POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Politecnico di Torino

Master's Degree Thesis

Human Pose Estimation aboard Nano-drones Using Tiny Vision Transformers

Supervisors

Prof. Daniele JAHIER PAGLIARI

Prof. Alessio BURELLO

Beatrice Alessandra MOTETTI

Candidate

Ovidiu Ioan JITARU

December 2024

Abstract

Nowadays drone usage is increasing due to their technological advancements, versatility, and broad applicability for a wide range of tasks. Improved capabilities in the drone sector and tech miniaturization such as the development of smaller and more powerful embedded systems allow AI to be integrated and efficiently run aboard them. Standard-size drones can be equipped with powerful Graphics Processing Units (GPUs) that allow the use of complex neural networks to solve perception tasks. However, nano-drones, with their extremely small dimensions and power envelope, have major limitations in terms of supported computational capabilities. Thus, there is a need to develop more optimized solutions to enable onboard AI, preserving the real-time response of the perception system while maximizing the performance of the considered task.

At the moment the state of art (SoA) for computer vision tasks on nano-drones makes use of lightweight CNN architectures which can be effectively executed by the MCU-class processor aboard without the need for high-performance GPUs. However, many recent works in the computer vision field have proved how the Vision Transformer (ViT) architecture frequently outperforms SoA CNNs in several tasks. Thus, optimizing the architecture of ViTs by reducing their size and latency for real-time applications, making them suitable for deployment on nano-drones, represents an interesting challenge.

The task for which the drone is challenged consists of human pose estimation, which is a computer vision task aimed at identifying the position and orientation of a person. In this particular case, the drone's objective is to position itself in front of a person and follow them while keeping a constant distance from the subject.

The work of the thesis consists of analyzing and developing through optimization techniques an efficient ViT model to be deployed on nano-drones that mount a GAP8 AI deck. Two datasets containing images obtained from two separate laboratories are used to train and assess the performance of distinct perception modules. Multiple approaches are explored, comprising the evaluation of different ViT's architecture configurations and the assessment of the benefits of a pre-training step prior to the fine-tuning of our task. Finally, comparisons of the selected ViT performances with the MobileNet are considered, a CNN that achieves SoA results on the considered benchmark, and in the end, structured pruning is applied to reduce the model's size, while preserving its original performance.

Comparisons between the ViT and MobileNet networks are assessed considering the Mean Absolute Error (MAE) between the predicted $\hat{x}$, $\hat{y}$, $\hat{z}$, and $\hat{\phi}$ coordinates and the ground-truth ones. We achieve with the ViT network an overall loss over all 4 predicted axes 12% lower than the MobileNet. Finally, by applying structured pruning techniques we reach a 30% compression of the model in terms of number of parameters with an increase of the overall loss to 1.3 with only a 12% decrease in performances.

Acknowledgements

This thesis marks the culmination of a long academic journey, one filled with both professional challenges and personal hardships. I extend my gratitude to my supervisors, Professor Daniele Jahier Pagliari, Alessio Burello, and Beatrice Alessandra Motetti, for their support and insightful guidance throughout my research journey. Their profound dedication to academic excellence and meticulous attention to detail have been instrumental in shaping this dissertation.

To my dear parents, I owe infinite gratitude for their unwavering support and steadfast presence during even the most difficult moments. Though this path was often solitary, I never felt alone when looking back, for they were always there for me. To my brother, whose boundless energy and lightheartedness have brightened my grayest days and brought countless smiles to my face, thank you for being my constant source of joy.

To my beloved girlfriend, who entered my life during the most challenging phase of this journey. Thank you for your unwavering encouragement, for motivating me to persevere, and for loving me unconditionally.

To my lads, whom I met in the good times and who are still by my side today, thank you for being there and for our unique and special way of being there for each other. Zoi, Netta, Lambe and Pimpy, your friendship has been a source of strength and I hope our bond remains as strong as ever.
To my friends who shared the struggles of endless study sessions, daunting exams, and moments of anxiety and uncertainty: you Pigeon, Cat, Raccoon, Dani, Ja, Granny, Princess, Cozzo, Little Seal, and Diddi, thank you for walking this road with me and making it less lonely.
To my colleagues, who are no longer colleagues but have long since become dear friends, Luca, Giorgio and Matteo. I am incredibly grateful for your support in critical moments. Your companionship and encouragement have been invaluable, and I am truly grateful to have known you.

To everyone who shared even the smallest meaningful moments with me during this journey, thank you for your presence and kindness. Each interaction has added to this unforgettable experience.

Finally, to the version of myself who embarked on this path long ago, thank you for persevering despite every challenge and setback. You deserve this accomplishment and so much more. Best of luck in all that lies ahead.

Figure 1: Ringraziamenti.

Table of Contents

List of Tables

List of Figures

Acronyms

AI
Artificial intelligence

CV
Computer Vision

CNN
Convolutional Neural Networks

DL
Deep Learning

DNN
Deep Neural Network

FNN
Feedforward Neural Network

FPD
Fast Pose Distillation

GPU
Graphical Processing Unit

HCI
Human-Computer Interaction

HW
Hardware

LLM

Large Language Models

MAE

Mean Absolute Error

MCU

Microcontroller Unit

ML

Machine Learning

MLP

Multi Layer Perceptron

MHSA

Multi Head Self-Attention

NAS

Neural Architecture Search

NLP

Natural Processing Language

NN

Neural Network

PAF

Part affinity fields

PULP

Parallel Ultra Low Power

SoA

State of Art

SoC

System on Chip

Chapter 1

Introduction

In recent years, the adoption and application of drones, or Unmanned Aerial Vehicles (UAVs), have expanded dramatically, driven by significant technological advancements in areas such as electronics, machine learning, and robotics.
Drones are increasingly being used across a wide range of industries, from agriculture and infrastructure inspection to public safety, search-and-rescue missions, and entertainment. The increasing versatility of drones, their ability to carry out tasks in difficult or hazardous environments, and advancements in their autonomous capabilities, have accelerated their growth in usage. At the heart of this development, we can find the integration of artificial intelligence (AI) and machine learning techniques, which enable drones to perform complex tasks such as object detection, navigation, and autonomous decision-making with high precision and speed.

A key enabler of these advances is the miniaturization of powerful embedded systems, allowing AI algorithms to be run directly on board drones. Standard-size drones, for example, can be equipped with high-performance hardware, such as Graphics Processing Units (GPUs), which provide the computational power needed to run complex neural networks for tasks like computer vision and perception. However, while these advancements have enhanced the capabilities of standard drones, nano-drones which are lightweight drones with almost insect-sized forms, designed for constrained environments, face significant challenges. Due to their limited size and power supply, nano-drones cannot accommodate the same powerful hardware as their larger counterparts. This makes the deployment of AI models on nano-drones much more challenging, as their limited computational resources cannot support the execution of standard AI algorithms in real-time.

As nano-drones become more prevalent in applications such as surveillance, indoor navigation, and disaster response, the need for efficient, real-time onboard intelligence becomes critical. Traditional AI models, particularly those used in

computer vision tasks, are computationally expensive and memory-size heavy, thus impractical for direct deployment on resource-constrained platforms like nano-drones. To address this challenge, researchers and engineers focus on optimized, lightweight models that can deliver high performance while operating within the constraints of small devices. One of the key research directions in this field is the development of deep learning architectures that balance the trade-off between accuracy and computational efficiency.

Historically, Convolutional Neural Networks (CNNs) have been the dominant architecture for computer vision tasks. CNNs have proven highly effective at recognizing and interpreting visual data, from simple object classification to more complex tasks like human pose estimation. For nano-drones, lightweight versions of CNNs have been successfully deployed due to their relatively low computational demands. However, recent advancements in the field of computer vision have introduced a new model architecture known as the Vision Transformer (ViT), which has been shown to outperform traditional CNNs on a variety of tasks. The ViT model, which leverages transformer-based architecture mechanisms for processing image data, offers several advantages in terms of accuracy and flexibility. Despite its potential, the transformer-based model's high computational complexity and size such as the ViT, limits its application in real-time, resource-constrained environments such as nano-drones.

The central objective of this research is to address this gap by optimizing the Vision Transformer architecture to make it suitable for real-time deployment on nano-drones. Specifically, the focus is on the task of human pose estimation, a challenging computer vision problem where the goal is to identify the position and orientation of a person from visual data like pictures or video. The use case for this thesis involves a nano-drone that autonomously positions itself in front of a person and follows them while maintaining a constant distance. This requires the drone to continuously process visual data and adjust its position in real time, a task that is computationally intensive and must be accomplished within the limited processing power of a nano-drone's onboard hardware.

To achieve this, the research explores various optimization techniques aimed at reducing the computational demands of the ViT model while maintaining its accuracy and effectiveness for the task of human pose estimation. The nano-drone used in this study is equipped with a GAP8 AI deck, a low-power, microcontroller unit (MCU) designed for energy-efficient AI processing. Given the limited resources of the GAP8, optimizing the ViT model to run efficiently on this hardware is crucial.

The study employs two datasets obtained from separate laboratories to train and

evaluate the performance of different neural network architectures. By comparing various configurations of the ViT model and experimenting with pre-training and fine-tuning techniques, the research seeks to identify the most effective configuration for real-time deployment. In addition, the thesis benchmarks the performance of the optimized ViT model against MobileNet, a lightweight CNN architecture that has demonstrated state-of-the-art results on similar tasks. The MobileNet model has been widely adopted in edge computing applications due to its efficiency and performance, making it an ideal candidate for comparison.

One of the major contributions of this work is the application of structured pruning techniques to the ViT model. Structured pruning involves removing less important parameters from the neural network, thereby reducing the model's size and computational load without significantly degrading its performance. This process allows the ViT model to be compressed and optimized for deployment on nano-drones, where both storage capacity and real-time processing are constrained.

In conclusion, this thesis shows the feasibility of deploying an optimized Vision Transformer model on resource-constrained nano-drones. The results of this research show that, with proper optimization, ViT models can serve as a powerful alternative to CNNs for real-time computer vision tasks on low-resource devices like nano-drones.

Chapter 2

Background

This chapter provides background on Human Pose Estimation (HPE) in section 2.1, Drones and Nano-Drones in Section 2.2, Transformers in Section 2.3, Vision Transformers 2.3.2, and Network Optimization techniques in Section 2.4.

2.1 Human-to-drone pose estimation

Human Pose Estimation (HPE) is a computer vision task that identifies the key body joints of a human in images and videos to determine their pose. These key points typically include critical joints such as the shoulders, elbows, knees, and ankles, which together form a skeleton-like representation of the human body. HPE methods use deep learning models to detect, localize, and track these points across video streams or static images.

This task has various applications across different domains, for example in Human-Computer Interaction (HCI) [1] for gesture recognition, enabling systems to interpret human gestures and translate them into commands to aid people with communication disabilities. In the field of healthcare and sports [2] it is used to keep track of posture and movement, helping in rehabilitation with unobtrusive monitoring of patients, in robotics [3] for assisting robots understanding human movements and helping human-aware robot navigation, and in surveillance [4] of groups or crowds of people increase the complexity of the task.

Despite its versatility over different fields, HPE faces several technical challenges that compromise its accuracy and robustness in real-world applications.
For example, body alterations due to different forms of clothing, occlusions when certain parts of the body are hidden or blocked, either by other objects or people

increase the complexity of the task.
Highly dynamic or cluttered environments, where multiple moving objects or people are present or extreme perspectives, such as those from above or below, pose additional difficulty in correctly identifying key points. This last challenge is especially relevant in drone-based HPE, where the camera views are not stationary and are subject to different orientations and angles. In these situations, the prediction of the human pose is extremely challenging.

HPE's core consist of the identification of key points in the human body and determining their exact spatial positions over an input image or video stream. The relationships between these key points are then used to infer the human body's overall posture, producing a skeletal representation of the individual.
Once key points are identified, the next step is to determine their spatial positions in 2D or 3D relative to the input image, creating a coherent representation of the human skeleton to interpret posture or activity. In 2D estimation, each joint is positioned using (x, y) coordinates, whereas 3D pose estimation adds depth, using (x, y, z) coordinates to capture the position in three-dimensional space.

2.2 Nano-Drones

Nano-drones are extremely small unmanned aerial vehicles (UAVs) designed for specialized applications where size, weight, and precision are critical factors. These drones have an extremely small size $\sim 10cm$ and lightweight designs, making them distinct from larger drone categories. Despite their miniature form, these drones are equipped with various functionalities such as cameras, sensors, and wireless communication systems that enable them to operate efficiently in a range of environments. Table 2.1 depicts a detailed taxonomy of UAVs by size, weight, power, and HW device.

