This paper has been accepted for publication in the DATE 2024 conference
©2024 IEEE.

# Adaptive Deep Learning for Efficient Visual Pose Estimation aboard Ultra-low-power Nano-drones

Beatrice Alessandra Motetti*, Luca Crupi†, Mustafa Omer Mohammed Elamin Elshaigi*, Matteo Risso*,
Daniele Jahier Pagliari*, Daniele Palossi†, Alessio Burrello*
* Politecnico di Torino, Turin, 10129, Italy
† Dalle Molle Institute for Artificial Intelligence, USI and SUPSI, Lugano, 6962, Switzerland
Emails: name.surname@polito.it, name.surname@idsia.ch

arXiv:2401.15236v1 [cs.CV] 26 Jan 2024

***Abstract*—Sub-10 cm diameter nano-drones are gaining momentum thanks to their applicability in scenarios prevented to bigger flying drones, such as in narrow environments and close to humans. However, their tiny form factor also brings their major drawback: ultra-constrained memory and processors for the onboard execution of their perception pipelines. Therefore, lightweight deep learning-based approaches are becoming increasingly popular, stressing how computational efficiency and energy-saving are paramount as they can make the difference between a fully working closed-loop system and a failing one. In this work, to maximize the exploitation of the ultra-limited resources aboard nano-drones, we present a novel adaptive deep learning-based mechanism for the efficient execution of a vision-based human pose estimation task. We leverage two State-of-the-Art (SoA) convolutional neural networks (CNNs) with different regression performance vs. computational costs trade-offs. By combining these CNNs with three novel adaptation strategies based on the output's temporal consistency and on auxiliary tasks to swap the CNN being executed proactively, we present six different systems. On a real-world dataset and the actual nano-drone hardware, our best-performing system, compared to executing only the bigger and most accurate SoA model, shows 28% latency reduction while keeping the same mean absolute error (MAE), 3% MAE reduction while being iso-latency, and the absolute peak performance, i.e., 6% better than SoA model.**

***Index Terms*—Nano-drones, Adaptive Inference, TinyML**

## I. Introduction

Nano-sized Unmanned Aerial Vehicles (UAVs) have emerged as a groundbreaking technology thanks to their under-10 cm size and less than 40 g weight. They have opened new frontiers for applications in GPS-denied environments, confined spaces, and close to humans [1]. However, their miniaturization comes at the price of simplified equipped sensory, mechanical, and computational subsystems. The power available for computing, typically less than 100 milliwatts, is very limited compared to standard-sized commercial drones equipped with powerful CPUs or even mobile GPUs. Consequently, nano-UAVs typically mount Microcontroller unit (MCU)-class processors. This poses a significant challenge when executing resource-intensive perception, planning, and control algorithms. At the same time, offloading these tasks to a remote server through a continuous stream of raw sensor data is restrictive given the instability of the wireless connection and its limited range and does not guarantee a predictable response latency [2].

Therefore, to enable the onboard intelligence on nano-UAVs, lightweight Convolutional Neural Networks (CNNs) offer a possible solution to address these challenges in perception tasks. They have been shown to work effectively even with minimal sensor data, accommodating ultra-low-power cameras with limited resolution and dynamic range [3], [4]. Furthermore, they offer predictable and fixed computational and memory requirements during inference, enabling real-time operation without wireless coverage for offloading.

So far, the State-of-the-Art (SoA) autonomous navigation of nano-UAVs is characterized by deploying one static CNN [4], [5], acting as a high-level perception module. If, on the one hand, this approach offers a viable solution, on the other hand, it limits the level of accuracy and computational efficiency of the onboard perception. For some deployment scenarios, this design choice means addressing a simple task with a too-complex CNN – wasting computational resources and, therefore, energy – while in other cases, a static CNN might underperform due to a too-complex environment.

Recently, adaptive machine learning has gained traction as a method to dynamically control the trade-off between accuracy and complexity in neural network-based applications [6]. To cope with variable-complexity inputs, adaptive approaches build *ensembles of models* that are conditionally executed based on environmental conditions: in the simplest embodiment, a more lightweight, less memory demanding, and faster model is employed when a *task-specific policy* function deems the processed datum to be relatively easy, while a larger, more powerful model is used for more challenging inferences.

