Departamento de Informática de Sistemas y Computadores

Analysis design and implementation of artificial intelligence techniques in edge computing environments

Author:
Daniel Hernández Vicente

Advisor:
Dr. Jose María Cecilia Canales

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Valencia, España

Enero, 2023

Acknowledgements

First of all, I would like to express my most sincere appreciation and gratitude to Prof. José María Cecilia Canales (Chema), my thesis director, for his guidance during my university education as a student and researcher. Thank you for your trust in me since the first projects we carried out during these years. This thesis is the result of his support and suggestions that have been key to the development of the content of this thesis. Thank you for the countless advice, ideas, and understanding, both on a professional and personal level.

To Professors Carlos Tavares Calafate, Pietro Manzoni, Juan Carlos Cano and to all my colleagues in the Computer Networking Group (GRC) at polytechnic university of valencia (UPV). They have provided the perfect breeding ground for this research. The human quality of this place is unparalleled.

To my father, Aniceto and my brother Jose Antonio for their patience and continuous support. To my mother, Mª Esperanza, for her trust and encouragement since my youth; without her commitment, this study would not exist. To my fiancée Claudia who has remained tirelessly by my side all this time. If she still wants to marry me after this experience, she is the right one. And to Mario and Antonella, my Italian family on the other side of the Mediterranean.

I would like to mention all those who have been part of my training path in computer science, from the teachers of the multiplatform application development training cycle at the high school in Alquerías

who showed me that computer science is not only used, it is created. To the department of crazy ideas of Telefónica, directed by Sergio De Los Santos, who gave me my first professional opportunity and whose experience inspired me to continue my university studies. To the computer science department of the Saint Anthony Catholic University of Murcia, where I attended for a degree in computer engineering, and to the systems department of the University of Murcia, where I participated in a master's degree in data science. All of them laid the foundations before the beginning of my research career.

Finally, I would like to acknowledge the project Development of High-Performance IoT Infrastructures against Climate Change based on Artificial Intelligence (GLOBALoT). Funded by Ministerio de Ciencia e Innovación (RTC2019-007159-5), of which this thesis is part.

Resumen

Edge Computing es un modelo de computación emergente basado en acercar el procesamiento a los dispositivos de captura de datos en las infraestructuras Internet of things (IoT). Edge computing mejora, entre otras cosas, los tiempos de respuesta, ahorra anchos de banda, incrementa la seguridad de los servicios y oculta las caídas transitorias de la red. Este paradigma actúa en contraposición a la ejecución de servicios en entornos cloud y es muy útil cuando se desea desarrollar soluciones de inteligencia artificial (AI) que aborden problemas en entornos de desastres naturales, como pueden ser inundaciones, incendios u otros eventos derivados del cambio climático. La cobertura de estos escenarios puede resultar especialmente difícil debido a la escasez de infraestructuras disponibles, lo que a menudo impide un análisis de los datos basado en la nube en tiempo real. Por lo tanto, es fundamental habilitar técnicas de IA que no dependan de sistemas de cómputo externos y que puedan ser embebidas en dispositivos de móviles como vehículos aéreos no tripulados (VANT), para que puedan captar y procesar información que permita inferir posibles situaciones de emergencia y determinar así el curso de acción más adecuado de manera autónoma.

Históricamente, se hacía frente a este tipo de problemas utilizando los VANT como dispositivos de recogida de datos con el fin de, posteriormente, enviar esta información a la nube donde se dispone de servidores capacitados para analizar esta ingente cantidad de información. Este nuevo enfoque pretende realizar todo el procesamiento y la obtención de

resultados en el VANT o en un dispositivo local complementario. Esta aproximación permite eliminar la dependencia de un centro de cómputo remoto que añade complejidad a la infraestructura y que no es una opción en escenarios específicos, donde las conexiones inalámbricas no cumplen los requisitos de transferencia de datos o son entornos en los que la información tiene que obtenerse en ese preciso momento, por requisitos de seguridad o inmediatez.

Esta tesis doctoral está compuesta de tres propuestas principales. En primer lugar se plantea un sistema de despegue de enjambres de VANTs basado en el algoritmo de Kuhn Munkres que resuelve el problema de asignación en tiempo polinómico. Nuestra evaluación estudia la complejidad de despegue de grandes enjambres y analiza el coste computacional y de calidad de nuestra propuesta. La segunda propuesta es la definición de una secuencia de procesamiento de imágenes de catástrofes naturales tomadas desde drones basada en Deep learning (DL). El objetivo es reducir el número de imágenes que deben procesar los servicios de emergencias en la catástrofe natural para poder tomar acciones sobre el terreno de una manera más rápida. Por último, se utiliza un conjunto de datos de imágenes obtenidas con VANTs y relativas a diferentes inundaciones, en concreto, de la DANA de 2019, cedidas por el Ayuntamiento de San Javier, ejecutando un modelo DL de segmentación semántica que determina automáticamente las regiones más afectadas por las lluvias (zonas inundadas).

Entre los resultados obtenidos se destacan los siguientes: 1- la mejora drástica del rendimiento del despegue vertical coordinado de una red de VANTs. 2- La propuesta de un modelo no supervisado para la vigilancia de zonas desconocidas representa un avance para la exploración autónoma mediante VANTs. Esto permite una visión global de una zona concreta sin realizar un estudio detallado de la misma. 3- Por último, un modelo de segmentación semántica de las zonas inundadas, desplegado para el procesamiento de imágenes en el VANTs, permite la obtención de datos de inundaciones en tiempo real (respetando la privacidad) para una reconstrucción virtual fidedigna del evento.

Esta tesis ofrece una propuesta para mejorar el despegue coordinado de drones y dotar de capacidad de procesamiento de algoritmos de deep learning a dispositivos edge, más concretamente UAVs autónomos. Servicios como la detección de incendios, zonas inundadas y personas en peligro, serán ejemplos de la utilidad que se puede obtener. Gracias

a esta investigación, será posible desarrollar sistemas que permitan la coordinación de grandes conjuntos de drones y el procesamiento de imágenes sin necesidad de dispositivos adicionales. Esta flexibilidad hace que nuestro enfoque sea una apuesta de futuro y proporciona una vía de desarrollo para cualquiera que esté interesado en desplegar un sistema de vigilancia y actuación basado en drones autónomos.

Abstract

Edge Computing is an emerging computing model based on bringing data processing and storage closer to the location needed to improve response times and save bandwidth. This new paradigm acts as opposed to running services in cloud environments and is very useful in developing artificial intelligence (AI) solutions that address problems in natural disaster environments, such as floods, fires, or other events of an adverse nature. Coverage of these scenarios can be particularly challenging due to the lack of available infrastructure, which often precludes real-time cloud-based data analysis. Therefore, it is critical to enable AI techniques that do not rely on external computing systems and can be embedded in mobile devices such as unmanned aerial vehicles (UAVs) so that they can capture and process information to understand their context and determine the appropriate course of action independently.

Historically, this problem was addressed by using UAVs as data collection devices to send this information to the cloud, where servers can process it. This new approach aims to do all the processing and get the results on the UAV or a complementary local device. This approach eliminates the dependency on a remote computing center that adds complexity to the infrastructure and is not an option in specific scenarios where wireless connections do not meet the data transfer requirements. It is also an option in environments where the information has to be obtained at that precise moment due to security or immediacy requirements.

This study consists of three main proposals. First, we propose a

UAV swarm takeoff system based on the Kuhn Munkres algorithm that solves the assignment problem in polynomial time. Our evaluation studies the takeoff complexity of large swarms and analyzes our proposal's computational and quality cost. The second proposal is the definition of a Deep learning (DL) based image processing sequence for natural disaster images taken from drones to reduce the number of images processed by the first responders in the natural disaster. Finally, a dataset of images obtained with UAVs and related to different floods is used to run a semantic segmentation DL model that automatically determines the regions most affected by the rains (flooded areas).

The results are 1- The drastic improvement of the performance of the coordinated vertical take-off of a network of UAVs. 2- The proposal of an unsupervised model for the surveillance of unknown areas represents a breakthrough for autonomous exploration by UAVs. This allows a global view of a specific area without performing a detailed study. 3- Finally, a semantic segmentation model of flooded areas, deployed for image processing in the UAV, allows obtaining real-time flood data (respecting privacy) for a reliable virtual reconstruction of the event.

This thesis offers a proposal to improve the coordinated take-off of drones, to provide edge devices with deep learning algorithms processing capacity, more specifically autonomous UAVs, in order to develop services for the surveillance of areas affected by natural disasters such as fire detection, segmentation of flooded areas or detection of people in danger. Thanks to this research, services can be developed that enable the coordination of large arrays of drones and allow image processing without needing additional devices. This flexibility makes our approach a bet for the future and thus provides a development path for anyone interested in deploying an autonomous drone-based surveillance and actuation system.

Resum

Edge Computing és un model de computació emergent basat a acostar el processament als dispositius de captura de dades en les infraestructures Internet of things (IoT). Edge computing millora, entre altres coses, els temps de resposta, estalvia amplades de banda, incrementa la seguretat dels serveis i oculta les caigudes transitòries de la xarxa. Aquest paradigma actua en contraposició a l'execució de serveis en entorns cloud i és molt útil quan es desitja desenvolupar solucions d'intel·ligència artificial (AI) que aborden problemes en entorns de desastres naturals, com poden ser inundacions, incendis o altres esdeveniments derivats del canvi climàtic. La cobertura d'aquests escenaris pot resultar especialment difícil a causa de l'escassetat d'infraestructures disponibles, la qual cosa sovint impedeix una anàlisi de les dades basat en el núvol en temps real. Per tant, és fonamental habilitar tècniques de IA que no depenguen de sistemes de còmput externs i que puguen ser embegudes en dispositius de mòbils com a vehicles aeris no tripulats (VANT), perquè puguen captar i processar informació per a inferir possibles situacions d'emergència i determinar així el curs d'acció més adequat de manera autònoma.

Històricament, es feia front a aquesta mena de problemes utilitzant els VANT com a dispositius de recollida de dades amb la finalitat de, posteriorment, enviar aquesta informació al núvol on es disposa de servidors capacitats per a analitzar aquesta ingent quantitat d'informació. Aquest nou enfocament pretén realitzar tot el processament i l'obtenció de resultats en el VANT o en un dispositiu local complementari. Aquesta

aproximació permet eliminar la dependència d'un centre de còmput remot que afig complexitat a la infraestructura i que no és una opció en escenaris específics, on les connexions sense fils no compleixen els requisits de transferència de dades o són entorns en els quals la informació ha d'obtindre's en aqueix precís moment, per requisits de seguretat o immediatesa.

Aquesta tesi doctoral està composta de tres propostes principals. En primer lloc es planteja un sistema d'enlairament d'eixams de VANTs basat en l'algorisme de Kuhn Munkres que resol el problema d'assignació en temps polinòmic. La nostra avaluació estudia la complexitat d'enlairament de grans eixams i analitza el cost computacional i de qualitat de la nostra proposta. La segona proposta és la definició d'una seqüència de processament d'imatges de catàstrofes naturals preses des de drons basada en Deep learning (DL).L'objectiu és reduir el nombre d'imatges que han de processar els serveis d'emergències en la catàstrofe natural per a poder prendre accions sobre el terreny d'una manera més ràpida. Finalment, s'utilitza un conjunt de dades d'imatges obtingudes amb VANTs i relatives a diferents inundacions, en concret, de la DANA de 2019, cedides per l'Ajuntament de San Javier, executant un model DL de segmentació semàntica que determina automàticament les regions més afectades per les pluges (zones inundades).

