This paper has been accepted for publication in the IEEE DCOSS 2021 conference

Improving the Generalization Capability of DNNs for Ultra-low Power Autonomous Nano-UAVs

Elia Cereda*, Marco Ferri*, Dario Mantegazza*, Nicky Zimmerman†, Luca M. Gambardella*, Jérôme Guzzi*, Alessandro Giusti*, and Daniele Palossi*‡
* Dalle Molle Institute for Artificial Intelligence – USI and SUPSI, Switzerland
† Photogrammetry Laboratory – University of Bonn, Germany
‡ Integrated Systems Laboratory – ETH Zürich, Switzerland
name.surname@idsia.ch

***Abstract*—Deep neural networks (DNNs) are becoming the first-class solution for autonomous unmanned aerial vehicles (UAVs) applications, especially for tiny, resource-constrained, nano-UAVs, with a few tens of grams in weight and sub-ten centimeters in diameter. DNN visual pipelines have been proven capable of delivering high intelligence aboard nano-UAVs, efficiently exploiting novel multi-core microcontroller units. However, one severe limitation of this class of solutions is the generalization challenge, i.e., the visual cues learned on the specific training domain hardly predict with the same accuracy on different ones. Ultimately, it results in very limited applicability of State-of-the-Art (SoA) autonomous navigation DNNs outside controlled environments. In this work, we tackle this problem in the context of the human pose estimation task with a SoA vision-based DNN [1]. We propose a novel methodology that leverages synthetic domain randomization by applying a simple but effective image background replacement technique to augment our training dataset. Our results demonstrate how the augmentation forces the learning process to focus on what matters most: the pose of the human subject. Our approach reduces the DNN's mean square error — vs. a non-augmented baseline — by up to 40%, on a never-seen-before testing environment. Since our methodology tackles the DNN's training stage, the improved generalization capabilities come at zero-cost for the computational/memory burdens aboard the nano-UAV.**

***Index Terms*—Deep Neural Network, Domain Generalization, Autonomous UAVs, Nano-drones**

I. Introduction

Palm-sized autonomous unmanned aerial vehicles (UAVs) are rapidly evolving, getting even smarter and faster thanks to end-to-end deep neural network-based (DNN) algorithms running directly on their limited onboard processors [1]–[3]. With their agility and reduced form factor, i.e., a few tens of grams in weight and sub-10 cm diameter, these robotic platforms (also called nano-UAVs) are ideal candidates for exploring extremely narrow environments and safely operating in humans' surroundings. As an example, DNN-powered autonomous nano-UAVs have been proven able to explore indoor environments [2], avoiding collisions with dynamic obstacles, flying at high-speed [3], and precisely following human subjects [1]. However, to date, one significant limitation of this robotics vision-perception class of solutions is the lack of generalization capabilities; i.e., the visual cues learned from the DNN's training domain result in a model that hardly predicts with the same accuracy in different environments. Solving this challenging problem would ultimately lead to broader adoption of these versatile robotic platforms, making them operate outside of controlled environments.

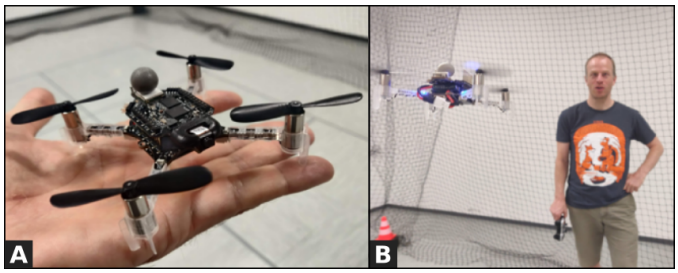

Fig. 1. The palm-sized robotic platform (A) we envision for the deployment of our work, performing the human-pose estimation task (B).

Enabling full autonomous navigation on nano-sized UAVs is an ambitious goal, as they can not rely on any remote computation or off-board aids. Therefore, real-time requirements, e.g., DNNs' inference throughput, must be matched using minimal computational/memory resources due to the stringent onboard power budget and payload. Since nano-UAVs have a total power of a few Watts, of which only $5-15\%$ is available for onboard processing [4], this results in a few hundreds of mW, at most, for the onboard computation. For this reason, the typical processing device these tiny UAVs can host aboard is relegated to ultra-low-power (ULP) microcontroller units (MCUs). Given this challenging scenario, our search for improved generalization capabilities of DNNs must aim at methodologies that minimize the onboard demand for additional computational power and memory.