Table 2.1: UAV's Taxonomy by size [5]

Vehicle class	⌀ :Weight[cm:kg]	Power[W]	Onboard device
standard-size [6]	~ 50:≥ 1	≥ 100	Desktop
micro-size [7]	~ 25:~ 0.5	~ 50	Embedded
nano-size [8]	~ 10:~ 0.01	~ 5	MCU
pico-size [9]	~ 2: ≤ 0.001	~ 0.1	ULP

One of the significant advantages of nano-drones is their mobility and ability to access hazardous environments. Due to their compact size and maneuverability, they can navigate through tight or cluttered spaces, making them highly effective for precision tasks such as indoor navigation, search-and-rescue missions in collapsed

buildings, while being under highly constrained power budgets [10]. This capability is particularly valuable in situations where human access would be dangerous or impossible.
Another notable benefit is the reduced risk of damage to the surroundings or injury to people. Because of their lightweight construction, nano-drones are less likely to cause significant harm if they collide with objects or people, making them safer to deploy in environments where fragile equipment or sensitive infrastructure is present.

Nano-drones are increasingly being utilized in various fields due to their versatility and adaptability. In surveillance [11], they are ideal for monitoring indoor and confined spaces where larger UAVs cannot operate effectively. Their small size allows for discreet and non-intrusive observation.
In search and rescue missions [12], nano-drones play a critical role by providing real-time situational awareness in disaster-stricken areas, such as collapsed buildings or areas affected by natural disasters. Their ability to navigate through debris and confined spaces helps rescuers locate survivors and assess structural damage, improving the overall efficiency and safety of rescue operations.

In many drone applications, AI processing is handled remotely due to the limited onboard computational power. The typical process involves drones collecting data such as images or video, transmitting them to a remote server where AI models process the information, and then sending the actionable insights back to the drone. This remote AI processing allows for the use of complex algorithms and deep neural networks (DNNs) that the drone itself cannot execute due to hardware constraints.

This approach, thanks to the use of remote servers allows to run large and complex models that require significant memory size and computational resources, which are usually unavailable on drone devices. In this way, drones can leverage cutting-edge AI models that are updated and managed remotely, allowing for more frequent software updates without the need for drone hardware upgrades.
On the other side, data transmission to and from the server introduces delays, which can be problematic in time-sensitive applications like real-time navigation or obstacle avoidance, and the drone depends on stable communication networks, which may not always be available, especially in remote or hazardous environments.

Advances in embedded systems and hardware efficiency allow to mount of high-performance HW, such as GPUs, enabling support for complex deep learning models. This enables the possibility of running DNNs directly on standard drones.

One of the primary advantages of integrating AI models on board of drones and nano-drones is the ability to perform onboard computations, eliminating the need

for constant data transmission to cloud servers. This not only reduces operational costs which translates into higher energy efficiency but also enhances data security, as locally processed data is less vulnerable to interception during transmission. Furthermore, processing data onboard minimizes the significant power drain associated with continuous communication with remote servers, and allows the drones to make real-time decisions with reduced latency.
However, even with improvements, the onboard processors of drones are still limited compared to remote servers, restricting the complexity of AI models that can be deployed. Another significant issue consist on the fact that onboard AI consumes significant battery power, reducing the drone's operational time.

For what concerns nano-drones, their benefits are accompanied by substantial challenges.
While nano-drones offer significant advantages in terms of cost efficiency, agility, and safety, their extremely reduced size means restricted space where to place high-performance HW.
Energy efficiency is a critical issue for nano-drones because large battery packs cannot be mounted, which directly limits the flight time and operational lifetime of the drone, and in particular also the computing power of nano-drones.
For the same reason, they mount MCU-class processors, which limits the complexity of neural networks that can be implemented on board.

Another significant limitation is the memory capacity of these drones. Given their compact structure, nano-drones have limited onboard memory, which poses problems in implementing models that require substantial memory for the increasing amount of parameters of DNNs. In addition, inference latency, the delay between input processing and output generation, becomes a critical factor in real-time applications, as the number of computations, due to the number of parameters, slows down processing in inference, where the timeliness of responses is critical. For example, in applications such as object tracking or navigation in dynamic environments, any delay in inference can hinder performance and compromise the success of the task.

To address the challenges posed by limited space, computational resources, and energy constraints in nano-drones, researchers have developed lightweight models specifically designed for resource-constrained environments.

2.3 Transformers

While CNNs have been the cornerstone of CV, in recent years, the introduction of transformer-based models for Natural Language Processing (NLP) has revolutionized various domains of AI. Transformers architecture allows the model to capture long-range dependencies in the input data. This is particularly useful for computer vision, where understanding both local and global contexts is essential for accurately interpreting visual information.
The success achieved by transformer-based models inspired their applications in CV tasks in place of CNN architectures with the emerging Vision Transformers.

2.3.1 Transformer Architecture

Transformer neural networks are a class of models initially designed for NLP, proposed as opposed to the then-dominant sequence transduction models, such as Recurrent Neural Networks (RNNs) or LSTMs.

Introduced by Vaswani et al. (2017) in their paper Attention is All You Need [13] the Transformer architecture revolutionized the way models process sequential data relying on the self-attention mechanism.

The Transformer's architecture is particularly impactful in NLP tasks like translation, text generation, and language modeling. Its key innovation lies in the self-attention mechanism.
The attention is the core of Transformer models, unlike traditional RNN, which process data sequentially and struggle with long-range dependencies, the attention mechanism allows the model to focus on different parts of the input simultaneously, capturing both local and global dependencies more effectively, thus making it both more efficient and capable of handling larger contexts.

The input sequence of the attention is mapped to a Q query, K key, and V value vectors by applying three linear transformations to the input data.
The Q elements represent what are the elements of interest for the model in the input data, the K vectors are used to determine whether the token is relevant to the query, and V vectors are the actual data from the input that the model will attend to based on the relevance scores. With this three matrices are then computed the self-attention.

One problem which arises is that the transformer processes the entire input at once, unlike other models like RNNs. It treats all elements in the sequence equally, meaning it does not understand the order of words in a sentence, and without

understanding the order, the model would lose important context.
To address this limitation positional encoding is being added to the input sequence to give the model information about the position of tokens in the input sequence. Figure 2.1 shows a simplified visualization of the usage of positional encoding, in the upper example, two phrases contain the same information but lack positional distinctions, while in the lower example, positional encoding differentiates the contexts, allowing the model to interpret them as distinct sequences.

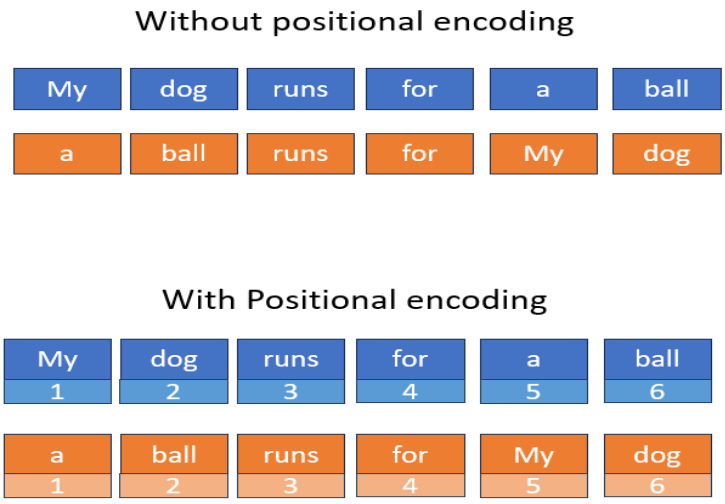

Figure 2.1: Positional encoding for sentences.

After applying positional encoding and mapping the input to the three vectors $\mathcal{Q}, \mathcal{K}, \mathcal{V}$, the scaled dot-product attention is computed, followed by a softmax activation to generate weights for the values.
Mathematically the attention can be written as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where d_k correspond to the dimensionality of vectors.

The architecture of the self-attention mechanism is illustrated in Fig. 2.2 In this setup, $\mathcal{Q}$ and $\mathcal{K}$ vectors are first multiplied to compute the dot product, which captures the similarity between tokens. The result is then scaled and passed through a softmax activation to produce attention weights. An optional mask can be applied to this dot-product output, setting certain positions to zero or -∞ to ignore specific values. The resulting attention weights are then multiplied by the $\mathcal{V}$ vector, yielding the final output vector.

The softmax function assigns higher weights to words that the model identifies as more relevant, thus amplifying their values in the output, while lower scores diminish the influence of less relevant words.

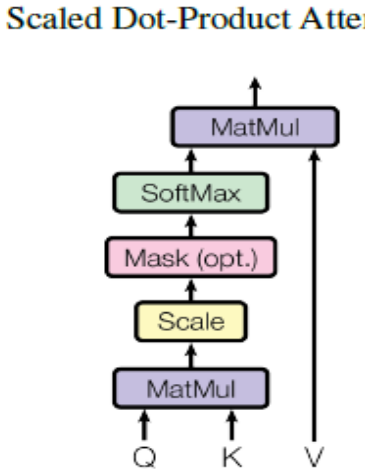

Figure 2.2: Workflow of $\mathcal{Q}, \mathcal{K}, \mathcal{V}$ in the attention mechanism.

In order to focus on different parts of the input the self attention is computed multiple times in parallel, thus creating the multi-head self-attention (MHSA) mechanism which is an extension of self-attention. Each attention head focuses on different parts of the input, capturing different aspects of the relationships between elements. These attention heads work independently and then combine their results to provide a richer understanding of the input data.
Fig. 2.3 shows the combination of multiple attentions to create the multi-head attention.

The complete transformer architecture, in its most general structure, shown in Fig. 2.4 consists of two main components: an encoder and a decoder.

On the left of the structure there is the encoder, composed of N identical layers, each containing two key sub-layers: a Multi-Head Self-Attention Mechanism and a Feedforward Neural Network (FFN).
At the end of the encoder stack, we have a sequence of context-aware representations of the input that encapsulate both local and long-range dependencies.
On the right of the structure, we find the decoder block, which is similarly composed

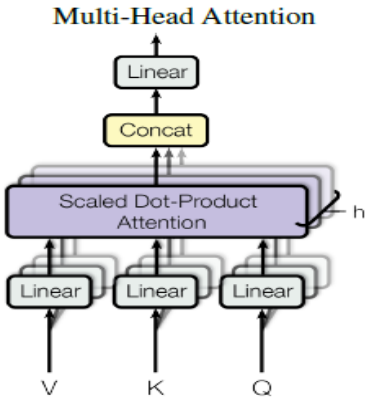

Figure 2.3: Extention of the self-attention into a multi-head attention.

of N identical layers. However, the decoder has an additional encoder-decoder attention sub-layer that allows it to focus on relevant parts of the encoder's output.

The encoder and decoder work together but have distinct responsibilities.
The encoder's goal is to extract meaningful patterns from the input sequence, converting raw input into a set of rich, context-aware vectors.
The decoder in contrast, aims at generating the output sequence by iteratively attending to both the encoded input and the output it has generated so far.

2.3.2 Vision Transformers

Vision Transformers (ViTs) are an adaptation of the Transformer architecture specifically for CV tasks. They represent a shift from traditional CNNs, which have long dominated the field of computer vision.
ViTs are based on the core concept of the Transformer architecture, particularly the self-attention mechanism, but adapted for image data. Rather than processing sequences of words, ViTs treat images as sequences of smaller image patches, transforming them into embeddings and processing them similarly to how a standard Transformer processes textual tokens.

With ViTs, the input image is divided into fixed-size patches, typically 16x16 pixels. Each patch is flattened and then linearly embedded into a high-dimensional vector.
Just like in text-based Transformers, ViT needs to maintain spatial information about the image, if the image patches are mixed the meaning of the original image

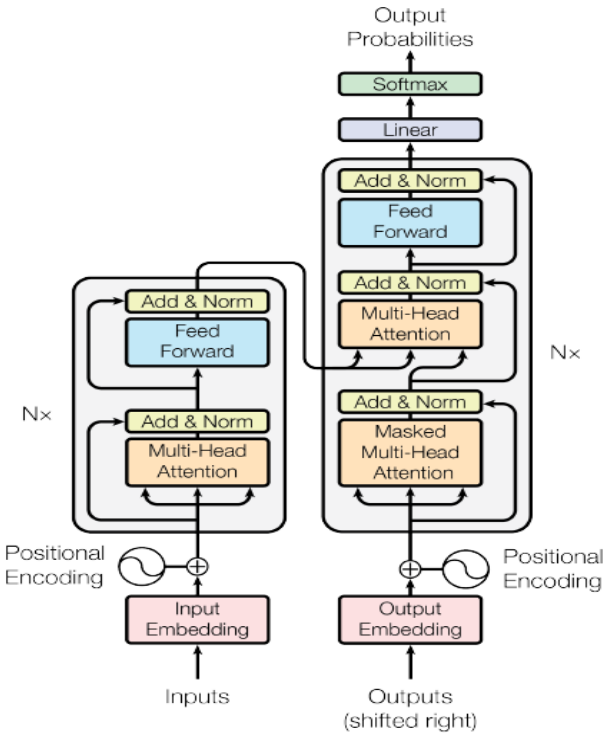

Figure 2.4: General structure of transformer architecture. [13]

is lost. Thus positional encodings are added to the patch embeddings to preserve the 2D spatial structure of the image.