This work uses this paradigm to demonstrate the potential for optimizing a vision-based human pose estimation task onboard a nano-UAV: the fundamental algorithmic building block for "people monitoring" and "follow-me" applications. To the best of our knowledge, our work proposes the first employment of an adaptive system capable of adjusting its complexity based on the mission's environment. Our detailed contribution is as follows:

- We propose two distinct adaptive ensembles, which combine three models from [5]. The two reported less accurate networks are used as *small* models in our ensembles, and the most accurate network as *big* model in both ensembles. Further, we propose three novel adaptation

This publication is part of the project PNRR-NGEU which has received funding from the MUR – DM 118/2023.

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

policies based on the output's temporal consistency of pose estimation and on auxiliary tasks explicitly tailored for visual human pose estimation aboard nano-UAVs.
- We demonstrate that our adaptive algorithms, i.e., the ensembles combined with our proposed policies, outperform the SoA results on two datasets for the same robotic task.
- We demonstrate that, compared to SoA static CNNs, our approach generates a broader solution space with numerous intermediate points that provide a good trade-off between accuracy and complexity compared to static models. Compared to the most accurate SoA network, we reduce the Mean Absolute Error (MAE) on the estimated pose by 3.15% while maintaining a constant latency. When we consider iso-MAE performance, we instead reduce the latency of execution by 28.03%.

## II. Background & Related Work

### A. Human Pose Estimation aboard Nano-Drones

Our work is in the context of drone navigation tasks aimed at following a human subject, maintaining a constant distance [4]. The input data are grey-scale images captured by an onboard camera, while the outputs are poses then converted into set-points for the low-level control. More specifically, we focus on the perception sub-task, which consists of a relative pose estimation between a human subject and a nano-UAV, approached using a CNN model. While SoA computer vision techniques [7]–[9] have achieved remarkable accuracy on this problem, they remain beyond the practical capabilities of nano-drones due to their high computational demands ($\sim 10^{10}$ Multiply and ACcumulate - MAC - operations) [9]. In contrast to these algorithms, which estimate complete skeletal structures [8] or dense 3D mesh representations of individuals [9], CNNs deployed on nano-drones focus their predictions on a minimal set of elements: the head position in 3D space $(x, y, z)$ and a rotation angle relative to the gravity z-axis ($\phi$).

However, deploying CNNs on nano-UAVs still presents substantial challenges due to the limited hardware resources, as mentioned in Sec. I. Traditional solutions relying on off-the-shelf MCUs, such as those in [10], [11], can only accommodate simple neural architectures to achieve real-time performance. For example, the network in [11] performs a modest 27 kMAC per frame at a rate of 100 Hz. Consequently, these approaches are primarily suitable for processing low-dimensional input signals and ill-suited for the computational demands of camera image processing. Thus far, the PULP-Frontnet CNN and its optimized variants [4], [5] stand as one of the few instances where real-time visual-based human pose estimation tasks have been successfully executed aboard a nano-UAV. These methods leverage general-purpose ultra-low-power multi-core System-on-Chip (SoC) architectures, tailored software deployment pipelines, and fine-tuned CNN architectures to carry out the task efficiently. PULP-Frontnet has a complexity of 14.7 MMAC and achieves an inference rate of 45 Hz with a power consumption of 92.2 mW when deployed on a Crazyflie 2.1 nano-drone [4]. Subsequently, in [5], Neural Architecture Search (NAS) was employed to further statically optimize this CNN, obtaining a latency reduction of 13.2% [5] without accuracy degradation.

All advancements in executing CNN-based vision tasks on nano-drones have focused on *static* (data-independent) optimizations. This approach misses an important opportunity since, in typical drone navigation tasks, the environment can often change, e.g., from indoor to outdoor, from empty spaces to crowded ones, etc, calling for a perception method that can adapt to changes in the input data distribution. This adaptability can lead to a faster inference for "easy" input data, which provides a higher average throughput and, consequently, a faster response from the control part.

### B. Adaptive Machine Learning

The challenge of optimizing Deep Learning (DL) models for execution on ultra-low-power edge nodes, trading off slight reductions in accuracy for substantial gains in latency, energy efficiency, or memory usage, has been the subject of extensive research recently [12], [13]. One significant categorization of DL optimization techniques distinguishes them as *static* and *dynamic* [14]. The former involves compressing a model before deployment, either during the training phase or post-training. Static methods include quantization, pruning, and NAS [13], now commonly used to optimize neural networks. A critical limitation of these approaches is their inability to adapt the inference complexity at runtime when environmental conditions evolve.