In this work, we tackle the task of human-drone pose estimation, training a lightweight DNN for visually estimating the relative pose of a person from a low-resolution image acquired by a nano-UAV flying nearby. A standard workflow for coping with this task [1], [5] involves: *i*) acquiring training sequences in a room equipped with a motion capture (mocap)

This work has been partially supported by the Secure Systems Research Center (SSRC) of the UAE Technology Innovation Institute (TII), by the Swiss National Science Foundation (SNSF) through the NCCR Robotics, and by the European Union's Horizon 2020 Research and Innovation Programs under Grant No. 871743 (1-SWARM).

system which tracks the absolute pose of both the drone and the subject; *ii*) building a set of training instances, each consisting in a camera frame and the corresponding ground truth pose of the subject relative to the drone; *iii*) training a regression model that, given an input image, estimates the components of the relative pose. Such workflow perfectly works when the training domain and the deployment one match [1], [5]. However, the poor variability of the training data — always collected in the same mocap-equipped room — does not promote the DNN's generalization ability, limiting its effectiveness in never-seen-before deployment environments.

This problem can be mitigated at the price of the additional complexity of the onboard intelligence, such as by augmenting the in-mission inference with methods of *uncertainty estimation* [6]. This family of solutions improves the system's generalization capability, for example, by assessing the level of confidence on the prediction of an ensemble of DNN models, speculating multiple predictions via on-the-fly data augmentation of the input image [7]. Despite the proven efficacy of these techniques, their price in additional in-mission computation and memory is a hard match for the available resources aboard nano-UAVs.

A second strategy to improve the DNN's generalization capability, which is widely used in traditional computer vision and machine vision tasks (e.g., object detection/classification), relies upon increasing the number and the variability of environments represented in the training samples. Thanks to relatively simple labels and the massive availability of *in-the-wild* images/labels, this approach has shown great potential when abundant public datasets [8], [9] are available. However, this solution hardly applies to robotics tasks, such as ours, as they require precise physical-world labels that cannot be collected without expensive ad-hoc infrastructure, such as a mm-accurate mocap system.

One common approach to overcome this limitation consists in generating training datasets in simulation [10], which enables the acquisition of huge amounts of data with known ground truth. In this case, one can leverage *domain randomization* techniques [11], which promote generalization by programmatically randomizing various aspects of the simulated environments. Nevertheless, learning exclusively from simulated data suffers from the well-known *simulation-to-reality* gap [12], with no guarantee that the resulting models will work in the real world. Moreover, either with real-world training data or a simulated one, the need for expensive ad-hoc infrastructure — the same used for acquiring the real-world data — is unavoidable for precise and quantitative final evaluation.

The main contribution of this work is a novel methodology of dataset augmentation to improve the generalization capabilities of a state-of-the-art (SoA) DNN — named PULP-Frontnet — for human pose estimation on nano-UAVs [1]. This DNN has been proven to enable autonomous navigation within the limited computational power aboard a Crazyflie 2.1 nano-quadrotor[1] extended with a commercial ULP multi-core System-on-Chip (i.e., the GreenWaves Technologies GAP8[2]), making the nano-UAVs capable of following a free-moving person. In this work, we improve the PULP-Frontnet generalization ability, adopting a novel augmentation methodology inspired by *domain randomization* [11]. The proposed methodology hinges on: *i*) segmenting the images of the PULP-Frontnet training dataset, to obtain a binary mask of pixels belonging to the closest person to the camera; *ii*) employing such a mask to substitute the image background with a random one from a large collection [9], resulting in a new augmented dataset which we use to retrain the PULP-Frontnet DNN.

Our results demonstrate improved generalization performance of the augmented DNN, w.r.t. the original PULP-Frontnet, when predicting human poses on a never-seen-before test dataset, collected in a different place from the one used for training. Our model shows a reduced mean squared error (MSE), compared to the PULP-Frontnet baseline, with a peak reduction of almost 40%. Similarly, the R^2 metric on the DNN's output improves from 0.09 to 0.45. Ultimately, since our approach exclusively focuses on the training phase of the DNN, the improvements on the model's generalization come at *zero cost* on the final nano-UAV, maintaining the same real-time performance of the original PULP-Frontnet, i.e., $\sim$20-$48\,\mathrm{frame/s}$ within $\sim$25-$96\,\mathrm{mW}$.

The rest of the paper is structured as follows: Section II introduces the SoA in both autonomous navigation for nano-UAVs and DNN generalization; Section III presents the proposed method and its integration in our data augmentation pipeline; Section IV shows our experimental evaluation; finally, Section V concludes the paper.

II. Related Work

Methodologies based on deep learning have become the leading approach for solving autonomous navigation tasks on nano-UAVs [1]–[3], [13]–[15], as the computational/memory requirements of more traditional approaches, such as the *localization-mapping-planning* cycle [16], would result unaffordable for the onboard processing power of this class of vehicles. When autonomous navigation capabilities come to the nano-size class of UAVs, there are three main categories of solutions: *i*) offloading the computation to some external power-unconstrained base-station [13], [17]; *ii*) reducing the onboard workload's complexity to minimal functionalities [14]; *iii*) extending the onboard brain either through application-specific processors [18] or general-purpose ULP heterogeneous multi-core SoCs [1]–[3].