In Fig. 2.5 it is shown how 2 different images, with patch positional-encoding similar to NLP tasks, will result in 2 different images.

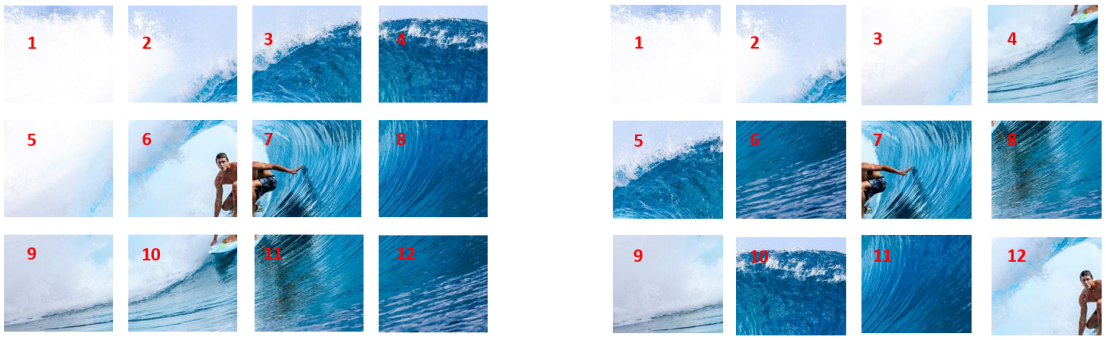

Figure 2.5: The position embedding applied on the patches allow to understand the differences between the two and not treat them as the same.

Figure 2.6 illustrates the new architecture presented in [14] with the new way of processing images applied to transformers. The last part of the architecture shows a Multi-layer Perceptron (MLP) as the head of the architecture used for the purpose of classification.

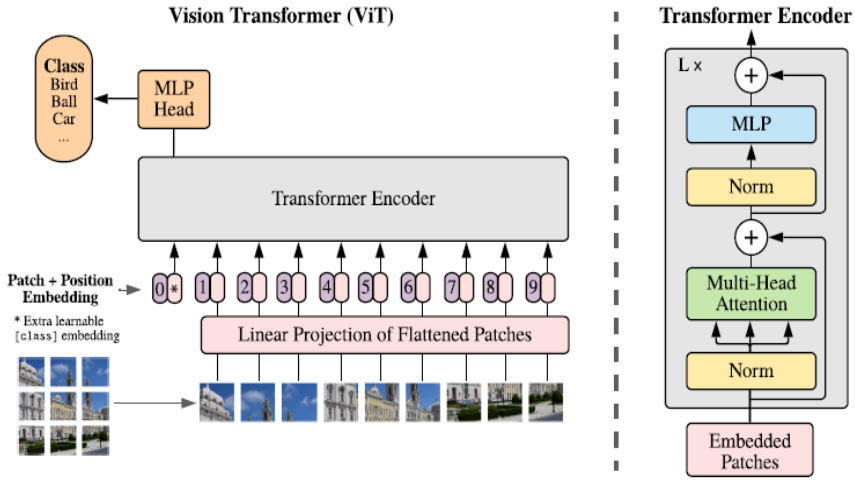

Figure 2.6: ViT general structure [14]

In the case of ViTs, only the encoder is used because ViTs are primarily used for image classification tasks, where the goal is to map a given input image to a single output label. This is a one-to-one mapping problem, where the input is an image, and the output is a classification label.

One of the key challenges with ViTs is their requirement for large datasets to achieve optimal performance. Unlike traditional CNNs, which can generalize well on relatively smaller datasets, Vision Transformers tend to perform poorly when trained on small datasets.
A common strategy to overcome this problem consists of transfer learning. ViTs can be pre-trained on large, publicly available datasets and then fine-tuned on smaller datasets for specific tasks. This allows the model to leverage knowledge gained from a large dataset. After pre-training, the model is fine-tuned to a smaller, task-specific dataset, such as drone imagery for human pose estimation. Finally, fine-tuning helps the model learn domain-specific features without requiring as much data as would be needed to train from scratch.

Deploying ViTs on nano-drones is a complex challenge due to the high computational, memory, and data requirements of ViTs. ViTs typically have a large number of parameters thus the size of ViT models can be a significant barrier for

deployment on nano-drones.
There are ongoing research into Tiny Transformers, and optimized solutions for compressed models that may help bridge this gap, enabling the powerful capabilities of ViTs to be used in resource-constrained environments like nano-drones.

2.4 Network Optimization

Network optimization in deep learning refers to a set of techniques and strategies aimed at improving the efficiency and performance of deep neural networks, as deep learning models grow increasingly complex, with millions or even billions of parameters.

In resource-constrained environments, where there are significant limits on computational power, energy, memory, and storage, network optimization is essential for achieving faster training, reducing inference time, and deploying models in resource-constrained environments like mobile devices, embedded systems, or nano-drones. Techniques like pruning, quantization, and model compression enable models to run in real-time, consume less energy, and fit within the hardware limitations of devices like nano-drones.
Reducing computational complexity is a crucial aspect of network optimization, particularly in resource-constrained environments. The aim is to make deep learning models more efficient by minimizing their memory requirements through compression and speeding up inference while maintaining accuracy.

Model compression allows to reduce the size and memory footprint of a model, making it more suitable for environments with limited storage and memory, such as nano-drones, mobile devices, or embedded systems.

Another important aspect consists of reducing inference latency which is critical for real-time applications, such as autonomous navigation, surveillance, or gesture recognition. The goal is to enable the model to process input data and generate predictions as quickly as possible, which is often a challenge for deep learning models due to their computational complexity.

2.4.1 Neural Architecture Search

Neural Architecture Search (NAS) [15] is an advanced technique in deep learning aimed at automating the process of designing neural network architectures. Traditionally, neural networks are hand-designed through trial and error, which can

be labor-intensive and time-consuming. NAS, however, leverages algorithms to automatically search for the optimal architecture based on a given objective, such as maximizing accuracy while minimizing computational cost, memory usage, or inference time.

In the context of network optimization, NAS plays a crucial role in discovering efficient models that can perform well in resource-constrained environments, such as mobile devices, embedded systems, and nano-drones. By optimizing the network architecture for specific constraints, NAS enables the creation of models that balance performance with efficiency, without the need for manual design intervention.

NAS comprises three primary components: the search space, the search strategy, and the evaluation of architectures. These components work together to explore various network architectures, assess their performance, and identify the most efficient designs.

The search space in NAS defines all the potential neural network architectures that can be explored during the search process. It includes various design choices that can be made for a neural network, such as the number of layers, types of layers, filter sizes, activation functions, and skip connections.
The richness and flexibility of the search space are critical because they determine the variety of architectures that NAS can explore. A broader search space allows for more experimentation, potentially leading to the discovery of novel architectures that outperform manually designed networks.

The search strategy is the core component of NAS that determines how the algorithm navigates the search space. Since it is impractical to evaluate every possible architecture in a vast search space, the search strategy must be efficient in finding promising candidates while avoiding exhaustive exploration. The search strategy can rely on differentiable or evolutionary algorithms, or on Reinforcement Learning.

The last step, once candidate architectures are generated, is to evaluate their performance. Each architecture must be trained and validated to assess its effectiveness, typically using metrics such as accuracy, loss, or error rate. However, in the context of network optimization, it is not enough to only focus on accuracy. The evaluation process must also account for other critical factors like computational complexity, memory usage, and inference latency, especially when targeting resource-constrained environments.
Mathematically the NAS optimization technique can be written as [15]

$$\Big\{ argmin = \quad \mathcal{L}(\hat{A}, D_{train}, D_{val}) \quad s.t. \hat{A} \in A$$

where $\hat{A}$ corresponds to one potential DNN from the search space A architectures, the $Loss(\cdot)$ is the measure of the performance of $\hat{A}$ on the validation dataset D_{val} and the training dataset D_{train}.

2.4.2 Pruning

Pruning is another important technique in network optimization that focuses on reducing the complexity of deep neural networks by systematically removing unnecessary parameters such as weights, neurons, or filters, while maintaining a high level of performance. By eliminating parameters that contribute little to the network's overall performance, pruning allows for reduced model size and improved inference speed. This is especially useful in scenarios where models need to be deployed on devices with limited hardware resources, like edge devices or drones.

Pruning significantly reduces the memory footprint of deep learning models. Since many DNNs, particularly transformer-based models used for complex tasks like image classification or human pose estimation, contain millions of parameters, their storage and processing require substantial memory. By pruning unimportant weights, the size of the model is reduced, allowing it to be stored and executed more efficiently in environments with limited memory, such as embedded systems or small-scale devices like nano-drones.

One of the critical benefits of pruning is the creation of more energy-efficient models.
In devices with limited battery life, such as nano-drones or mobile platforms, reducing energy consumption is a primary concern. This results in longer operational times for compact battery-powered devices and lower energy costs.

It is particularly important for deploying deep learning models in resource-constrained environments like embedded systems, and nano-drones, where memory, computational power, and energy efficiency are limited.

The goal of pruning is to create a smaller, more efficient model with fewer parameters, while still achieving near-original accuracy. This makes it possible to deploy deep learning models on hardware with limited resources without sacrificing much performance. There are several types of pruning techniques, each designed to optimize different parts of the neural network.

There are two main types of pruning that can be applied, unstructured pruning, and structured pruning, each offering different advantages and trade-offs depending on the specific deployment environment.
Unstructured pruning [16] involves removing individual weights from the neural network based on their magnitude or contribution to the final output. Typically, weights with values close to zero are pruned, creating a sparse network with fewer active connections between neurons. This form of pruning is highly flexible and can result in significant reductions in the number of parameters, but the resulting sparse matrix is often less efficient to process on standard hardware, which prefers dense matrix operations.

Structured pruning [17] operates at a higher level of abstraction than unstructured pruning, removing entire components of the network, such as neurons, channels, or even layers. By eliminating entire filters or neurons, structured pruning leads to a more hardware-friendly model with a reduced number of computational operations.

Pruning can also be performed in an HW-aware [18] manner, taking into account the specific hardware on which the pruned model will be deployed. It is tailored to optimize the model's performance on a particular platform, such as a mobile device, GPU, or edge AI accelerator. The goal is to prune the network in a way that maximizes efficiency for the target hardware's architecture, considering factors such as memory, computation power, and bandwidth.

One of the primary challenges of pruning is the trade-off between accuracy and compression. Aggressive pruning can lead to significant reductions in model size and computation but may result in degraded accuracy if important weights or structures are removed. Striking the right balance between model compression and maintaining accuracy is critical to the success of pruning, especially in applications where precision is paramount, such as in human pose estimation for drones or medical imaging.

Chapter 3

Related works

This chapter provides SoA approaches and techniques explored in recent years for HPE 3.1, then we will present efficient models for this task deployable to constraint-resources devices 3.2 and finally explore pruning techniques applied to ViT 3.3.

3.1 Human Pose Estimation Models

The problem of recognizing human pose has been studied extensively in recent literature. Early breakthroughs laid the foundation for modern deep learning approaches, transitioning from traditional methods to powerful neural network-based solutions.

Introduced by Google in 2014, DeepPose [19] is a DNN-based approach for human pose estimation by formulating it as a regression problem towards body joints. The method employs a cascade of DNN regressors that significantly improve the precision of pose estimation. This approach leverages advances in deep learning and offers SoA performance across several benchmarks.
Rather than depending on part-based models, which are constrained in terms of expressiveness, the DNN method leverages the entire image to predict joint positions, eliminating the necessity for predefined feature detectors or graphical models.
The model captures contextual information about the entire pose by using the full image rather than focusing on individual body parts in isolation, this approach avoids complex handcrafted features and part-based models, making it a more straightforward and scalable solution for pose estimation.