Entre els resultats obtinguts es destaquen els següents: 1- la millora dràstica del rendiment de l'enlairament vertical coordinat d'una xarxa de VANTs. 2- La proposta d'un model no supervisat per a la vigilància de zones desconegudes representa un avanç per a l'exploració autònoma mitjançant VANTs. Això permet una visió global d'una zona concreta sense realitzar un estudi detallat d'aquesta. 3- Finalment, un model de segmentació semàntica de les zones inundades, desplegat per al processament d'imatges en el VANTs, permet l'obtenció de dades d'inundacions en temps real (respectant la privacitat) per a una reconstrucció virtual fidedigna de l'esdeveniment.

Aquesta tesi ofereix una proposta per a millorar l'enlairament coordinat de drons i dotar de capacitat de processament d'algorismes de deep learning a dispositius edge, més concretament UAVs autònoms. Serveis com la detecció d'incendis, zones inundades i persones en perill, seran exemples de la utilitat que es pot obtindre. Gràcies a aquesta investigació, serà possible desenvolupar serveis que permeten la coordinació de grans conjunts de drons i el processament d'imatges sense necessitat de dispositius addicionals. Aquesta flexibilitat fa que el nostre enfocament

siga una aposta de futur i proporciona una via de desenvolupament per a qualsevol que estiga interessat a desplegar un sistema de vigilància i actuació basat en drons autònoms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Natural disasters have increased in frequency, complexity, scope, and destructive capacity. During the past two decades, earthquakes, hurricanes, tsunamis, floods, volcanic eruptions, wildfires, and other disasters, have caused millions of deaths, adversely affected the lives of at least one billion people, and caused enormous economic losses. Generally, the poorest and most underdeveloped countries suffer the most significant losses in terms of human, social and economic lives, as their resources, infrastructures, and disaster protection and prevention systems are poorly developed. The 2020 "World Disasters Report" [1] of the International Federation of Red Cross and Red Crescent Societies shows that the majority of disasters in the last ten years were caused by extreme weather or climate events, such as floods, storms, and heat waves. The number of such disasters has risen since the 1960s and has increased by almost 35% since the 1990s. It continued to increase, although the total number of catastrophes remained stable, accounting for 76% of catastrophes between 2001 and 2010 and 83% from 2011 to 2020. However, this figure could be much lower than the actual number due to poor data collection in many countries.

Using new and emerging technologies to develop new and efficient approaches for monitoring natural disasters is an actual demand to reduce the impact of natural disasters on society. In particular, technologies developed at the intersection between Artificial Intelligence (AI) and the Internet of Things (IoT) are offering great answers to these phenomena, providing novel technology that provides real-time response [2]. Within the umbrella of AI, Machine Learning (ML), and especially the usage of one of its branches, deep learning has become an essential tool for generating knowledge through predictive analysis of large amounts of data. However, ML algorithms are complex algorithms that require significant computational infrastructures to obtain results in a time frame that allows actions to be taken in response to a given problem. These algorithms have traditionally been executed in large data centers (cloud computing), where performance prevails over energy efficiency. However, this approach implies a continuous sending of raw information that can lead to a series of security-privacy issues, transient connection failures, bandwidth limitations, scalability, and high energy consumption. To avoid these problems, another more decentralized approach has recently been proposed, where some tasks are executed near (or on) the capture devices, thus reducing the amount of information to be sent to the cloud. This emerging paradigm is known as Edge Computing (EC) and is limited by the computational resources available at these energy-constrained levels of the network.

This PhD thesis designs, develops and evaluates a transversal technology capable of collecting and processing information in real time in the context of natural disasters in order to help authorities to efficiently manage disasters derived from these continouly increasing dissasters. To this end, the thesis is based on different research lines within the field of computer science such as artificial intelligence (AI), the Internet of Things (IoT) and high-performance computing (HPC). Particularly, Several artificial vision algorithms are proposed to be executed at the edge, on low-power but high-efficient devices, embedded in unmanned aerial vehicles (UAVs), a type of aircraft that does not operate with a pilot on board and, depending on the type, is either remotely piloted by a human or operated autonomously with on-board computers that are small devices.

Historically, to face this type of problem that requires large amounts of computation, UAVs were used as data collection devices to send this

information to the cloud, where HPC servers are available to deal with this data deluge. This approach seeks to perform all the information processing in the device or a supplementary local device. This allows us to get rid of the dependence on a remote processing center that adds complexity to the infrastructure and may not be an option in specific scenarios where the wireless connections do not meet the requirements of data transfer or are scenarios where the processing has to be done at that precise moment for security or immediacy requirements.

However, having a single drone equipped with this technology is insufficient to respond to the consequences of a natural disaster such as a wildfire or flood. Therefore, this thesis proposes developing cooperative strategies between drones or swarms of drones that efficiently coordinate to cover the largest area in the shortest possible time. Particularly, this thesis proposes the solution to particular challenges of coordination of a drone swarm, surveillance of a previously unknown area, and real-time mapping of a flooded area that will require the use of traditional artificial intelligence algorithms to the latest implementations established to date of computer vision and massive information processing, as well as a computational study of the hardware that will be responsible for the final deployment of the different proposed solutions.

1.2 Objectives

The three main objectives of this Phd thesis are the following:

1. To develop a hardware-software cross-cutting technology that efficiently analyzes large amounts of data and images in IoT environments. To achieve this efficient analysis, artificial intelligence algorithms combined with mechanisms that will provide IoT environments with high-performance computing capabilities typical of large data centers will be used.

2. To optimize the synchronized take-off time of a swarm of drones with a new take-off scheme that improves the current time to determine positions in the air based on an evaluation that studies the take-off complexity of large swarms and analyzes the computational and quality trade-off of our proposal.

3. To design a UAV-based solution that will allow real-time monitoring of natural disasters through the use of AI techniques to automate the process of image and video interpretation that will reduce or avoid possible losses, provide the necessary attention to the victims, or speed up decision making during this type of events.

1.3 Structure

This thesis is structured into seven Chapters. The first chapter introduced the PhD challenges and main objectives. Then, the second chapter provides the contextualization in which the thesis has been developed by presenting the most relevant standards and technologies on which the work has been based on.

Chapters 3, 4, and 5 include the papers associated with this thesis. The main topics included in this section are The Kuhn-Munkres algorithm for the efficient vertical takeoff of UAV swarms, AI-enabled autonomous drones for fast climate change crisis assessment, Flood Detection Using Real-Time Image Segmentation from Unmanned Aerial Vehicles on Edge-Computing Platform.

Following the publications representing this study, Chapter 6 provides a brief overview of all the research conducted in the thesis, reviewing and discussing the entire set of results obtained.

Finally, Chapter 7 concludes the thesis also summarizing the research contributions and discussing future works.

Chapter 2

Background

This thesis involves different fields within computer science, including Artifical Intelligence, edge computing and UAV technology development. In this chapter, the main technologies and related works are introduced to let the reader better understand the proposals here presented.

2.1 Internet of Things and drone-based edge computing infrastructures

The concept of the Internet of Things (IoT) is based on connecting any electronic device to the network to collect information from a particular context and make decisions based on the analysis of this information[3]. IoT is a network of networks that includes devices interconnected through the Internet and people, processes, and data, to transform structured and unstructured information into richer user experiences, tangible business values, and autonomous behaviors based on real-time analytics. Two key factors underpinning the IoT revolution are (1) data, which can carry hidden patterns, correlations, as well as other valuable information, and (2) real-time analytics, as knowledge is often time-context sensitive and only valuable for a specific time frame [4]. Therefore, timely analysis of data coming from IoT infrastructures is crucial for usefully transforming

this deluge of data into knowledge that can add value to the particular application domain. With the evolution of IoT, the requirements of these systems increased, and it was sought that small interconnected devices could perform more complex tasks, usually associated with a high need for computation and that these devices are limited by both consumption and hardware limitations.

An IoT device is any machine that historically has been designed to automate a particular task offline and that, over the years, has benefited from the technological development of both compactness and network communications in order to interconnect with other devices or increase their capabilities of use by the end user [5]. These devices are found both by home users, such as washing machines, lighting systems, and intelligent air conditioning, and by industrial users to increase security in quality control systems or security. The device that has evolved substantially in the last decade has been drones, or UAVs [6]. This thesis focuses on using these devices as mobile IoT devices that can perform a multitude of tasks autonomously if equipped with the necessary hardware and software. A UAV can be a very complete and versatile mobile IoT information-gathering device since, if a camera is installed, it can perform surveillance tasks from different heights in areas that are difficult to access or where a panoramic view of the area from the sky is required being able to send information in real-time to the interested parties, [7].

Some of the proposals suggest the use of drones for rescue tasks and searching for survivors; early work in the area [8] studies the use of a single drone for this task. Years later, in the work [9], drones have been used to analyze the effects of a landslide in Tibet, comparing the terrain profile before and after the catastrophe. More recently, [10] has proposed a technique for combining aerial imagery quickly and efficiently applicable specifically to natural disasters, which has direct applicability in the case of using a swarm of drones to obtain such imagery.

This PhD proposal proposes drones as computing unitis that are able to run heavy workloads such as machine learning and deep learning algorithms that, in a typical scenario, would run on a dedicated cloud server. The IoT device, in this case, the drone, would be responsible for collecting the necessary data and sending the information in the form of images or tabular data to the server, in which the inference tasks would be performed to obtain the desired result, and this would send the information to the interested parties. The proposal discards the use of

this server and seeks to provide the UAV with the ability to obtain the results autonomously in a local environment without relying on external agents beyond the devices interconnected at the edge. This action of delegating the server work to the data collection system without sending any information is called edge computing [11].

An edge system refers to an environment that meets the following criteria: dense Geographical Distribution, Mobility Support, Location Awareness, and Proximity [12]. Edge computing brings low latency, high mobility, and location awareness to delay-sensitive applications. There has been a considerable amount of research in the area of edge computing, which is conducted and reviewed in terms of the latest developments, such as mobile edge computing, Cloudlet, and fog computing, resulting in bringing researchers a better understanding of current solutions and future applications [13]. Usually, systems that require a large computational capacity or hardware accelerators have been executed in servers in charge of processing large amounts of information or, in the last few years, in the cloud. Where hardware is available as a service and network, memory and computes, limitations are blurred compared to the capabilities of a local device to perform the designated task.

Edge computing extends the service and capabilities of cloud computing towards the end user and is featured by fast processing and application fast response time. Today's evolving Internet-enabled applications such as surveillance, virtual reality, and real-time traffic monitoring demand fast computing processing and fast response time [14]. Edge computing drives computational data, applications, and services away from remote servers to the edge of a network. Content providers and application developers can deploy EC systems by offering users services closer to their needs. Some of their features are high bandwidth, ultra-low latency, and real-time access to network information that multiple applications can utilize. Some examples of edge system implementations can be seen in [15] where a deploying approach of computing resources placed nearby the devices in the factory was introduced. The approach suggested helped to provide real-time information about the installations. The research also looked at deploying an intelligent computing system composed of data centers, gateways, and Edge devices in a logistics center. The study further provided an integer programming model that would minimize the total installed cost of Fog-enabled devices.

