Optimal Deep Learning model for UAVs: A Case Study

1st Chandan Kumar  
Dept. of Computer Science  
Iowa State University  
Ames, IA, USA  
chandan@iastate.edu

2nd Ali Jannesari  
Dept. of Computer Science  
Iowa State University  
Ames, IA, USA  
jannesar@iastate.edu

Abstract—Unmanned Autonomous Vehicles (UAVs), particularly autonomous flying drones, have garnered significant attention in the field of Artificial Intelligence. Advancements in electronic technology, marked by smaller, cheaper, and more efficient components, have led to substantial progress in UAV research. These versatile devices find applications in diverse fields, ranging from flood monitoring and algae spread detection in water bodies to forest trail identification.

In this work, we focus on autonomous flying drones, conducting a comprehensive case study to assess the efficiency, robustness, and accuracy of UAVs. Our research is substantiated through a series of experiments, providing a detailed analysis of the software and hardware architecture employed in the study. We present the implementation algorithms and compare three state-of-the-art approaches: TrailNet, InceptionResnet, and MobileNet, in terms of accuracy, robustness, power consumption, and inference time.

Our study reveals that MobileNet outperforms the other algorithms, offering superior results with minimal computational requirements and power consumption. Additionally, we address the challenges encountered during our research and discuss potential avenues for enhancing safety features and overall performance in future work.

Index Terms—Edge Computing, UAVs

I. INTRODUCTION

Unmanned Autonomous Vehicles (UAVs) have gained immense popularity, particularly in the field of autonomous flying drones, owing to their ability to navigate without physical path limitations. These aerial objects have become a topic of great interest across various domains, including traffic monitoring [2], agriculture [3], inventory management [4] [15], surveillance [1], disaster response [5], object detection [14] and data mining. As their applications expand, the need for well-suited algorithms for these vehicles becomes increasingly crucial.

Different applications may demand varying levels of precision and endurance from drones. Some scenarios, like surveillance, require prolonged operation without extreme accuracy, while others, such as item delivery, necessitate precise execution within a shorter time frame. Over the last decade, substantial advancements have been made in the field of autonomous motion planning for UAVs. However, the complexities inherent in aerial vehicles, such as differential constraints, uncertainty in vehicle state, and limited environmental knowledge, pose challenges to precise pre-computed planning.

As a result, numerous approaches and techniques have emerged for planning the motion of UAVs. However, the diverse nature of these algorithms and their inherent intricacies call for a comprehensive benchmarking study to enhance accuracy, robustness, power consumption, safety, and inference time, thus fine-tuning them for optimal performance.

This paper presents a pioneering comparative study of three algorithms aiming to achieve better motion control of drones for trail detection. Precise evaluation of each algorithm’s precision, robustness, power consumption, and inference time is essential for meticulous comparison. Additionally, considering the distinct application areas is vital in forming a reasonable opinion about an algorithm based on specific requirements.

In this study, we focus on the recent develop-
ments and algorithms employed in the domain of trail detection by UAVs. We aim to comprehensively review the existing literature and shed light on the key characteristics and prerequisites relevant to UAVs. The core design of UAVs incorporates acceleration and velocity constraints, along with higher-order differential constraints related to the equation of motion. However, the ultimate objective remains the same - guiding the vehicle toward a goal.

By conducting this comparative analysis, we seek to provide valuable insights into motion planning algorithms for UAVs, advancing the field of autonomous flying drones and paving the way for further developments and applications in various domains.

II. RELATED WORKS

Unmanned Aerial Vehicles (UAVs) and drones have found diverse applications in various fields, each with its unique challenges. Prior research by [8] explored a Micro Aerial Vehicles (MAV) system for autonomous trail detection. They introduced Trailnet, a deep neural network trained to estimate the view orientation and lateral offset of the MAV concerning the trail center. Running their vision systems in real-time on NVIDIA Jetson TX1, their work provided valuable insights into MAV navigation. Building upon their research, we present a comprehensive study aiming to enhance accuracy and reduce training time.