Dynamic (or "adaptive") inference techniques, including the approaches presented in this paper, aim to address this limitation. An adaptive inference's complexity is reduced when the input data is "simple", and a lower amount of computation suffices to ensure high accuracy (e.g., for a vision task, the relevant object is well-centered in the frame, not blurred, and against a uniform background). Conversely, more compute budget is allocated for "challenging" inputs (e.g., a moving or partially obstructed object) [6], [15]–[17]. The easiest way to implement an adaptive system is by appropriately combining the outputs of two independent static models [6]: a smaller, less complex model for easy-to-process inputs and a larger, more accurate one for the others. Other alternatives also exist, such as using only parts of the channels of a complex model to obtain the lightweight one [15], [16], or early-exiting after a subset of layers [17].

Regardless of the specific scheme, a crucial component for all adaptive DL systems is a discrimination *policy*, which determines which model to activate for a given input data [6]. An effective policy should provide an acceptable loss of accuracy compared to the most complex model while resulting in a substantial computational saving. Notably, while policies for classification tasks have been extensively explored, the same is not true for regression tasks. To the best of our knowledge, we are the first to propose two novel policies that consider the temporal correlation between images captured by a camera and an auxiliary task to gauge the input's complexity. In Sec. III, we delve into a detailed analysis of these policies.

TABLE I
STATIC MODELS METRICS EXTRACTED FROM [5].

| Network | MAE | | | | | Params | MAC |
|---|---|---|---|---|---|---|---|
| | $x$ | $y$ | $z$ | $\phi$ | sum | | |
| $\mathbf{F}^1$ | 0.27 | 0.27 | 0.28 | 0.52 | 1.34 | 14.8 k | 4.51 M |
| $\mathbf{F}^2$ | 0.21 | 0.18 | 0.24 | 0.46 | 1.10 | 44.5 k | 7.09 M |
| $\mathbf{M}^{1.0}$ | 0.19 | 0.14 | 0.23 | 0.48 | 1.04 | 46.8 k | 11.42 M |

## III. MATERIALS & METHODS

### A. Static Neural Networks

Inspired by [6], we build our adaptive system from an ensemble of two independent "static" CNNs of different complexity and accuracy. Namely, we consider two ensembles by combining three SoA models proposed in [5], whose MAE, number of parameters, and NMACs are reported in Tab. I. $\mathrm{F}^1$ and $\mathrm{F}^2$ are two versions of a convolutional network based on PULP-Frontnet composed by 7 convolutional layers, the first with a $5 \times 5$ filter and the others with a $3 \times 3$ filter. The difference between the two networks is in the number of filters (or channels) of each layer. $\mathrm{M}^{1.0}$ is a reduced version of a classical MobileNet v1.0, pruned by a SoA NAS algorithm [13] to minimize the number of filters. All networks have been trained on the open source dataset of [5]. For our experiments, we combine these three CNNs in pairs to create two different ensembles, fixing the big model to be $\mathrm{M}^{1.0}$ and changing the small model between the other two.

### B. Adaptive Inference for Visual Pose Estimation

Adaptive inference for regression problems must be formulated differently from the case of classification. Most adaptive policies rely on the class probabilities estimated by a softmax classifier function to compute "confidence" scores [6], [14]–[17]. This cannot be directly extended to a regressor, which produces a pointwise estimate of continuous variables (in our case, $x$, $y$, $z$, and $\phi$) without associated confidence. Given this limitation, we introduce ad-hoc policies to manage inference adaptively based on our specific problem's characteristics. Specifically, we introduce two novel policies, *Output-based Partitioning* and *Auxiliary Task-based Partitioning*, to realize adaptive inference for visual pose estimation on nano-drones.

Note that an *ideal* policy should execute the big model ($\mathrm{M}^{1.0}$) if and only if it provides a lower prediction error than the small model on the currently considered input. At the same time, the policy itself should have negligible computational complexity compared to the execution of the two models to avoid nullifying the gains.

