POLITECNICO DI TORINO
Repository ISTITUZIONALE

Optimized Deployment of Deep Neural Networks for Visual Pose Estimation on Nano-drones

Original
Optimized Deployment of Deep Neural Networks for Visual Pose Estimation on Nano-drones / Risso, M., Daghero, F., Motetti, B.A., JAHIER PAGLIARI, D., Macii, E., Poncino, M., Burrello, A.. - ELETTRONICO. - 32:(2024), pp. 304-309. (ERF 2024 | European Robotics Forum Rimini (IT) March, 13 - 15, 2024) [10.1007/978-3-031-76424-0_54].

Availability:
This version is available at: 11583/2992684 since: 2025-04-10T07:29:00Z

Publisher:
Springer

Published
DOI:10.1007/978-3-031-76424-0_54

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

arXiv:2402.15273v1 [cs.CV] 23 Feb 2024

Optimized Deployment of Deep Neural Networks for Visual Pose Estimation on Nano-drones

Matteo Risso[1], Francesco Daghero[2], Beatrice Alessandra Motetti[1], Daniele Jahier Pagliari[1], Enrico Macii[2], Massimo Poncino[1], and Alessio Burrello[2]

[1] Department of Control and Computer Engineering,
[2] Interuniversity Department of Regional and Urban Studies and Planning,
Politecnico di Torino, Turin 10129, Italy,
first_name.first_surname@polito.it

Abstract. Miniaturized autonomous unmanned aerial vehicles (UAVs) are gaining popularity due to their small size, enabling new tasks such as indoor navigation or people monitoring. Nonetheless, their size and simple electronics pose severe challenges in implementing advanced on-board intelligence. This work proposes a new automatic optimization pipeline for visual pose estimation tasks using Deep Neural Networks (DNNs). The pipeline leverages two different Neural Architecture Search (NAS) algorithms to pursue a vast complexity-driven exploration in the DNNs' architectural space. The obtained networks are then deployed on an off-the-shelf nano-drone equipped with a parallel ultra-low power System-on-Chip leveraging a set of novel software kernels for the efficient fused execution of critical DNN layer sequences. Our results improve the state-of-the-art reducing inference latency by up to 3.22× at iso-error.

Keywords: Nano-drones, TinyML, NAS, CNN, Fused Layers

1 Introduction and Related Works

Nano-sized unmanned aerial vehicles (UAVs), commonly named "nano-drones", are increasingly used for navigation in GPS-denied environments and to operate near humans, thanks to their small size (sub-10 cm) and low weight (sub-40 g). However, their limited on-board computational and memory capacity pose substantial challenges to achieving full autonomy. In particular, they make it impossible to utilize large deep learning models for perception [10, 11]. A particularly relevant task for nano-UAVs is *human pose estimation* [11], which enables applications such as "people monitoring" and "follow-me". Recent research on this task has concentrated on optimizing tiny Convolutional Neural Networks (CNNs) to operate within tight nano-UAVs hardware constraints, marking a significant stride towards obtaining reasonable perceptual performance on such platforms [3, 9, 11]. The work of [3], in particular, underscored the critical role of automated optimization methods such as *Neural Architecture Search* (NAS) in

This paper has been accepted for publication in the ERF 2024 conference.

facilitating the design of efficient architectures that balance computational efficiency with task performance. However, [3] only scratched the surface of a potentially vast research direction, applying a NAS algorithm aimed at shrinking an input CNN architecture (called "seed") through the elimination of unimportant feature maps [12], closely resembling *structured pruning*. Other NAS methods allow exploring broader (yet less fine-grained) search spaces, e.g., by selecting between different alternatives for each layer of the CNN [2,7,8,13]. In this work, we show that the sequential application of these two families of NASs (layer selection and model shrinking) can yield superior results in terms of the pose estimation accuracy versus latency and memory trade-offs.

In addition to an optimized network architecture, another key component to enable real-time CNN-based perception on nano-UAVs is the availability of efficient low-level software kernels, fully exploiting the hardware available onboard. To this end, this work proposes the usage of optimized kernels for the execution of *fused DepthWise (DW) and PointWise (PW) convolution* on the Parallel Ultra Low-Power (PULP) multi-core clusters available in recent nano-UAV platforms [10]. Sequences of DW and PW layers are common in tiny CNNs (inspired by MobileNets [6]), and fusing them significantly reduces the amount of intermediate memory transfers, thus improving end-to-end latency compared to single-layer kernels for the same hardware, such as the ones in [5].

Through the combined optimization of a CNN architecture (with two chained NAS steps) and of the corresponding inference software stack, we outperform the state-of-the-art (SoTA) on human pose estimation for nano-UAV-class devices [3,9], obtaining up to 13.78% lower Mean Absolute Error (MAE), or reducing latency by up to 3.22× at iso-error. Our work highlights the key importance of multi-level automated deployment flows for tinyML on tiny drones.