For our study, we selected InceptionResnet [6] and MobileNet [7] as alternatives to Trailnet [8]. The Inception architecture [6] has demonstrated excellent performance at a relatively low computational cost. The integration of residual connections with a traditional architecture achieved state-of-the-art results in the 2015 ILSVRC challenge, rivaling the latest Inception-v3 network. Residual Inception networks have even shown a slight advantage over similarly expensive Inception networks without residual connections.

In contrast, MobileNets [7] introduced a class of efficient models tailored for embedded vision applications. Utilizing depth-wise separable convolutions, they constructed lightweight deep neural networks. The introduction of two global hyper-parameters allowed for an efficient trade-off between latency and accuracy. MobileNets primarily rely on depth-wise separable convolutions, previously employed in Sif and Inception models [9], to reduce computation in the initial layers.

These works provide a foundation for our study, where we seek to build upon their contributions to further optimize UAV motion planning for trail detection. By evaluating and comparing InceptionResnet and MobileNet against the proposed Trailnet, we aim to advance the field of autonomous flying drones and enhance the efficiency and performance of UAVs in trail detection applications.

III. METHODOLOGY

This paper presents a comparative study aimed at identifying a well-suited algorithm for UAVs (Unmanned Aerial Vehicles) [13]. For this purpose, we selected three state-of-the-art algorithms, namely Trailnet, InceptionResnet, and MobileNet. InceptionResnet and MobileNet were chosen due to their demonstrated excellent performance in classification tasks.

The primary objective of our research is to train these algorithms offline and then deploy them on the drone. Subsequently, we utilize the output commands generated by these algorithms to accurately navigate the drone through various trails. To evaluate their robustness over extended paths, we simulated the trajectories generated by these algorithms using the Udacity simulator.

By conducting these experiments and comparative analyses, our study aims to identify the most suitable algorithm that can efficiently guide UAVs through complex trails with a high degree of reliability and accuracy.

A. System Architecture

In our research, a key objective is to showcase the efficacy of low-cost systems in accomplishing the complex task of autonomous drone flight. In this section, we present the architecture we adopted, encompassing both hardware and software components, to achieve our desired outcomes.

For the computing aspect, we chose the Jetson TX2 as the central processing unit. The decision to opt for the Jetson TX2 was driven by its impressive computational capabilities and power efficiency, making it well-suited for our drone’s onboard processing needs. The Jetson TX2’s embedded GPU and CPU provide the necessary
computational muscle to execute sophisticated algorithms while maintaining a low power profile, which is vital for autonomous drones with limited energy resources.

In terms of software, we employed ROS (Robot Operating System) in conjunction with Ubuntu L4T (Linux for Tegra) as the foundation for our drone’s control and communication framework. ROS offers a robust set of tools and libraries for controlling autonomous systems, enabling us to design, implement, and test our algorithms efficiently. Ubuntu L4T provides a specialized version of the Linux operating system optimized for NVIDIA’s Tegra processors, seamlessly integrating with the Jetson TX2’s hardware capabilities and maximizing its performance.

Regarding the hardware configuration of our drone, we carefully selected components that strike a balance between cost-effectiveness and performance. The precise hardware details, including the choice of motors, sensors, and communication modules, will be detailed in this section. The overall design aims to minimize costs without compromising the drone’s ability to execute complex flight tasks with precision and stability.

By combining the Jetson TX2, ROS, Ubuntu L4T, and our well-thought-out hardware setup, we successfully demonstrate that a low-cost system can achieve remarkable results in the domain of autonomous drone flight. This approach opens up opportunities for developing cost-effective and efficient solutions for various drone applications, paving the way for broader adoption of autonomous drones in real-world scenarios.

1) **Hardware Architecture:**

- **Drone Setup:** Our drone consists of a DJI Flame Wheel F450 quadcopter with an open-source Pixhawk autopilot mounted on an Orbitty carrier board.