Another interesting work that follows a different approach for 2D HPE is [20] where the authors propose a bottom-up approach for real-time multi-person 2D pose estimation using Part Affinity Fields (PAFs), a non-parametric representation, that allows learning the association between detected body parts and individuals, enabling pose estimation to scale efficiently, irrespective of the number of people in the image. The method demonstrates state-of-the-art accuracy while maintaining a satisfactory real-time response latency.
The proposed architecture is designed to learn both body part detection and the relationships between body parts simultaneously, creating a dual-task framework that enhances the overall accuracy of pose estimation. By optimizing these tasks together, the model achieves more precise results. A key innovation in this approach is its novel representation, which encodes not only the location but also the orientation of limbs. The model operates through multiple stages, progressively refining its detection and association of body parts at each step. This iterative process improves accuracy with every pass.
The approach's strengths include scalability, as it efficiently handles crowded scenes by decoupling runtime from the number of people. It offers real-time performance with constant inference time, making it ideal for live applications. Additionally, its use of Part Affinity Fields (PAFs) ensures robustness, enabling accurate pose estimation despite occlusions, scale variations, and complex interactions

In [21] the first single-network approach for whole-body 2D pose estimation is presented, it can estimate key points for the body, face, hands, and feet from in-the-wild images of multiple people. This new method is a significant improvement over the original OpenPose, which required multiple networks and suffered from runtime issues proportional to the number of people detected. The proposed system operates in real-time and improves accuracy, especially for low-resolution and occluded keypoints, making it a more efficient and accurate system for various applications.
In this research the proposed method combines the detection of body, face, hands, and feet key points into a single network, eliminating the need for multiple separate detectors for different single key points. This makes the system much faster and more scalable for multi-person scenarios.
The single-network model greatly reduces the inference time, especially in scenes with multiple people, providing nearly constant runtime regardless of the number of individuals. Furthermore, the network shows significant improvements in detecting key points in challenging conditions, such as occlusions, motion blur, and small objects.

All these three research address the HPE task in a 2D fashion, and thanks to DNNs, OpenPose, and the PAF-based approach aim for real-time performances.

DensePose [22] represents a novel approach to human pose estimation by establishing dense correspondences between an RGB image and a 3D surface model of the human body.
The proposed architecture combines fully convolutional networks with region-based Mask-RCNN architectures, improving accuracy in detecting and mapping dense human poses. This system is optimized to work in real-time, operating at around 20-26 frames per second for lower-resolution images, making it suitable for applications such as video processing, augmented reality, and interactive graphics.
This model achieve to handle real-world complexities like occlusion, clothing, and scale variation, outperforming prior work that focused on key points detection.

HoloPose [23] integrates multi-task learning, training the model to predict both 2D and 3D key points and integrating DensePose-style mapping with additional 3D pose and shape estimation capabilities. It refines the model through an iterative optimization process that aligns top-down 3D model predictions with bottom-up CNN-based pose estimates. This allows for both high spatial accuracy and global consistency, operating at over 10 frames per second, even in real-world settings with occlusion and clutter.
Unlike previous monolithic approaches that apply CNNs over an entire object's bounding box, HoloPose uses part-based modeling. This model extracts features around specific joints, ensuring more localized information and improved robustness to articulation, occlusion, and global translations.
By using a part-based feature extraction approach, HoloPose handles occlusions and complex poses more effectively than monolithic models. This results in improved accuracy in real-world images with multiple people or complex interactions, and can process images at more than 10 frames per second, making it suitable for interactive applications or video analysis in dynamic environments, achieving state-of-the-art performance on various benchmarks.

In [24] propose HRNet, an innovative method for human pose estimation, focusing on maintaining high-resolution representations throughout the network, rather than recovering resolution from lower levels as seen in other approaches.

Unlike traditional methods that rely on downsampling and later upsampling to recover high-resolution representations, HRNet maintains high-resolution features throughout the network. The architecture, by connecting multiple subnetworks of different resolutions in parallel, facilitates repeated multi-scale fusion. Each high-to-low resolution subnetwork continuously exchanges information with others, enriching high-resolution representations with multi-scale information.

The ability to retain high-resolution information throughout the process leads to more spatially accurate predictions, making HRNet one of the top performers in human pose estimation.

3.2 Efficient HPE models

All the models discussed in the previous section introduce novel architectures and paradigms for achieving high accuracy and real-time HPE. However, these models typically require substantial computational power and energy, making them impractical for deployment on resource-constrained devices. To address this limitation, recent advancements have focused on developing efficient HPE models that maintain strong performance while reducing resource demands. This section will present models specifically optimized for deployment on edge devices and nano-drones, highlighting approaches that enable real-time HPE in limited-resource environments.

An innovative platform developed to enable high-performance computing at ultra-low power levels consists of the Parallel Ultra Low Power (PULP) system family, ideal for energy-constrained devices such as nano-UAVs, IoT nodes, and edge computing applications. PULP systems like the GAP8 System-on-Chip (SoC) integrate tightly coupled cores and memory hierarchies, allowing for efficient data movement and computation.

There have been already several works like MobileNet [25], FrontNet [5] and Dronet [26] addressing unique challenges in human-drone pose estimation and interaction, autonomous navigation, and network optimization, demonstrating the versatility and power of PULP-based architectures in energy-constrained environments.
MobileNet is an efficient neural network model designed specifically for mobile and embedded applications. Its lightweight architecture uses depthwise separable convolutions, significantly reducing computational cost without sacrificing accuracy.
FrontNet is designed for robust front-view perception, addressing challenges in tasks like human pose estimation and obstacle detection from a forward-facing perspective. Tailored for edge devices, FrontNet's architecture is streamlined to handle the high-speed, real-time demands of drone-based applications.
DroNet targets autonomous navigation in urban environments, focusing on predicting safe paths for drones using a convolutional neural network trained to handle urban traffic patterns. It employs end-to-end learning to make navigation decisions based on visual input, enabling drones to interpret and respond to urban dynamics in real time.

In their work with the use of GAP8 hardware demonstrates the potential for deploying complex AI tasks on ultra-low-power devices, opening doors for small-scale, intelligent drones.
All papers focuses on enabling real-time inference on nano-UAVs, emphasizing the need for fast, onboard visual processing for navigation and human interaction. The CNN models are optimized to handle visual data in real-time, meeting frame rate requirements for smooth and responsive operation during flight.
To fit within the limited computational resources of nano-UAVs, all three papers employ aggressive model compression techniques, such as quantization and pruning, reducing both memory footprint and computational load, and each paper showcases how different compression methods can maintain high levels of accuracy and inference speed, a necessity for small-scale robotics.

Another work that tackles the challenge of efficient HPE is *Fast Human Pose Estimation* [27], that proposes a Fast Pose Distillation (FPD) approach, a novel training strategy that leverages knowledge distillation specifically for human pose estimation. In this approach, a large teacher model is used to train a smaller student network, which distills knowledge through a mimicry loss function that aligns the outputs of the student model with the teacher's predictions.

The authors design a compact variant of the Hourglass [28] network by reducing the depth and width of layers. This architecture achieves efficient inference with minimal loss in accuracy. Their proposed FPD model achieves near SoA accuracy on different datasets with only a fraction of the computational cost of the Hourglass model.

Adaptive [29] approaches are explored as well where authors propose an adaptive deep learning framework that combines two SoA CNNs with novel adaptation strategies to optimize HPE on nano-drones. By switching between CNNs based on the assessed complexity of visual input, this adaptive system maintains high accuracy while reducing computational latency and power consumption.

In [30], the authors explore HPE for nano-drones equipped with GAP8 SoC and limited sensory resources. By integrating depth and camera images using a CNN trained in a simulated environment, they achieve robust real-world performance.
The proposed model improves pose estimation accuracy significantly over camera-only baselines, making it suitable for challenging real-world conditions where reliable 3D positional data is essential.

3.3 Efficient Vision Transformers

The efficient models presented in the previous section primarily utilized CNN-based architectures and were specifically optimized for deployment on low-resource hardware, such as the GAP8 SoC. Considering the success of these CNN approaches on hardware like GAP8, the next section will explore research on efficient ViTs, aimed at extending the capabilities of Transformer-based architectures to resource-limited devices.

In [31], the authors propose an approach to prune Vision Transformers, addressing the high computational and storage demands of ViTs, which limit their deployment on mobile and embedded devices. The authors propose a pruning method that enforces sparsity on the dimension-wise components of each layer within the transformer. By learning importance scores for each dimension, the model identifies and prunes less significant dimensions, achieving a high compression ratio with minimal accuracy loss.

This pruning method reduces computational demands, enabling ViTs to run on mobile and embedded devices where computational resources are limited, and the strategy is designed to identify and preserve essential dimensions, maintaining high accuracy while achieving significant compression.

In [30], the authors addresses the high computational costs of ViTs by introducing a patch slimming technique that prunes redundant patches at each layer of the transformer, starting from the final layer and moving backward. The approach calculates each patch's impact on the model's final output, allowing the removal of patches that contribute minimally.

The algorithm calculates an importance score for each patch by assessing its contribution to the model's final feature representation. Patches with lower scores are identified as redundant and pruned. After pruning, the model's architecture forms a pyramid-like structure, where fewer patches are preserved in deeper layers. This structure aligns with the increased redundancy found in deeper transformer layers, where the attention mechanism aggregates features across patches.

Patch slimming significantly reduces the number of computations required for ViTs, making them more viable for resource-constrained environments, such as mobile and IoT devices.

Chapter 4

Methodology

This chapter outlines the methodology employed in this study, in Section 4.1 an analysis of the performance of the ViT architecture on the Visual Pose Estimation task is presented. In Section 4.2 an overview of the architectural modifications proposed to improve the performance-cost trade-off is reported. In Section 4.3 a description of the structured pruning techniques adopted to optimize the size of the model is provided.

4.1 Visual Pose Estimation benchmarks

As outlined in Section 2.1, the HPE task in this study involves predicting the 3D positional coordinates (x, y, z) and the rotational angle ϕ of the subject with respect to the drone's camera. This task is framed as a multi-output regression problem, aiming to estimate both the relative 3D spatial pose and rotational orientation of a human target with respect to the drone position in its environment.
The drone's objective is to locate and recognize a subject by detecting their face, then position itself directly in front of the target's face, dynamically maintaining this alignment as the person moves.

Figure 4.1 depicts a schematic representation of the task.
The figure provides a visualization of the drone's perception and prediction process during the human pose estimation task. On the right, the grayscale image represents the frame captured by the camera mounted on the drone. On the left, the drone's perceived perspective is illustrated, with the blue triangular region indicating the area of interest within the drone's field of view.
The green point represents the ground-truth coordinates of the target in space, while the blue point corresponds to the predicted coordinates generated by the model based on the captured image. The arrows illustrate the angular relationships:

the arrow for the green point depicts the true angle relative to the drone, while the arrow for the blue point represents the predicted angle based on the model's output. This visualization highlights the spatial reasoning applied by the drone to map its environment and predict human pose coordinates.

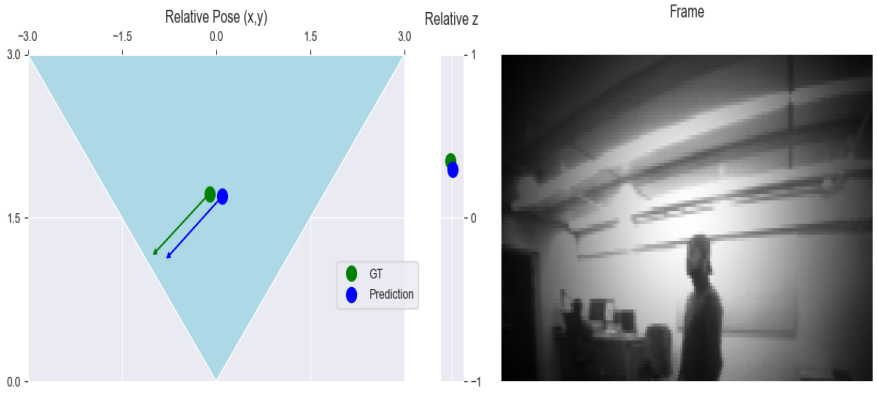

Figure 4.1: Schematic representation of the vision task from the drone point of view

Dataset

The dataset used in this work consists of images extracted from short videos recorded in two laboratories, Manno and Viganello.
All the image frames have been captured by a grayscale camera mounted aboard the nano-drone. Unlike standard RGB cameras that capture images using three color channels (Red, Green, and Blue), grayscale cameras use a single intensity channel, recording variations in light intensity without color information. This single-channel format captures essential visual details but omits the color data present in RGB images.

Although the videos were recorded in two different laboratories, the number of frames available as images was limited. To address this, data augmentation was employed—a technique that expands the size and diversity of a dataset by applying various transformations to the existing data. This approach enhances model generalization and performance by making the model more adaptable and robust, reducing the risk of overfitting, and improving performance on unseen data, which is particularly beneficial when data is limited.

The specific data augmentation techniques applied included cropping, noise injection, blurring, vignetting, and exposure adjustments. These transformations increased the variety within the dataset, allowing the model to train on a broader range of visual scenarios.

Figure 4.2 illustrates the application of different augmentation techniques to a frame sampled from the dataset. This process effectively expands the dataset, offering the model a broader diversity of training examples to improve learning and generalization.

