-bit LPDDR4
Storage	32GB eMMC
Video Decode	HW Decode: 1080p60, H.265/4, VP9, VP8
Video Encode	HW Encode: 1080p60, H.265/4
ISP	Dual 1080p@30 ISP (12MP, 375Mpix/s)
GPU	16 GFLOPS (high-precision), OpenGL ES 3.1/3.0, Vulkan, OpenCL 1.2 FP, OpenVG 1.1
Secure Element	NXP SE050 (FIPS 140-2 L3)
Thermal	-40°C to +85°C (entire SBC)
OS	YOCTO: Kirkstone LTS / Scarthgap LTS
Power	Min 4W, Max 8W (depending on workload), standby / powersave mode available
Form Factor	RaspberryPi Compute Module 4 (CM4)
Size	55 × 40 mm

Table 1: Astrial Hardware Specifications

A summary of the Astrial features are reported in Table 1.

5. Experiments

5.1. Aerial Image Disaster Event Recognition (AIDER) dataset

The AIDER [12] dataset is a collection of images acquired by different aerial vehicles and coming from various digital repositories, such as the world-wide-web (YouTube, newspaper web pages, the image indexing sections provided by major search engines, etc.) and via real-world drone acquisitions. The dataset consists of a set of multi-resolution images of complex scenes composed of natural territories, urban environments, large road infrastructure, rural or coastal areas, and so on. It includes scenes acquired in different seasons and at distinct times of the day, trying to offer a truthful overview of the real world, despite its small size. The hand-collected images contain scenes of emergency situations such as floods, collapsed buildings, traffic accidents, and fires. The first version of the dataset features 320 images for fires/smoke, 370 images for floods, 320 images for collapsed buildings/rubble, 335 images for traffic accidents, and 1200 images for the normal class, *i.e.* aerial images free of the various types of disasters. The dataset has a first limited version in size and was later expanded in [64]. Unfortunately, this latest version has been reduced due to copyright infringement, and a rather high number of images have been removed. The image composition of the AIDER dataset is detailed in Table 2, which enumerates the total number of images available along with a breakdown of the distribution between individual classes. For a fair comparison, we split the dataset into training and testing sets, using a proportion of 70/30 for all classes except Normal, split with a proportion of 65/35, keeping the exact test proportions as [70] and [64].

Due to the scarcity of data, we did not leave room for a standalone validation set, which we replaced with a k-fold cross-validation method performed on the training set, with $k = 5$.

Class	Train	Test	Total
Collapsed Building	365	146	511
Fire/Smoke	373	148	521
Flood	376	150	526
Traffic Accidents	346	139	485
Normal	2,850	1,540	4,390
Total Per Set	**4,310**	**2,123**	**6,433**

Table 2: Class distribution across training and testing sets of AIDER dataset [12] (latest available version).

5.2. Aerial Image Disaster Event Recognition Version 2 (AIDERv2) Dataset

Similarly to AIDER, AIDERv2 [13] is a collection of aerial images representing multiresolution emergency situations. The dataset collects urban, suburban, rural, coastal or lagoon scenes, and natural parks, taken at different times and seasons. Unlike the previous, the dataset is fully available and official training, validation, and test sets are provided, where the split ratio is set to 80/10/10, respectively, as expressed in Table 3.

5.3. Continual Learning Benchmark for Remote Sensing Image Scene Classification (CLRS)

To further validate the quality of our model and deepen the comparison with its competitors, we in-

Figure 7: The variety of some of the images in the AIDER [12] dataset grouped by each class. In detail, from the left, collapsed buildings, fire/smoke, floods, traffic incidents, and normal classes are shown.

corporated an evaluation on the CLRS dataset [14]. This dataset comprises 15k remote sensing images evenly distributed across 25 classes – airport, bareland, beach, bridge, commercial, desert, farmland, forest, golf-course, highway, industrial, meadow, mountain, overpass, park, parking, playground, port, railway, railway-station, residential, river, runway, stadium, and storage-tank. Each class thus contains 600 samples, each with a base resolution of 256×256 pixels. The spatial resolution of the images ranges from 0.26m to 8.85m. Although originally published for continual learning research, the CLRS dataset's characteristics render it an excellent benchmark for models trained to recognize high altitude imagery. We partitioned the dataset into training, validation, and test subsets following an 70:10:20 ratio, yielding 10.5k training, 1.5k validation and 3.0k test samples.

Class	**Train**	**Validation**	**Test**	**Total**
Earthquakes	1,927	239	239	2,405
Flood	4,063	505	502	5,070
Fire	3,509	436	436	4,384
Normal	3,900	487	477	4,864
Total Per Set	**13,399**	**1,670**	**1,654**	**16,723**

Table 3: AIDERv2 [13]: class distribution across training, validation, and testing sets.

5.4. Training Settings

In the initial stage of the training pipeline, image augmentation is employed as a pre-processing step to promote pattern diversity and increase variance within the limited datasets. The augmentation techniques, consistent with those described in [64], include color shifting, blurring, geometric transformations (translations, rotations, mirroring), random cropping, sharpening, shadowing, illumination changes, and zooming. Each transformation is applied with a randomly selected probability ranging from 0.05 to 0.5, thereby preventing the model from learning spurious variations as inherent characteristics of the dataset. In addition, all images are normalized and resized to a fixed resolution specific to each dataset: 240×240 pixels for AIDER, 224×224 pixels for AIDERV2, and 256×256 pixels for CLRS. This choice ensures consistency with the original papers and enables the evaluation of model performance across varying input resolutions.

All the training phases are conducted on a system equipped with NVIDIA RTX 4070 Ti Super 16GB and Intel i7-12700F. TakuNetV1 as well as TakuNetV2 are trained using the RMSProp [78] optimizer with decay 0.9, momentum 0.9, and an initial learning rate of 1×10^{-3}, along with a L_2-regularization term of 1×10^{-5}. Both TakuNetV1 and TakuNetV2 are trained using PyTorch's Automatic Mixed Precision (AMP) framework, with `float16` and `bfloat16` as the respective compute dtypes. Under AMP, forward and backward passes are executed in the reduced-precision dtype while the optimizer retains a full-precision FP32 master copy of the parameters throughout training; the exported model checkpoint therefore contains FP32 weights in both cases, independently of the compute dtype chosen. The choice of bf16 for TakuNetV2 is motivated by its wider dynamic range: bf16 allocates eight exponent bits (identical to FP32) against the five exponent bits of fp16, eliminating the risk of gradient underflow that can arise in architectures with denser stems and parallel depthwise branches without requiring explicit loss scaling. The shallower stem and single-branch depthwise operators of TakuNetV1 do not exhibit this sensitivity, making fp16 sufficient for that model. Deployment precision is determined independently by the target runtime: TensorRT applies FP16 on platforms without native bf16 support (e.g., Jetson Nano) and exploits native bf16 on Ampere-class devices and ARMv8.6-A cores, with no modification to the exported FP32 graph required in either case. Training is performed over 300 epochs with a batch size of 64, using a step learning

rate scheduler to adjust the learning rate at each epoch, as defined by the following formula:

$$\eta_t = \eta_0 \times \gamma^{\lfloor t/\text{step\_size} \rfloor} \tag{11}$$

where η_t represents the learning rate at epoch t, γ is the multiplicative factor for the learning rate, and $step\_size$ is the period of learning rate decay. We use a γ value of 0.975 with a $step\_size$ of 2. Finally, the random seed is set to 22 and a cross-entropy loss function is employed.