- **Jetson TX2:** We have integrated the NVIDIA Jetson TX2 nvi into our system, which is recognized as one of the fastest supercomputing edge computing devices for interfacing. As NVIDIA’s second-generation CUDA-capable edge device, the Jetson TX2 shares similarities with its predecessor, the TX1. It operates on Ubuntu Linux4Tegra (L4T) and JetPack-L4T 3.2. The Jetson TX2 boasts impressive hardware specifications, including an integrated 256 CUDA core NVIDIA Pascal GPU, a hexcore ARMv8 64-bit CPU complex, and 8GB of LPDDR4 memory with a 128-bit interface. The CPU complex, illustrated in Fig, blends a quad-core ARM Cortex-A57 with a dual-core NVIDIA Denver 2. For our training purposes, we also utilized a Windows 10 system powered by an Intel(R) Xeon(R) Silver 4114 CPU @2.20GHz (20 CPUs), 2.2GHz, and 32GB RAM, along with an Nvidia Titan Xp GPU.

- **ZED Camera:** In our setup, we integrated the ZED stereo camera, a standard 3D USB camera designed for object detection and image capturing. This camera offers remarkable features, including depth perception, positional tracking, and 3D mapping, achieved through sophisticated sensing technology inspired by human stereo vision.

- **Battery:** The Venom 35C, 14.8V, 5000 mAh lithium polymer battery is the optimal choice for our drone’s power source.

- **GPS:** Our drone is equipped with the Holybro Ublox Neo-M8N GPS module, renowned for its high sensitivity, minimal acquisition time, and low system power consumption.

2) **Software Architecture:**

- **Jetpack:** Jetpack, an SDK developed by NVIDIA, stands out as the most comprehensive solution for creating cutting-edge artificial intelligence applications in recent years (jet, 2014). In our project, we began by flashing the latest OS image, Ubuntu 16.04 (Xenial), onto our Jetson Developer Kit.
(TX2) installed on the drone. To streamline our development environment, we leveraged the JetPack installer to set up essential developer tools for both the host PC and the drone’s developer kit. The installation process included libraries like TensorRT, cuDNN, CUDA, and OpenCV, among others, ensuring that our development environment was efficiently initiated. For the specific focus of our case study, we employed JetPack-L4T 3.2, which further enhanced our ability to harness the full potential of the Jetson platform and its advanced capabilities in artificial intelligence and autonomous systems development.

- **RO.S.**: ROS, short for Robot Operating System, is a comprehensive collection of Linux-based frameworks designed to serve as robotics middleware. In our research, we utilized ROS Kinetic Kame for handling the hardware abstraction of both the Joystick and the Controller on the drone. Additionally, ROS plays a crucial role in controlling devices at a low-level and facilitating data transfer. A typical ROS system comprises multiple independent nodes, and in our specific setup, these nodes include MAVROS, Control Node, DNN, Camera, and Joystick. These nodes effectively communicate with each other using the subscribe or publish messaging model, ensuring seamless data exchange and coordination among various components. To facilitate communication between the nodes, they are all registered with the master node, MAVROS in our case. This registration enables them to locate and establish communication with one another efficiently. MAVROS serves as the bridge that enables the utilization of the MAVLink (Micro Air Vehicle Link) protocol, allowing communication with the on-board PX4 (Flight Control Unit) of the drone. In summary, ROS acts as the middleware, orchestrating the communication and coordination among the different components in our drone’s system, thereby facilitating its autonomous operation.

- **Communication**: We enabled the Wi-Fi access point on the on-board Jetson TX2 before installing it on the drone, allowing the host PC to establish a wireless connection and access it remotely. Through this wireless connection, the host PC gained control of the Jetson TX2 by sending commands via the terminal, using the Secure Socket Shell (SSH) network protocol.

- **Udacity Simulator**: Indeed, the process of learning about UAVs (Unmanned Aerial Vehicles) involves inherent risks, given the potential for crashes and costly failures. Various aspects, such as testing new hardware, familiarizing oneself with controlling the flight controller unit of the drone, and refining algorithms, can lead to challenges and the possibility of accidents. To mitigate these risks and avoid expensive damages, we took advantage of a simulator provided by Udacity as part of its Flying Car nanodegree program. By using the simulator, we were able to create a virtual environment where we could experiment with different hardware configurations, control algorithms, and flight maneuvers without the risk of damaging physical drones or endangering people and property. The use of simulators in UAV development and training is a valuable approach to minimize potential risks, reduce costs associated with physical crashes, and accelerate the learning process. It enables researchers and developers to gain practical experience in a risk-free virtual environment before deploying their algorithms and systems on real-world hardware.