Currently, we are witnessing the data revolution and thus Machine

Learning algorithm are gaining acceptance in developing novel application. Thus, edge computing and AI intersection is a must to provide novel and real-time intelligence services. However, the computational requirements of this type of algorithm are very high, limiting their application to problems where time requirements are very restricted. Many ML models are trained offline, using High-Performance Computing (HPC) involving the use of hundreds or even thousands of multi-core processors, together with hardware accelerators such as Graphics Processing Units (GPUs) or Field-Programmable Gate Arrays (FPGAs) as programmable computing platforms to accelerate some phase of the computation of these algorithms. One of the challenges faced in this thesis is the adaptation of these models to edge computing and goes one step further trying to use UAV hardware-accelerated sensor IoT devices to perform complex computer vision (CV) and drone swarm takeoff coordination tasks.

2.2 Efficient information processing techniques using swarms of drones

Unmanned aerial vehicles (UAVs) of the multirotor type, commonly known as drones, have recently experienced unprecedented growth. However, if complex tasks such as assessing the effects of a natural disaster and searching for survivors are to be carried out in a short time, the deployment of swarms of UAVs has become an interesting option, improving the current efficiency of individual UAV systems. Using a group of UAVs offers many advxtantages, such as extending the coverage of a mission and improving the performance of an operation through cooperation between the different UAVs [16].

Designing a swarm of UAVs involves a series of tasks such as defining the number of devices that will make up the swarm and the formation in which it will operate, defining the assignment of the UAVs on the ground to the swarm positions when flying, getting them off the ground safely and efficiently, moving the swarm in a coordinated way avoiding collisions, and finally, landing the formation safely in a designated area for that purpose. Every UAV should be assigned to positions in the swarm so as to minimize their flight time by preventing UAVs close to a ground edge from assuming positions on the opposite edge in the formation; also, the takeoff sequence should be such that it is sufficiently fast, while ensuring

that UAVs do not collide with each other during the process.

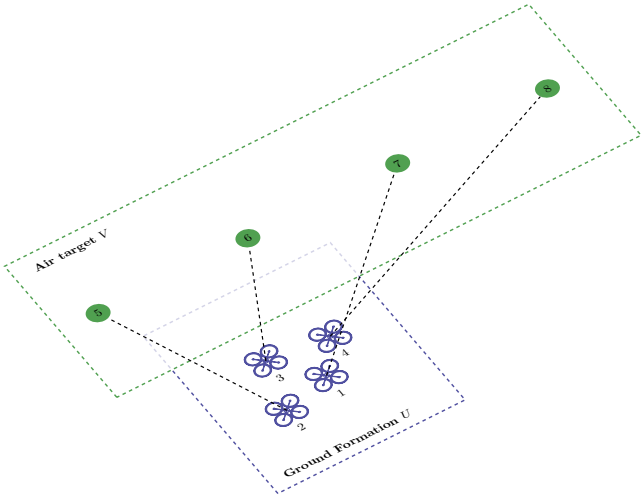


Figure 2.1: Diagram of airborne position allocation for vertical takeoff of UAV swarms.

To provide a graphical outline of the complexity of the swarm takeoff problem, Figure 2.1 shows a hypothetical scenario in which multiple UAVs arranged on the ground are to be launched in such a way that they end up forming an aerial formation for the start of the mission, following targets depicted above them. The UAVs are to be assigned positions in the swarm so that they are ready to complete a mission, in order to optimize their flight time, trying to prevent UAVs close to one edge on the ground from assuming positions on the opposite edge in the formation due to performance reasons, and also to minimize collision risks.

For the optimal allocation that ensures the minimum total distance traveled by the UAVs to their target position in the formation, a brute-force method can be used. This algorithm analyses the entire universe of possible solutions by calculating all possible assignment combinations between the ground positions and the airborne formation, having a cost that grows as $O(n! \cdot n^2)$. The main disadvantage of this approach is that searching for all possible solutions results in an exponential growth of computational complexity. The computation times become prohibitive even for a low number of UAVs (<20).

A second proposed solution includes a heuristic to allocate the UAVs rapidly on their swarm position [17], resulting in an efficient and safe takeoff. This work provides a solution similar to the results obtained by the brute force algorithm but with a significantly reduced time delay overhead. Its main point is determining a central position on the ground concerning the deployed UAVs. Then, that central position is used to calculate the distance to all positions of the desired flight formation, sorted in descending order. Using this list, the closest UAV is assigned to each of these positions. In terms of computational cost, it has a cost that grows with $O(n^2)$. It is relevant to note that the solutions provided by this heuristic might not be optimal, as will be demonstrated in section 3, where a proposal will be shown that tries to improve this position allocation algorithm based on the kunh munkres algorithm.

After the technical proposal of the solution, an experimental implementation will be carried out on ardusim software [18] [1], which is a real-time flight simulator aimed at the development of flight coordination protocols for multicopters, performing planned missions or making a swarm. It is also capable of simultaneously simulating up to multiple UAVs. The number of UAVs which can be simulated is mainly restricted by the PC being used. For this reason, it is crucial to optimize the take-off algorithm, as it dramatically limits the number of devices to be used. ArduSim also simulates an Ad-hoc wireless network for communications between UAVs. ArduSim generates the route followed by each UAV in OMNeT++ and NS2 format to provide mobility traces, perform a simulation, or even run ArduSim on a real multi-copter.

2.3 Image semantic segmentation based on convolutional neural networks

Semantic segmentation is the task of differentiating parts of images that are part of the same target class. Semantic segmentation remains a classification task. Most algorithms work with a fixed set of classes; some even work only with binary classes, such as front and background. There is a set of image segmentation algorithms based on a traditional approach and therefore do not apply neural networks and make intensive

[1] https://github.com/GRCDEV/ArduSim

use of domain knowledge. Although new image processing techniques are replacing them, they are still used in the computer vision community. Some of these algorithms are Watershed Segmentation [1], which is defined as a region-based segmentation approach, Markov Random Fields (MRF) [19] that are are undirected probabilistic graphical models, or random forest (RF) [20] trained for quantitatively evaluate a range of potential image segmentation scale alternatives in order to identify the segmentation scale(s) that best predict land cover classes of interest.

Convolutional neural networks (CNN) architectures have evolved rapidly and, in recent years, have achieved results that were previously only considered possible through human execution/intervention. Depending on the task to be performed and the corresponding constraints, a wide variety of architectures are adapted to the problem domain.

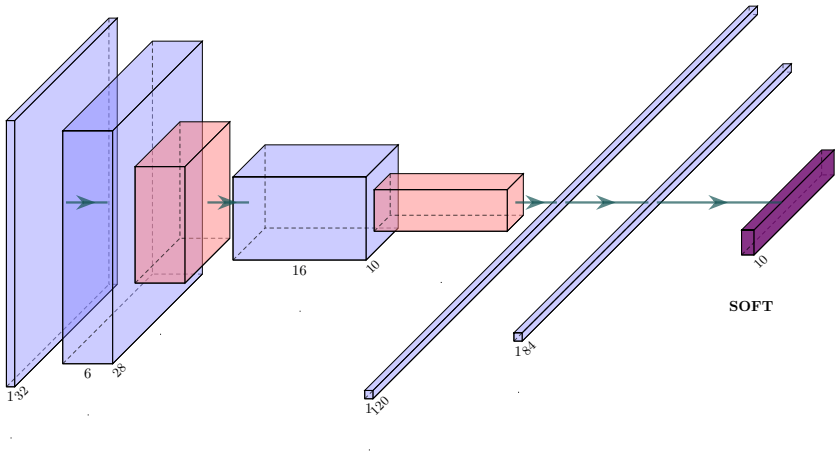


Figure 2.2: LeNet deep learning model architecture diagram.

LeNet [21] is one of the pioneering architectures in using neural networks in computer vision. It was developed between 1989 and 1998 to solve the task of handwritten digit recognition. His proposal is based on the fact that gradient-based learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns,

such as handwritten characters, with minimal image preprocessing.

LeNet is composed of the basic units of these neural networks: convolutional, the pooling layer, and the fully connected layer. It is also known as leNet-5 because of its nature, as shown in figure 2.2. The network has a total of 60,000 parameters structured as follows: it has a first convolution layer followed by a pooling layer, then repeats this structure once again and links to a fully connected layer followed by the final fully connected layer that connects to the 10-class classification output.

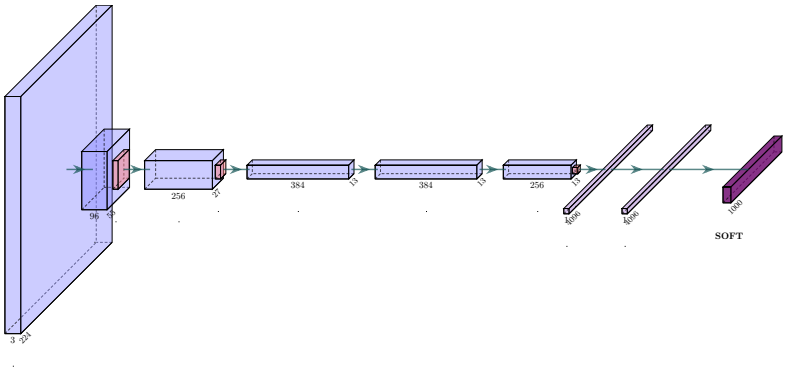


Figure 2.3: AlexNet deep learning model architecture diagram.

AlexNet [22] was developed to classify the 1.2 million high-resolution images from the ImageNet ILSVRC-2010 competition into 1000 different classes. Imagenet was one of the triggers of the revolution of CNNs as it provided the developers with a sufficiently large and well-structured dataset to be used as a benchmark for the development of new computer vision models. Thanks to AlexNet, the developers won first place in the competition and laid the foundations for deep neural networks that took advantage of the GPU's matrix computing capabilities since the main feature of this model is the inclusion of more processing layers involving a larger number of trainable parameters that depend on a large computational capacity to be trained.

AlexNet consists of eight layers shown in Figure 2.3. The first five were convolutional layers, some of them followed by max-pooling layers, and the last three were fully connected layers. Instead of using a tang or sigmoid activation function, it uses the ReLU unsaturated activation function, which provides better training performance. Alexnet represented

a significant increase in the model size compared to LeNet, with Alexnet having 60 million parameters.

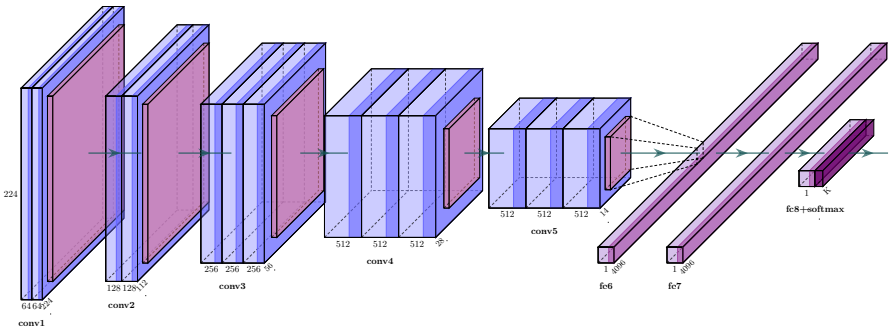


Figure 2.4: VGG16 deep learning model architecture diagram.