5.5. Evaluation and Performance Metrics

Following previous work in the literature [64, 70], we validate the accuracy of the proposed method using the F1-score metric, which is particularly effective for assessing the classification capability of the model on even unbalanced datasets. We also include parameters that are essential to assess how well a model is compatible with deployment on an embedded device. In particular, we report the number of model parameters and the amount of memory used by the model during inference. As a measure of computational complexity, we use FLOPS, representing the number of floating-point operations a system can perform per second. Although widely adopted, FLOPS have been criticized in the literature [79, 55] as often being limited to accurately reflect model performance on real-world platforms. Furthermore, the efficiency of neural network components can vary considerably depending on the computational characteristics of the underlying hardware, particularly the CPU. To overcome these limitations, we complement FLOPS with practical performance evaluation by reporting the framerate, in terms of frames per second (fps), achieved on real-world embedded devices. For each board we also report the Thermal Design Power (TDP), *i.e.* the maximum amount of heat a processor or other component is designed to dissipate under normal operating conditions. This empirical assessment captures model latency more explicitly, providing a clearer indication of its suitability for real-world deployment.

5.6. Comparison with the literature

We perform a comparative analysis against several well-established computer vision models. For the sake of understanding, in the tables we categorize these architectures into two families.

The first family comprises widely recognized models, such as VGG [18] and ResNet [20], mainly developed to improve performance in general computer vision tasks. These models have been developed without regard for their potential use on embedded boards: indeed, they have Giga-FLOPS values and therefore require powerful GPUs to be deployed. We also include additional models, such as MobileNet [49] and SqueezeNet [46], that have been tailored for a single specific embedded hardware accelerator and then they have not been directly optimized for efficiency or speed in multiple embedded settings. As previously discussed (see Sect. 2), they achieve enhanced performance at the expense of increased complexity and greater hardware requirements. Both of these architectures are presented in the upper sections of the tables to serve as performance references on the considered datasets.

The second family includes architectures explicitly designed for both embedded environments and aerial image recognition, such as EmergencyNet [64], TinyEmergencyNet [70] and TakuNet [11]. These models are listed in the lower sections of the tables, providing direct benchmarks to evaluate the performance of TakuNetV1 and TakuNetV2 on the respective datasets. EmergencyNet and TinyEmergencyNet are implemented using the PyTorch[6] framework, following the instructions detailed in the respective articles [64, 70]. During this process, we observe minor discrepancies in the publicly available implementation of EmergencyNet, while no official implementation of TinyEmergencyNet is found online. Other architectures included in the comparison are sourced directly from TorchVision [7] and Timm [8] libraries, ensuring consistency and reproducibility in the benchmarking.

For the competing architectures, we retain the optimization hyperparameters and learning-rate policies reported in the respective original publications. We deliberately did not train the competing architectures using the optimizer and step-decay schedule adopted for TakuNetV2. In general, applying a common training recipe would not necessarily yield a fairer comparison, since optimization hyperparameters can be architecture-dependent (*i.e.*if obtained through grid-search algorithms or similar techniques). If the resulting performance were lower than that obtained with the original training policies, the comparison would unfairly penalize the competing models. Conversely, if the modified recipe improved their performance, the resulting optimization strategy would constitute an additional contribution beyond the scope of this work. Therefore, each competing model was evaluated using the training policy reported in its original publication.

[6] https://pytorch.org
[7] https://pytorch.org/vision/stable
[8] https://huggingface.co/timm

6. Results

In this section, we evaluate TakuNetV1 and TakuNetV2 on three aerial image recognition benchmarks of increasing complexity and conduct a detailed ablation to isolate the contribution of each architectural component. Before presenting individual results, we state the central empirical thesis that the full set of results collectively supports: *as dataset complexity increases — measured by the number of semantic categories and degree of intra-class variability — the accuracy advantage of TakuNetV2 over TakuNetV1 grows monotonically, while both models remain strictly Pareto-optimal among all compared architectures in the accuracy-versus-parameter-count space*. On AIDER (4 classes, small dataset), TakuNetV1 achieves the highest F1-score among micro-scale models; on AIDERv2 (17 classes, large dataset), TakuNetV2 overtakes TakuNetV1 and reaches the top overall score; on CLRS (25 classes, balanced), TakuNetV2 achieves state-of-the-art accuracy among all models regardless of scale. This progression directly validates the architectural motivation of TakuNetV2: its richer intra-block feature representation is most beneficial precisely where fine-grained discrimination among many similar categories is required. On hardware, a complementary pattern emerges: TakuNetV1 dominates on efficiency-first hardware where operation-count minimality is paramount (NPU, legacy CPU), while TakuNetV2 demonstrates superior throughput-per-watt on modern hardware with high arithmetic intensity (Raspberry Pi 4/5, Jetson platforms), reflecting the alignment of its denser computational blocks with the parallel execution capabilities of contemporary processors.

The evaluation on AIDER [12] provides a test of model generalization under severe data scarcity, given its limited size and four-class structure. As shown in Table 4, TakuNetV1$_{\text{FP=16}}$ achieves the highest F1-score (0.943) among all micro-scale models, outperforming EmergencyNet (0.936) while using fewer than half its parameters (37.7K vs. 90.9K) and lower FLOPS (35.93M vs. 77.34M). Both TakuNet variants establish a new Pareto frontier in the accuracy-efficiency space, with TakuNetV1 achieving the best accuracy at the smallest model size among all compared architectures.

TakuNetV2 attains an F1-score of 0.936, matching EmergencyNet while requiring half its parameters and two-thirds its FLOPS. The marginally lower score relative to TakuNetV1 (-0.007) is consistent with a well-understood phenomenon in compact model design: on low-complexity tasks with few classes and limited training data, architectures with higher representational capacity may not fully exploit their added expressiveness, and the implicit regularization of a simpler model can provide a marginal benefit [44]. AIDER's four-class structure provides insufficient discriminative challenge to reveal the advantages of TakuBlockV2's multi-scale channel-aware processing. This interpretation is directly confirmed by the progression across datasets: the performance gap between V2 and V1 reverses sign and grows as class count increases (-0.007 on AIDER $\rightarrow$ $+0.002$ on AIDERv2 $\rightarrow$ $+0.033$ on CLRS). Further architectural or training optimization on AIDER is not pursued, as the observed performance gap is marginal and the dataset limited. We believe that additional tuning would likely overfit to this four-class benchmark, which is not representative of the intended deployment regime of TakuNetV2. In practice, TakuNetV1 is preferable for simpler classification tasks, while TakuNetV2 is better suited to scenarios with higher semantic complexity.

The evaluation on AIDERv2 [13], a substantially larger yet class-imbalanced dataset, reveals pronounced gains for TakuNetV2. As shown in Table 5, TakuNetV2$_{\text{FP=16}}$ attains the highest F1-score (0.959), surpassing both the original TakuNetV1$_{\text{FP=32}}$ (0.957) and EmergencyNet (0.955), while maintaining a low parameter count (45.7k) and moderate computational cost (45.2M FLOPS). Compared to TakuNetV1$_{\text{FP=16}}$ (0.948), V2 achieves an 1.1 pp improvement, demonstrating that its enhanced stem and dual-branch TakuBlockV2 effectively leverage the increased data volume to extract finer-grained and more specialized features.

Moreover, TakuNetV2's superior performance relative to other ultra-light- weight models such as TinyEmergencyNet (0.882) and MobileNetV3 (0.919), confirms that its architectural refinements scale favorably with dataset size, improving robustness to class imbalance and intra-class variability. These results underscore that, as the amount of training data grows, the benefits of a denser stem and advanced feature fusion within TakuNetV2 become more pronounced, validating its design as an optimal solution for real-time embedded aerial image recognition.