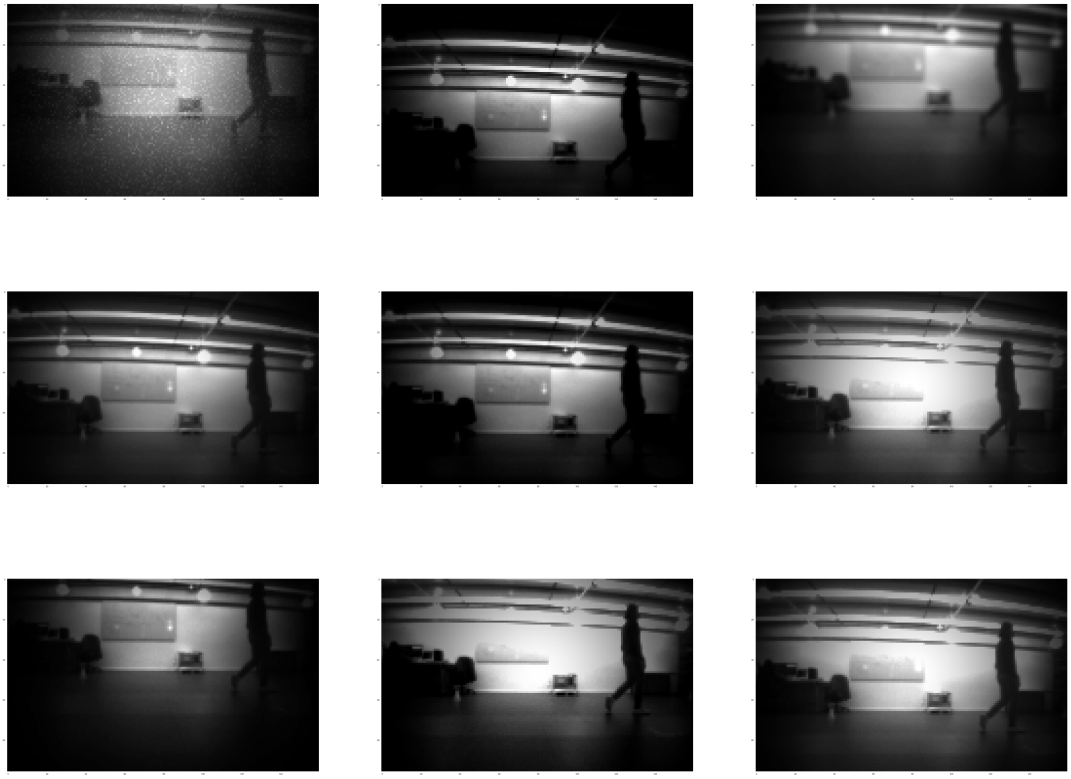

Figure 4.2: Effect of the application of multiple augmentation techniques on a frame of the dataset.

In machine learning, the dataset is first divided into training and test sets, with the test set kept completely separate to ensure unbiased evaluation of the model's performance on unseen data. The training set is then further split into training and validation subsets to facilitate monitoring and optimization during the training process.
Typically, 25% of the training data is randomly sampled to create the validation

set. This validation set plays a crucial role in assessing the model's generalization ability during training and in tuning hyperparameters to prevent overfitting. The remaining 75% of the training data is used to train the model, enabling it to learn underlying patterns and relationships by optimizing its parameters. This structured allocation ensures that the model is exposed to sufficient data for learning while the validation set provides continuous feedback.
The independent test set is reserved to evaluate the model's final performance on unseen data. This 75/25 structure, along with a distinct test set, ensures effective learning, real-time validation, and an unbiased final assessment of model performance.

To ensure consistency across the dataset, the target values the model aims to predict were carefully inspected within the training, validation, and test sets. This review confirmed that target values are coherent and evenly distributed across these subsets, providing the model with consistent data for both training and evaluation. Figure 4.3 illustrates this data consistency, highlighting uniformity across all dataset portions.

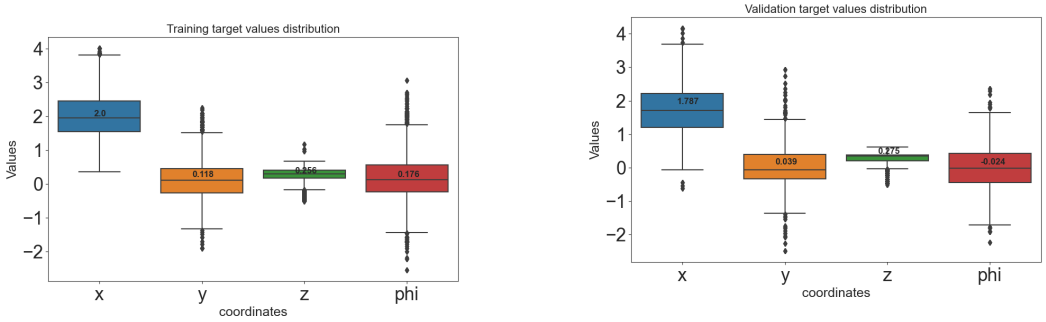

Figure 4.3: Distribution of the target values in training and validation are very similar.

Platform

To fully understand the resource constraints inherent to this study, the key characteristics of the target platform under investigation will be briefly outlined. Figure 4.4 provides an illustration of the nano-drone used in this research.
Its characteristics and specifications are the following as detailed in its datasheet [1].

[1]Datasheet link

1. **Onboard MCUs**
   (a) STM32F405 main application MCU (Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash)
   (b) nRF51822 radio and power management MCU (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash
   (c) On-board LiPo charger with 100mA, 500mA and 980mA modes available
   (d) 8KB
2. **Phisical specifications**
   (a) Takeoff weight: 27g
   (b) Size (WxHxD): 92x92x29mm (motor-to-motor and including motor mount feet)
3. **Flight specifications**
   (a) Flight time with stock battery: 7 minutes
   (b) Charging time with stock battery: 40 minutes
   (c) Max recommended payload weight: 15 g

Figure 4.4: Crazyflie 2.1 nano-drone

4.2 Vision Transformer Baselines

In this work, transfer learning [32] was utilized to improve model performance despite limited data and computational resources. This technique involves adapting a model pre-trained on a large dataset for a related task by fine-tuning it for a new, specific task. By leveraging the knowledge learned by the pre-trained model, transfer learning enables better results with fewer training examples and shorter training time.
Before stepping into the fine-tuning phase, the pre-trained Vision Transformer model used was retrieved with the timm library, which loads model weights from the Hugging Face hub.

To establish baselines, adjustments were made to ensure compatibility between the pre-trained models and our dataset. The pre-trained models were originally trained on color images with three channels (R, G, B), while our dataset consists of grayscale images with only one channel. This difference in channel count made the data incompatible with the pre-trained model's architecture. To resolve this, a preprocessing step was applied, replicating the single grayscale channel across all three RGB channels, thereby aligning the data with the model's input structure.

The original ViT architecture was designed for classification tasks, incorporating an MLP head with two fully connected layers and a GeLU activation function. However, our task involves regression predicting four values, the subject's three spatial coordinates (x, y, z) and its angle relative to the drone (phi). To adapt the model for this purpose, the MLP head was replaced with a single fully connected layer containing four output nodes corresponding to these coordinates and angle. No activation function was applied, allowing the model to produce continuous values suitable for regression.

To assess the effectiveness of training without transfer learning, we also trained a model from scratch on the available dataset, without using pre-trained weights or channel replication as required for pre-trained models. Training this model independently allows us to evaluate the impact of transfer learning on performance and determine if our dataset alone provides enough data for effective learning.

This approach establishes a baseline comparison between the two models, one that uses transfer learning with adapted input channels and the other trained solely on our dataset. Comparing their performances enables us to assess the impact of pre-training and identify any limitations in training effectiveness when relying exclusively on our data. This evaluation ultimately helps determine whether the dataset is large and diverse enough to support robust learning without external data sources.

The selection of the ViT model was based on a search through pre-trained models available in the timm library, which loads model weights from the Hugging Face hub. Among the smallest ViT models, the parameter counts range from 5.7 million to 12 million, highlighting the stark contrast between transformer-based and CNN-based models. ViT models carry a substantial parameter load, while MobileNet, by comparison, has only 47k parameters. This difference underscores the significant computational resources typically required by ViT architectures compared to CNNs.

The final choice of pre-trained model weights focused on the smallest available model [2]. The model is specifically designed to process images at a resolution of 224x224 and, after further training and fine-tuning on our dataset, has a total of 5,499,076 parameters which represents an increase in parameters of almost 11,600%.

4.3 Vision Transformer Optimization

To compress and optimize the ViT model structured pruning has been applied.
As discussed earlier in 2.3.2, key components of ViTs are MHSA, FFN, and layer normalization. Structured pruning aims to remove either entire modules or sub-structures to improve efficiency. Furthermore, structured pruning can provide direct acceleration by reducing computations.
For example, pruning attention heads reduces the dimensionality of the key, value, and query projections. This decreases the overall scaled dot product attention costs.

Here structured pruning is implemented through a training scheme with learnable binary masks. The core component of this approach involves applying learnable masks to transformer self-attention layers called Block Movement Pruning for MHSA.

Block Movement Pruning for MHSA involves masking less important components in the self-attention mechanism based on computed importance scores. This approach ensures that only the most significant portions of each query, key, and value matrix are maintained, thereby enhancing efficiency without heavily impacting performance.

Each attention head i is assigned an importance score S_H^i, and each column j in the query and key projections, as well as each column z in the value projections,

[2] *vit_tiny_patch16_224.augreg_in21k_ft_in1k*: initially trained on ImageNet-21k and later fine-tuned on ImageNet-1k with additional augmentation strategies

has scores S_{QK}^j and S_V^z respectively. Using a threshold τ, binary masks M_H^i, M_{QK}^j, and M_V^z are created as:

$$M_H^i = \mathbb{1}(S_H^i \geq \tau), \quad M_{QK}^j = \mathbb{1}(S_{QK}^j \geq \tau), \quad M_V^z = \mathbb{1}(S_V^z \geq \tau),$$

where $\mathbb{1}$ denotes the indicator function, setting the mask to 1 if the score is above the threshold and 0 otherwise.

Once the masks are applied, the attention head i is computed as [17]:

$$\text{Head}_i(X) = M_H^i \cdot \text{Softmax}\left(\frac{\left(XW_Q^i \odot M_{QK}\right)\left(XW_K^i \odot M_{QK}\right)^T}{\sqrt{\sum_j M_{QK}^j}}\right) \cdot \left(XW_V^i \odot M_V\right),$$

where $\odot$ indicates element-wise multiplication and W_Q^i, W_K^i, W_V^i are the query, key, and value weight matrices for head i. Only the unmasked elements contribute to the attention score, thus reducing the computation while retaining essential components.

Along with pruning for MHSA, pruning for normalization layers was also employed.

Block Movement Pruning for Layer Normalization (LN) prunes dimensions within each token representation, making LN more efficient in terms of computation and memory.
LN normalizes token representations across channels independently per token, for input $X \in \mathbb{R}^{L \times d}$ with L tokens of dimension d, LN normalizes each token X_i as:

$$\text{LN}(X)_i = \gamma \cdot \frac{X_i - \mu_i}{\sigma_i} + \beta,$$

where μ_i and σ_i are the mean and standard deviation of token i, calculated over its d dimensions, and $\gamma, \beta \in \mathbb{R}^d$ are learnable parameters.

To each dimension c in X is assigned a score S^c, and a mask M^c is applied as:

$$M^c = \mathbb{1}(S^c \geq \tau),$$

where τ is a threshold. This mask zeroes out dimension c if its score is below the threshold, and the mean and standard deviation are recalculated excluding the

pruned dimensions [17]:

$$\mu_i = \frac{\sum_c M^c X_{i,c}}{\sum_c M^c}, \quad \sigma_i = \sqrt{\frac{\sum_c M^c (X_{i,c} - \mu_i)^2}{\sum_c M^c}}.$$

After masking, the normalized output for each token becomes:

$$\text{Pruned LN}(X)_i = \left(\frac{X_i - \mu_i}{\sigma_i} \cdot \gamma + \beta \right) \odot M,$$

where only the active (unmasked) dimensions contribute to the normalization, achieving a more compact and efficient LN layer.

In the structured pruning approach employs mask sharing to ensure consistent pruning across residual-connected components, enabling efficient and structurally coherent pruning. The default strategy, referred to as full sharing, applies a single pruning mask across key components of the model, including the patch embedding layer, positional embeddings, attention outputs, FFN outputs.
Full sharing ensures that once a dimension is pruned in one component, it is consistently removed across all subsequent layers, aligning with the architecture's residual connections. This strategy achieves significant parameter reduction and computational savings while maintaining structural consistency. However, it reduces pruning granularity, as pruning a dimension impacts all layers sharing the mask. Despite this limitation, full sharing is the default approach due to its balance between efficiency and simplicity, making it ideal for resource-constrained environments like nano-drones.