*1) Output-based Partitioning:* The Output-based Partitioning (*OP*) policy is based on assessing changes in predictions across consecutive time steps. As illustrated in Fig. 1, at each new time stamp $t$, we first run the small model and monitor variations, compared to time $t-1$, in the predicted, min-max scaled (thus dimensionless) $x$, $y$, $z$, and $\phi$ values. Specifically, we compute:

$$O_{\text{sum}} = x + y + z + \phi, \quad OP_t = O_{\text{sum},t} - O_{\text{sum},t-1} \tag{1}$$

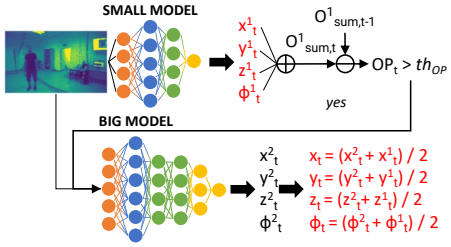


Fig. 1. OP policy: the small model is always executed to compute a first set of outputs, that are possibly corrected by the big model if they differ significantly from the prediction at the previous time stamp.

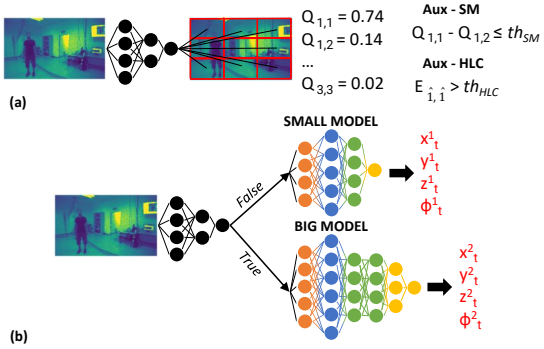


Fig. 2. Auxiliary task-based partitioning. (a) A first auxiliary CNN is executed to localize the head, and one of the two policies is applied. (b) Based on the policy outcome, one of the two models of the ensemble is executed.

A higher $OP_t$ score indicates that the relative position between human and drone is changing rapidly, necessitating the drone to change its location. This is associated with more challenging predictions due to the moving camera and the subject's distance from the center of the image. Conversely, a lower $OP_t$ signifies a slowly moving or stationary drone.

Based on these considerations, we build an adaptive system that works as follows: when $OP_t \leq th_{OP}$, the predictions of the small model are used "as is". Vice versa, the *big* network is executed when $OP_t > th_{OP}$, and the final outputs are computed as the *average* between the small and big model predictions, based on the observation that using both models concurrently, as an ensemble, leads to enhanced accuracy. Notably, $th_{OP}$ is a tunable threshold that can be set at runtime to manipulate the prediction error vs. complexity trade-off (lower $th$ implies more big model executions, and vice versa).

The critical limitation of this scheme is that the small model is always executed since its outputs are needed to compute $OP_t$. Numerically, the total cost is:

$$C = C_{small} + N_{big}(th_{OP}) \cdot C_{big} \tag{2}$$

where $C_{small}$/$C_{big}$ are the NMACs or latency of the small/big models, respectively, and $N_{big}(th_{OP}) = \sum_t \mathbb{1}(OP_t > th_{OP})$, with $\mathbb{1}(\cdot)$ being the indicator function, is the number of times the big model is called.

*2) Auxiliary Task-based Partitioning:* To avoid the constant execution of the small model, we propose an alternative policy

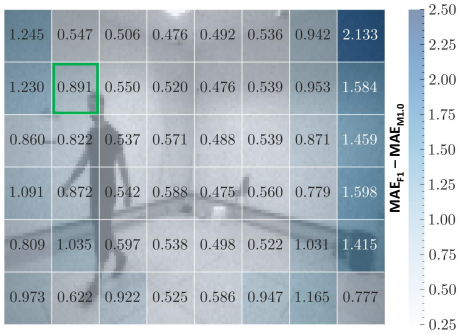


Fig. 3. 8×6 grid division of the input images. In each quadrant, the difference between the MAE of $F^1$ and $M^{1.0}$ is reported. A green square marks the quadrant to which the head of the person belongs.

that leverages an auxiliary classification task executed before the pose estimation. As shown in Fig. 2, we train a small additional CNN to locate the human head in one of the grid cells superimposed to the input image. We test grids of dimensions $2 \times 2$, $3 \times 3$, and $8 \times 6$.

We build the auxiliary CNN as a strongly reduced version of PULP-Frontnet, with additional stride-2 pooling layers to rapidly shrink the activation tensors. We start from 4 convolution+pooling blocks with 8, 16, 32, and 64 filters, respectively, and a final fully connected layer for just 656 kMACs. We then prune unimportant filters from this network using the mask-based method from [13].