Relying on powerful off-board computers is a proven strategy to demonstrate advanced capabilities in UAVs of all sizes, and in particular on resource-constrained nano-UAVs [13], [17]. For example, obstacle avoidance has been demonstrated on nano-UAVs, taking advantage of vision-based DNNs running on wireless-connected commodity laptops [13].

[1]https://www.bitcraze.io/products/crazyflie-2-1

[2]https://greenwaves-technologies.com/gap8_gap9

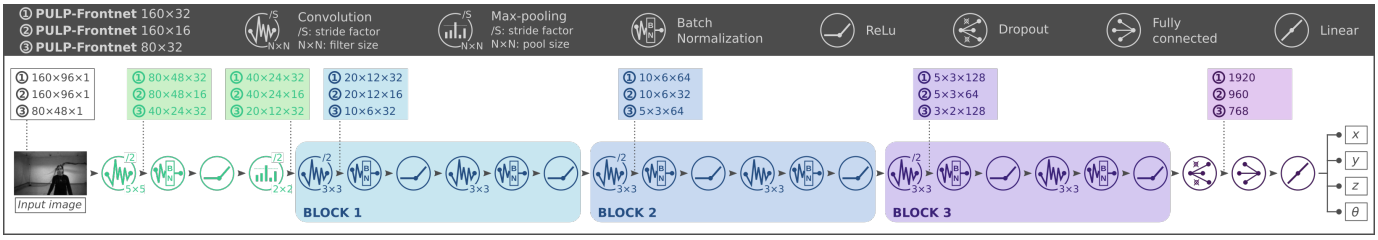

Fig. 2. Architecture of the original PULP-Frontnet convolutional neural network [1].

Similarly, in [17], a fuzzy logic position controller and vision-based position estimator have been demonstrated on a nano-drone, offloading all the intensive computation to a powerful Intel i7 processor via radio streaming of the images acquired on board. However, this category of solutions shows multiple drawbacks on the communication channel, such as latency, noise, and security vulnerabilities, but also on the UAV's power budget, with the power consumption overhead for the radio streaming [19].

Given the typical computational unit available aboard a commercial-off-the-shelf nano-UAV, many solutions pay a severe price on the system's functionalities due to the absence of sufficient computing power, as the onboard device is limited to simple single-core MCUs [14], [20]. In [14], a DNN is proposed to improve the localization accuracy on the z-axis of a nano-drone swarm utilizing sensor biases from a localization infrastructure. Despite the effectiveness of the proposed solution, the onboard processing device (i.e., ARM Cortex M4) relegates the workload to a simple DNN (i.e., $\sim$27 kMAC per inference), which leads to a prediction capability of the only z-component of the drone's pose. Similarly, also in [20], the visual DNN proposed for improving the localization accuracy of a nano-drone runs on the onboard ARM Cortex M4 MCU, which restricts the computational burden to $2.7\,\mathrm{kMAC/frame}$ to run in real-time (200 Hz). Compared to the complexity of the vision-based DNN we refer to in this work [1], i.e., $14.1\,\mathrm{MMAC/frame}$, the DNNs in [14] and in [20] require, respectively, three and four orders of magnitude fewer operations per inference.

The last class of solutions for enabling complex autonomous navigation algorithms on nano-sized UAVs is embodied by the extension of the onboard computational unit either with application-specific integrated circuits (ASICs) [21] or adopting general-purpose ULP multi-core SoCs [1]–[3]. Co-design of custom hardware and algorithms can enable significant breakthroughs, such as in [21] where a high complexity visual simultaneous localization and mapping (SLAM) task is addressed in a highly tight power envelope (24 mW). However, ASIC designs are costly and feature fixed/limited autonomous navigation functionalities, handling only part of the overall system's needs. Therefore, they can only be employed as a co-processor of less energy-efficient devices. General-purpose ULP multi-core SoCs, conversely, have the advantage of delivering high compute performance within a limited power budget while retaining flexibility thanks to their programmable nature. The GAP8 SoC, which we also refer to in this work, is the commercial embodiment of the parallel ULP (PULP) heterogeneous paradigm [22]. Its "workhorse" capabilities have already been demonstrated aboard nano-drones, executing complex DNNs [2], [3] for lane-detection and obstacle avoidance tasks ($6\,\mathrm{frame/s}$, $40\,\mathrm{MMAC/frame}$ @ 64 mW), as well as the PULP-Frontnet DNN [1] this work is based on ($20\,\mathrm{frame/s}$, $14\,\mathrm{MMAC/frame}$ @ 25 mW).

Even assuming unbounded computational power aboard UAVs, ensuring that DNN models perform well when deployed in the real world remains challenging. One crucial aspect is that of generalization: training models that perform well in environments different from those shown in training data. Many of the breakthroughs of deep learning in computer vision have achieved this through a large-scale in-the-wild collection of training data followed by manual annotation by humans [8], [9]. In the context of robot perception, this is not always feasible: most applications require task-specific datasets, so the amount of available data is often limited.