2 Materials and Methods

Target Platform: We focus on the Bitcraze Crazyflie 2.1 nanodrone equipped with a greyscale camera and the GAP8 SoC [4]. GAP8 comprises a single-core fabric controller (FC) and an 8-core PULP cluster (CL). The FC orchestrates memory transfers and delegates demanding computations to the CL. CL cores share a 64 kB L1 memory and the SoC includes a 512 kB L2 memory. A Direct Memory Access (DMA) unit handles transfers between the two memories.

Complexity-driven Architecture Search: Fig. 1 (left) shows the proposed architecture optimization flow, which combines two SotA NASs, *Supernet* [8] and *PIT* [12], using the implementations of the PLiNIO open-source library [7]. Both are so-called Differentiable NASs (DNASs), i.e., they jointly train (with gradient descent) the standard weights of the network W and some additional parameters θ, which control architectural choices such as the type of layer (for *Supernet*) or the number of output features of each layer (for *PIT*). The red box of Fig. 1 shows the loss function minimized by both NASs, where $\mathcal{L}$ is the task-specific loss (e.g., Mean Squared Error) and $\mathcal{C}$ is an added complexity term such as the

number of parameters or operations of the network, as a function of θ. The balance between the two is controlled by the scalar λ. In particular, we used the size complexity defined in [12].

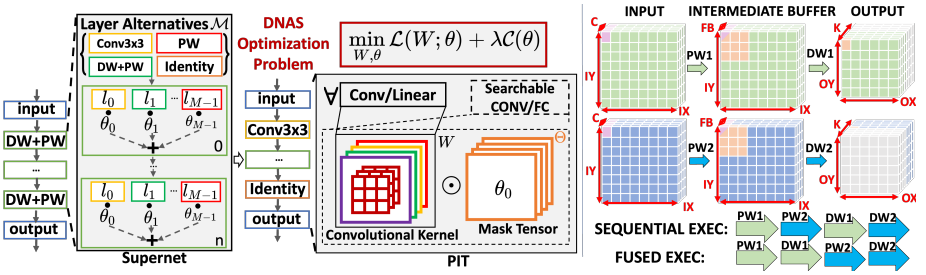


Fig. 1: NAS-based optimization flow (left); Optimized PW+DW kernel (right).

We use a MobileNetV1 architecture as blueprint for the SuperNet step, since it demonstrated SotA performance on human-to-nanodrone pose estimation [3, 9]. For each DW+PW block of the original network, the NAS selects between: i) The original block; ii) A single PW layer; iii) A standard 2DConv with 3×3 filter; iv) A "no-operation" to optionally skip the block. As depicted in Fig. 1, each of the $\mathcal{M} = 4$ alternatives is coupled with a θ_i parameter. At the end of training, the alternative associated with the largest θ_i is selected. The architectures obtained with the Supernet are further optimized with PIT, which applies a fine-grained structured pruning, eliminating unimportant output channels from each layer. Namely, the weight tensor of each Conv or linear layer is paired with a set of binary trainable masks θ, which are trained to control whether a specific feature is removed ($\theta_i = 0$) or kept ($\theta_i = 1$).

Fused Kernel for efficient inference: DW layers are notoriously difficult to accelerate due to their more limited data reuse options compared to standard 2D Conv. PULP-NN [5], a SoTA open-source kernel library for GAP8, handles this by changing the input data layout from Height-Width-Channels (HWC) to Channels-Height-Width (CHW). However, this causes an increase in data transfers, as the re-ordering operation has to be repeated multiple times when tiling the layers (i.e., loading parts of the data in L1 memory before performing computations) [1]. To reduce this overhead, we implement a new fused kernel, computing a PW+DW sequence entirely in L1, as shown in Fig. 1 (right). We accelerate PW+DW (and not DW+PW) sequences, as this enables exploiting the independence of DW computations across input/output channels (C/K). Specifically, we compute a subset (FB) of the PW output channels storing them in an additional L1 buffer of size IX*IY*FB (IX/IY = input rows/columns); then, we execute the DW operation on such buffer. The whole process is repeated until all K output channels have been produced. Since the DW is parallelized over channels, we set $FB = 8$, i.e., equal to the cores of GAP8, to maximize utilization with the minimum possible L1 memory overhead.

3 Experimental Results

We train and test our CNNs on the dataset introduced in [11], and with the same data splits of [9] Fig. 2 shows the results obtained with the cascaded application of the Supernet and PIT NASs compared with the SotA networks (red circles) of [9] in the N. of Parameters vs. MAE plane. All the results are uniformly quantized to INT8 data format using [7]. We use a slightly modified version of the test set, as in [9], where the labels have been adjusted to compensate for a calibration inaccuracy of the measurement tools detected through manual inspection.

We apply the SuperNet NAS using two different MobileNetV1 variants as blueprints: a standard one (denoted as *Large*) and one with a width-multiplier of 0.25× (*Small*). The orange and blue triangles of Fig. 2 denote two promising output architectures obtained with this first NAS step. In particular, we selected the blue triangle (SuperNet Small) as the smallest network achieving a MAE < 1. Conversely, we selected the orange one (SuperNet Large) as the network achieving the lowest MAE. The found architectures details are as follows: SuperNet Large substitutes the first three DW+PW layers of the vanilla MobileNetV1 with standard 2DConv layers. Conversely, SuperNet Small substitutes only the first DW+PW block with a 2DConv, while skipping entirely the last DW+PW block.