Formulating head localization as a multi-class classification allows us to derive two adaptive inference policies. The first uses the SoA score margin method for confidence estimation (Aux-SM), whereas the second is based on domain knowledge and is called Head Localization-Class (Aux-HLC).

**Aux-SM** assumes that if the auxiliary CNN's confidence is low in classifying the head position, then the original pose regression task will be harder for a given input (e.g., because the image is blurry). We estimate the auxiliary task confidence with the score margin:

$$SM_t = \max(Q_{i,j,t}) - \text{second\_max}(Q_{i,j,t}) \tag{3}$$

where $Q_{i,j,t}$ represents the probabilities assigned to the $(i, j)$ grid cell for frame $t$. Higher confidence values indicate that the prediction is more reliable; therefore, the input image is easier, and pose estimation can be done with the small model. In this case, we define a tunable threshold $th_{SM}$ and invoke the big model if and only if $SM_t \leq th_{SM}$.

**Aux-HLC** assumes that pose estimation is harder when the head is in specific locations of the image (i.e., borders/corners vs center). To create a policy from this hypothesis, we rely on a validation set and create an error map, which indicates how much the big model outperforms the small one in each grid cell. Namely, each error map location contains $E_{i,j} = MAE_{small,i,j} - MAE_{big,i,j}$, that is, the average MAE difference computed on all validation samples for which the ground truth head location is in cell $(i, j)$. The policy invokes the big model if and only if $E_{\hat{i},\hat{j}} > th_{HLC}$, where $(\hat{i}, \hat{j})$ is the grid cell predicted by the auxiliary CNN at time $t$. Fig. 3 shows the error map for models $F^1$ and $M^{1.0}$. As shown, the difference increases at the edges and even more at the corners.

For both *Aux* policies, the total cost is:

$$C = C_{Aux} + (1 - N_{big}(th)) \cdot C_{small} + N_{big}(th) \cdot C_{big} \tag{4}$$

where $th \in \{th_{SM}, th_{HLC}\}$ and depends on the policy, and $C_{Aux}$ is the cost of the auxiliary CNN. Given that the latter is lower than $C_{small}$, it is evident that these policies incur a lower total cost compared to the OP policy when the big model is invoked frequently. However, with the auxiliary policies, the small and big model outputs are not ensembled for hard inputs, possibly leading to a higher error.

### C. Target Platform

The target robotic platform is the Crazyflie 2.1, a nano-drone weighting only 27 g produced by Bitcraze. The nano-drone mounts an AI-deck board featuring a camera and the GAP8 SoC by GreenWaves Technologies [18]. The camera acquires grayscale images up to 320x320 pixels, while the SoC extends the computational capabilities offered by the onboard STM32 processor for control purposes. GAP8 is a parallel ultra-low power platform [19] with hardware support for integer-only arithmetic. It features a single-core fabric controller (FC) and an 8-core cluster (CL). The FC orchestrates transfers between memories and offloads computationally intense tasks to the CL. The SoC also includes a 64 kB L1 memory shared between the CL cores and a 512 kB L2 memory within the FC. The off-chip memories consist of an 8 MB DRAM and a 64 MB FLASH. The transfers between DRAM and L2 are operated autonomously through a programmable micro-DMA. Transfers between L2 and L1 are performed by a DMA that is in the CL domain.

To deploy CNNs on this platform, we exploit two tools: PLiNIO [13] and DORY [20]. PLiNIO is a multi-optimization library that we leverage for shrinking the auxiliary CNN channels, as well as for applying Quantization-Aware Training (QAT) to all models to obtain integer-only implementations at 8-bit precision deployable on GAP8. The DORY compiler automatically converts the trained integer models from Python to C code, leveraging an optimized kernel library. It applies hardware-aware tiling and schedules asynchronous memory transfers between DRAM and L2, or L2 and L1, to orchestrate the execution of the entire network.

All CNN inferences are executed on GAP8 at a frequency of 170 MHz. Predictions are transferred via UART to the STM32 MCU, which hosts the other parts of the closed-control loop system. The control loop involves the same four tasks of [4]: (i) the CNN model pose estimation, the most computationally intensive task, (ii) a Kalman filter to smooth the sequences of poses, (iii) the velocity controller, and (iv) the low-level control for motor actuation and stabilization. Note that this paper focuses solely on the perceptual task, although it is expected that reducing its latency and error could also enhance the control loop's performance.