The evaluation on the CLRS dataset [14], summarized in Table 6, provides a rigorous test of model expressivity and generalization, given its 25 semantically similar classes and balanced sample distribution. In this challenging setting, TakuNetV2$_{\text{FP=16}}$ achieves an F1-score of 0.838, outperforming not only the original TakuNetV1$_{\text{FP=16}}$ (0.805) and TakuNetV1$_{\text{FP=32}}$ (0.791), but also all other ultra-lightweight competitors, includ-

Model	F1-score	Parameters	Memory	FLOPS
VGG16 [18]	0.601	14,840,133	59.36	17.62G
ResNet50 [20]	0.917	23,518,277	94.07	4.83G
SqueezeNet [46]	0.890	**737,989**	**2.95**	845M
EfficientNet B0 [44]	**0.950**	4,013,953	16.06	479M
MobileNetV2 [43]	0.930	2,230,277	8.92	371M
MobileNetV3 [50]	0.915	4,208,437	16.83	265M
ShuffleNet [47]	0.908	1,258,729	5.03	**176M**
EmergencyNet [64]	0.936	90,963	0.36	77.34M
TinyEmergencyNet [70]	0.895	39,334	0.16	36.30M
TakuNetV1$_{\text{FP=16}}$ [11]	**0.943**	**37,685**	**0.15**	**35.93M**
TakuNetV1$_{\text{FP=32}}$ [11]	0.938	**37,685**	**0.15**	**35.93M**
TakuNetV2$_{\text{FP=16}}$	0.936	45,945	0.18	51.80M

Table 4: Comparison of model F1-score, parameters, memory footprint (MB), and FLOPs on the AIDER [12] dataset. Images are set to 240×240 pixels. Elements highlighted in **bold** refer to the best results, while those underlined refer to the second-best results.

Model	F1-score	Parameters	Memory	FLOPS
ConvNext Tiny [80]	0.940	27,820,000	111.29	4.46G
GCVit XXtiny [81]	0.932	11,480,000	45.93	1.94G
Vit Tiny [29]	0.873	5,530,000	22.1	1.08G
Convit Tiny [82]	0.871	5,520,000	22.07	1.08G
MobileViT s [83]	0.855	4,940,000	19.76	1.42G
DiRecNetV2 [84]	**0.964**	799,380	3.20	1.09G
MobileViT xs [83]	0.837	1,930,000	7.74	710M
EfficientNet-B0 [44]	0.934	4,010,000	16.05	410M
MobileVit V2 050 [85]	0.834	1,110,000	4.46	360M
MobileNetV2 [43]	0.934	2,230,000	8.92	330M
MobileViT xxs [83]	0.859	950,000	3.81	260M
SqueezeNet [46]	0.845	**730,000**	**2.90**	260M
ShuffleNet V2 [72]	0.852	1,260,000	5.03	150M
MobileNetV3 [50]	0.919	930,000	3.72	**60M**
EmergencyNet [64]	0.955	90,704	0.36	61.96M
TinyEmergencyNet [70]	0.882	39,075	0.16	31.57M
TakuNetV1$_{\text{FP=16}}$ [11]	0.948	**37,444**	**0.15**	**31.38M**
TakuNetV1$_{\text{FP=32}}$ [11]	0.957	**37,444**	**0.15**	**31.38M**
TakuNetV2$_{\text{FP=16}}$	**0.959**	45,704	0.18	45.23M

Table 5: Comparison of F1-score, parameters, memory footprint (MB), and FLOPs on the AIDERv2 [13] dataset. Input image size is set to 224. Elements highlighted in **bold** refer to the best results, while those underlined refer to the second-best results.

ing EmergencyNet (0.815), TinyEmergencyNet (0.710). This improvement is driven by two principal architectural advances: the dense stem, which enhances initial channel mixing and spatial context aggregation, and the TakuBlockV2 modules, whose dual-branch depthwise convolutions and channel-aware pathways enable multi-scale feature extraction and refined inter-channel interactions. These design choices allow TakuNetV2 to generate richer and more discriminative feature maps, directly translating into superior performance on complex aerial scene classification tasks. Notably, while larger models – such as ResNet50 (0.831) – remain competitive, they require orders of magnitude more parameters and computational resources. The same applies

also to lightweight vision models such as MobileNets V2 (0.733) and V3 (0.799), EfficientNet (0.752), or one the group-best-performing model SqueezeNet (0.814), showing their unsuitability for these specific settings.

Across all the three benchmarks AIDER, AIDERv2, and CLRS, the original TakuNetV1 consistently balances accuracy and efficiency, establishing state-of-the-art performance among ultra-lightweight models. TakuNetV2 builds on this foundation, demonstrating modest accuracy trade-offs on the smallest dataset (AIDER) and significant improvements on larger and more complex datasets in terms of classes (AIDERv2 and CLRS). In particular, V2 leverages increased data volume to maximize the benefits of its enhanced Stem and TakuBlockV2, resulting in the best overall F1-scores (0.959 on AIDERv2, 0.838 on CLRS) while preserving real-time inference capabilities and low resource consumption. These results validate TakuNetV2 as a versatile and effective solution for embedded aerial image recognition under diverse data regimes.

The ablation study in Table 7 traces the incremental transition from TakuNetV1 to the final TakuNetV2 configuration, with each row adding a single architectural modification. Unless otherwise noted, all intermediate configurations inherit the fp16 AMP training regime of TakuNetV1; the switch to bf16 is introduced as an explicit, isolated step in row 7. Starting from the complete TakuNetV1 baseline (0.805), the first step removes the Refiner module (row 1), which causes a substantial accuracy drop to 0.761 (−4.4 pp). The Refiner — a single non-activated convolutional layer designed to numerically stabilize pre-logit activations — accounts for approximately 6.8% of TakuNetV1's total parameters. Its strong contribution to TakuNetV1's accuracy reflects the fact that, within that architecture, it compensates for the limited discriminative quality of the purely depthwise features produced upstream: without it, the classifier receives insufficiently processed representations. Importantly, this does not imply that the Refiner is an indispensable component in general; rather, it masks the representational deficit of TakuBlocksV1. The subsequent introduction of TakuBlocksV2 (row 5) and the richer stem (row 4) directly addresses this deficit at its source, rendering the Refiner functionally redundant: the final TakuNetV2 achieves 0.838 without it. The ablation therefore demonstrates that the architectural changes introduced in TakuNetV2 are not merely additive improvements, but constitute a genuine redesign in which the role previously played by the Refiner is absorbed into the feature extraction pipeline itself, yielding both higher accuracy and a more principled architecture.

The removal of the Global Response Normalization (GRN) layer (test #2), which relies on computationally expensive operations such as norms and square roots, results in only a minor further decrease in accuracy (to 0.793). This suggests that, although GRN can be beneficial in certain contexts, its contribution is limited in lightweight architectures and its latency overhead on embedded hardware outweighs its marginal performance gains, especially in complex scenarios like CLRS.

The introduction of the LeakyReLU activation function (test #3), introduced to mitigate the dead neuron effect, does not lead to any measurable improvement in accuracy. This suggests that activation sparsity does not introduce undesired bottlenecks for this architecture under the current experimental conditions. Although the LeakyReLU neither improves nor degrades the final accuracy, it is retained also due to its comparable latency characteristics. However, in other contexts, the use of LeakyReLU may still offer advantages given its potential to alleviate neuronal inactivity.

A significant leap in performance is observed with the adoption of a denser stem (test #4), which increases both the number of parameters and FLOPS (to 44,365 and 58.87M, respectively), but also raises accuracy to 0.813 (+2.5%). The stem, composed of two dense convolutional layers, is critical for early-stage feature extraction and channel mixing. Its dense computational nature is particularly well-suited for acceleration on GPUs and NPUs, making the trade-off between complexity and accuracy highly favorable in practical deployments.

The introduction of TakuBlocksV2 (test #5) marks another substantial improvement, elevating accuracy to 0.827 and further increasing the parameter count and computational cost. These blocks employ dual-branch depthwise convolutions and channel-aware pathways, which enable multi-scale feature extraction and refined inter-channel interactions. This design directly addresses the channel-wise limitations of the original TakuNetV1, allowing the network to capture more specialized and discriminative features essential for complex aerial scene classification.

Further improvements involve reducing the number of layers in the initial stages, where features are less discriminative, and increasing the group size in pointwise convolutions within the downsampler modules (test #6). These changes yield a modest accuracy gain (to 0.831) and, importantly, a substantial reduction in FLOPS, demonstrating the effectiveness of parameter redistribution and architectural streamlining.