VGGNet [23] was developed in 2014 to increase the depth of the CNNs to increase the model's performance under the premise that as the number of layers in the CNN increases, so does the model's ability to fit more complex functions. Therefore, a higher number of layers promises better performance. Two sizes were proposed, 16 and 19 layers. Figure 2.4 shows the design of the 16-layer architecture, better known as VGG16. At the time of its publication, it achieved a 92.7% accuracy in the Imagenet test dataset represented a new leap in the development of deep networks.

Having 16 layers, VGG16 is quite an extensive network with 138 million parameters. Even by modern industry standards, it is a vast network. However, the VGGNet16 infrastructure's simplicity makes the network more attractive. The structure consists of a few convolution layers followed by a pooling layer that reduces the height and width. There are about 64 filters available which can be duplicated up to about 128 and then up to 256. In the last layers, one can use 512 filters. Due to this structure, the number of filters used is doubled in each step or each stack of the convolution layer. This is one of the main principles used to design the network architecture. Consequently, one of the disadvantages of the VGG16 network is that it is a massive network, which means that more time is needed to train its parameters. Training a single net on a system equipped with four NVIDIATitan Black GPUs took 2–3 weeks,

depending on the architecture.

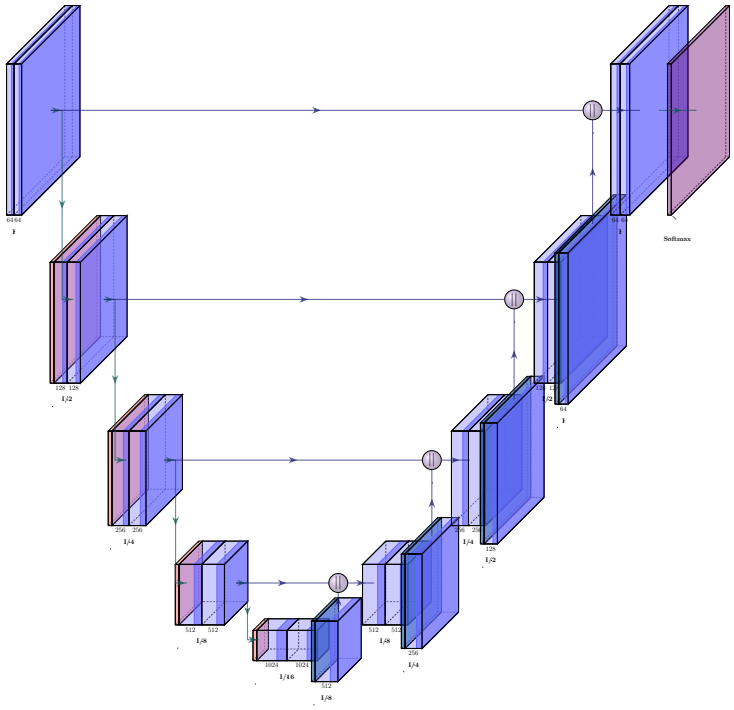


Figure 2.5: Unet deep learning model architecture diagram.

With the evolution of CNN architectures oriented to image classification and object detection and the development of hardware accelerators to accommodate these increasingly larger models, the options opened up for a new type of network that would not only classify images as a whole. However, they would classify each pixel of the image, thus allowing a much more precise detection of the target class, since in this case, it would not obtain a bounding box that positions an object, but it could obtain the complete silhouette of the object, this task is commonly referred to as dense prediction. In 2015 a method for semantic segmentation of biomedical images was proposed. This model was called Unet [24] and was surprising for its results and simplicity.

As mentioned at the beginning of this section, the network output in a semantic segmentation task is not just a class label or bounding box parameters. Indeed, the output is a complete high-resolution image in

which all pixels are classified. By using a typical convolutional network with clustering layers and dense layers, the information *"WHERE"* will be lost, and only the information *"WHAT"* will be retained, which is not what is desired. In the case of segmentation, Both *"WHAT"* and *"WHERE"* information is needed. Thus, it is necessary to upsample the image after having stored all the information in the latent space in the center part of the model as shown in Figure 6.1, i.e., to convert a low-resolution image into a high-resolution image to retrieve the *"WHERE"* information. For this task, historically, many techniques have been developed to increase the size of an image. Some are bilinear interpolation, cubic interpolation, and nearest neighbor interpolation. However, in most state-of-the-art networks, transposed convolution is the preferred option for sampling an image, the latter being the main contribution of Unet. The main proposal of this network, which sets it apart from the rest, is Transposed convolution is a technique for upsampling an image with trainable parameters. At a high level, transposed convolution is exactly the opposite process of a normal convolution, i.e., the input volume is a low-resolution image, and the output volume is a high-resolution image with the particularity that It does not use a predefined interpolation method; it has learnable parameters.

2.4 Image clustering by using deep learning techniques at the edge

Thanks to the deep learning approach, some works successfully combine feature learning and clustering into a single unified framework that can directly cluster the original images with even higher performance.

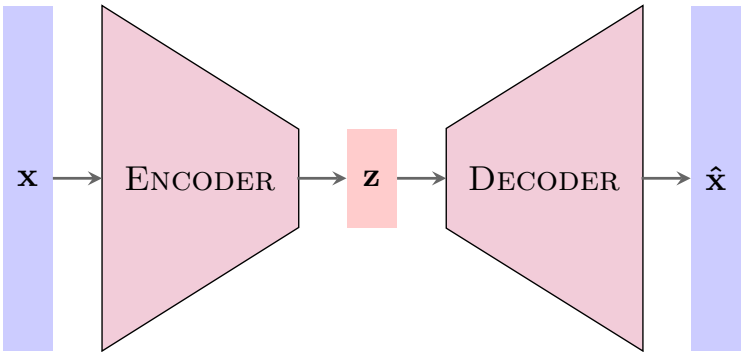


Figure 2.6: High-level view of the Autoencoder schema.

Clustering is a type of unsupervised machine learning method, which means that it is a method in which information is extracted from input data sets without labeling the responses. In general, it is used as a process to find meaningful structure, underlying explanatory processes, generative features, and groupings inherent in a set of examples. Properly implemented clustering allows for intrinsic grouping among the unlabeled data present. Intuitively, clustering is the task of partitioning the population or data points into a subset of groups such that data points in the same groups are more similar to other data points in the same group and less similar to data points in other groups. Essentially, this is a set of objects based on similarities and dissimilarities. Some well-known clustering algorithms are:

- K-means [25] clustering algorithm is the simplest unsupervised learning algorithm that solves the clustering problem. The K-means algorithm partitions n observations into k clusters where each observation belongs to the cluster with the closest mean that serves as the cluster prototype.
- Hierarchical clustering [26] is an algorithm that builds a hierarchy of clusters. This algorithm starts with all data points assigned to a cluster of their own. Then, two closer clusters are merged into the same cluster. In the end, this algorithm terminates when only a single cluster remains.
- Density-based spatial clustering of noisy applications (DBSCAN) [27] is a data clustering algorithm proposed by It is a non-parametric

density-based clustering algorithm: given a set of points in some space, it groups points that are close together (points with many close neighbors), marking as outliers points that are alone in regions of low density (whose nearest neighbors are too far away).

Before the clustering stage, a technique is often used to reduce the dimensionality of the data to be processed and thus perform a virtual grouping of the data in a smaller information space. One of the most used approaches to resolve the problem of dimension reduction is the principal component method (PCA). However, it can only be applied to rectilinear data. If large objects are considered, the probability that they are well separated is small; nevertheless, if they constitute a mixture of things belonging to normal distributions with different parameters. Alternatively, there is the option of developing a convolutional autoencoder (CAE)2.6, which has been developed that can be end-to-end trained to learn features from unlabeled images. The unsigned CAE is superior to stacked autoencoders by incorporating the spatial relationships between pixels in the pictures. It is shown that the convolutional layer, the convolutional transpose layer, and the fully connected layer are sufficient to structure an effective CAE. The main key factor of the CAE is the aggressive restriction of the dimension of the embedded layer. If the embedded layer is large enough, the network may be able to copy its input to the output, leading to learning useless features. The intuitive way to avoid identity mapping is to control the dimension of latent representations. Learning such sub-complete representations forces the autoencoder to capture the most salient features of the data. Thus, the dimension of the embedded space is forced to equal the number of clusters in the data set. In this way, the network can be trained comprehensively even without regularizations such as Dropout or Batch Normalization.

2.5 Summary

An existing problem in vertical take-off has been shown. The issue of vertical take-off of drone aerial vehicles, which so far requires a previous calculation of the assignment of each UAV to its destination and which significantly limits the number of devices that can be integrated into the aerial vehicle and the changes that can be made on the ground, since due to the structure of the algorithm used, the calculation of the

assignment would take a long time and would require a data center or a team that would be computing for hours or days, and the requirement for the solution to be able to run in the desired environment which serves as the basis for the proposal in the Kunh Munkres chapter.

This section has gone through the historical inclusion of autonomous UAVs as IoT devices that can perform surveillance tasks and how they can become edge computing devices if they integrate graphics acceleration hardware that allows the execution of deep learning models of adequate size and structure to obtain the correct results but also process the images in the device itself.

We have shown the evolution of computer vision techniques and how the feedback between hardware advances in graphics acceleration and research in convolutional neural networks have resulted in a remarkable growth of the capabilities of these image processing models, which will allow in the following chapters to show its application with a self-contained framework in the edge of swarms of drones to perform surveillance tasks in flooded areas that will enable semantic segmentation tasks and obtaining the most relevant images by using clustering techniques.

Chapter 3

The Kuhn-Munkres algorithm for efficient vertical takeoff of UAV swarms

Hernández, D., Cecilia, J. M., Calafate, C. T., Cano, J. C., Manzoni, P. (2021, April). The Kuhn-Munkres algorithm for efficient vertical takeoff of AV swarms. In 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring) (pp. 1-5). IEEE. doi: 10.1109/VTC2021-Spring51267.2021.9448873

3.1 Abstract

The field of Unmanned Aerial Vehicles (UAVs) is gaining momentum thanks to the amazing capabilities of these flying devices. In particular, small aircrafts using vertical takeoff and landing (VTOL) are among the preferred solutions in the civilian sector thanks to their low cost, simplicity of operation, and the ability to carry powerful sensing devices. When combined to create a swarm, the potential of such UAVs is fur-

ther extended by allowing to perform more complex missions efficiently. However, as the number of UAVs involved becomes higher, many issues arise that can result into mission failures. In this paper, we specifically address the swarm takeoff problem from an optimization perspective. We propose a new takeoff scheme based on the Munkres algorithm that solves the assignment problem in polynomial time. Our evaluation studies the taking off complexity of large swarms and analyze the computational and quality trade-off of our proposal. Experiments show that the Munkres algorithm offers optimal solution with a low computation overhead.

3.2 Introduction

Unmanned Aerial Vehicles (UAVs) are autonomous unmanned aircrafts that are widely used for different applications and tasks. They have gradually shifted from more established areas such as aerial photography and video, to new areas such as precision agriculture, border surveillance, package delivery, thermal inspections, and air cabs to name few. [1]. The use of UAV swarms increase their potential capabilities in many cases. UAV smarms flying autonomously in cooperation can create or improve networks (e.g. solving infrastructure problems), monitor weather, transfer information or traffic, etc.