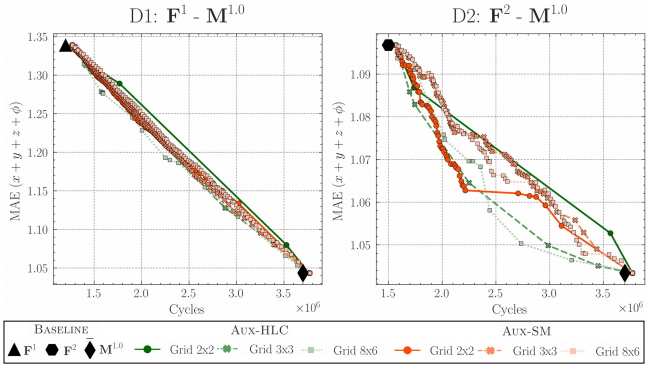


Fig. 4. Auxiliary task-based policies comparison on the Known dataset.

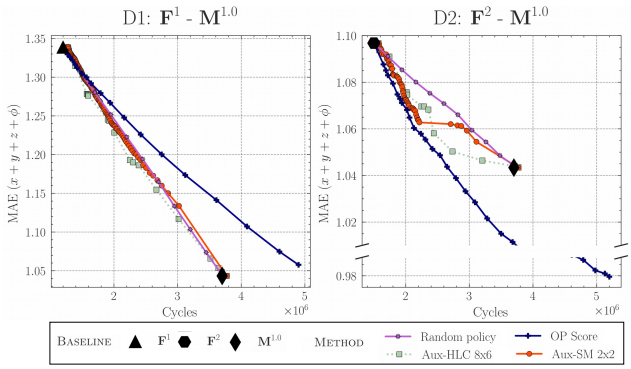


Fig. 5. OP and Aux policies comparison on the Known dataset.

## IV. Experimental Results

### A. Setup

For our experiments, we employed the dataset introduced in [5], which served as the benchmark for evaluating the performance of the reference static models. It comprises a total of 30.3k images, divided into training, validation, and test sets in the proportions of 70%, 20%, and 10%, respectively. We used the validation set to select the most accurate pruned auxiliary CNN and build the error map for Aux-HLC, whereas all results are reported on the test set. To show the generalization of our method, we also test on a second dataset introduced in [5]. Similarly to the SoA one, this dataset is collected with a real nano-drone in a different laboratory environment and with different subjects. This dataset comprises 45k images divided into 72% for training, 18% for validation, and 10% for testing. We will henceforth refer to these two datasets as the "Known Dataset" and the "Unseen Dataset."

Our comparison baselines are the three static CNNs ($F^1$, $F^2$, and $M^{1.0}$) since they constitute the SoA for this task. In the following sections, we refer to the adaptive solution that employs $F^1$ as the small model as D1 and the one that utilizes $F^2$ as D2.

### B. Auxiliary Task Exploration

Fig. 4 presents a comprehensive comparison of the auxiliary task policies, considering both CNN pairs and different grid dimensions. We did not consider finer grids because doing so would compromise the network's ability to predict the correct quadrant, resulting in nearly random outputs. The different policies are compared in terms of total MAE on the four regressed variables versus the average number of clock cycles per inference on GAP8, over the Known dataset's test set. Different points for the same policy refer to different settings of the respective tunable thresholds ($th_{SM}$ and $th_{HLC}$).

While all policies perform similarly on D1, the best results are obtained with Aux-HLC, using an $8 \times 6$ grid. For D2, the top-2 policies are Aux-HLC with an $8 \times 6$ grid and Aux-SM with a $2 \times 2$ grid, with Aux-HLC achieving optimal performance in lower MAE regions and Aux-SM performing better in the low cycles regime. For instance, with the Aux-HLC policy, we can obtain a MAE of 1.05, which is only 0.57% higher than the error of the big model, while simultaneously reducing the inference cycles by 26.07%. In the following sections, we will only consider these two optimal combinations of type and grid size for the Aux policies results.

### C. Output vs. Auxiliary Task Policies

In this section, we compare Aux and OP policies on the Known dataset. As a baseline for our analysis, we consider a Random policy, that is, a "trivial" dynamic model in which a zero-cost policy randomly selects between the execution of the small or the big model. Fig. 5 reports the comparison results for the two adaptive systems, revealing two distinct trends.