Finally, the training compute dtype is switched from fp16 to bf16 mixed precision (test #7). Both dtypes pre-

Model	F1-score	Parameters	Memory	FLOPS
VGG16 [18]	0.762	15,341,913	61.37	20.05G
ResNet50 [20]	**0.831**	23,559,257	94.24	5.37G
SqueezeNet [46]	0.788	**748,249**	**2.99**	968M
EfficientNet B0 [44]	0.752	4,039,573	16.16	522M
MobileNetV2 [43]	0.733	2,255,897	9.02	408M
MobileNetV3 [50]	0.799	4,234,057	16.94	292M
ShuffleNet [47]	0.814	1,279,229	5.12	**193M**
EmergencyNet [64]	0.815	96,143	0.38	82.31M
TinyEmergencyNet [70]	0.710	44,514	0.18	42.63M
TakuNetV1$_{\text{FP=16}}$ [11]	0.805	**42,505**	**0.17**	**40.95M**
TakuNetV1$_{\text{FP=32}}$ [11]	0.791	**42,505**	**0.17**	**40.95M**
TakuNetV2$_{\text{FP=16}}$	**0.838**	50,765	0.20	59.07M

Table 6: Comparison of F1-score, parameters, memory footprint (MB), and FLOPS on CLRS [14], where images are set to 256×256 pixels. Elements highlighted in **bold** refer to the best results, while those underlined refer to the second-best results.

#	Arch. element	F1-score	Parameters	Memory	FLOPS
	TakuNetV1[11]	0.805	42,505	0.16	40.95M
1	w/o refiner	0.761	39,625	0.15	40.91M
2	w/o GRN [74]	0.793	38,185	0.15	40.91M
3	LeakyReLU	0.793	38,185	0.18	40.91M
4	stem	0.813	44,365	0.18	58.87M
5	TakuBlocksV2	0.827	48,605	0.19	63.31M
6	#layers, #groups	0.831	50,765	0.20	59.07M
7	bf16-mixed prec.	0.838	50,765	0.20	59.07M

Table 7: Ablation study on architectural difference between TakuNetV1 and TakuNetV2 on the CLRS [14] dataset . The last row correspond to the final version of the proposed TakuNetV2.

serve a FP32 master copy of the weights under PyTorch AMP, so the exported model is format-identical in either case and incurs no deployment incompatibility on platforms lacking native bf16 support. The accuracy gain ($0.831 \rightarrow 0.838$) is attributable to the wider dynamic range of bf16, whose eight exponent bits match those of FP32 and eliminate the gradient underflow risk that fp16's narrower five-exponent-bit range can introduce in architectures with dense stems and parallel depthwise branches, precisely the structures added in rows 4 and 5. That TakuNetV1 does not require this switch confirms that fp16 instability is an emergent property of the denser architecture rather than a dataset artifact: the simpler single-branch depthwise operators and shallower stem of TakuNetV1 keep gradient magnitudes within fp16's representable range throughout training.

In summary, the ablation study validates the rationale behind each architectural decision. The removal of non-essential components, the strategic enhancement of the stem, the introduction of advanced TakuBlocksV2, and the careful tuning of layer depth and group size collectively drive the model's performance to state-of-the-art levels for ultra-lightweight aerial image recognition. These results highlight the importance of holistic architectural optimization, where even minor changes can have significant impacts on both accuracy and efficiency, especially in resource-constrained environments.

7. Performance on Real Embedded Devices

Although floating-point operations per second (FLOPS) are commonly used in the literature to estimate computational complexity and serve as a rough proxy for inference latency, their predictive

value is often limited in real-world embedded scenarios. FLOPS do not account for hardware-specific optimizations, memory access patterns, parallelization bottlenecks, or the heterogeneous support for neural network operations across different device architectures. Consequently, models with similar theoretical complexity may exhibit markedly different latency and throughput when deployed on actual hardware, especially in energy-constrained environments.

To provide a realistic assessment of deployability, we report direct latency and throughput measurements for all models on a diverse set of embedded platforms. The experimental setup includes ARM-based single-board computers without dedicated neural accelerators, such as the Raspberry Pi family, as well as devices equipped with general-purpose GPUs, including NVIDIA Jetson Nano and Jetson Orin Nano. Additionally, the astrial board featuring the specialized Neural Processing Unit (NPU) Hailo 8 for low-power, high-throughput inference is considered. This selection encompasses the spectrum of commercially available embedded solutions, ranging from cost-effective, general-purpose boards to advanced platforms optimized for deep learning workloads. All devices operate within a low-power envelope, typically between 4 and 15 watts, ensuring relevance for both scientific benchmarking and practical deployment in UAVs.

For each platform, models were exported to ONNX format and evaluated on batches of 2,500 images, with 1,000 images used for Raspberry Pi 3 due to memory constraints. Where supported, floating-point-16 models, such as TakuNetV1 and TakuNetV2, were further optimized using TensorRT, enabling efficient utilization of GPU and NPU resources without loss of accuracy. Inference times were measured end-to-end, excluding data transfer and pre/post-processing overhead, to reflect the real latency for each model.

To provide a complete understanding of model behavior, we report three complementary metrics: FPS, FPS/δ-Power, and FPS/Power. FPS quantifies raw inference throughput, reflecting real-time capability. FPS/δ-Power captures the true computational efficiency of the model by considering only the additional power required above the idle state, thus isolating the effectiveness of the architecture itself. Conversely, FPS/Power expresses the global efficiency with respect to the device's total power consumption, which is especially relevant for deployment and battery life estimation. Together, these metrics offer a balanced picture of both intrinsic model efficiency and practical energy requirements for embedded deployment.

The results presented in Tables 8, 9, and 10 provide a detailed comparison of state-of-the-art ultra-lightweight architectures across heterogeneous embedded hardware, highlighting the interplay between model design, hardware acceleration, and real-world efficiency. This analysis enables a nuanced understanding of the trade-offs involved in selecting and deploying neural networks for energy- and latency-constrained applications.

7.1. CPU-Based Platform Performances

The results in Table 8 provide a comprehensive view of model performance across three generations of Raspberry Pi boards, each representing a different tier of embedded CPU capability. On the Raspberry Pi 3B, all models are constrained by limited computational resources and memory bandwidth. Here, TinyEmergencyNet achieves the highest frame rate (11.3 FPS), yet this performance is still inadequate for real-time operation on edge devices, and its accuracy is notably lower than that of the TakuNet family. The original TakuNetV1 ($\text{TakuNetV1}_{\text{FP=16}}$) and TakuNetV2 ($\text{TakuNetV2}_{\text{FP=16}}$) both deliver competitive throughput (9.17 and 7.64 FPS, respectively), with TakuNetV2 offering a superior balance of power efficiency and accuracy due to its denser stem and improved feature extraction. Notably, TakuNetV2 matches EmergencyNet in power consumption while providing higher FPS and better classification results, underscoring the effectiveness of its architectural refinements even on legacy hardware.

Moving to the Raspberry Pi 4, the increased processing power and memory bandwidth allow all models to achieve substantially higher frame rates. TinyEmergencyNet again leads in raw FPS (38.8), but TakuNetV2 narrows the gap, reaching 31.4 FPS with a power draw of just 5.25 W. Importantly, TakuNetV2 outperforms both EmergencyNet and the original TakuNetV1 in terms of FPS per watt and FPS per delta-power, reflecting its superior energy efficiency and suitability for sustained, real-time operation. The denser stem and optimized block structure of TakuNetV2 enable it to better exploit the available hardware, translating architectural improvements directly into practical gains.