Generally speaking, the number of drones in a swarm is an important factor in determining its application capabilities. More drones in a swarm means that the swarm has a greater potential to increase the tracking area, offering fault tolerance and different imaging perspectives. However, having more drones in a swarm means more variables to control, which translates into a higher computational complexity. This could affect the swarm's behavior as well as critical decisions such as those related to preventing drones from crashing into each other. Actually, the management of UAV swarms, and specifically those of the Vertical Take-Off and Landing (VTOL) type, to accomplish joint tasks is also being addressed by different research groups worldwide [2, 3, 4]. For instance, Intel used 500 drones to create the first UAV-based Light Show with such a massive number of UAVs[1] in 2016. EHANG[2] deployed 1180

[1] https://www.tokyo-motorshow.com/en/press_release/20191018.html
[2] https://www.ehang.com/ehangaav

drones to create a similar show in China[3] in December 2017. This number of drones was even increased to 1374 drones in April 2018[4]. In July 2018, Intel designed a coordinated swarm with up to 2018 drones to break the Guinness World Record of the largest number of unmanned aerial vehicles simultaneously flying[5].

All these experiments were centrally managed and very strict deployment conditions were applied to ensure their success. The management of this number of drones as a whole requires algorithms for autonomous decision making by the swarm. As the number of drones increases, the computational overhead also increases, which limits the success of these procedures beyond 100 drones in a swarm. This computational problem is even worse in real case scenarios where the swarm can include any number of UAVs, under any conditions, and for any swarm layout. Only few works address the specific issue of achieving a safe takeoff for a large number of UAVs participating in a swarm. In [5], authors only consider three simple takeoff schemes for a swarm: manual, sequential and simultaneous. However, when the swarm is large, and when the formation in the air remains unrelated to positions on the ground, these techniques can take too much execution time (manual, sequential), or be prone to cause collisions between UAVs (simultaneous), especially when UAVs are packed together on the ground. In our previous work [6], we proposed an heuristic to provide a near-optimal assignments of UAV positions in the swarm formation selected optimally. In this paper, we go a step further by addressing this problem from the perspective of a assignment problem. We rely on the Munkres algorithm (also known as the Hungarian algorithm) which offers optimal solutions to the take-off problem. We analyze the problem in different real-life scenarios, i.e. using different formations on the ground and in the air and scaling the number of drones in the swarm. Finally, the results of this algorithm are compared with other takeoff algorithms, analyzing the different advantages and disadvantages of each approach and demonstrate the validity of our proposal.

The remainder of this paper is organized as follows: in section 3.3 we provide a more detailed overview of the problem, and describe the ideal solution for the swarm position assignment. Section 3.4 approach the

[3]https://www.popsci.com/china-drone-swarms/

[4]http://www.ehang.com/news/365.html

[5]https://newsroom.intel.com/news/intel-breaks-guinness-world-records-title-drone-light-shows-celebration-50th-anniversary/

vertical takeoff problem of Drones from the point of view of an assignment problem and proposes its resolution using the Kuhn-Munkres algorithm. Then, in section 3.5.2, we assess the performance of the solution using different formations and compare it with existing algorithms for this purpose. Finally, section 7 concludes the paper, and discusses future works.

3.3 Problem overview

Creating a swarm of UAVs is a challenging task that involves defining the swarm size and layout, defining the assignment of ground UAVs to positions on the swarm when flying, making UAVs takeoff in a safe and yet effective manner while introducing small delays, moving the swarm in a coordinated manner and avoiding collisions while executing a coordinated mission, and finally landing the UAVs safely and in a small area.

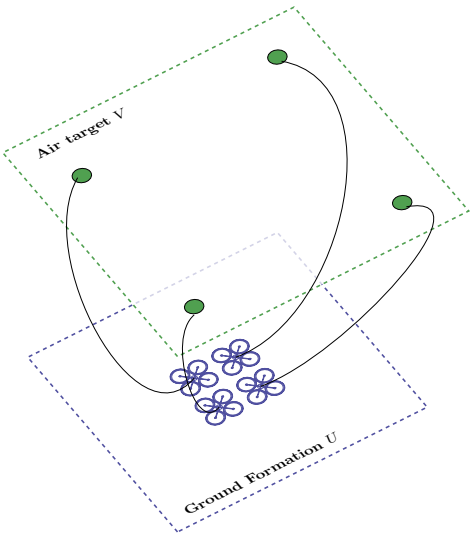


Figure 3.1: Overall problem description.

To have a better insight into the complexity of the swarm takeoff problem, Figure 3.1 shows a scenario where several UAVs on the ground participate in a swarm following the layout depicted above them. In general, to have the UAV swarm ready for completing a mission, UAVs should be assigned positions on the swarm so as to minimize their flight

time, e.g. avoiding that UAVs near one edge on the ground assume positions on the opposite edge in the formation, and also to minimize collision risks.

With regard to the requirement of the optimal allocation that guarantees the minimal overall distance travelled by UAVs to their position, a brute-force algorithm can be used. This algorithm analyses the whole solution space by computing all possible combinations of assignment between ground positions and air formation, having a cost that grows as $O(n! \cdot n^2)$. The main drawback of this technique is that searching through all the possible solutions makes calculation complexity to have an exponential growth. In fact, as shown later in section 3.5.2, calculation times become prohibitive even for a low number of UAVs (<20).

Other proposed solution includes a heuristic [6] to quickly assign UAVs to their position in the swarm, achieving an efficient and safe takeoff. This proposal encompasses a heuristic for making an efficient UAV-to-swarm position assignment that offers a solution that comes near to that obtained by the brute-force algorithm, but with a significantly lower time overhead. Basically, it consists in determining a location on the ground which is central with regard to the UAVs deployed, as shown in the Figure 3.2. Then, such central position is used to compute the distance towards all the positions in the desired flight formation, which are then sorted in descending order. Using this list, the UAV closer to each of these positions is then assigned to it. In terms of computational cost, it has a cost that grows with $O(n^2)$. It is important to note the solutions provided by this heuristic could not be the optimal as it will shown in Section 3.5

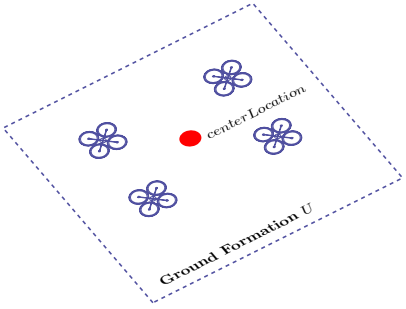


Figure 3.2: Heuristic formation graph

3.4 The Kuhn-Munkres takeoff algorithm

3.4.1 The assignment problem

The assignment problem is one of the fundamental combinatorial optimization problems. Many real world problems can be categorized as an assignment problem, such as the UAV takeoff problem. This problem consists in finding a maximum weighted matching in a weighted bipartite graph. In this problem, the main goal is to minimize the distance. Given an n x n matrix $W = (w_{ij})$, a permutation φ of the integers $1, 2, ..., n$ must be found to minimize Equation 3.1.

$$\sum_{i=1}^{n} w_{i\varphi(i)} \tag{3.1}$$

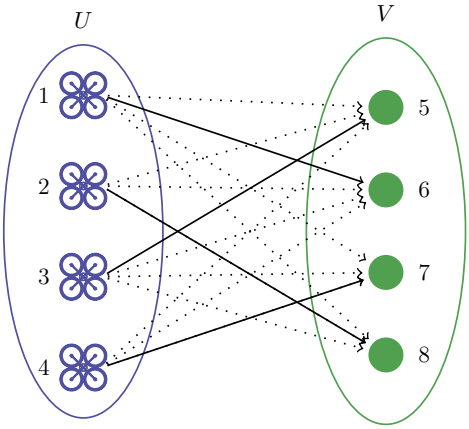


Figure 3.3: Bipartite graph.

The UAVs takeoff problem can be seen as an assignation problem in a complete weighted bipartite graph (see Figure 3.3). The vertices are divided into two subgroups, such that every edge of the graph joins a vertex in one set to a vertex in the other set. Hence, there is a (finite) minimal number of vertices with the property that every edge of the graph ends in one of these vertices. Based on the bipartite graph, it is possible to calculate the weight matrix using an algorithm to minimize the Equation 3.1. Within the umbrella of linear programming, different algorithms have been proposed to solve the assignment problem. The

auction algorithm [7] and the Kuhn-Munkres algorithm [8] are the most widely used in this field. We have focused on the the latter since it offers better performance.

3.4.2 The Kuhn-Munkres Algorithm

This method, also known as the Hungarian method as it was largely based on the earlier works of two Hungarian mathematicians in 1916 [9] and 1931, was developed and published later on, in 1955, by Kuhn [8]. The striking advantage of Kuhn's algorithm is that it is strongly polynomial $O(n^3)$ [10], and for this reason, and because of the simplicity of its implementation, it has been used in several combinatorial optimization procedures in areas such as UAV task assignment [11], network flows, matroids, and matching theory[12].

Algorithm 1 distanceMatrixCalc(numUAVs, groundLocations, flightFormation, errorMatrix)

Require: $groundLocations\ of\ size\ numUAVs \wedge$
$flightFormation\ of\ size\ numUAVs \wedge$
$errorMatrix\ of\ size\ numUAVs * numUAVs$

1: **for** $i \in \{0, \ldots, numUAVs\}$ **do**
2: **for** $j \in \{0, \ldots, numUAVs\}$ **do**
3: $errorMatrix[i][j] \leftarrow d(groundLocations[i], flightFormation[j])^2$
4: **end for**
5: **end for**=0

To adapt the takeoff problem to the Munkres algorithm, the distance matrix will be calculated using Algorithm 1, where d function is the Euclidean distance. Then, all the steps shown in Algorithm 2 are followed. The result is a vector with the minimum possible allocation between the drones on the ground and the desired aerial formation.

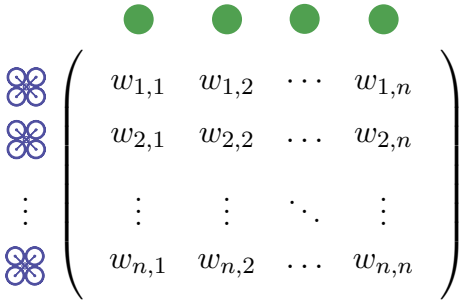


Figure 3.4: Distance matrix between all the nodes

3.5 Evaluation

3.5.1 Scenarios

Three different swarm formations have been used during the study. Each one starts from a set of ground source coordinates and a set of target coordinates, where a ground position will be assigned to a target position using the different proposed algorithms, always starting from a swarm of UAVs located on the ground in the form of random points in a square of side 400 meters, with a minimum separation of 7.5 meters between UAVs for the sake of security during takeoff. The air formation is defined for an altitude of 50 meters above ground. Both on the ground and in the air, the center point of each formation is located on position (x,y)=(0,0). Figure 3.5 shows the three swarm formations used in this experiment, whose characteristics are the following:

- **Circular formation**: Given the number of UAVs, a circumference is drawn with the smallest possible radius so that each drone is placed on its edge, and at least a 10-meter euclidean distance is established between the two drones closest to it.
- **Lineal formation**: Starting from the center, a UAV is added by varying the distance 10 meters from its closest drone, adding in case the drone is positioned to the right of the formation, or reducing in case it is placed on the left.
- **Matrix formation**: To achieve the matrix shape independently of the number of drones in the formation, each drone is added

Algorithm 2 KunhnMunkres(numUAVs, groundLocations, flightFormation)

Require: $groundLocations.size = numUAVs \wedge$
$flightFormation.size = numUAVs$

Step 1 $\rightarrow$ Set a matrix of size $numUAVs * numUAVs$ and fill it out using $distanceMatrixCalc()$

Step 2 $\rightarrow$ **Subtract row minima:** Subtract the smallest entry in each row from each entry in that row in the distance matrix.