For D1, the most effective policy is Aux-HLC. Interestingly, the OP policy performs worse than the random one in this case. This can be ascribed to the low accuracy of the small model, which causes a misalignment between the $OP_t$ score and the actual need to invoke the big model. The decision to employ the big model becomes then somewhat random. This, together with the high cost incurred by the OP policy when the big model is invoked frequently (both models executed for most frames), yields a higher latency without accuracy benefits.

Conversely, for D2, OP significantly outperforms the other policies, resulting in the attainment of new SoA results. At iso-MAE with the static big model, we observe a substantial reduction of 28.03% in inference cycles, whereas at iso-latency, we reduce the MAE by 3.15%. Finally, our approach yields the best known MAE (0.98) for this task, surpassing the current state of the art by 6.13%.

### D. Crazyflie Deployment

In Tab. II, we present a detailed breakdown of some of the Pareto solutions from Fig. 5 deployed on the Crazyflie 2.1. We select the $th_*$ value that maximizes the latency benefit from adaptive inference for each ensemble, comparing the corresponding solutions with the Random policy at iso-MAE.

For the D1 ensemble, the most substantial gain is achieved by the Aux-HLC policy, invoking the big model for 39.1% of the predictions. In comparison to the Random Policy, we achieve reductions in latency and energy consumption of 8.1% and 8.8%, respectively. Furthermore, compared to exclusively running the big model, at the cost of a slightly higher total

TABLE II
DEPLOYMENT OF DIFFERENT DYNAMIC MODELS AND POLICIES ON THE CRAZYFLIE 2.1.

| Models | Method | MAE | Latency | % Big | Energy | Memory |
|---|---|---|---|---|---|---|
| $\mathbf{F}^1$ | Static | 1.34 | 7.06 ms | 0 | 0.57 mJ | 153 kB |
| $\mathbf{F}^2$ | Static | 1.10 | 8.82 ms | 0 | 0.71 mJ | 183 kB |
| $\mathbf{M}^{1.0}$ | Static | 1.04 | 21.76 ms | 100 | 1.92 mJ | 235 kB |
| D1 | Random | 1.19 | 14.41 ms | 50.0 | 1.25 mJ | 250 kB |
| D1 | Aux-HLC 8x6 | 1.19 | 13.24 ms | 39.1 | 1.14 mJ | 289 kB |
| D2 | Random | 1.04 | 21.76 ms | 100 | 1.92 mJ | 280 kB |
| D2 | OP | 1.04 | 15.66 ms | 31.4 | 1.32 mJ | 280 kB |

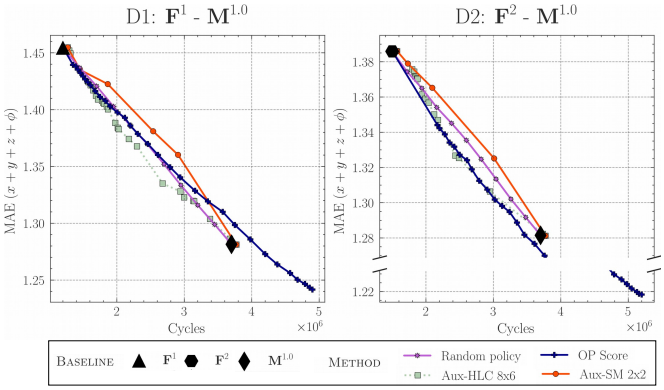


Fig. 6. Comparative analysis of the proposed adaptive policies on the Unseen dataset.

MAE (+0.15), we reduce the latency and energy consumption by 39.1% and 40.6%.

For the D2 ensemble, the OP policy significantly outperforms the Random policy. The most interesting gains emerge when the adaptive approach is compared to the static big model. Indeed, running the big model only 31.4% of the time with our proposed adaptive system is sufficient to maintain the same MAE while concurrently achieving reductions in latency and energy consumption of 28.03% and 31.25%, respectively.

Clearly, all adaptive systems incur a memory overhead since 2 or 3 (in the case of Aux policies) CNNs are deployed on the nano-drone instead of one. However, as shown in Tab. II, even the largest adaptive ensemble easily fits the 512 kB L2 memory of GAP8. Note that the memory results in the table are computed as the sum of the weights of all the deployed networks and of the biggest activation buffer required to run inference, making the total memory for D1 and D2 less than the sum of the composing static models.