Moreover the optimization process seeks to balance model pruning with task performance, formulated as a combined objective that minimizes two types of losses: the task-specific loss, denoted $\mathcal{L}_{\text{task}}$, and a regularization term, $\mathcal{L}_{\text{cost}}$, which induces sparsity by encouraging the model to prune less important components. The objective function can be expressed as [17]:

$$\min_{W,S} \ \mathcal{L}_{\text{task}}(W; S) + \lambda \mathcal{L}_{\text{cost}}(S),$$

where W represents the model's weights, S is a set of binary masks applied to specific structures, and λ is a hyperparameter that plays a critical role in balancing the two loss components. Each component has a distinct purpose: $\mathcal{L}_{\text{task}}$ measures the model's accuracy on the primary task, while $\mathcal{L}_{\text{cost}}$ guides the pruning process to improve efficiency.

The hyperparameter λ serves as a weighting factor for the sparsity-inducing regularization term $\mathcal{L}_{\text{cost}}(S)$. This weight fundamentally influences how the optimization

balances task accuracy against model sparsity, acting as a control mechanism for the pruning process.

When λ is set to a high value, the regularization term $\mathcal{L}_{\text{cost}}(S)$ gains more influence relative to $\mathcal{L}_{\text{task}}$, shifting the optimization's focus toward achieving greater sparsity. In this case, the model aggressively prunes components with lower importance scores, even if it may slightly impact task accuracy. The high emphasis on regularization leads to a more compact model with reduced computational demands, making it suitable for deployment in environments with very limited resources. However, if λ is set too high, the model may end up pruning critical components, leading to a drop in accuracy and potentially compromising its effectiveness on the primary task.

On the other hand, a small λ value diminishes the weight of $\mathcal{L}_{\text{cost}}(S)$ in the optimization process, resulting in a model that prioritizes task accuracy over sparsity. In this configuration, fewer parameters are pruned, and the model retains more of its original structure, leading to minimal impact on accuracy. This choice is often preferable when maintaining high task performance is crucial and there is flexibility regarding the computational resources available. In scenarios where task accuracy is of utmost importance, a lower λ helps ensure that pruning does not sacrifice any vital elements of the model's structure.

Finding the optimal λ value is essential to balance accuracy with efficiency. In practice, this value is typically selected through hyperparameter tuning, where various values of λ are tested, and the model's performance on validation data is evaluated. In this way, it is possible to obtain a Pareto front of architectures, each representing a Pareto-optimal model in terms of size vs accuracy on the task. By effectively adjusting λ, the optimization process can yield a model that retains the components necessary for high accuracy while significantly reducing the computational and memory footprint through selective pruning.

Chapter 5

Results

This chapter presents the results of our experiments, focusing on the performance of the Vision Transformer model in comparison to MobileNet.
We begin by evaluating the ViT models trained on our dataset, assessing their accuracy and generalization ability in section 5.1. Next, these results are directly compared to MobileNet's performance in section 5.2, highlighting differences in learning efficiency and model robustness under identical conditions.
Finally, section 5.3 examines the impact of compression optimization techniques applied to the ViT models. This comparative analysis offers insights into the trade-offs between model complexity, efficiency, and predictive performance.

5.1 Training and evaluations

During the training phase, multiple optimizers and learning rate schedulers were evaluated to determine the optimal configuration. The optimizers tested included SGD [33], Adam [34], and AdamW [35], while the learning rate schedulers considered were ReduceLROnPlateau, CosineAnnealing, and CosineAnnealingWarmRestarts [36]. The final configuration selected Adam as the optimizer for its high performance and CosineAnnealingWarmRestarts as the learning rate scheduler. CosineAnnealingWarmRestarts combines cosine annealing with warm restarts, promoting faster convergence and enhanced performance in deep neural network training.

Early stopping was also implemented to ensure efficient training. This technique halts the training process if there is no improvement in the validation loss for a specified number of epochs. In this case, early stopping was configured with a patience of 10 epochs.

Initially, the dataset collected in Manno was used for exploration and analysis. While the MobileNet model performed well on this dataset, the ViT model showed significant overfitting. This was evident from the high loss values observed on the training set and the low loss values on the validation set; however, the behavior was inconsistent, with a substantial gap between the training and validation losses. Further confirmation of overfitting was provided by the model's poor performance on the test set. Figure 5.1 illustrates the overfitting trend in the loss curves observed on the Manno dataset.

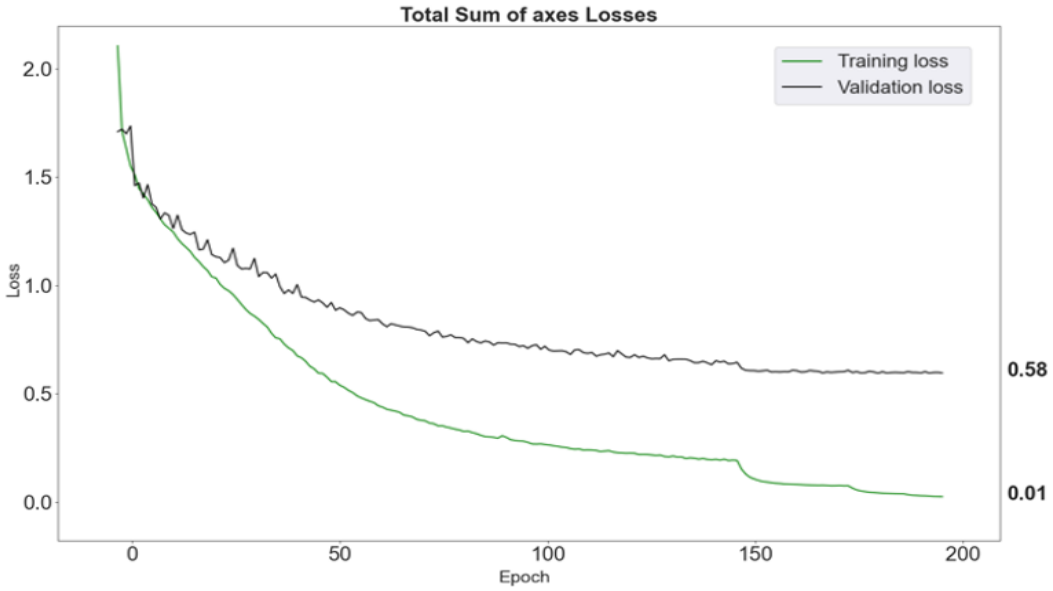

Figure 5.1: The plot illustrates the aggregated losses of the target variables. The green line represents the training loss, which exhibits significant overfitting, while the black line represents the validation loss. The validation loss plateaus, at a far distance from the training loss, indicating limited generalization.

To address the overfitting issue, several strategies were explored. Initial efforts involved experimenting with mixed combinations of training and validation data from the two datasets (Manno and Viganello).
To further expand the training data, the datasets were merged into a single, balanced dataset, however, these adjustments did not result in significant improvement.

Since the validation set was originally created through random sampling from the training set, and given that each dataset comprised frames extracted from video recordings, it was suspected that the random selection of frames might allow for easier interpolation, potentially contributing to overfitting on the training set.

To address this, the dataset was reorganized with a new structure for the training and validation sets. In the final approach, the first contiguous 75% of the dataset was allocated for training, while the remaining contiguous 25% was used for validation. This contiguous segmentation minimized frame overlap between the sets, aiming to reduce overfitting. Additionally, the test set was restructured by merging test data from both datasets, resulting in a new, combined test set.

This new combination of dataset structure and scheduler resulted in a marked improvement, as evidenced by the alignment in the training and validation loss curves, indicating more stable model learning. Figure 5.2 illustrates the updated loss curves, where both the training and validation losses follow a consistent decreasing trend. This marks a significant improvement compared to previous results, where the validation loss plateaued while the training loss continued to decrease, indicating overfitting.
With the new configuration, the training loss increased to 0.187, up from 0.01, reflecting a more balanced learning process. At the same time, the validation loss showed a substantial 35% improvement, decreasing from 0.58 to 0.3771. This improvement in validation loss highlights better generalization, suggesting that the updated setup effectively mitigated overfitting while maintaining strong performance.

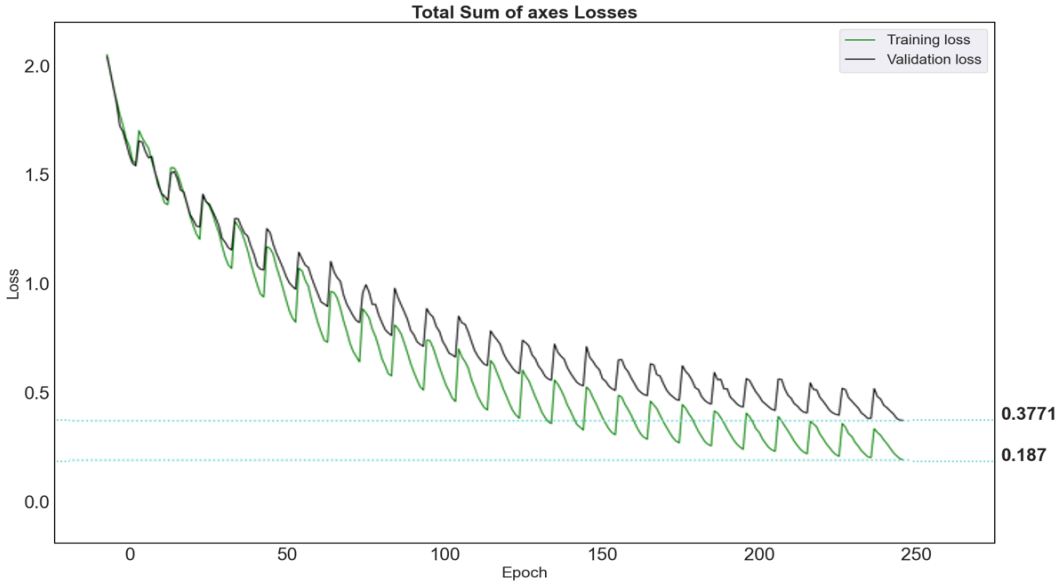

Figure 5.2: The plot illustrates the aggregated losses of the target variables. The cosine similar trend is due to the cosineAnnealinghWarmRestart. The green line represents the training loss while the black line represents the validation loss.

The ViT model architecture follows a structured sequence, beginning with a patch embedding layer, followed by a series of MHSA blocks, and concluding with a head layer specifically adapted for regression. By stacking multiple MHSA blocks, the model is able to learn progressively complex and hierarchical representations, with each block building upon the outputs of the previous one to refine its understanding of image structure. However, increasing the depth of these blocks also increases the model's complexity and parameter count, which can lead to overcomplexity and potential overfitting, especially on smaller datasets.
To address this issue, we experimented with reducing the number of MHSA blocks from the original 12, decrementing by two at each stage. This exploration helped identify an optimal model depth that minimized validation error while keeping complexity manageable. Through this process, we determined that a configuration with 4 MHSA blocks achieved the best balance between performance and efficiency. The parameter count was reduced by 65%, decreasing from 5,499,076 parameters in the model with 12 blocks to 1,940,164 in the model with 4 blocks. While the validation loss increased from 0.1673 to 0.4748, the test loss showed a significant improvement of 16%, decreasing from 1.3189 to 1.1059 when the original dataset was considered. This highlights the ability of the 4-block configuration to generalize better despite a modest increase in validation loss, making it an efficient choice for resource-constrained environments.

This exploration was further enhanced by applying data normalization, scaling pixel values of the images to a range between 0 and 1, to assess whether the ViT model would benefit from this preprocessing step.
Based on previous experience and experimentation, normalization had shown neither positive nor negative effects on the MobileNet model, with its performance remaining consistent across different training scenarios.
In contrast, normalization produced mixed results for the ViT model. When using the new dataset structure compared to the original, the training and validation loss curves showed partial degradation, with losses generally around 1.34, compared to values below 0.5 in the original dataset. Despite this increase in loss during training and validation, the test results indicated improved generalization, with losses ranging between 1.12 and 1.19, whereas the original dataset yielded higher test losses, peaking at 1.32. This suggests that normalization, combined with the new dataset structure, enhances the model's ability to generalize to unseen data. This finding implies that, although normalization may not directly improve ViT's training or validation performance, it enhances the model's robustness on unseen data, offering a promising approach for achieving better overall generalization.

The best-performing ViT model was then compared to the MobileNet model, with both models trained on the same updated dataset version and with the

application of data normalization to ensure consistent training conditions. This normalization, which scaled input values to a standard range, enabled a fair comparison that highlighted differences attributable solely to model architecture.
Results from these comparisons indicate that MobileNet's performance remains stable regardless of data normalization. In contrast, the ViT model struggles to learn meaningful features without normalization, while its performance improves significantly when normalization is applied, even exceeding the performance achieved by MobileNet. Figure 5.3 illustrates that, in the absence of data normalization, MobileNet exhibits significantly lower losses compared to the various ViT configurations. However, when normalization is applied, the ViT models outperform MobileNet in nearly all configurations, demonstrating their superior ability to leverage normalized data for improved performance.