Step 3 $\rightarrow$ **Subtract column minima:** Subtract the smallest entry in each column from each entry in that column in the distance matrix.

Step 4 $\rightarrow$ **Cover all zeros with the minimum number of lines:** Using the smallest number of lines possible, draw lines over rows and columns in order to cover all zeros in the matrix.

Step 5 $\rightarrow$ If the minimum number of covering lines is n, **an optimal assignment of** *zeros* **is possible and the process is finished**. If the minimum number of covering lines is less than n, an optimal assignment of *zeros* is not yet possible. In that case, proceed to *Step 6*

Step 6 $\rightarrow$ Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Then return to *Step 4*. =0

alternately between the quadrants of two perpendicular axes and, when the square is formed, 10 meters are added in each axis; the process continues until all the coordinates are assigned.

In the experiments, and for each formation, different scenarios are designed varying the number of drones from 2 to 1000. In the following sections, we show the results of execution time (depending on the algorithm, the formation, and the number of drones that integrate it), and

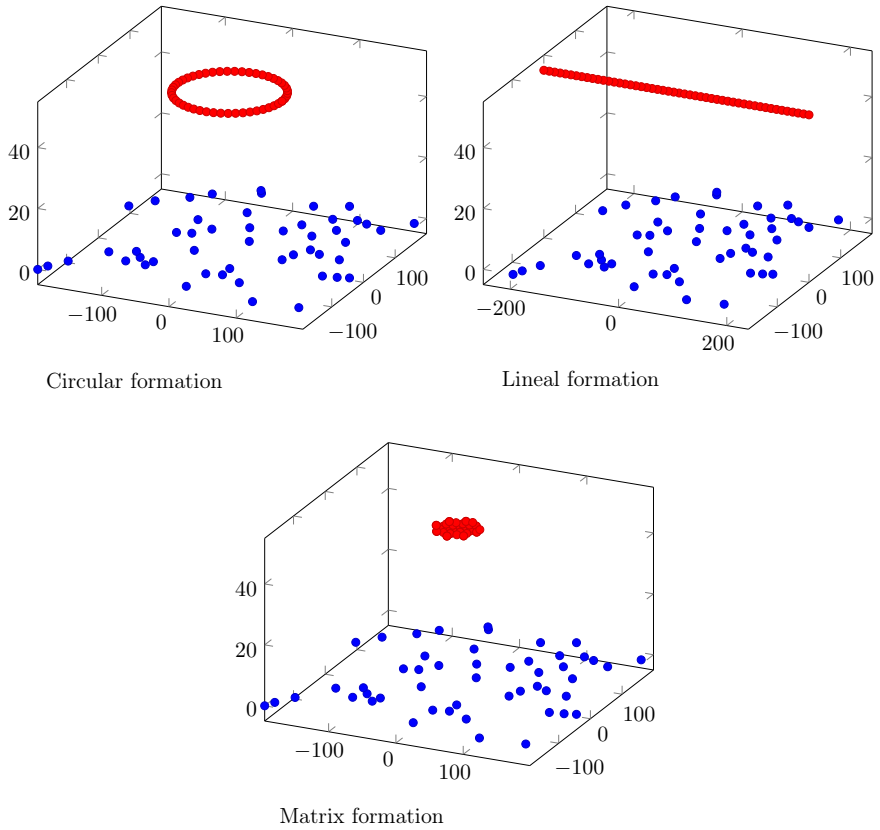


Figure 3.5: UAVs formations for the experiment.

the total distance traveled after making the assignment.

3.5.2 Performance evaluation

In this section, we compare our proposal with the execution time of the three algorithms mentioned in this document, showing the performance scalability when the formation and number of UAVs are varied. The experiments evaluates the execution time needed to find the solution with the total number of drones in a scale from 2 to 1000 UAVs. It is important to note that the ideal algorithm, which has been limited to executions with 2 and 10 since retrieving the solution becomes unfeasible with respect to the time overhead using standard computational resources.

The execution has been carried out in a Quad-Core Intel Core i7 CPU using a single core.

Figure 3.6 shows that the algorithm that consumes less execution time is the one that uses the heuristic approximation, followed by Munkres that, although its execution time is two orders of magnitude higher, remains a reasonable execution time given the amount of drones using in the experiment. In the executions of Mukres, a significant variability can be observed between the execution time needed to find the solution using the different formations for a same number of drones. This situation is not noticeable in the other two. This is due to the nature of the algorithm itself since, depending on the type of formation, the optimal solution will converge in a variable number of steps.

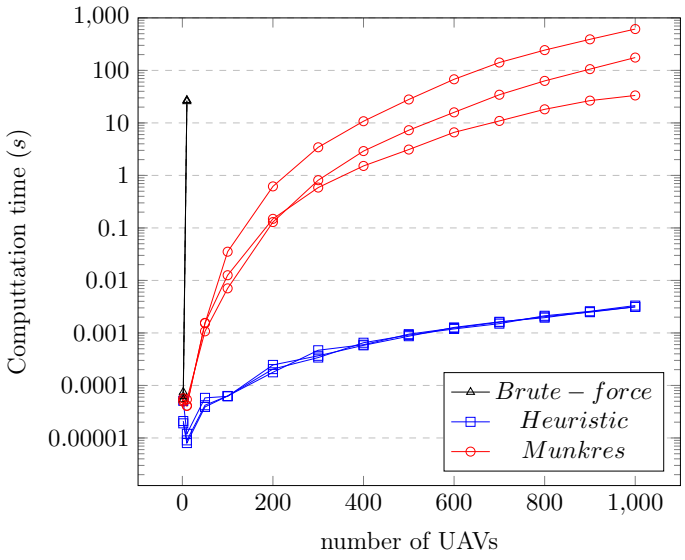


Figure 3.6: Computation time of each algorithm based on the number of UAVs.

3.5.3 Quality analysis

This section analyzes the quality of the solutions provided by the algorithms. The total distance travelled by UAVs for each of the three formations is measured, which actually depends on the actual number of

UAVs. The decision on the algorithm to be used in each formation will be made taking into account the heuristic and Munkres algorithms. The brute-force algorithm will not be used, since the result obtained is always the same as that obtained with Munkres, which as can be seen in figure 3.6 is much faster. Figure 3.7 shows the sum of the squared distance of each drone towards its assigned position, for each of the formations, and when varying the number of drones.

Starting from the circular formation, it can be observed that, when the number of drones grows, a difference starts to exist in favor of the Munkres algorithm. This difference, although appreciable, does not present a disruptive deviation and, therefore, the use of one or the other could be conditioned to other variables, such as the available computational capacity, or the minimization of the total space to be covered. For a formation of linear characteristics, the best option would be the use of the heuristic algorithm since the total distance traveled, although greater, is mostly imperceptible; in section 3.5.2 it can be appreciated how the Munkres algorithm provides a better performance at execution time.

3.5.4 Discussion

After experimentation, the brute-force algorithm is discarded in favor of the Kuhn-Munkres algorithm since, although both obtain the shortest possible total distance, the latter on drastically improves the total execution time.

The matrix formation allows us to see a greater difference between both algorithms under study. Notice that this aerial formation is where we have the highest contrast between the aerial formation area and the ground area, and thus the disadvantages of making a calculation based on the average of all positions on the ground can be better appreciated; in these cases, a calculation taking into account the unitary positions of each drone with respect to the final formation is more beneficial.

The actual choice between the heuristic technique and Kuhn-Munkres will depend on the chosen air formation, and we will have a clear trade-off between computational overhead and the total distance traveled. Results show that, for a linear formation, the best option would be to use the heuristic approach; however, for a matrix formation, the most consistent decision is the Kuhn-Munkres algorithm.

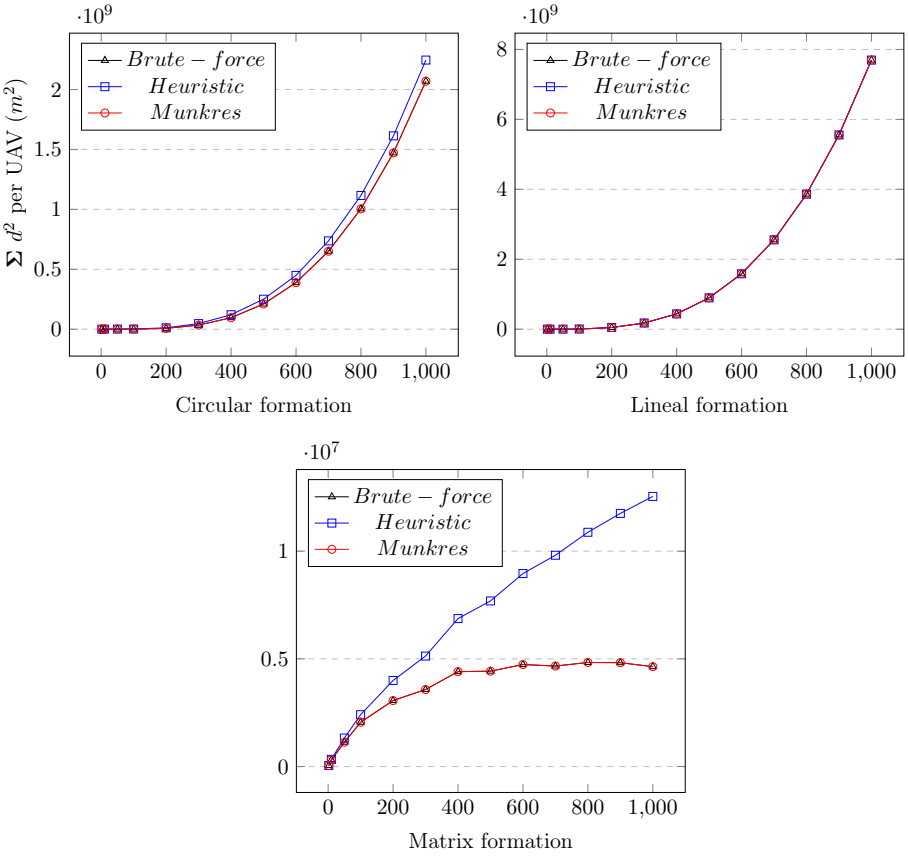


Figure 3.7: Distance matrix between all the nodes.

In the calculation of the quality of the algorithm, the total distance covered by the members of the swarm has been taken into account. However, our approach does not try to minimize the number of crossed routes that may interfere with each other, and thus potentially increase the total take-off time. Notice that those drones whose ground-to-air assignment segments intersect at some point will force a prioritization mechanism to be adopted in order to achieve collision avoidance.