### E. Unseen Dataset Results

Finally, Fig. 6 shows the generality of our policies by presenting results on the Unseen dataset. The same conclusions drawn from our analysis on the Known Dataset remain valid. For the D1 ensemble, the best-performing algorithm is Aux-HLC, which achieves the maximum latency reduction (9.2%) compared to the Random Policy baseline at a MAE of 1.33. In contrast, for the D2 ensemble, the OP continues to be superior. Two significant results it achieves are: (i) the best overall MAE of 1.22, improving the SoA by 4.9%, and (ii) a 6.49% latency reduction at iso-MAE with the big model.

## V. CONCLUSIONS

We have presented a practical use case of adaptive inference for perception tasks aboard nano-drones, focusing on visual human pose estimation. We have derived two adaptive systems from three SoA compact CNNs and proposed and compared different optimal adaptive policies in different working conditions. These policies have reduced the previous SoA error on the total regressive loss for human pose estimation on nano-drones obtained in [5]. On the other hand, fixing the desired MAE to 1.04 (the best result from [5]) and considering the same hardware platform, we achieved a lower latency (15.66 ms) within a power envelope of 90 mW. Additionally, as shown in Fig. 5 and 6, adaptive inference allows to obtain a rich set of intermediate Pareto-optimal solutions in the latency vs. MAE space, selectable at runtime by changing the value of a single tunable threshold. Future works will include the exploration of more advanced adaptive inference techniques.

## REFERENCES


[1] D. Palossi *et al.*, "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs," in *International Conference on DCOSS*, 2019.

[2] B. Varghese *et al.*, "Challenges and opportunities in edge computing," in *IEEE International Conference on Smart Cloud*, 2016.

[3] L. Lamberti *et al.*, "Tiny-pulp-dronets: Squeezing neural networks for faster and lighter inference on multi-tasking autonomous nano-drones," in *IEEE International Conference on AICAS*, 2022.

[4] D. Palossi *et al.*, "Fully onboard ai-powered human-drone pose estimation on ultralow-power autonomous flying nano-uavs," *IEEE IOTJ*, 2022.

[5] E. Cereda *et al.*, "Deep neural network architecture search for accurate visual pose estimation aboard nano-uavs," in *IEEE ICRA*, 2023.

[6] E. Park *et al.*, "Big/little deep neural network for ultra low power inference," in *International Conference on CODES+ISSS*, 2015.

[7] X. Sun *et al.*, "Integral human pose regression," in *IEEE ECCV*, 2018.

[8] D. C. Luvizon *et al.*, "2d/3d pose estimation and action recognition using multitask deep learning," in *IEEE CVPR*, 2018.

[9] R. A. Güler *et al.*, "Densepose: Dense human pose estimation in the wild," in *IEEE CVPR*, 2018.

[10] G. Shi *et al.*, "Neural-swarm: Decentralized close-proximity multirotor control using learned interactions," in *IEEE ICRA*, 2020.

[11] W. Zhao *et al.*, "Learning-based bias correction for time difference of arrival ultra-wideband localization of resource-constrained mobile robots," *IEEE Robotics and Automation Letters*, 2021.

[12] V. Sze *et al.*, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, 2017.

[13] D. J. Pagliari *et al.*, "PLiNIO: A user-friendly library of gradient-based methods for complexity-aware dnn optimization," in *Forum on Specification & Design Languages (FDL)*, 2023.

[14] F. Daghero *et al.*, "Energy-efficient deep learning inference on edge devices," in *Hardware Accelerator Systems for Artificial Intelligence and Machine Learning*. Elsevier, 2021.

[15] H. Tann *et al.*, "Runtime configurable deep neural networks for energy-accuracy trade-off," in *IEEE CODES*, 2016.

[16] F. Daghero *et al.*, "Human activity recognition on microcontrollers with quantized and adaptive deep neural networks," *ACM Trans. Embed. Comput. Syst.*, 2022.

[17] S. Teerapittayanon *et al.*, "BranchyNet: Fast inference via early exiting from deep neural networks," in *ICPR*, 2016.

[18] E. Flamand *et al.*, "Gap-8: A risc-v soc for ai at the edge of the iot," in *IEEE Int. Conf. on ASAP*, 2018.

[19] F. Conti *et al.*, "An iot endpoint system-on-chip for secure and energy-efficient near-sensor analytics," *IEEE TCAS-I*, 2017.

[20] A. Burrello *et al.*, "Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus," *IEEE Trans. Comput.*, 2021.