For each dataset configuration, a trivial model is created to serve as a baseline, capable of predicting only the mean value for each input. The mean values were calculated based on the test set of the two configurations, one derived from the original dataset and the other from the new dataset configuration. This baseline helps in assessing the models' ability to capture meaningful features beyond simple averages. Comparing each model to this baseline allows us to evaluate the extent to which they are learning relevant patterns in the data rather than merely approximating the mean.

Figure 5.3 presents the results of this analysis, illustrating the performance trends of the two models across different experimental setups. The bar plot compares the validation loss and test loss for the MobileNet (MNet) model and Vision Transformer (ViT) models with varying depths. In the notation ViT-N, N refers to the number of MHSA blocks in the architecture, providing a detailed comparison of performance across different configurations. In the upper row, the graphs display results obtained using the original dataset with random sampling for training and validation, while the lower row shows results with the restructured dataset, using a contiguous 75/25 split for training and validation, respectively. The left-hand side of the figure represents results without normalization, whereas the right-hand side shows results after data normalization within the [0,1] range.

Our experiments revealed notable differences in model performance depending on the dataset structure and the application of normalization. In the original dataset with random sampling (top-left graph), there was a clear tendency toward overfitting, particularly in the ViT architecture.

When the dataset was not normalized, the validation loss remained low for MobileNet and exceptionally low for ViT. However, the test loss was significantly

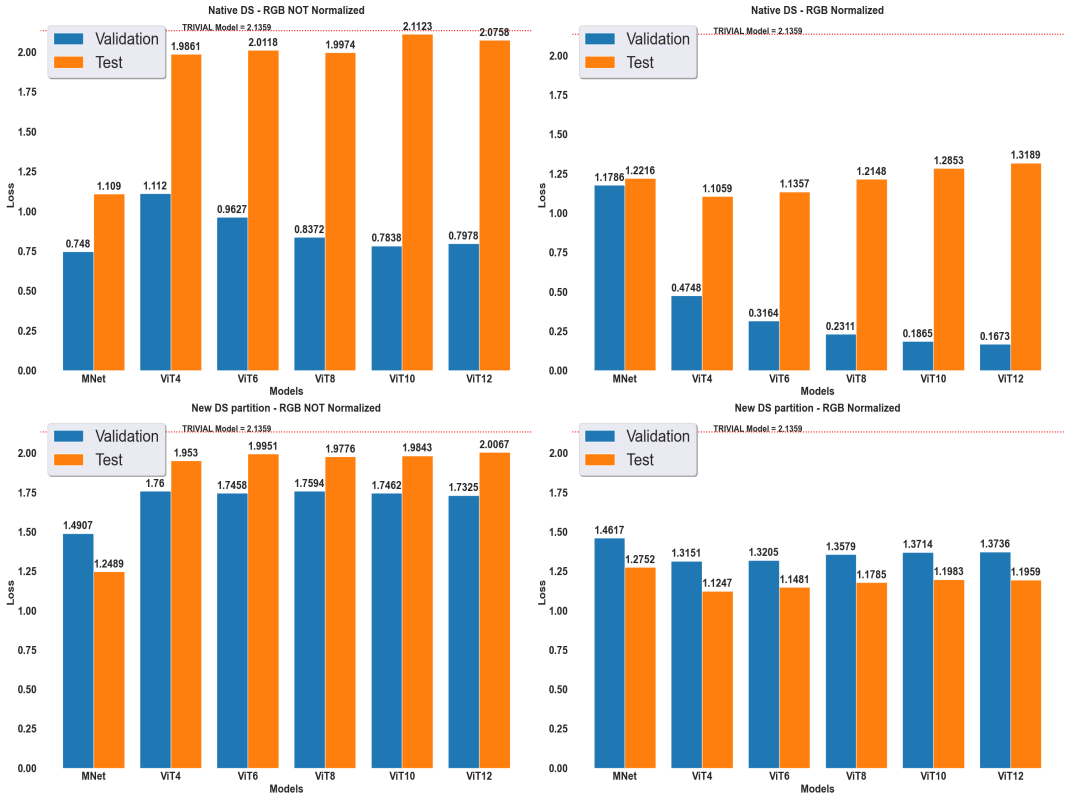

Figure 5.3: Regression performance comparison between the best ViT and MobileNet for each output coordinate.

higher, indicating poor generalization. For instance, in the ViT model with 12 MHSA blocks, the validation loss was 0.7978, while the test loss increased to 2.0758, representing a 160% performance degradation. For ViT models with reduced depths of 10, 8, 6, and 4 blocks, the corresponding performance deteriorations were 169%, 138%, 109%, and 78%, respectively.
When the dataset was normalized, the results demonstrated similar trends, though with slightly improved generalization. For the ViT model with 12 MHSA blocks, the validation loss was 0.1673, but the test loss rose to 1.3189, again indicating a 688% degradation. For models with 10, 8, 6, and 4 blocks, the performance degradations were 589%, 425%, 288%, and 132%, respectively.

These observations highlight that, despite normalization improving validation performance, the significant disparity between validation and test losses across all depths suggests limited generalization to unseen data.
The large discrepancy between validation and test performance suggests that the models were learning noise, patterns, and interpolations specific to the training

data rather than capturing generalizable features applicable to new samples. However, with the restructured dataset using a contiguous 75/25 train-validation split (bottom row), the models demonstrated improved generalization, though this adjustment alone did not significantly improve the test performance of the models.

The most significant improvement was observed when data normalization was applied, showed in the bottom-right bar plot. Normalizing the data to the [0,1] range led to a substantial enhancement in the performance of the ViT architecture, allowing it to generalize far more effectively. The ViT models trained on normalized data with the updated dataset structure outperformed the MobileNet model, particularly in test performance. For MobileNet, the validation loss was 1.4617, while the test loss decreased slightly to 1.2752, indicating stable but modest generalization. In contrast, the ViT model with 12 MHSA blocks achieved a validation loss of 1.3736 and an even lower test loss of 1.1959, reflecting a 13% improvement in performance. Similarly, for ViT models with reduced depths of 10, 8, 6, and 4 blocks, the corresponding performance improvements were 12.6%, 13%, 13%, and 14%, respectively.
Notably, the ViT models consistently achieved test losses that were lower than their validation losses, a strong indicator of robust generalization. These results demonstrate that normalization not only reduced overfitting but also enabled the ViT models to extract more meaningful patterns from the data, particularly when compared to the MobileNet model. Notably, the best-performing ViT model, ViT-4, outperformed MobileNet by 11.8%, further underscoring its superiority in terms of generalization and performance. This highlights the significant advantage of ViT architectures in effectively leveraging preprocessing techniques such as normalization to enhance both accuracy and generalization, even in shallower configurations.

In the end, the trivial model closely matched the performance of both models when trained on non-normalized datasets, indicating that, without proper preprocessing, the models struggled to learn meaningful features. However, once normalization was applied, the trivial model's performance fell significantly below that of the ViT and MobileNet models, underscoring the importance of dataset structure and preprocessing for effective learning in our HPE task.

5.2 Comparisons between Mobilenet and ViT

Both models were trained on the same dataset to ensure consistent experimental conditions. The primary metric for evaluating model performance was the Mean Absolute Error (MAE) for each of the four coordinate variables $(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \phi)$, as

well as the aggregated MAE across all target variables.
To provide a detailed view of model performance, Table 5.1 presents the regression results for each of the four output variables ($\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \phi$), comparing the best-performing ViT model (configured with a depth of 4 blocks) against the MobileNet model.

Table 5.1: MobileNet vs ViT results

Network loss	MAE				
	X	Y	Z	Φ	SUM
MNet Validation	0.4352	0.3019	0.2146	0.5093	1.4617
ViT Validation	0.3867	0.262	0.1648	0.503	1.3151
ViT improvements with respect to MNet	+14.8%	+13.21%	+23.2%	+1.23%	+10%
MNet Test	0.3269	0.1993	0.2647	0.4653	1.2752
ViT Test	0.3088	0.1677	0.1861	0.4588	1.1214
ViT improvements with respect to MNet	+5.53%	+15.85%	+29.69%	+1.4%	+12%

The results show that the ViT model outperformed MobileNet across all four output variables, achieving approximately a 10% improvement in validation MAE and a 12% improvement in test MAE. The performance gap was particularly notable in the test set, highlighting the ViT's superior generalization ability compared to MobileNet.

Results of our experiments, shown in Figure 5.4, indicate that the ViT models consistently outperformed MobileNet in terms of MAE across all four output variables. Data normalization proved to be a critical factor, significantly enhancing model performance by reducing overfitting and improving generalization. Due to its multi-head attention mechanism, the ViT model was able to learn more complex representations, achieving a 10-12% improvement over MobileNet, establishing it as the more suitable model for our task and preparing it for the next step, i.e. pruning. Additionally, the trivial model provided insights into the models' capacities, confirming that, without proper preprocessing, both models struggled to capture meaningful features.

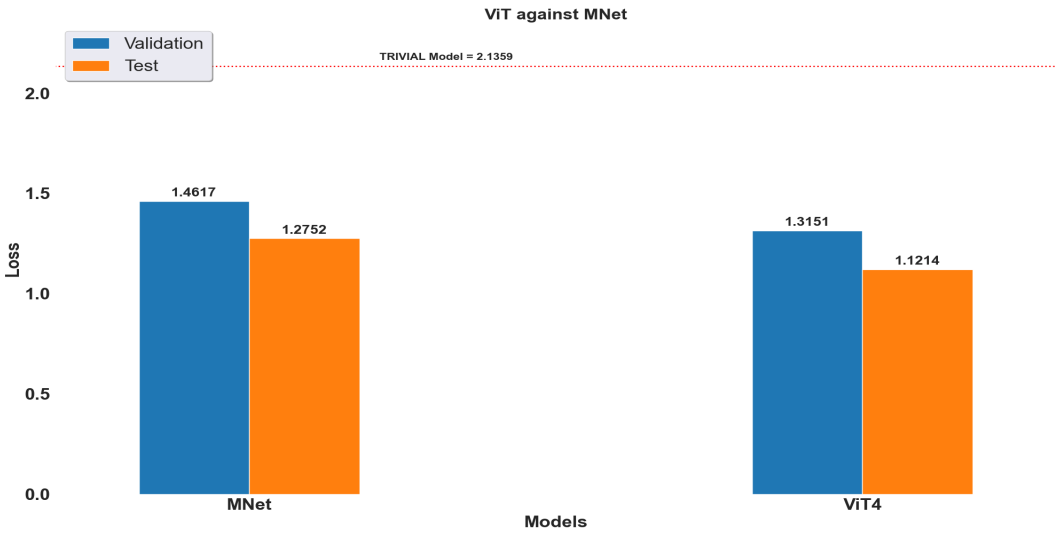

Figure 5.4: Summary comparisons between the selected ViT architecture against the MobileNet.

5.3 Pruning

Ultimately, the ViT model which demonstrated superior performance compared to MobileNet is selected for a pruning process to optimize its complexity and computational requirements while preserving or even enhancing its performance. This stage of the work focuses on identifying the most effective pruning strategies to achieve an optimal balance between model accuracy and efficiency, a balance that is essential for deployment in resource-constrained environments.

This portion of the research was conducted using the PLiNIO library [37], which provides a comprehensive suite of pre-built components designed for the development and evaluation of various pruning strategies. In this study, we focused on structured pruning. Our pruning approach employed a training algorithm that incorporates binary, learnable masks. Specifically, this algorithm introduces a binary mask importance score, $\mathcal{S}$, into the standard weight matrix of prunable weights, $\mathcal{W}$, enabling the model to identify which weights can be pruned [17].

In this study, a threshold value of 0.5 was applied during the structured pruning process. Importance scores were assigned to each mask, and components with scores greater than or equal to 0.5 were retained, as they were deemed essential to the model's performance. Conversely, components with scores below 0.5 were pruned by updating their masks to zero, effectively reducing the active parameters and simplifying the model architecture.

This threshold-based approach enables selective pruning of less important weights, resulting in a more efficient model with fewer parameters while preserving its learning capacity and performance. By pruning weights based on their relative importance, this method optimizes the model's structure, potentially enhancing its speed and reducing memory usage without significantly affecting accuracy.

To effectively optimize the two distinct loss functions involved in the pruning process, task loss and weight loss, two separate optimizers were utilized. These two optimizers worked on different masks, enabling each loss to independently guide the optimization of its respective objective. The final model optimization combined these two losses into an aggregated loss function, allowing precise control over both model accuracy and the degree of pruning.

A key factor in balancing these two losses is the hyparameter λ, which modulates the relative importance of the weight loss during training. The λ parameter effectively scales the weight loss to bring its magnitude in line with the task loss, preventing one loss from overpowering the optimization process. This scaling is essential, as the value of λ directly impacts the level of regularization applied to the weights, thereby influencing the extent of pruning. By adjusting λ, we can control the trade-off between model sparsity and task performance: larger values lead to more aggressive pruning, while smaller values retain a greater number of model parameters.