The simulation-to-reality transfer is a common strategy to sidestep the scarcity of real-world data. SoA results have been achieved by augmenting simulation-only training data with techniques of domain randomization [11], [23], even obtaining strong generalization to real life without any fine-tuning on a physical drone [10]. But building faithful simulations of the desired task is far from straightforward in many cases and requires significant development effort specific to the task itself. This is especially true for simulations involving complex physical interactions (such as soft objects, deformable ground, etc.) or, as in our case, when humans are involved. In fact, humans' appearance and their clothes require nontrivial modeling effort, and their motion and behavior are complex and difficult to predict.

Halfway between real data and the entirely synthetic data from simulators is *data augmentation*, a standard technique in deep learning to improve the generalization of DNNs. Starting from a real-world training set, it artificially increases the amount of data by randomly perturbing copies of each training instance so that the corresponding labels are not affected. Various augmentation strategies have been devised [24], the most common being to apply geometric transformations and color adjustments [25] or to erase parts of the image [26]. Recently, machine learning approaches have been proposed to explore automatically image augmentation strategies [27].

Our approach brings domain randomization to *real-world*

Fig. 3. Top row: original PULP-Frontnet training frames with object detection and segmentation (colored masks). Bottom row: resulting images after applying the background randomization.

data as a step in our data augmentation pipeline. A pre-trained Mask R-CNN [28] is used to extract a segmentation mask of the foreground person in each camera frame, which is then used to composite the person onto a random background selected from the Indoor Scene Recognition dataset [9]. Our methodology shows a significant improvement (up to 40%) w.r.t. the original baseline without domain randomization when a never-seen-before environment challenges the DNN's prediction.

III. Methodology

A. PULP-Frontnet and robotic platform

Our work leverages the *PULP-Frontnet* [1] convolutional neural network (CNN), a SoA DNN model for the human pose estimation task demonstrated on PULP processor. The model takes as input gray-scale low-resolution images from the camera onboard the nano-drone and produces as regression output the relative pose of the human subject w.r.t. the drone. The pose is composed of four independent variables, which correspond to the position components, represented as a point in 3D space (x, y, z), and one orientation component, the rotation angle w.r.t. the gravity z-axis, called ϕ or *yaw*. Relative rotations along the *roll* and *pitch* axes, on the other hand, are not considered by the model.

The CNN topology, shown in Figure 2, adopts the established pattern in which each convolution layer is followed by batch normalization and a ReLU activation function. The model is composed of seven convolutional, one max-pooling, and one fully connected layer. The first layer is a 5×5 stride-2 convolution, followed by a 2×2 max-pooling layer, both of which reduce the size of output feature maps by $4\times$. The central part of the CNN contains three repeated blocks, each composed of two 3×3 convolutions that double the number of output channels and divide the feature map size by $4\times$, each. Finally, dropout is applied for its regularization effect, followed by a fully connected layer that predicts the four regression outputs.

The reference deployment platform for both PULP-Frontnet and this work is a COTS palm-sized UAV, the Bitcraze Crazyflie 2.1 nano-quadrotor. It is open-hardware and open-source, and its large community makes it ideal for research. It can be coupled with a COTS pluggable printed circuit board (PCB) called AI-deck[3], to increase the onboard computational/memory resources. This PCB plays a crucial role in extending the drone's perception capabilities, providing a monochrome ULP Himax HM01B0 camera capable of delivering QVGA images at $60\,\mathrm{frame/s}$ with a power consumption of $\sim 4\,\mathrm{mW}$. In addition, the AI-deck provides unprecedented computational power for a nano-drone through a PULP-based GWT GAP8 octa-core SoC that peaks at about $2\,\mathrm{GOp/s}$. Lastly, this expansion board offers additional off-chip memory, i.e., $8\,\mathrm{MB}$ DRAM and $64\,\mathrm{MB}$ Flash storage, and an ESP32 WiFi module that allows real-time streaming of camera images to a computer, easing the dataset collection process.

Ground-truth data used to train the PULP-Frontnet model is acquired in a room equipped with a motion capture system. Ten sequences are recorded, each featuring a different human subject moving around the scene. The raw, gray-scale, 160×160 pixels images acquired from the drone's onboard camera are streamed to a nearby base station using the Wi-Fi module on the nano-drone and stored together with the drone's and human subject's absolute poses captured by the mocap system. After the recording is complete, the four-dimensional relative poses (x, y, z, ϕ) of the subject w.r.t. the drone, are computed for each image. Of the ten recorded sequences, four are kept to form the test set. This ensures that subjects are entirely disjoint between training and test set. The other six sequences are further split, randomly holding out 20% of their frames as a validation set and using the remainder as the training set. Training is performed over 100 epochs using the Adam optimizer (learning rate 10^{-4}) to minimize the L1 loss for the 4-component relative pose, (x, y, z, ϕ).