On the Raspberry Pi 5, which offers a significant leap in CPU performance, the advantages of the TakuNet family become even more pronounced. The original TakuNetV1 achieves the highest absolute FPS (141.0), demonstrating the scalability of its lightweight design. However, TakuNetV2, while slightly lower in raw FPS (101.5), delivers a markedly better trade-off between throughput, power consumption, and accuracy. Its architectural enhancements – particularly the improved

Raspberry Pi 3B	FPS	Power [W]	FPS/δ-Power	FPS/Power
idle		1.58		
EmergencyNet	7.3	2.81	5.98	2.61
TinyEmergencyNet	**11.3**	2.81	**9.23**	**4.03**
TakuNetV1$_{\text{FP=16}}$	9.17	4.28	3.39	2.14
TakuNetV2$_{\text{FP=16}}$	7.64	**2.55**	7.88	3.00
Raspberry Pi 4	FPS	Power [W]	FPS/δ-Power	FPS/Power
idle		1.53		
EmergencyNet	27.6	5.30	7.31	5.20
TinyEmergencyNet	**38.8**	**5.15**	**10.72**	**7.53**
TakuNetV1	27.3	5.36	7.14	5.10
TakuNetV2$_{\text{FP=16}}$	31.4	5.25	8.43	5.98
Raspberry Pi 5	FPS	Power [W]	FPS/δ-Power	FPS/Power
idle		3.21		
EmergencyNet	92.0	8.11	18.79	11.35
TinyEmergencyNet	125.0	**7.80**	27.23	16.02
TakuNetV1	**141.0**	8.16	**28.50**	**17.28**
TakuNetV2	101.5	8.00	21.20	12.69

Table 8: FPS performance of competing models on CPU embedded platforms, with **best** and second-best results highlighted. Values are averaged over 2,500 runs at batch size 1, except for Raspberry Pi 3 with 1,000 runs due to RAM limitations.

stem and TakuBlockV2 modules – enable it to maintain high efficiency and robust classification performance, making it the most balanced choice for deployment where both speed and accuracy are critical.

Overall, these results demonstrate that TakuNetV2 consistently delivers strong real-time performance and energy efficiency across a range of embedded CPU platforms. Its design choices, including a denser computational stem and advanced feature extraction modules, allow it to adapt effectively to varying hardware capabilities, outperforming or matching other ultra-lightweight models not only in accuracy but also in practical deployability. This highlights the importance of co-designing neural architectures with hardware characteristics in mind to achieve optimal results in real-world embedded applications.

7.2. GPU-Accelerated Platforms Performances

The results in Table 9 provide a detailed comparison of model performance on two generations of NVIDIA Jetson platforms, each equipped with a dedicated GPU accelerator optimized for edge deep learning workloads. On the Jetson Nano, which represents an entry-level GPU-accelerated embedded device, TakuNetV1 achieves the highest throughput at 108.0 FPS, with the best power efficiency (21.18 $\frac{FPS}{W}$) and FPS-per-delta-power (33.75). This demonstrates the exceptional suitability of the original TakuNet architecture for real-time inference on legacy edge GPU hardware, where both speed and energy consumption are critical. TinyEmergencyNet also performs well, reaching 87.3 FPS, but its lower accuracy compared to TakuNetV1 and TakuNetV2 limits its practical utility for demanding vision tasks. Notably, TakuNetV2 delivers 84.3 FPS (*i.e.*, it is comparable to TinyEmergencyNet), while offering a substantial improvement in classification accuracy thanks to its enhanced stem and TakuBlockV2 modules. The denser computational structure of TakuNetV2, although slightly heavier, is efficiently handled by the Jetson Nano's GPU, resulting in a strong balance between throughput and accuracy.

On the Jetson Orin Nano, which features a significantly more powerful GPU and improved memory bandwidth, all models achieve markedly higher frame rates. TakuNetV1 again leads, attaining an impressive 655.3 FPS with the highest energy efficiency (65.53 $\frac{FPS}{W}$) and FPS-per-delta-power (119.15). This scalability highlights the architectural efficiency of TakuNet, which is able to fully exploit the advanced hardware capabilities of the Orin Nano. TakuNetV2 achieves 560.3 FPS, ranking second in both raw throughput and efficiency metrics, and demonstrates that its architectural enhancements –particularly the dual-branch TakuBlockV2 – scale effectively with increased com-

Jetson Nano	FPS	Power [W]	FPS/δ-Power	FPS/Power
idle		1.9		
EmergencyNet	68.8	5.9	17.20	11.66
TinyEmergencyNet	<u>87.3</u>	5.7	22.97	15.32
TakuNetV1	**108.0**	**5.1**	**33.75**	**21.18**
TakuNetV2	84.3	<u>5.4</u>	<u>24.09</u>	<u>15.61</u>
Jetson Orin Nano	FPS	Power [W]	FPS/δ-Power	FPS/Power
idle		4.5		
EmergencyNet	400.5	14.5	40.05	27.62
TinyEmergencyNet	428.6	13.8	46.09	31.06
TakuNetV1	**655.3**	**10**	**119.15**	**65.53**
TakuNetV2	<u>560.3</u>	**10**	<u>101.87</u>	<u>56.03</u>

Table 9: FPS performance of competing models on GPU embedded platforms, with **best** and <u>second-best</u> results highlighted. Values are averaged over 2,500 runs at batch size 1

putational resources. EmergencyNet and TinyEmergencyNet, while competitive in terms of FPS (400.5 and 428.6, respectively), lag behind in efficiency or accuracy, underscoring the advantage of the TakuNet family for high-performance embedded deployment.

It is noteworthy that, despite the increased computational complexity of TakuBlockV2 (with roughly three times as many layers per block compared to the original), TakuNetV2 maintains excellent efficiency on both Jetson platforms. This is attributable to the hardware-aware design, which ensures that the added depth and channel interactions are well-matched to the parallel processing capabilities of modern GPUs. Overall, these results confirm that both TakuNetV1 and TakuNetV2 deliver state-of-the-art performance and energy efficiency across a range of GPU-accelerated embedded devices, with TakuNetV2 offering a compelling trade-off between throughput and classification accuracy for real-time aerial image recognition.

7.3. NPU-Based Platforms Performances

The results for the Astrial platform equipped with the Hailo-8 NPU, shown in Table 10, require careful interpretation in light of the Hailo-8's distinctive hardware architecture and its compiler behavior for non-standard topologies.

The Hailo-8 implements a proprietary structure-driven dataflow architecture in which the chip's compute, memory, and control blocks are physically distributed across the die and *statically allocated to individual network layers at compile time*. Crucially, the Hailo-8 integrates all required memory directly on the processor die, with no access to external DRAM . All intermediate activations and weights must therefore fit within the chip's fixed on-chip SRAM during a single inference pass. When the compiler determines that a model's resource requirements exceed what can be mapped within a single execution context,due to routing constraints, layer topology complexity, or on-chip memory pressure, it partitions the model into multiple sequential *contexts*, each of which is loaded and executed separately, with intermediate results transferred between contexts over the PCIe bus. When multiple contexts are required, additional transfers will need to occur over the PCIe bus in order to perform the context switching required to execute the entire model. This context-switching overhead is well-documented: models compiled to a single context consistently achieve substantially higher throughput than multi-context equivalents, regardless of their theoretical FLOPS count.

In the case of TakuNetV2, the Hailo Dataflow Compiler generates a two-context execution plan, while TakuNetV1 compiles to a single context. This outcome is paradoxical given the extremely small size of TakuNetV2 (~51K parameters, 45M FLOPS), which is orders of magnitude smaller than models the Hailo-8 routinely executes in a single context — such as ResNet-50 (25M parameters, 4.1G FLOPS). The root cause is therefore not model size but rather the *topological structure* of TakuBlockV2: the parallel dual-branch depthwise paths and the multi-input concatenation operation create a non-linear dataflow graph that the current version of the Hailo compiler cannot route within a single on-chip context allocation. This is a compiler limitation specific to the current Hailo Dataflow Compiler toolchain, not an intrinsic property of the model,

Astrial	FPS	Power [W]	FPS/δ-Power	FPS/Power
idle		3.99		
EmergencyNet	868.0	4.79	1085.00	181.16
TinyEmergencyNet	**1920.0**	5.27	**1500.00**	**364.33**
TakuNetV1	<u>1116.0</u>	<u>4.84</u>	<u>1312.94</u>	<u>230.58</u>
TakuNetV2	358.0	**4.46**	761.70	80.27

Table 10: FPS performance of competing models on NPU embedded platforms, with **best** and <u>second-best</u> results highlighted. Values are averaged over 2,500 runs at batch size 1.

and does not arise on any other platform tested in this work.