We then apply PIT to both these models, obtaining the rich collection of Pareto-optimal architectures, depicted in Fig. 2 with blue and orange stars. Noteworthy, our smallest model achieves a 9× size compression at iso-MAE w.r.t. the most accurate SotA model [9]. Conversely, in the large N. of Parameters regime, our solutions can improve the SotA by up to 13.78%. Note that these results showcase the benefits of the combined usage of both NASs, given that SotA networks have been obtained using PIT only [3].

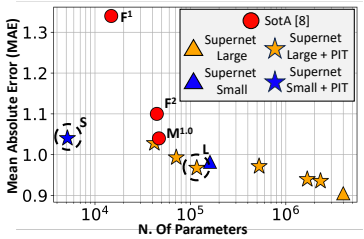


Fig. 2: Optimized architectures vs SotA from [9].

Model	Ker	MAE	Lat[ms]	Mem[kB]
SoTA				
F^1	U	1.34	7.06	14.8
F^2	U	1.1	8.82	44.5
M^1	U	1.04	21.76	46.8
Ours				
S	U	1.04	7.4	5.14
	F		6.76	
L	U	0.97	32.5	116.63
	F		31.43	

Table 1: Deployment results.

We deploy models on GAP8 mirroring the setup of [9]. Table 1 shows the deployment results of selected CNNs from Fig. 2 (those labeled with a letter) from our work and the SoTA. We deploy the smallest (S) and most accurate (L) CNNs found by our NAS chain, excluding models that do not fit the L2 memory of GAP8 (512 kB). For each model, we report the test set MAE, the weight memory footprint (Mem), and the latency (Lat). We also report the kernel used for all PW+DW sequences (Ker), which is either our proposed fused implementation (F) or an unfused one (U) from vanilla PULP-NN.

When considering the S model, we achieve the same MAE at far lower latency (-65%) w.r.t. the most accurate SoTA model (M1). The speedup increases to

68.1% (a further 8.6% reduction) when utilizing our fused PW+DW kernel. If we compare the same model to the least accurate SoTA one (F1), we achieve significantly better MAE (-0.3) with more than 30× fewer parameters. When using fused kernels, we also still reduce latency by 4.2%. The L model, on the other hand, outperforms the most accurate SoTA CNN in terms of MAE (6.98% reduction). Fused kernels are less beneficial for this model, as most of the latency is due to the initial standard convolutional layers. Nonetheless, they still grant a latency reduction of 3.27% compared to a standard deployment. Noteworthy, as shown in [3], reducing perception latency (i.e., improving the frames per second that the model can process) has been shown very beneficial to improve the performance of nano-drones control loops.

4 Conclusions

We have presented a multi-stage, fully-automated optimization pipeline for visual human-to-nanodrone pose estimation, including two chained NAS methods, respectively for layer selection and model pruning, and an optimized implementation of fused PW and DW convolutions to improve CNN inference latency, and consequently the drone's controller reaction time, by cutting intermediate memory transfers. Overall, our work serves to demonstrate the fundamental importance of deployment optimization pipelines for TinyML on tiny drones.

References

1. Burrello, A., et al.: Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus. IEEE Transactions on Computers pp. 1253–1268 (2021)
2. Burrello, A., et al.: Enhancing neural architecture search with multiple hardware constraints for deep learning model deployment on tiny iot devices. IEEE Transactions on Emerging Topics in Computing pp. 1–15 (2023)
3. Cereda, E., et al.: Deep neural network architecture search for accurate visual pose estimation aboard nano-uavs. In: IEEE ICRA (2023)
4. Flamand, E., et al.: Gap-8: A risc-v soc for ai at the edge of the iot. In: IEEE 29th ASAP, pp. 1–4 (2018)
5. Garofalo, A., et al.: Pulp-nn: A computing library for quantized neural network inference at the edge on risc-v based parallel ultra low power clusters. In: 26th IEEE ICECS, pp. 33–36. IEEE (2019)
6. Howard, A.G., et al.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
7. Jahier Pagliari, D., et al.: Plinio: A user-friendly library of gradient-based methods for complexity-aware dnn optimization. In: FDL, pp. 1–8 (2023)
8. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
9. Motetti, B.A., et al.: Adaptive Deep Learning for Efficient Visual Pose Estimation aboard Ultra-low-power Nano-drones. arXiv:2401.15236 (2024)
10. Palossi, D., et al.: An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs. In: 15th DCOSS, pp. 604–611 (2019)

11. Palossi, D., et al.: Fully onboard ai-powered human-drone pose estimation on ultralow-power autonomous flying nano-uavs. IEEE IoTJ **9**(3), 1913–1929 (2022)
12. Risso, M., et al.: Lightweight neural architecture search for temporal convolutional networks at the edge. IEEE Trans. Comp. (2022)
13. Tan, M., et al.: Mnasnet: Platform-aware neural architecture search for mobile. In: IEEE/CVF CVPR, pp. 2820–2828 (2019)