The PULP-Frontnet architecture is optimized to take advantage of the computational capabilities of the GAP8 SoC. Due to the absence of a dedicated floating-point unit, deployment on this platform requires a quantization step so that inference can be efficiently performed using 8-bit fixed-point arithmetic. In this work, we use the 160×32 version of PULP-Frontnet

[3]https://www.bitcraze.io/products/ai-deck

Fig. 4. A) An image sample from the original PULP-Frontnet dataset. B) Samples of the only data augmentation procedure without (w/o) background randomization. C) Sample images of the entire pipeline including the background randomization.

DNN, which has $303\,\mathrm{k}$ parameters and requires $14.1\,\mathrm{MMAC}$ and $499\,\mathrm{kB}$ of memory to perform inference on a single frame. PULP-Frontnet throughput peaks at $48\,\mathrm{frame/s}$ within $96\,\mathrm{mW}$ power consumption, an energy efficiency of $2\,\mathrm{mJ/frame}$.

B. Background replacement

To enable our augmentation pipeline (Section III-C), we perform *object instance segmentation* on the PULP-Frontnet training dataset employing the SoA model Mask R-CNN [28]. This process partitions a digital image into semantically consistent regions (i.e., groups of pixels), detecting the various "objects" visible in the scene. For every object in the input image, Mask R-CNN returns:

- **the semantic class of the object** (e.g., person, animal), i.e., the result of a classification process;
- **the bounding box**, i.e., the rectangle containing the object and defining its location and dimensions inside the image;
- **the binary segmentation mask**, marking which pixels inside the bounding box actually belong to the object.

Mask R-CNN's classification domain — the classes of objects that it can detect — depends on its training dataset. For this purpose, we use the Microsoft COCO [8] dataset, which contains the "person" class among ten others, spanning from animals to daily-life objects, vehicles, and more. Therefore, Mask R-CNN, trained on the COCO dataset, acts in our pipeline as a "person-detection" CNN (as shown in the top row of Figure 3), which is of paramount importance given our ultimate goal of human pose estimation. Contrary to simpler *semantic segmentation* models [29], we selected Mask R-CNN due to its capability to isolate every single instance of the same class; for example, multiple persons in the same scene are identified as separate objects. Finally, the "person object" with the biggest bounding box in the image is selected to be the subject of the relative pose prediction. All remaining pixels outside the human subject's binary mask are considered "background", including all "non-person" objects and other persons with smaller bounding boxes.

Once we have marked, for every image of the PULP-Frontnet training dataset, the pixels corresponding to the human subject and those referring to the background, we proceed with background replacement. Using the *Indoor Scene Recognition* [9] dataset, which features 15'620 images from 67 categories of indoor environments (e.g., homes, stores, libraries, and working places), we replace the background on all images. We first resize the new background image to have 160 pixels on the smallest dimension (e.g., the height) while preserving its aspect ratio. Then we crop it to 160×160 pixels, and lastly, we paste the human subject onto the new resized and cropped background image, using standard compositing operation. This process of background replacement is performed only once before training the PULP-Frontnet DNN to minimize the computational overhead. An example of the result of our background randomization pipeline is reported in the bottom row of Figure 3.

C. Data augmentation

Our data augmentation pipeline runs during the PULP-Frontnet DNN training, performing the following three steps for each training batch of images:

- **Background randomization**: as described in Section III-B, we randomly replace the background in all 160×160 gray-scale input images of the original training dataset while keeping the identified human subject in the foreground.
- **Pitch augmentation**: we crop each "background-randomized" image at random image's height, passing from an input size of 160×160 pixels to the target resolution of 160×96 pixels. This approach simulates variations in the pitch of the drone: for example, cropping a 96-rows image aligned to the top of the input image approximates a $+14°$ pitch w.r.t. the same cropping but in the center of the image. As the training dataset has been acquired with the nano-drone mounted on a movable cart at a fixed pitch [1], this augmentation brings the pitch variance of the training set closer to the one seen during the actual flight.
- **Photometric augmentation**: to increase the robustness of our DNN model, we introduce additional photometric and optical augmentation, which includes exposure, gamma correction, dynamic range reduction, blur, and additive noise, followed by a vignetting effect with random intensity, as depicted in Figure 4. Ensuring the presence of the vignetting effect also after the background replacement is particularly useful to match the photometric properties of the low-resolution camera used to collect the original training set, which is also the one available on the final deployment nano-drone.