3.6 Conclusions and future work

In this document we have evaluated the performance and quality of different algorithms for the problem of vertical takeoff of UAV swarms. We present a new approach to the problem by applying graph theory, and we propose using the Kuhn-Munkres algorithm to try to obtain the best possible allocation in an assumable execution time.

As future work we will seek to optimize the takeoff time by taking into account the potential crossing routes that may exist, and try to reduce the assignment time required by the Kuhn-Munkres algorithm by making a parallel implementation of it.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science and Innovation, under the Ramon y Cajal Program (Grant No. RYC2018-025580-I) and under grants RTI2018-096384-B-I00, RTC2017-6389-5 and RTC2019-007159-5.

References

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, et al. "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges". In: *IEEE Access* 7 (2019), pp. 48572–48634. DOI: 10.1109/ACCESS.2019.2909530 (cited on p. 20).

[2] M. Campion, P. Ranganathan, and S. Faruque. "UAV swarm communication and control architectures: a review". In: *Journal of Unmanned Vehicle Systems* 7.2 (2019), pp. 93–106. DOI: 10.1139/juvs-2018-0009 (cited on p. 20).

[3] N. Goddemeier, K. Daniel, and C. Wietfeld. "Role-Based Connectivity Management with Realistic Air-to-Ground Channels for Cooperative UAVs". In: *IEEE Journal on Selected Areas in Communications* 30.5 (June 2012), pp. 951–963 (cited on p. 20).

[4] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila. "Swarms of Unmanned Aerial Vehicles — A Survey". In: *Journal of Industrial Information Integration* (2019), p. 100106. ISSN: 2452-414X. DOI: https://doi.org/10.1016/j.jii.2019.100106 (cited on p. 20).

[5] N. Dousse, G. Heitz, and D. Floreano. "Extension of a ground control interface for swarms of Small Drones". In: *Artificial Life and Robotics* 21.3 (Sept. 2016), pp. 308–316. ISSN: 1614-7456. DOI: 10.1007/s10015-016-0302-9 (cited on p. 21).

[6] F. Fabra, J. Wubben, C. T. Calafate, J. C. Cano, and P. Manzoni. "Efficient and coordinated vertical takeoff of UAV swarms". In: (2020), pp. 1–5 (cited on pp. 21, 23).

[7] D. P. Bertsekas. "A distributed algorithm for the assignment problem". In: *Lab. for Information and Decision Systems Working, MIT* (1979) (cited on p. 25).

[8] H. W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97 (cited on p. 25).

[9] D. König. "Über graphen und ihre anwendung auf determinantentheorie und mengenlehre". In: *Mathematische Annalen* 77.4 (1916), pp. 453–465 (cited on p. 25).

[10] J. Munkres. "Algorithms for the assignment and transportation problems". In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38 (cited on p. 25).

[11] Z. Sui, Z. Pu, and J. Yi. "Optimal UAVs formation transformation strategy based on task assignment and Particle Swarm Optimization". In: *2017 IEEE International Conference on Mechatronics and Automation, ICMA 2017* (2017), pp. 1804–1809. DOI: 10.1109/ICMA.2017.8016091 (cited on p. 25).

[12] A. Frank. "On Kuhn's Hungarian method - A tribute from Hungary". In: *Naval Research Logistics* 52.1 (2005), pp. 2–5. ISSN: 0894069X. DOI: 10.1002/nav.20056 (cited on p. 25).

Chapter 4

AI-enabled autonomous drones for fast climate change crisis assessment

Hernández, D., Cano, J. C., Silla, F., Calafate, C. T., Cecilia, J. M. (2021). Al- enabled autonomous drones for fast climate change crisis assessment. IEEE Internet of Things Journal. doi: 10.1109/JIOT.2021.3098379.

4.1 Abstract

Climate change is one of the greatest challenges for modern societies. Its consequences, often associated with extreme events, have dramatic results worldwide. New synergies between different disciplines including Artificial Intelligence (AI), Internet of Things (IoT), and edge computing can lead to radically new approaches for the real-time tracking of natural disasters that are also designed to reduce the environmental footprint. In this article, we propose an AI-based pipeline for processing natural

disaster images taken from drones. The purpose of this pipeline is to reduce the number of images to be processed by the first responders of the natural disaster. It consists of three main stages, (1) a lightweight auto-encoder based on deep learning, (2) a dimensionality reduction using the t-SNE algorithm and (3) a fuzzy clustering procedure. This pipeline is evaluated on several edge computing platforms with low-power accelerators to assess the design of intelligent autonomous drones to provide this service in real time. Our experimental evaluation focuses on flooding, showing that the amount of information to be processed is substantially reduced whereas edge computing platforms with low-power GPUs are placed as a compelling alternative for processing these heavy computational workloads, obtaining a performance loss of only 2.3x compared to its cloud counterpart version, running both the training and inference steps.

4.2 Introduction

Climate has dramatic consequences all over the world, with effects having noticeably negative results [1]. The consequences of floods are undoubtedly one of the most dramatic ones among the many natural disasters, as they encompass loss of human life and loss of natural ecosystems. Floods also cause economic losses. Indeed, the effects of natural disasters have consequences where immediacy in decision-making is essential. Improving preparedness for an effective response to these events is essential in situations where every minute counts. In this regard, technological advances can help to achieve this efficiency in response times where sustainability, efficiency, and ubiquity should be the main ingredients of these developments [2].

Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, are autonomous unmanned aircrafts that are widely used for different applications and tasks. The use of drones has gradually evolved from more recreational areas such as photography and video, to more technical ones such as border surveillance, precision agriculture and infrastructure inspections, just to name a few [3]. Drones are also playing an important role in emergency and response protocols. They are currently widely used in the response stage and are also used, albeit to a lesser extent, in the other stages of a natural disaster, i.e., prevention, preparation and

recovery [4].

Natural disaster management situations are very stressful, and the use of technological tools such as drones could be helpful. However, using drones requires qualified personnel that can monitor and process the information generated by these tools. In particular, drones can generate an enormous amount of video and images that need to be analysed by experts in conditions where it is very easy to make mistakes. The probability of errors can be reduced by using image processing techniques for the detection of potential risks in a natural disaster, including Machine Learning techniques (ML) and Deep Learning (DL), which can automate the process of image interpretation and clustering in order to speed-up decision making by managers, and to avoid possible human errors. However, these AI-techniques, particularly their training, is a compute-intensive process, and although there is an industry-wide trend towards hardware specialization to improve performance and energy consumption [5], traditionally these workloads have been executed in a cloud-fashion approach. Nevertheless, the rescue of people, the identification of affected areas, and the prevention of the secondary effects of a natural disaster are all emergency tasks, and therefore the information should be processed in real time, or at least, as quickly as possible.

An alternative that is emerging in the last decade is edge computing [6]. In edge computing, data processing is performed, totally or partially, on the devices that are at the edge of the network; i.e., at those devices that are closest to mobile devices or sensors. This distributed way of computing provides energy savings, scalability and responsive web services for mobile computing, and offers a mechanism for data privacy in the IoT context. In addition, it offers the possibility to mask transient cloud outages. Edge computing devices should be designed in such a way that energy efficiency is the main objective. To this end, leading processor companies are developing energy-efficient solutions with low power consumption and high performance. In particular, the Nvidia Jetson family of embedded systems can be highlighted, which include low-power graphics accelerators (GPUs) that deliver good performance for massively parallel applications with power consumption between 7.5 and 10 watts [7].

In this paper, we propose an AI-based pipeline for the identification of the drone-based images that are related to floods. We use a deep-learning-based auto-enconder to highlight the main features of the images

taken from the drones. Then, those features are reduced and clusterized to help first responders of natural disasters in dealing with large datasets. We also evaluate the AI-based pipeline in different low-power GPU-based edge computing platforms to figure out if they would be a compelling alternative to the main aim of developing an AI-based autonomous drone for emergency situations. Hence, the major contributions of this paper are:

1. A deep-learning based lightweight auto-encoder is proposed to identify the main features of aerial flood images.
2. An AI-based pipeline to reduce the amount of information to be supervised by first responders in natural disasters is designed.
3. An in-depth performance evaluation of different low-power GPU-based edge computing devices is provided to assess the feasibility of autonomous AI drones in natural disasters.
4. A particular case study that targets flooding scenarios is under study.

The rest of the article is structured as follows. Section 4.3 provides the required knowledge related to the main research areas of this work. Section 4.4 introduces the general infrastructure of the AI-based pipeline to deal with aerial images of natural disasters. Section 4.5 shows the experimental setup before showing the performance and quality evaluation of our approach in Section 4.6. Finally, Section 5.6 shows conclusions and directions for future work.

4.3 Background and related work

This section provides the main background on the different topics related to this paper: UAVs, DL for artificial vision, and edge computing.

4.3.1 UAVs in natural disasters

UAVs have recently experienced unprecedented growth, with countless areas of application foreseen for the coming years [8]. Regarding works proposing the use of drones for rescue and survivor search tasks, the first

works in the area [9] studied the use of a single drone for this endeavour. Years later drones have been used to analyse the effects of a landslide in Tibet [10], comparing the profile of the terrain before and after the catastrophe. More recently, Mehrdad et al. [11] proposed a technique specifically applicable to natural disasters that is able to quickly and efficiently combine aerial images, and which has direct applicability in the case of using a swarm of drones to obtain such images. However, if we are attempting to perform complex tasks in a short time, such as assessing the effects of a natural disaster and searching for survivors, the deployment of swarms of UAVs is a very interesting alternative. Drone swarms can improve the efficiency of individual systems by offering many advantages, including the possibility of extending mission coverage in a short time, thanks to the cooperation between UAVs [12].

To date, very few tests have been conducted with outdoor multicopter swarms, and even fewer have been conducted on large scale, the most notable to date being the test conducted by China, which managed to coordinate up to 1000 UAVs for the first time ever [13]. This lack of works in the literature is due to the fact that using a swarm of drones collaborating with each other to perform a cooperative task presents significant communication, synchronisation and quality of service issues [14].

4.3.2 Artificial vision in natural disasters

Natural disasters present characteristics where immediacy in decision-making is fundamental. There are image processing techniques for the detection of potential risks in a natural disaster, including ML techniques such as Support Vector Machines (SVM), Bayesian non-parametric Models, Genetic Algorithms (GA), Random Forest (RF), Fuzzy Clustering (FC) or K-nearest Neighbours (KNN), that are used for image classification. In [15], authors proposed an extended motion diffusion-based (EMD) to detect changes in airport ground. Its method was verified from the Airport Ground Video Surveillance (AGVS) benchmark test by obtaining positive results in situations such as fog and camouflage. In [16], a genetic algorithm combined with a neural network was proposed to classify images coming from a flood; this proposal was compared with three FC methods, being that the proposed algorithm was able to obtain better results. The authors of [17] proposed a hybrid framework composed

of a Deep Learning algorithm, specifically a convolutional neural network (CNN), and a feature extraction algorithm, to classify images of natural disasters such as avalanches, cyclones, tornadoes and fires, among others. The data used to test this framework was an artificial dataset created by the authors. The proposed CNN is compared with RF, SVN, and KNN techniques, obtaining the CNN the best result. The same happens with the CNN proposed in [18]. In that study, the authors proposed the use of a CNN for flood image classification using images obtained from a UAV, producing such offline image classification. The authors made a comparison with a SVM technique, obtaining better results with the proposed CNN-based technique. In short, the techniques based on Deep Learning are the ones that achieve the best performance in image classification of natural disasters so far.