**B. Controls**

In our system, we employ standard UART (Universal Asynchronous Receiver/Transmitter) communications for control purposes. The transmitting UART takes parallel data from a controlling device, such as a CPU, and converts it into a serial format for transmission. This serial data is then received by another UART, which converts it back into parallel form for the receiving device. The communication flow between all the modules and the interaction among them are illustrated in Figure 1.

**C. Algorithm**

For our purposes, we have used 3 different algorithms namely MobileNet [7], Inception-Resnet

D. Dataset

For our experiments, we utilized the IDSIA trail dataset [12], which was collected on Swiss Alps forest trails. This dataset contains approximately 15GB of image data acquired using various cameras. The dataset is organized into 15 parts, represented by folders numbered from 000 to 014.

To train our model, we selected folders 001, 002, 004, 005, 006, 007, and 009, while folders 003, 008, and 010 were used for validation. For testing, we ran our experiments on folder 012. We excluded folders 000, 013, and 014 from our experiments as they contained only preliminary test data.

IV. EXPERIMENTS

In this section we first go through the model selections that we made by considering their performances on ImageNet Challenge [9]. ImageNet project in an ongoing large visual database designed for use in visual object recognition software research. Since 2010, ImageNet has been running an annual competition in visual recognition where participants are provided with 1.2 Million images belonging to 1000 different classes. Several deep learning architectures have been proposed since then and we have considered two of them for our experiments.

A. Model Choices

We have chosen the models based on their performance on Imagenet Challenge [9] considering accuracy and computation as major metrics. Accuracy is one of the critical aspect in building deep learning models. It depends on network architecture and the amount of data available for training. Also most of the convolutional neural networks (ConvNets) have huge memory and computational requirements especially during training. This is an important concern specifically for our purpose as memory and space footprint is limited in embedded AI devices such as drones. Also size of the final trained model becomes important to consider as we will be deploying the models to run locally on drone. In general, more computationally intensive networks usually tends to produce more accuracy. Hence, there is always a trade-off between accuracy and computational cost. Apart from these, there are many other factors which are important in selecting models such as training time, ability of a network to generalize well, inference time etc. Considering all of these factors, we have chosen MobileNet [7] and Inception-Resnet pretrained [6] on ImageNet [11] for our experiments. Table 1 compares these model factors on ImageNet project.

B. Transfer Learning and Fine Tuning

Transfer learning is a Machine Learning technique to store knowledge gained on solving one problem and applying it to a different related problem. The three major transfer learning scenarios are using ConvNet as a fixed feature extractor, fine-tuning the ConvNet, and using pre-trained models. The two most important factors that help us to decide on what type of transfer learning we should perform on the new dataset are the size of the dataset and its similarity to the original dataset. Since our dataset is large enough and different from ImageNet [11] dataset, we have decided to fine-tune our models. Fine-tuning is a transfer learning strategy to replace the final layers of the ConvNets and tweak them so that they can learn more robust features relevant to our problem. We then retrain the classifier on the new dataset and fine-tune the weights of the pre-trained network by continuing the backpropagation. Usually, the earlier features of a ConvNet contain more generic features (like edge detectors, color detectors, etc.) which are useful for many image detecting tasks. Later layers of ConvNets extract minute detailed features specific to the classes contained in our problem. We have fine-tuned both InceptionResnet and MobileNet by initializing them with their weights pretrained on ImageNet [11] and retrain the model from scratch.

V. RESULTS AND DISCUSSIONS

In this section, we discuss the results of our experiments with comparisons in terms of the size of the models, accuracy, inference time, and power consumption.