Like in [1], we apply these transformations generating ten new images from every input one, where all the augmentation

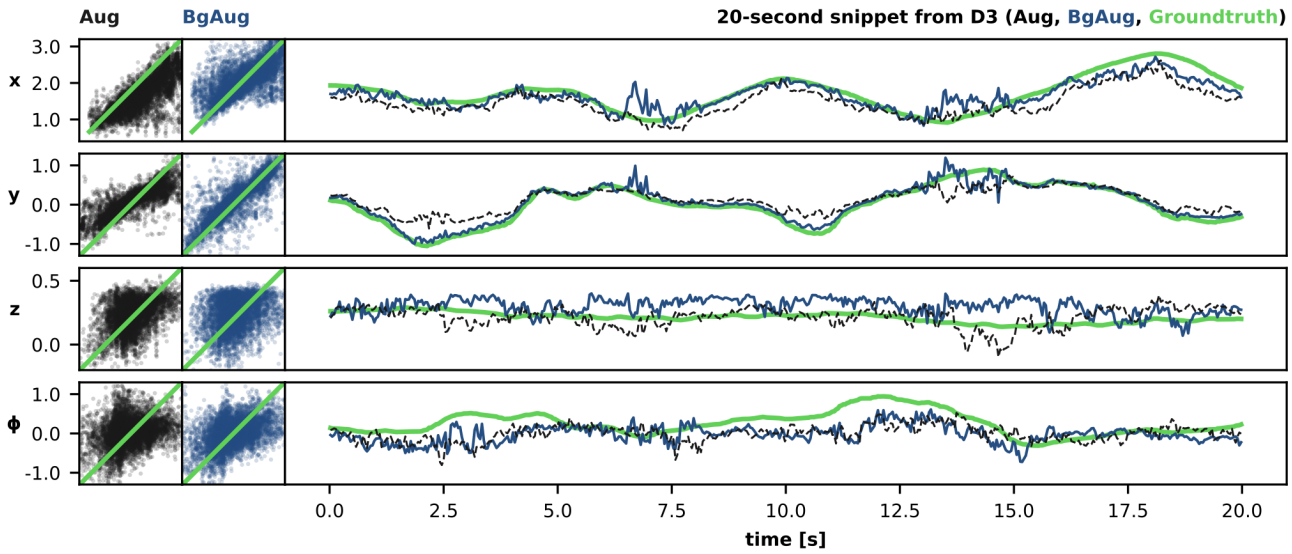

Fig. 5. For each output variable (rows), we report the ground truth and predictions for Aug (black) and BgAug (blue) models. Scatter plots on the left compare ground truth (x axis) to predictions (y axis) of the two models on all frames in D3. Line plots on the right show a time series of the ground truth (thick green) and the two predictions (Aug: dashed black thin line; BgAug: blue thin line) for a 20 s segment from D3.

parameters (e.g., background, pitch, gamma correction, etc.) are randomly selected every time. Therefore, we increase the training dataset from 2'629 to 26'290 images in each epoch. Lastly, since we rerun the whole augmentation procedure for each training epoch, the ultimate number of different images grows up to 2'629'000 when iterating over 100 epochs, like in our case.

IV. Experimental Results

A. Dataset and DNN training

We consider two different environments (EnvA and EnvB) for our experiments, both equipped with a 12-camera OptiTrack motion capture system. Both environments are indoor laboratories of a similar size (approximately 10×10 meters); still, they have significantly different physical characteristics, such as walls, furniture, floor texture, ceiling colors, and lighting, as shown in Figure 6. We record several camera sequences featuring different human subjects moving around the scene in each environment and employing the QVGA gray-scale camera available on the target nano-drone. We manually handle the drone, keeping the subject inside the camera's field of view while capturing different parts of the room as backgrounds. In EnvA, the drone rests flat at a fixed altitude on a cart that is pushed around (so to enforce a constant pitch of almost $0°$), while in EnvB, we move the handheld drone freely in space, constantly changing its pitch and roll, to mimic in-flight conditions.

The raw, gray-scale, 160×160 pixels images acquired by the drone camera are streamed to a PC via the onboard WiFi and stored together with the absolute poses of the drone and the subject captured by the motion tracking system, both running at the same frequency (i.e., 60 Hz). After recording, for each image, we compute the four-dimensional relative pose (x, y, z, ϕ) of the subject w.r.t. the drone, and we extend the PULP-Frontnet original dataset, resulting in three disjoint datasets:

- **D1**: is the original PULP-Frontnet training and validation dataset (2'629 images);
- **D2**: is the original PULP-Frontnet test dataset (1'119 images);
- **D3**: is a new test dataset (8'737 images), collected in a never-seen-before environment.

D1 and D2 are acquired in EnvA with two different sets of subjects, instead D3 is acquired in EnvB, with yet another set of subjects.

In the following, we use the D1 dataset to train two CNN regressors, which differ only by the type of augmentation applied during training:

- **Aug**: the baseline approach introduced in [1], which only uses pitch and photometric augmentation;
- **BgAug**: our approach which combines both background randomization, and pitch/photometric augmentation.

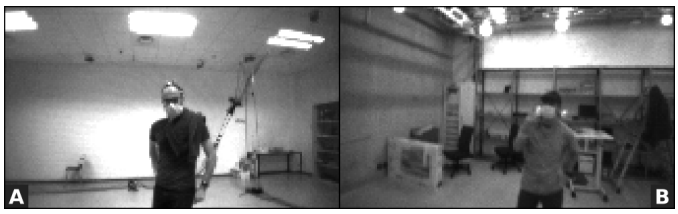

Fig. 6. A) Sample image from dataset D1 in EnvA and B) a sample image from D3 in EnvB.