The pruning procedure was applied to the best-performing model identified in prior experiments, with adjustments to both the learning rate for the pruning optimizer and the regularization strength λ. The learning rate was set to 1e-5, while λ was varied across a range from 1e-3 to 1e-9 to explore different pruning intensities
Figure 5.5 presents the Pareto curve containing the pruned models, illustrating how variations in the λ parameter impact the trade-off between model size and performance.
This curve visually represents the balance between task loss and model complexity quantified in terms of number of parameters. By modifying λ, we observe the model's behavior under different regularization strengths—higher λ values result in more aggressive pruning with reduced task loss, while lower values retain more parameters at the expense of reduced sparsity. Adjusting λ in this way generates neural networks with varied architectures and performance levels, depending on the desired trade-off during optimization.

For example, with a regularization strength of λ = 1e-3, the pruning process was highly aggressive, resulting in an approximately 99% reduction in the number of

parameters. However, this level of pruning led to a considerable decrease in model performance, with an aggregated MAE of 3.7, as it removed weights essential for the model's ability to generalize effectively. Conversely, when λ was set to 1e-9, the pruning process was much more conservative, preserving a greater number of parameters around 800K and retaining more of the model's original performance with an aggregated MAE of 1.72.

From the Pareto analysis, shown in Figure 5.5, several optimal configurations were identified, offering the optimal balance between accuracy and number of parameters, with λ = 1e-6 providing the most favorable trade-off between model sparsity and accuracy. In this range, we achieved a parameter reduction of approximately 30% with respect to the original model, while incurring only a modest performance loss of around 14%. This finding demonstrates that, for our task, a substantial degree of pruning can be applied without significantly impacting model performance, presenting an efficient solution for reducing computational and memory overheads.

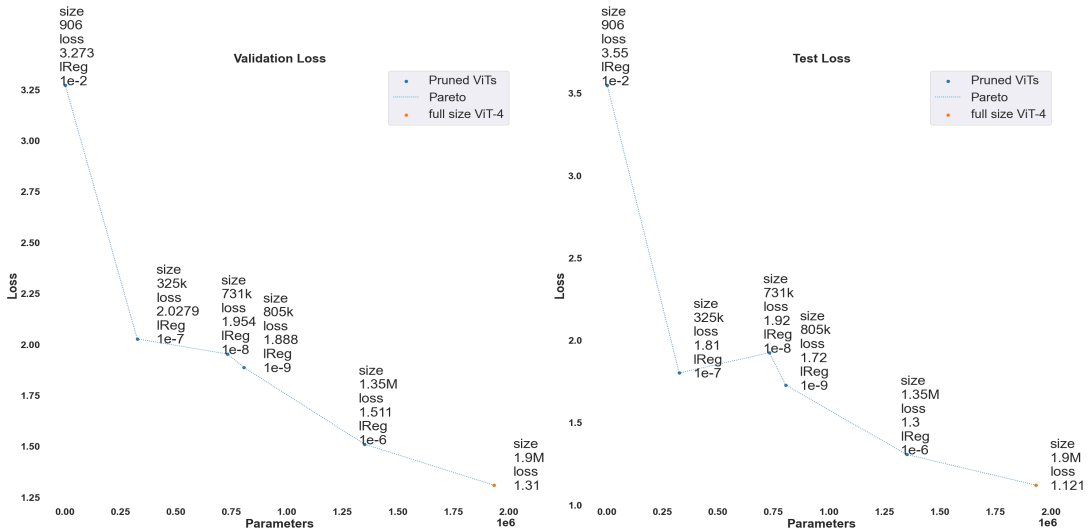

Figure 5.5: Pareto curve of the pruned models.

Chapter 6

Conclusions

This thesis successfully demonstrates the feasibility of optimizing Vision Transformer (ViT) models for real-time human pose estimation aboard resource-constrained nano-drones. By focusing on structured pruning techniques and leveraging advanced optimization strategies, the study addresses the challenges posed by the limited computational and energy capacities of nano-drones.

The research highlights the importance of tailoring ViT architectures to fit the constraints of nano-drones, such as a Crazyflie 2.1 equipped with a GAP8 AI deck for onboard computation. The application of structured pruning, particularly in Multi-Head Self-Attention (MHSA) and Layer Normalization (LN) components, achieves significant reductions in computational complexity while maintaining competitive accuracy. The results showcase a 30% reduction in the number of model parameters, with only a minimal decrease in predictive performance.
The comparative analysis between ViT and MobileNet emphasizes the potential of ViTs as a superior alternative for tasks requiring precision and adaptability. Despite MobileNet's efficiency, the optimized ViT model outperforms it in terms of accuracy, demonstrating the transformative impact of structured pruning and tailored optimization for edge computing.

The study's contributions extend beyond immediate applications, providing a robust framework for deploying deep learning models in resource-constrained environments. This work paves the way for further research into Tiny Transformers and lightweight model adaptations for other critical drone applications, such as navigation and environmental monitoring.
In conclusion, this thesis achieves its primary objective of making ViTs viable for deployment on nano-drones, advancing the state of the art in computer vision and deep learning for embedded systems.

Bibliography

[1] Soren Lenman, Lars Bretzner, and Bjorn Thuresson. «Computer vision based hand gesture interfaces for human-computer interaction». In: *Royal Institute of Technology, Sweden* (2002) (cit. on p. 4).

[2] Bappaditya Debnath, Mary O'brien, Motonori Yamaguchi, and Ardhendu Behera. «A review of computer vision-based approaches for physical rehabilitation and assessment». In: *Multimedia Systems* 28.1 (2022), pp. 209–239 (cit. on p. 4).

[3] Mikael Svenstrup, Soren Tranberg, Hans Jorgen Andersen, and Thomas Bak. «Pose estimation and adaptive robot behaviour for human-robot interaction». In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 3571–3576 (cit. on p. 4).

[4] Thomas Golda, Tobias Kalb, Arne Schumann, and Jürgen Beyerer. «Human pose estimation for real-world crowded scenarios». In: *2019 16th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE. 2019, pp. 1–8 (cit. on p. 4).

[5] Daniele Palossi, Nicky Zimmerman, Alessio Burrello, Francesco Conti, Hanna Müller, Luca Maria Gambardella, Luca Benini, Alessandro Giusti, and Jérôme Guzzi. «Fully Onboard AI-powered Human-Drone Pose Estimation on Ultra-low Power Autonomous FlyingNano-UAVs». In: (2021). arXiv: 2103.10873 [cs.RO]. URL: https://arxiv.org/abs/2103.10873 (cit. on pp. 5, 21).

[6] Nikolai Smolyanskiy, Alexey Kamenev, Jeffrey Smith, and Stan Birchfield. «Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness». In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4241–4247 (cit. on p. 5).

[7] Inkyu Sa, Zetao Chen, Marija Popović, Raghav Khanna, Frank Liebisch, Juan Nieto, and Roland Siegwart. «weednet: Dense semantic weed classification using multispectral images and mav for smart farming». In: *IEEE robotics and automation letters* 3.1 (2017), pp. 588–595 (cit. on p. 5).

[8] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. «A 64-mw dnn-based visual navigation engine for autonomous nano-drones». In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 8357–8371 (cit. on p. 5).

[9] Robert J Wood, Benjamin Finio, Michael Karpelson, Kevin Ma, Nestor O Pérez-Arancibia, Pratheev S Sreetharan, Hiro Tanaka, and John P Whitney. «Progress on "pico" air vehicles». In: *Robotics Research: The 15th International Symposium ISRR*. Springer. 2017, pp. 3–19 (cit. on p. 5).

[10] Beatrice Alessandra Motetti, Luca Crupi, Mustafa Omer Mohammed Elamin Elshaigi, Matteo Risso, Daniele Jahier Pagliari, Daniele Palossi, and Alessio Burrello. «Adaptive Deep Learning for Efficient Visual Pose Estimation Aboard Ultra-Low-Power Nano-Drones». In: *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2024, pp. 1–6. DOI: 10.23919/DATE58400.2024.10546577 (cit. on p. 6).

[11] Iman Ostovar. «Nano-Drones: Enabling Indoor Collision Avoidance With a Miniaturized Multi-Zone Time of Flight Sensor». PhD thesis. Politecnico di Torino, 2022 (cit. on p. 6).

[12] S Karam, F Nex, O Karlsson, J Rydell, E Bilock, M Tulldahl, M Holmberg, and N Kerle. «Micro and macro quadcopter drones for indoor mapping to support disaster management». In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 1 (2022), pp. 203–210 (cit. on p. 6).

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (cit. on pp. 8, 12).

[14] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929 (cit. on p. 13).

[15] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. «A survey on evolutionary neural architecture search». In: *IEEE transactions on neural networks and learning systems* 34.2 (2021), pp. 550–570 (cit. on pp. 14, 15).

[16] Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. «Can Unstructured Pruning Reduce the Depth in Deep Neural Networks?» In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 1402–1406 (cit. on p. 17).

[17] Leonardo Tredese. «Structured Pruning of Vision Transformers at Training Time». Master's thesis. Politecnico di Torino, 2023 (cit. on pp. 17, 31, 32, 42).

[18] Hengyi Li and Lin Meng. «Hardware-aware approach to deep neural network optimization». In: *Neurocomputing* 559 (2023), p. 126808 (cit. on p. 17).

[19] Alexander Toshev and Christian Szegedy. «Deeppose: Human pose estimation via deep neural networks». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1653–1660 (cit. on p. 18).

[20] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. «Realtime multi-person 2d pose estimation using part affinity fields». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7291–7299 (cit. on p. 19).

[21] Ginés Hidalgo Martınez. «Openpose: Whole-body pose estimation». PhD thesis. Carnegie Mellon University Pittsburgh, PA, USA, 2019 (cit. on p. 19).

[22] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. «Densepose: Dense human pose estimation in the wild». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7297–7306 (cit. on p. 20).

[23] Riza Alp Guler and Iasonas Kokkinos. «Holopose: Holistic 3d human reconstruction in-the-wild». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 10884–10894 (cit. on p. 20).

[24] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. «Deep high-resolution representation learning for human pose estimation». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5693–5703 (cit. on p. 20).

[25] Elia Cereda, Luca Crupi, Matteo Risso, Alessio Burrello, Luca Benini, Alessandro Giusti, D Jahier Pagliari, and Daniele Palossi. «Deep neural network architecture search for accurate visual pose estimation aboard nano-uavs». In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 6065–6071 (cit. on p. 21).

[26] Lorenzo Lamberti, Vlad Niculescu, Michał Barcis, Lorenzo Bellone, Enrico Natalizio, Luca Benini, and Daniele Palossi. *Tiny-PULP-Dronets: Squeezing Neural Networks for Faster and Lighter Inference on Multi-Tasking Autonomous Nano-Drones*. 2024. arXiv: `2407.02405 [cs.RO]`. URL: `https://arxiv.org/abs/2407.02405` (cit. on p. 21).

[27] Feng Zhang, Xiatian Zhu, and Mao Ye. «Fast human pose estimation». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3517–3526 (cit. on p. 22).

[28] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. «Image-based localization using hourglass networks». In: *Proceedings of the IEEE international conference on computer vision workshops*. 2017, pp. 879–886 (cit. on p. 22).

[29] Beatrice Alessandra Motetti, Luca Crupi, Mustafa Omer Mohammed Elamin Elshaigi, Matteo Risso, Daniele Jahier Pagliari, Daniele Palossi, and Alessio Burrello. «Adaptive deep learning for efficient visual pose estimation aboard ultra-low-power nano-drones». In: *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2024, pp. 1–6 (cit. on p. 22).

[30] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. «Patch slimming for efficient vision transformers». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12165–12174 (cit. on pp. 22, 23).

[31] Mingjian Zhu, Yehui Tang, and Kai Han. «Vision transformer pruning». In: *arXiv preprint arXiv:2104.08500* (2021) (cit. on p. 23).

[32] Asmaul Hosna, Ethel Merry, Jigmey Gyalmo, Zulfikar Alom, Zeyar Aung, and Mohammad Abdul Azim. «Transfer learning: a friendly introduction». In: *Journal of Big Data* 9.1 (2022), p. 102 (cit. on p. 29).

[33] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. «On the importance of initialization and momentum in deep learning». In: *International conference on machine learning*. PMLR. 2013, pp. 1139–1147 (cit. on p. 34).

[34] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980 (cit. on p. 34).

[35] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101 (cit. on p. 34).

[36] Ilya Loshchilov and Frank Hutter. «Sgdr: Stochastic gradient descent with warm restarts». In: *arXiv preprint arXiv:1608.03983* (2016) (cit. on p. 34).

[37] Daniele Jahier Pagliari, Matteo Risso, Beatrice Alessandra Motetti, and Alessio Burrello. *PLiNIO: A User-Friendly Library of Gradient-based Methods for Complexity-aware DNN Optimization*. 2023. arXiv: 2307.09488 [cs.LG]. URL: https://arxiv.org/abs/2307.09488 (cit. on p. 42).