1) Size and Accuracy: Table 1 compares the architectural complexity, training time, and accuracy (total accuracy, top-1 % accuracy, and...
<table>
<thead>
<tr>
<th>Model</th>
<th>No. of Layers</th>
<th>No. of Parameters (in millions)</th>
<th>Training Time</th>
<th>Accuracy (in %)</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trailnet</td>
<td>18</td>
<td>10M</td>
<td>13 hours</td>
<td>84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inception-Resnet</td>
<td>27</td>
<td>54M</td>
<td>8 hours</td>
<td>93</td>
<td>80.4</td>
<td>95.3</td>
</tr>
<tr>
<td>MobileNet</td>
<td>28</td>
<td>3.5M</td>
<td>2 hours</td>
<td>89</td>
<td>70.1</td>
<td>89.9</td>
</tr>
</tbody>
</table>

top-5 % accuracy) of TrailNet, Inception-Resnet, and MobileNet on our dataset. It is evident from our observations that both Inception-Resnet and MobileNet have an accuracy better than TrailNet. Also, the training time and computational cost involved for MobileNet is much less than that of the other two models because of its less complexity, even though its accuracy is on par with the Inception-Resnet model. Our tests were performed by running the models on Udacity simulator. In this simulation environment, we have used 2000 images from IDSIA dataset which consists of images pertaining to a trail path of approximately 100 meters. The command to the simulated drone was provided by the output of the algorithm Figure 2 shows the path traversed by the drone autonomously on the test data environment, where

- (a) is the path traversed by drone manually controlled using simulator. This also set as the ground truth path.
- (b) is the path traversed when the simulator is controlled with output of Inception-Resnet.
- (c) is the path traversed by drone when controlled with output of MobileNet.

It can be noticed that the drone is robust under both the algorithms and approximately followed the ground truth path in both scenarios.

2) Inference Time and Power Consumption:

We have measured the inference time (i.e., the time taken by the model to predict the output on the test dataset) on the Jetson TX2 machine onboard the drone and the following table shows the comparison. From our observations in Table II, MobileNet has a very less inference time (approximately 45 % lesser inference time than the other models). Hence, Mobilenet is a more optimal model where energy consumption (power saving) is more important and where high frame rate is desired. Since this model uses much less energy than the other models, it is evident to boost the flight time of the drone. In our study, we applied transfer learning to the Trailnet model, leading us to report solely on the inference power draw. For calculating the training power draw, we measured the power draw on the Nvidia Titan XP GPU, which served as the training machine, capable of a maximum power draw of 300 Watts. The average power consumption of this GPU during idle periods was 15 Watts. As for the inference power draw, we utilized the Venom battery integrated onboard the drone. To ensure consistency throughout our experiments, we maintained a fixed batch size of 32 for both training power draw and inference power draw across all models.

![Fig. 2. Path traversal by the drone simulator under different algorithm](image)

<table>
<thead>
<tr>
<th>Model</th>
<th>Inference Time (ms)</th>
<th>Training Power Draw (Watts)</th>
<th>Inference Power Draw (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trailnet</td>
<td>49.6</td>
<td>NA*</td>
<td>110</td>
</tr>
<tr>
<td>Inception-Resnet</td>
<td>55.15</td>
<td>260</td>
<td>120</td>
</tr>
<tr>
<td>MobileNet</td>
<td>21.73</td>
<td>210</td>
<td>65</td>
</tr>
</tbody>
</table>

VI. CONCLUSION AND FUTURE WORK

This paper presents a comprehensive comparison of three algorithms - TrailNet, Inception-
ResNet, and MobileNet, with regards to accuracy, computational cost, power consumption, inference time, and robustness. Although higher accuracy is always desirable, the selection of an algorithm for UAVs depends on various factors. In this study, we focused on crucial factors that facilitate a reasonable comparison for algorithm selection. Among the three algorithms, MobileNet stood out as the most promising choice, showcasing superior performance with significantly lower computational requirements and power consumption. Consequently, we believe that MobileNet is better suited for drones and other embedded devices compared to TrailNet and InceptionResNet.

Safety constitutes a major concern in drone operations, encompassing potential issues such as collisions with objects, external disturbances like winds, drones straying beyond manual control zones, battery-related concerns, risks of theft, and other safety hazards. In our future work, we plan to address these drone-related safety features, implementing measures to enhance the safety and reliability of UAV operations.

REFERENCES