Despite this penalty, TakuNetV2 achieves 358 FPS on the Astrial platform — substantially exceeding any real-time deployment requirement for aerial image classification — and maintains a competitive FPS/δ-Power ratio of 761. Notably, TakuNetV2 achieves this while consuming the lowest absolute power of any model tested on this platform (4.46W), which can be attributed to the PCIe transfer overhead not drawing additional compute power from the NPU array itself. The dual-context execution paradoxically results in a model that is *thermally lighter* on the NPU die, since each context utilizes fewer on-chip resources per pass.

We note that this limitation is specific to the current generation of Hailo toolchain support and is actively evolving. As documented by Hailo, newer compiler versions improve multi-context allocation and routing for complex topologies. Furthermore, the successor architecture Hailo-10H introduces direct DDR memory access, which would eliminate the on-chip memory pressure that triggers multi-context compilation entirely. The TakuNetV2 architecture is therefore not constrained by any fundamental NPU incompatibility, but rather sits at the frontier of what current compiler toolchains can optimally map — a position that reflects the forward-looking nature of its design rather than a deployment deficiency.

TakuNetV1, compiling to a single context, achieves 1,116 FPS on the Astrial with an FPS/δ-Power of 1,312.94 — the second-highest energy efficiency of any model on any platform tested in this work. This confirms TakuNetV1 as the preferred choice when NPU deployment with maximal throughput is required.

7.4. Dual-context compilation on Hailo-8

Plausible root cause. The Hailo-8 neural core is a structured-dataflow streamer processor designed circa 2017 and optimised for dense full convolutions. Its compute fabric consists of eight processing clusters interconnected by a Network-on-Chip (NoC) bus, backed by 32 MB of on-chip Static Random-Access Memory (SRAM) shared among compiled model weights, intermediate activations, and I/O tensors [86]. The Dataflow Compiler (DFC) allocates these resources greedily to maximise throughput subject to per-context utilisation thresholds on compute, memory, and control logic; when any threshold is exceeded the graph is automatically partitioned into consecutive execution *contexts*, each fitting within the on-chip envelope. Within this framework, depthwise convolutions impose a structural overhead that standard convolutions do not. A dense convolution maps every input channel to every output channel, keeping all MAC lanes active and allowing activations to stream continuously through the NoC from one cluster to the next. A depthwise convolution is architecturally equivalent to a group convolution in which the group count equals the number of input channels, with exactly one filter per group. Because the Hailo-8 subcluster processes features in multiples of eight, a depthwise layer satisfies the hardware's grouping constraint (OF $<$ 8 outputs per group) but activates only a single MAC lane per group, leaving the remaining seven idle. Crucially, the valid per-channel results must be gathered and re-ordered before they can be forwarded downstream: the DFC must allocate an additional routing unit for this purpose, consuming control and on-chip memory resources beyond those required by the convolution itself. The cost is quantitatively visible in the DFC's supported-parameter tables: a standard 3×3 convolution supports dilation factors up to 16×16 at unit stride, whereas the optimised depthwise 3×3 path reaches only 4×4, a fourfold reduction that directly reflects the tighter resource budget consumed by the extra routing stage [86]. The DFC profiler also exposes this overhead through the `mac_layers_util` metric, which records the fraction of subclusters computing valid output features *excluding* intermediate routing operations, confirming that depthwise operators occupy hardware resources without contributing proportionally to useful computation.

TakuNetV2 applies this operator class under conditions

that compound the routing pressure along two independent axes. First, the revised stem comprises two successive 3×3 dense convolutions with stride 2, processing the full-resolution input before any spatial downsampling. The dense dataflow of standard convolutions means that every compute and weight-memory budget entry is consumed at maximal channel utilisation at this stage, leaving a reduced on-chip footprint for the subsequent blocks. Second, each TakuBlockV2 instance applies a grouped pointwise projection followed by a channel-wise split of the output into two equal halves, each processed by a separate depthwise branch: one with a standard 3×3 kernel and one with a dilated 3×3 kernel. Both branches must retain their intermediate activation tensors simultaneously in on-chip L3 or L4 memory until they are combined, preventing the producer-to-consumer streaming that the NoC is designed to exploit and requiring twice the buffer allocation of a single-branch depthwise design. Each branch independently triggers the extra routing-unit allocation intrinsic to depthwise operators on this architecture; their structural parallelism therefore doubles the routing overhead relative to a single depthwise path. The cumulative effect, a memory-intensive dense stem occupying the initial resource budget, followed by repeated stages of parallel depthwise branches each carrying routing overhead and competing for the same 32 MB of on-chip SRAM, exhausts the available resource envelope before the full graph can be allocated within a single context, causing the DFC to partition TakuNetV2 into two consecutive execution contexts. It should be noted that the dual-context behaviour is an end-to-end property of the compiled graph as a whole; while the denser stem introduced in TakuNetV2 increases the initial resource consumption, the DFC output does not provide sufficient granularity to attribute the context split to any single architectural modification in isolation, and the parallel depthwise branches of TakuBlockV2 may constitute a stronger plausible contributing factor considering how the NPU is designed to work.

Power measurement boundary conditions under dual-context execution. The dual-context configuration introduces two sources of execution overhead that do not arise in single-context networks: the transfer of intermediate activation tensors between the two contexts over the PCIe interface to host DRAM and back, and the reconfiguration of the accelerator's internal NoC routing tables and cluster assignments before the second context can begin execution. Of these two components, reconfiguration is the dominant term: its duration is largely independent of tensor size and is instead determined by the number of layers and subcluster assignments that must be reprogrammed, whereas the PCIe transfer time scales with the size of the inter-context activation maps. The reported power information encompass the complete inference execution, including both contributions. This is an intentional worst-case boundary condition: all energy consumed during context switching is energy that is genuinely required to execute TakuNetV2 on this accelerator in its as-compiled form, and reporting it in full provides an honest assessment of the effective deployment cost on the selected hardware [86]. The idle baseline, measured while the accelerator is powered but not executing inference, excludes context-switching activity by construction: no contexts are active, and therefore neither PCIe transfers nor reconfiguration events occur. Both components of the dual-context overhead consequently appear exclusively in the active-inference energy, and the idle power correctly captures only the quiescent chip state. No available Hailo profiling tool decomposes the PCIe transfer power from the reconfiguration power within the context-switch interval; an estimate derived from inter-context tensor sizes, bus voltage, and link-utilisation parameters could in principle be constructed, but it would rely on unverifiable assumptions about the PCIe PHY power model and would not improve the accuracy of the reported figures. For these reasons, and because the reconfiguration overhead is itself non-negligible, the dual-context penalty is reported entirely at the system level. Its most tangible manifestation is an increase in end-to-end latency and energy per inference, both of which are captured by the measurements reported in Table 10.

In summary, while TakuNetV2 introduces a more expressive and accurate architectural design, its advanced modularity and interconnections reveal limitations in current NPU hardware and compiler support, such as the dual-context execution required on Hailo-8. This limitation paradoxically favors large, dense models and penalizes efficient, state-of-the-art lightweight architectures that otherwise excel on other embedded platforms. Moreover, the TakuNet family of architectures demonstrates inherent superiority: TinyEmergencyNet, derived from pruning EmergencyNet, must retrain beginning from pretrained weights and effectively undergoes approximately twice the number of training epochs to reach a reasonable, albeit significantly lower, accuracy level. We hope that future NPU designs and toolchains will evolve to better support these modern, efficient models, enabling them to reach their full potential for real-time, low-power edge AI applications.