TABLE I
MAE AND MSE REGRESSION PERFORMANCE FOR AUG AND BGAUG MODELS.

Network	MAE [$\cdot 10^{-3}$]				MSE [$\cdot 10^{-3}$]			
	x	y	z	θ	x	y	z	θ
Aug	464	231	**128**	492	362	125	**52**	502
BgAug	**321**	**182**	153	**436**	**218**	**95**	62	**419**

The two models use the same lightweight architecture introduced in PULP-Frontnet (named 160×32) and discussed in Section III-A. The two models are trained for 100 epochs (without early stopping) on a dedicated workstation equipped with Nvidia GeForce RTX 2080 Ti GPUs. The training data are split into 329 batches for each epoch, with each one containing 64 images randomly sampled from D1 and augmented using either the Aug or BgAug strategy. On average, the models see ten randomly augmented versions of each of the 2'629 training images during one epoch. 20% of samples are kept for validation from the training set, being manipulated with the same augmentation strategy we use for training.

Since the proposed methodology only affects the training process, during inference, both models have the same execution time and power consumption, identical to those of the original PULP-Frontnet 160×32 model [1]. As we can configure the GAP8 SoC's frequencies to best match the application's needs, the models' throughput can span from $\sim$20 to $48\,\mathrm{frame/s}$ within $\sim$25 to $96\,\mathrm{mW}$, respectively. This leaves enough computational/memory headroom on the GAP8 SoC for additional tasks to run together with the proposed augmented PULP-Frontnet.

B. Regression performance

In the following, we present a quantitative analysis of the two models, i.e., Aug and BgAug, on both D2 and D3 test datasets. The scatter plots, on the left of Figure 5, compare, for each output variable (rows), the relationship between the ground truth and prediction of each of the two models on all frames of D3. Furthermore, it also shows the evolution of these variables for a $20\,\mathrm{s}$ segment of D3. Additionally, we report in Table I the regression performance for the same dataset D3, in terms of mean absolute error (MAE) and mean squared error (MSE). We observe that, qualitatively, predictions from BgAug match the ground truth more closely than predictions from the Aug model, scoring a lower MAE and MSE for all output components except for the z one, where BgAug shows a slightly increased error (MAE from 0.13 to 0.15 and MSE from 0.05 to 0.06).

This observation is further quantified in Figure 7, which reports the R^2 metric for each output variable (x, y, z, ϕ) of each model, for both D2 and D3. The R^2 value indicates the fraction of variance in the target variable that the model explains; it reaches 1.0 for an ideal regressor, whereas it has a value of 0.0 for regressors that consistently predict the average of the target variable on the testing set. As expected, for both

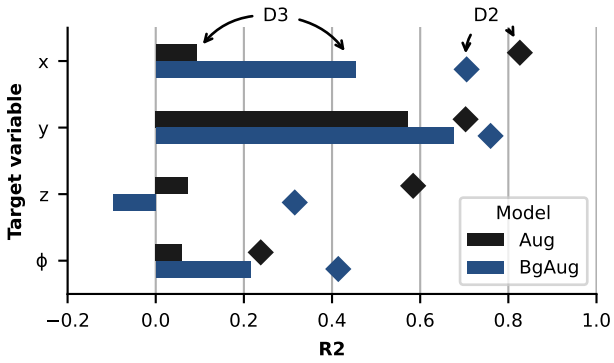

Fig. 7. R^2 on D3 (bars) and D2 (diamonds) for each output variable (y-axis) for Aug (black) and BgAug (blue) models.

models, the performance on D2 (diamonds) is always better (higher R^2 score) than in D3 (bars) — for all four outputs. This results from the higher similarity of the test set D2 to the training one (D1), both acquired in EnvA.

Focusing only on D2 (Figure 7, diamond markers) background augmentation (BgAug) is surprisingly beneficial for the x, ϕ outputs, while it reduces the performance on the other outputs. The small reduction in performance on the x output (i.e., the subject's distance) is in the expected order of magnitude, i.e., from 0.83 to 0.71, as Aug has been trained exclusively on EnvA: the same environment of the test set D2. Instead, the drop in performance on the z variable requires a more thorough discussion, as the R^2 score drops from 0.58 to 0.31.

To explain this, we note that because of pitch augmentation, estimating z from a single image (without any explicit input representing the actual pitch at which it was acquired) is very challenging. For example, observing the subject's head in the bottom part of the image might mean either that the head is lower than the quadrotor ($z < 0$), while the quadrotor has a 0° pitch, or that the head is at the same height as the quadrotor ($z = 0$), while the quadrotor has a positive pitch (i.e., is looking up). Thus, we hypothesize that the model, to predict z better, indirectly learns to estimate the camera pitch from the image background. This approach works well as long as the background is consistent between the training and evaluation datasets (e.g., for the Aug model in D2). When this is not true, e.g., with background randomization, performance on the z output deteriorates. In practice, precise estimation of the subject's relative height is not critical for most real-world applications, such as those where the altitude needs to be constant or obtained through other means.