ReLU [87]	**ReLU6** [49]	**LReLU** [76]	**PReLU** [88]
30.08s	30.84s	30.87s	32.87s
ELU [89]	**CELU** [90]	**GELU** [77]	**SELU** [91]
168.56s	169.27s	401.49s	169.00s

Table 11: Latency of common activation functions on a Raspberry Pi 4, measured during execution on a (10000, 100) tensor.

7.5. Activation Functions Cost on ARM CPU

The latency measurements reported in Table 11 highlight the substantial differences in computational cost among common activation functions on embedded hardware. While ReLU, ReLU6, and LeakyReLU exhibit nearly identical and minimal latency, more complex activations such as ELU, CELU, and GELU are significantly slower—by more than an order of magnitude. This result justifies the adoption of LeakyReLU in our architecture, as it provides robust nonlinearity without incurring the latency penalties associated with advanced activation functions, ensuring efficient inference on resource-constrained devices.

8. Conclusion

This work presents TakuNet advanced ultra-lightweight neural architectures specifically designed for aerial image recognition on embedded platforms. In particular, by introducing a denser stem and the new TakuBlockV2 units, hybrid Ghost blocks with parallel standard and dilated depthwise convolutions, TakuNetV2 significantly enhances feature diversity and spatial representation, enabling robust generalization across heterogeneous and complex datasets, even more than the first version TakuNetV1. Extensive experiments on public benchmarks and real embedded hardware – including CPUs, GPUs, and NPUs – demonstrate that the TakuNet architecture design consistently achieves state-of-the-art accuracy while maintaining a minimal parameter count, memory footprint, and inference latency.

The hardware-aware design choices, such as the use of low-latency activations and dense initial convolutions, prove critical for maximizing efficiency in various edge devices. In particular, TakuNetV1and TakuNetV2deliver strong performance not only on conventional platforms but also on energy-constrained systems, confirming its versatility for real-world deployment in scenarios such as emergency response and autonomous UAV operations.

However, our analysis also reveals current limitations in NPU hardware and compiler toolchains, where routing and context constraints can penalize modular and efficient models, paradoxically favoring larger and denser architectures. This finding underscores the need for a closer co-evolution between neural network design and hardware acceleration technologies, so that future NPUs and compilers can fully support advanced lightweight models.

Future research will focus on evaluating the effectiveness of TakuNet architecture as a backbone for more complex computer vision tasks, including object detection and multi-object tracking in aerial imagery. Given its demonstrated efficiency and strong feature extraction capabilities, TakuNet is well positioned to serve as the core of lightweight detection and tracking pipelines, particularly in resource-constrained embedded environments. By extending its application beyond classification, we aim to assess its ability to support real-time, end-to-end perception systems for UAVs and other edge devices, further validating its versatility and impact for next-generation embedded deep vision tasks.

References

[1] P. Villalobos, J. Sevilla, T. Besiroglu, L. Heim, A. Ho, M. Hobbhahn, Machine learning model sizes and the parameter gap, arXiv preprint arXiv:2207.02852 (2022). 1

[2] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al., Laion-5b: An open large-scale dataset for training next generation image-text models, Advances in neural information processing systems 35 (2022) 25278–25294. 1

[3] X. Wang, L. L. Zhang, Y. Wang, M. Yang, Towards efficient vision transformer inference: A first study of transformers on mobile devices, in: Proceedings of the 23rd annual international workshop on mobile computing systems and applications, 2022, pp. 1–7. 1

[4] P. Daponte, L. De Vito, L. Glielmo, L. Iannelli, D. Liuzza, F. Picariello, G. Silano, A review on the use of drones for precision agriculture, in: IOP conference series: earth and environmental science, Vol. 275, IOP Publishing, 2019, p. 012022. 1

[5] D. Gallacher, Drone applications for environmental management in urban spaces: A review, International Journal of Sustainable Land Use and Urban Planning 3 (4) (2016). 1

[6] N. M. K. Dousai, S. Loncaric, Detection of humans in drone images for search and rescue operations, in: Proceedings of the 2021 3rd Asia Pacific Information Technology Conference, 2021, pp. 69–75. 1

[7] T. Furutani, M. Minami, Drones for disaster risk reduction and crisis response, Emerging Technologies for Disaster Resilience: Practical Cases and Theories (2021) 51–62. 1

[8] V. Mersheeva, G. Friedrich, Multi-UAV monitoring with priorities and limited energy resources, in: Proceedings of the International Conference on Automated Planning and Scheduling, Vol. 25, 2015, pp. 347–355. 2

[9] S. Oh, M. Kim, D. Kim, M. Jeong, M. Lee, Investigation on performance and energy efficiency of cnn-based object detection on embedded device, in: 2017 4th International Conference on

Computer Applications and Information Processing Technology (CAIPT), IEEE, 2017, pp. 1–4. 2

[10] K. J. Lee, Architecture of neural processing unit for deep neural networks, in: Advances in computers, Vol. 122, Elsevier, 2021, pp. 217–245. 2

[11] D. Rossi, G. Borghi, R. Vezzani, Takunet: an energy-efficient cnn for real-time inference on embedded UAV systems in emergency response scenarios, in: Proceedings of the Winter Conference on Applications of Computer Vision, 2025, pp. 376–385. 2, 3, 11, 20, 22, 24

[12] C. Kyrkou, T. Theocharides, Deep-learning-based aerial image classification for emergency response applications using unmanned aerial vehicles., in: CVPR workshops, 2019, pp. 517–525. 3, 8, 18, 19, 21, 22

[13] D. Shianios, C. Kyrkou, P. S. Kolios, A benchmark and investigation of deep-learning-based techniques for detecting natural disasters in aerial images, in: International Conference on Computer Analysis of Images and Patterns, Springer, 2023, pp. 244–254. 3, 8, 18, 19, 21, 22

[14] H. Li, H. Jiang, X. Gu, J. Peng, W. Li, L. Hong, C. Tao, Clrs: Continual learning benchmark for remote sensing image scene classification, Sensors 20 (4) (2020) 1226. 3, 19, 21, 24

[15] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks 3361 (10) (1995) 1995. 3

[16] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012. 4

[17] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems 25 (2012). 4, 6, 9

[18] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014). 4, 9, 20, 22, 24

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9. 4

[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. 4, 9, 11, 15, 20, 22, 24

[21] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708. 4, 11, 15

[22] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500. 4

[23] S. Zagoruyko, N. Komodakis, Wide residual networks, in: British Machine Vision Conference 2016, British Machine Vision Association, 2016. 4

[24] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, P. Torr, Res2net: A new multi-scale backbone architecture, IEEE transactions on pattern analysis and machine intelligence 43 (2) (2019) 652–662. 4

[25] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141. 4

[26] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, Q. Hu, Eca-net: Efficient channel attention for deep convolutional neural networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11534–11542. 5

[27] S. Woo, J. Park, J.-Y. Lee, I. S. Kweon, Cbam: Convolutional block attention module, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 3–19. 5

[28] A. Vaswani, Attention is all you need, Advances in Neural Information Processing Systems (2017). 5

[29] A. Dosovitskiy, An image is worth 16x16 words: Transformers for image recognition at scale, arXiv preprint arXiv:2010.11929 (2020). 5, 9, 22

[30] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, in: International conference on machine learning, PMLR, 2021, pp. 10347–10357. 5

[31] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, R. Girshick, Masked autoencoders are scalable vision learners, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 16000–16009. 5

[32] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 10012–10022. 5, 10

[33] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, L. Zhang, Cvt: Introducing convolutions to vision transformers, in: Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 22–31. 5

[34] W. Xu, Y. Xu, T. Chang, Z. Tu, Co-scale conv-attentional image transformers, in: Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 9981–9990. 5

[35] N. C. Thompson, K. Greenewald, K. Lee, G. F. Manso, et al., The computational limits of deep learning, arXiv preprint arXiv:2007.05558 10 (2) (2020). 5, 6

[36] R. Schwartz, J. Dodge, N. A. Smith, O. Etzioni, Green ai, Communications of the ACM 63 (12) (2020) 54–63. 5, 6

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255. 5

[38] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: European conference on computer vision, Springer, 2014, pp. 740–755. 5

[39] C. Sun, A. Shrivastava, S. Singh, A. Gupta, Revisiting unreasonable effectiveness of data in deep learning era, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 843–852. 5

[40] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, L. Van Gool, Ai benchmark: All about deep learning on smartphones in 2019, in: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), IEEE, 2019, pp. 3617–3635. 6

[41] E. Strubell, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in nlp, in: Proceedings of the 57th annual meeting of the association for computational linguistics, 2019, pp. 3645–3650. 6

[42] J. Pan, A. Bulat, F. Tan, X. Zhu, L. Dudziak, H. Li, G. Tzimiropoulos, B. Martinez, Edgevits: Competing light-weight cnns on mobile devices with vision transformers, in: European conference on computer vision, Springer, 2022, pp. 294–311. 6

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520. 6, 14, 22, 24

[44] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, 2019, pp. 6105–6114. 6, 7, 10, 11, 16,

21, 22, 24

[45] H. Cai, L. Zhu, S. Han, Proxylessnas: Direct neural architecture search on target task and hardware, arXiv preprint arXiv:1812.00332 (2018). 6, 7, 10

[46] F. N. Iandola, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size, arXiv preprint arXiv:1602.07360 (2016). 6, 20, 22, 24

[47] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848–6856. 6, 11, 22, 24

[48] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258. 6

[49] A. G. Howard, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017). 6, 11, 16, 20, 30

[50] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., Searching for mobilenetv3, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1314–1324. 7, 14, 22, 24

[51] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, H. Adam, Netadapt: Platform-aware neural network adaptation for mobile applications, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 285–300. 7

[52] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, H. Hajishirzi, Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation, in: Proceedings of the european conference on computer vision (ECCV), 2018, pp. 552–568. 7

[53] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, C. Xu, Ghostnet: More features from cheap operations, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 1580–1589. 7, 14

[54] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710. 7

[55] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, K. Keutzer, Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 10734–10742. 7, 20

[56] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Advances in neural information processing systems 28 (2015). 7

[57] J.-H. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5058–5066. 7

[58] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2736–2744. 7

[59] P. Helber, B. Bischke, A. Dengel, D. Borth, Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 12 (7) (2019) 2217–2226. 8

[60] M. A. Wulder, J. C. White, T. R. Loveland, C. E. Woodcock, A. S. Belward, W. B. Cohen, E. A. Fosnight, J. Shaw, J. G. Masek, D. P. Roy, The global landsat archive: Status, consolidation, and direction, Remote Sensing of Environment 185 (2016) 271–283, landsat 8 Science Results. doi:https://doi.org/10.1016/j.rse.2015.11.032. 8

[61] A. Toker, L. Kondmann, M. Weber, M. Eisenberger, A. Camero, J. Hu, A. P. Hoderlein, Ç. Şenaras, T. Davis, D. Cremers, et al., Dynamicearthnet: Daily multi-spectral satellite dataset for semantic change segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 21158–21167. 8, 9

[62] A. Krizhevsky, V. Nair, G. Hinton, Cifar-10 and cifar-100 datasets, URl: https://www. cs. toronto. edu/kriz/cifar. html 6 (1) (2009) 1. 9

[63] G. Cheng, J. Han, X. Lu, Remote sensing image scene classification: Benchmark and state of the art, Proceedings of the IEEE 105 (10) (2017) 1865–1883. 9

[64] C. Kyrkou, T. Theocharides, Emergencynet: Efficient aerial image classification for drone-based emergency monitoring using atrous convolutional feature fusion, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 13 (2020) 1687–1699. 9, 10, 15, 18, 19, 20, 22, 24

[65] M.-D. Yang, K.-S. Huang, Y.-H. Kuo, H. P. Tsai, L.-M. Lin, Spatial and spectral hybrid image classification for rice lodging assessment through UAV imagery, Remote Sensing 9 (6) (2017) 583. 10

[66] M. Mafanya, P. Tsele, J. Botai, P. Manyama, B. Swart, T. Monate, Evaluating pixel and object based image classification techniques for mapping plant invasions from UAV derived aerial imagery: Harrisia pomanensis as a case study, ISPRS Journal of Photogrammetry and Remote Sensing 129 (2017) 1–11. 10

[67] Q. Bi, K. Qin, H. Zhang, J. Xie, Z. Li, K. Xu, Apdc-net: Attention pooling-based convolutional network for aerial scene classification, IEEE Geoscience and Remote Sensing Letters 17 (9) (2019) 1603–1607. 10

[68] W. F. Hendria, Q. T. Phan, F. Adzaka, C. Jeong, Combining transformer and cnn for object detection in UAV imagery, ICT Express 9 (2) (2023) 258–263. 10

[69] S. Qiao, L.-C. Chen, A. Yuille, Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 10213–10224. 10

[70] O. M. Mogaka, R. Zewail, K. Inoue, M. S. Sayed, Tinyemergencynet: a hardware-friendly ultra-lightweight deep learning model for aerial scene image classification, Journal of Real-Time Image Processing 21 (2) (2024) 51. 10, 18, 20, 22, 24

[71] J. P. S. Schuler, S. Romani, M. Abdel-Nasser, H. Rashwan, D. Puig, Grouped pointwise convolutions reduce parameters in convolutional neural networks, in: Mendel, Vol. 28, 2022, pp. 23–31. 11

[72] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 116–131. 11, 14, 15, 22

[73] S. Ioffe, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167 (2015). 11

[74] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, S. Xie, Convnext v2: Co-designing and scaling convnets with masked autoencoders, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 16133–16142. 12, 24

[75] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, arXiv preprint arXiv:1511.07122 (2015). 14

[76] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al., Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, Vol. 30, Atlanta, GA, 2013, p. 3. 15, 30

[77] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), arXiv preprint arXiv:1606.08415 (2016). 15, 30

[78] T. Tieleman, G. Hinton, Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning, COURSERA Neural Networks Mach. Learn 17 (2012). 19

[79] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q. V. Le, Mnasnet: Platform-aware neural architecture search for mobile, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 2820–2828. 20

[80] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, S. Xie, A convnet for the 2020s, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 11976–11986. 22

[81] A. Hatamizadeh, H. Yin, G. Heinrich, J. Kautz, P. Molchanov, Global context vision transformers, in: International Conference on Machine Learning, PMLR, 2023, pp. 12633–12646. 22

[82] S. d'Ascoli, H. Touvron, M. L. Leavitt, A. S. Morcos, G. Biroli, L. Sagun, Convit: improving vision transformers with soft convolutional inductive biases*, Journal of Statistical Mechanics: Theory and Experiment 2022 (11) (2022) 114005. doi:10.1088/1742-5468/ac9830. 22

[83] S. Mehta, M. Rastegari, Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer, arXiv preprint arXiv:2110.02178 (2021). 22

[84] D. Shianios, P. S. Kolios, C. Kyrkou, Direcnetv2: A transformer-enhanced network for aerial disaster recognition, SN Computer Science 5 (6) (Aug. 2024). doi:10.1007/s42979-024-03066-y. 22

[85] S. Mehta, M. Rastegari, Separable self-attention for mobile vision transformers, arXiv preprint arXiv:2206.02680 (2022). 22

[86] Hailo Technologies Ltd., Hailo Dataflow Compiler, version 3.33.1, Developer documentation.
URL https://hailo.ai/developer-zone/documentation/dataflow-compiler-v3-33-1/ 28, 29

[87] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814. 30

[88] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034. 30

[89] D.-A. Clevert, Fast and accurate deep network learning by exponential linear units (elus), arXiv preprint arXiv:1511.07289 (2015). 30

[90] J. T. Barron, Continuously differentiable exponential linear units, arXiv preprint arXiv:1704.07483 (2017). 30

[91] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, Advances in neural information processing systems 30 (2017). 30