Lastly, focusing on D3 (Figure 7, bar markers), i.e., the most challenging never-seen-before test set, BgAug outperforms Aug for x, y, and ϕ variables, confirming the effectiveness of the proposed approach. Even in this case, BgAug performance on the z component is marginally lower than Aug, i.e., from 0.07 to -0.09. Overall, the D3 test set performance shows that the proposed augmentation pipeline (including the background randomization) increases the model's generalization capabili-

ties to different environments and its robustness to varying pitch and roll of the camera.

V. Conclusions

In this paper, we tackle the limited generalization capabilities found in DNN-based visual pipelines for autonomous navigation. This DNN-based class of solutions is rapidly getting adopted among nano-UAVs, as the computation/memory requirements are lower than more traditional visual pipelines for autonomous navigation (e.g., SLAM) and, therefore, a better fit for their resource-constrained MCUs. However, one severe limitation of this class of solutions is the lack of generalization capabilities, i.e., the visual cues learned on the specific training domain hardly predict with the same accuracy on different ones. This work introduces a novel data augmentation methodology based on background randomization and image augmentation for training a visual human pose estimation CNN for autonomous UAVs. Experimental results on data from two different environments show that the augmented DNN yields better generalization ability than a baseline without background randomization. Our approach reduces the DNN's mean square error — vs. a non-augmented baseline — up to $\mathbf{40}\%$, on a never-seen-before testing environment, with zero-impact on the computational/memory/power burden aboard the nano-UAV. As the next step in the evolution of our work, we envision the in-field deployment of our augmented DNN, to perform further evaluation of the control accuracy of our closed-loop nano-drone, challenging it with a wide range of different environments (both indoor and outdoor).

References

[1] D. Palossi, N. Zimmerman, A. Burrello, F. Conti, H. Müller, L. M. Gambardella, L. Benini, A. Giusti, and J. Guzzi, "Fully onboard AI-powered human-drone pose estimation on ultra-low power autonomous flying nano-UAVs," *arXiv preprint arXiv:2103.10873*, 2021.

[2] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mW DNN-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.

[3] D. Palossi, F. Conti, and L. Benini, "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 604–611.

[4] R. J. Wood, B. Finio, M. Karpelson, K. Ma, N. O. Pérez-Arancibia, P. S. Sreetharan, H. Tanaka, and J. P. Whitney, *Progress on "Pico" Air Vehicles*. Springer International Publishing, 2017.

[5] D. Mantegazza, J. Guzzi, L. M. Gambardella, and A. Giusti, "Vision-based control of a quadrotor in user proximity: Mediated vs end-to-end learning approaches," *2019 IEEE International Conference on Robotics and Automation (ICRA)*, May 2019.

[6] S. H. Mallidi, T. Ogawa, and H. Hermansky, "Uncertainty estimation of DNN classifiers," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 283–288.

[7] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning," *arXiv preprint arXiv:2002.06470*, 2020.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*. European Conference on Computer Vision, September 2014.

[9] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 413–420.

[10] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019.

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[12] J. Collins, D. Howard, and J. Leitner, "Quantifying the reality gap in robotic manipulation tasks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6706–6712.

[13] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6008–6014.

[14] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung, "Neural-Swarm: Decentralized close-proximity multirotor control using learned interactions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3241–3247.

[15] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[16] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari, "Structure-SLAM: Low-drift monocular SLAM in indoor environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6583–6590, 2020.

[17] F. Candan, A. Beke, and T. Kumbasar, "Design and deployment of fuzzy PID controllers to the nano quadcopter Crazyflie 2.0," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2018, pp. 1–6.

[18] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A 2-mW fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, 2019.

[19] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of Threats? a survey of practical security vulnerabilities in real IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[20] W. Zhao, J. Panerati, and A. P. Schoellig, "Learning-based bias correction for time difference of arrival ultra-wideband localization of resource-constrained mobile robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3639–3646, 2021.

[21] J.-H. Yoon and A. Raychowdhury, "31.1 a 65nm 8.79TOPS/W 23.82mW mixed-signal oscillator-based NeuroSLAM accelerator for applications in edge robotics," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 478–480.

[22] F. Conti, D. Palossi, A. Marongiu, D. Rossi, and L. Benini, "Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 1201–1206.

[23] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Conference on Robot Learning*. PMLR, 2020, pp. 1162–1176.

[24] C. Shorten and T. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, 07 2019.

[25] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 6256–6268.

[26] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using DropConnect," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28. PMLR, 2013, pp. 1058–1066.

[27] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le, "Learning data augmentation strategies for object detection," in *European Conference on Computer Vision*. Springer, 2020, pp. 566–583.

[28] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[29] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.