There are some works in the literature where Deep Learning techniques are applied for image classification in general and, in particular, for images of natural disasters. However, these techniques consume a lot of computational resources, and image classification is performed offline. However, the rescue of people, the identification of affected areas, and the prevention of secondary effects of a natural disaster, are emergency tasks requiring information to be processed in real time. Hence, our paper explores the design of these techniques so that they can be executed in low-power processors that can be introduced into the UAV swarm; this allows performing real-time image processing from different perspectives (thanks to the swarm and coordination of the UAVs) to make effective and accurate decisions in real time.

4.3.3 Edge computing platforms

In the history of computing, the paradigms of centralised and decentralised computing have alternated over time. In the early days, computing was developed using centralised processing with batch and time-sharing techniques. The development of personal computers in the 1980s brought about a shift to a decentralised approach. This approach was re-centralised at the beginning of the 21st century with cloud computing. Cloud computing has now established itself as the most widely used approach, mainly driven by the rise of mobile devices and the IoT, for which cloud computing offers high-performance computing and storage services that are not available on these low-power, low-cost devices. However, the

nearest cloud infrastructure running mobile and IoT application services may be too far away from the source of data.

Satyanarayanan et al. [6] proposed a two-level architecture to pursue interactive performance of mobile applications. A first level consisting of a traditional cloud and a second level consisting of a network of cloudlets; i.e. dispersed elements containing state information cached from the first level [19]. In addition, Bonomi et al. also proposed a multi-tier architecture that they called fog computing. In this case, the authors designed this architecture motivated by the lack of scalability of IoT infrastructures [20]. As in the case of edge computing, the proximity of cloudlets (or fog nodes) to the nodes capturing data offers a number of benefits, in addition to the scalability benefits initially sought by the authors. These benefits include the availability of highly responsive cloud services, the reduction of end-to-end latency, the increased bandwidth and low jitter to services located at the edge, etc. In [21], authors studied computation offloading in fifth generation networks and proposed a distributed learning method to address the technical challenges arising from uncertainties and limited resource sharing in an multi-access edge computing (MEC) system. They provided a case study on resource orchestration to show the potential of the proposal, outperforming benchmark resource orchestration algorithms.

Edge computing provides computing power in close proximity to sensors or mobile devices. In fact, as mentioned above, there are compute-intensive applications for which this technology is opening up new development opportunities such as interactive mobile applications for augmented reality. Undoubtedly, the design of cloudlets has to be highly energy efficient, while providing the highest computational horsepower possible. Actually, microprocessor industry is releasing systems on chip (SoCs) that include low-power accelerators such as Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs). Among them, we may highlight the Nvidia's Jetson family [22], Intel's Movidius [23] or the Google's Coral project [24]. Thanks to these accelerators, the energy efficiency of edge devices can substantially increase.

Another important feature of edge/fog computing related to this work is the reduction of the amount of data that needs to be transferred to the cloud. This has great benefits such as reduction of network overhead, energy savings, cost reduction in the cloud, reduction of storage space, etc. For instance, Simeons et al. developed a video processing system, known as GigaSight [25], where videos obtained from mobile devices are processed

in the nearest cloudlet, sending only the results and some metadata to the cloud, drastically reducing the application's bandwidth and storage needs. This feature is of particular interest for the UAV environment, where autonomy is scarce. Furthermore, in the particular case of natural disasters, the reduction of data delivery through edge processing provides clarity in analysing the information for first responders.

4.4 AI-pipeline proposed for management of natural disasters

Natural disasters require immediacy so that decisions can be taken as quickly as possible to save lives. First-responders need tools that allow them to quickly assess the magnitude of the natural disaster. As previously mentioned, drones are capable of exploring wide areas inaccessible by first-responders, allowing a large number of images to be taken to assess the impact of the disaster. However, manually processing this large amount of information is very difficult for humans, even more in these types of critical scenarios.

This section introduces the AI-pipeline proposed to deal with unclassified drone-based images of natural disasters. The main objective of this AI-pipeline is to reduce the number of drone-generated images to be processed by the first-reponders. This is developed through an unsupervised process which identifies the main features of the images through a deep-learning based auto-encoder and reduces the dimensionality of these features to eventually perform a clustering process to group those images by similarity. This AI-pipeline outputs an image that represents each cluster. This image can be evaluated by first-responders to determine whether that group of images are of interest for decision making. Another important feature of natural disasters is that they usually occur in remote locations, where connectivity is limited. For this purpose, the last part of this work evaluates the complete execution of the proposed AI-pipeline on egde computing platforms, so that Internet connection will not be required to obtain the benefits of this system.

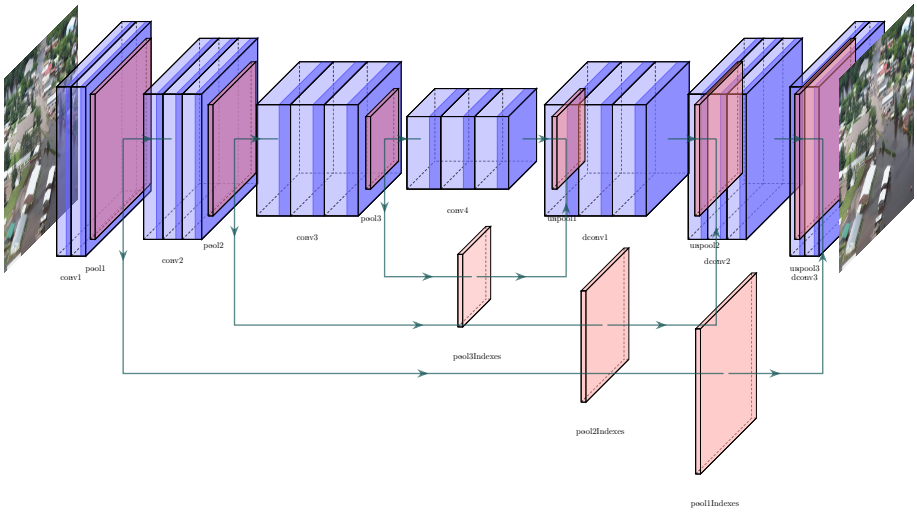


Figure 4.1: Deep learning based auto-enconder.

4.4.1 Deep learning based auto-encoder structure

An auto-encoder is a neural network architecture designed for feature learning from unlabeled data. It has a distinctive shape consisting of two layers; the first one is the encoder which is responsible for data compression so that it becomes smaller in size than its input by decreasing its dimensionality from the input layer to a central information layer. The other is the decoder, which attempts to regenerate the input data compressed by the first layer to regenerate the original input to the encoder phase as faithfully as possible.

The data to be processed is formed by a set of images, therefore the neural network that forms the auto-encoder is composed of convolutional networks that behave better for this type of scenario. This type of auto-encoder is known as convolutional auto-encoder (CAE). The structure of the proposed auto-encoder for this work is described in Figure 4.1. For its design, we have tried to prioritize simplicity and the reduction of the total number of parameters as much as possible, since this network will be trained and used for the clustering process entirely in the devices located at the edge, and therefore the limitations of memory and computational capacity have been taken into account.

One of the differences with traditional auto-encoders is that no fully-

connected layers have been used within the autoencoder. Moreover, average pooling operations are performed on the feature map extracted from the filters of the central layer for the feature extraction process. The blocks that compose the CAE are convolutional, as well as the pooling layers for the encoding and compression phase, and deconvolutional and unpooling blocks for the regeneration phase of the original input.

Convolutional blocks, called *"conv"* in Figure 4.1, are responsible for filtering an input to create a feature map that summarizes the presence of features detected in that input. In contrast, deconvolutional elements, called *"deconv"* in Figure 4.1, apply a 2D transposed convolution operator on an input image composed of several input planes. This operation can be viewed as the gradient of Conv2d with respect to its input, also known as fractional convolution. With this operation, we will decompress the abstract representation generated by the convolutional layers into something more visual.

Max Pooling, *"pool"* in Figure 4.1, is a sample-based discretization process designed to filter out noisy activations by retaining only the robust activations in the upper layers, but the spatial information is lost during pooling. This reduces its dimensionality and allows assumptions to be made about the features contained in the binned subregions. Unpooling, *"unpool"* in Figure 4.1, is the opposite operation. It captures example-specific structures by tracing the original locations with strong activations back to image space. As a result, it reconstructs the detailed structure that was done in the pooling phase. Pooling and unpooling layers do not have tuning parameters, although they will have to share parameters between them since indices generated by each of the pooling layers in the encoding part have to be sent to their respective unpooling layer when decoding in order to reconstruct the original dimensionality prior to the pooling operation. This operation can be seen in layers *poolxIndexes* in Figure 4.1.

4.4.2 t-SNE

The second main step of the proposed pipeline is a dimensionality reduction. The objetive of this step is two-fold, (1) for easy viewing of information and (2) for reducing the computational complexity. We use a t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm [26], which is a non-linear technique that reduces the number of dimensions

of the input data. More specifically, we use the t-SNE implementation made available by the scikit-learn Python library [27]. The algorithm searches for joint probabilities based on similarities between data points. In our case, each data point is the average flatten pooling obtained from the convolutional autoencoder. This input is a 240-feature vector, and we apply t-SNE to obtain a two-dimensional one before performing the clustering. The t-SNE algorithm attempts to minimise the so-called Kullback-Leibler (KL) divergence between the input data and the joint probabilities of the low-dimensional embedding. This divergence is a way to measure the difference between two distributions. The t-SNE procedure follows the following equations. First, Equation 4.1 shows the calculation of the conditional probability of point x_j to be next to point x_i.

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma^2\right)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma^2)} \tag{4.1}$$

Then, the joint probability distribution is calculated based on the conditional distributions (see Equation 4.2).

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_i - y_l\|^2)^{-1}} \tag{4.2}$$

Finally, the KL divergence is calculated for both distributions P and Q in the probability space of x to optimize their distribution (see Equation 4.3).

$$\mathrm{KL}(P\|Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)} \tag{4.3}$$

4.4.3 The clustering algorithm

Clustering algorithms are data analysis techniques that organise n-dimensional data-points into groups or *clusters*. Each cluster is composed of those points that are most similar to each other based on a metric defined in the algorithm [28]. Several clustering algorithms have been proposed in the literature as applied to different scientific applications [29, 30]. They are mainly divided into two main groups, i.e. hard and soft clustering techniques. In hard clustering techniques, such as the well-known k-means algorithm, a data-point can only belong to one group.

However, in soft clustering techniques, a probability of belonging to each group is assigned to each data point. Among the soft clustering techniques, we may highlight the fuzzy c-means (FCM) algorithm [31] and the fuzzy minimals (FM) algorithm [32]. Another important feature of clustering algorithms is the number of clusters to be generated; k-means and FCM require us to define the number of clusters to be developed. In contrast, FM does not need this parameter to be set in advance without the need for the clusters to be Compact Well-Separated (CWS). This feature enables unsupervised clustering that does not condition the sets of images to be obtained from each drone mission, and thus it is what motivated us to introduce FM at the final stage of the pipeline proposed in this article. In what follows, we briefly introduce the FM algorithm and refer the reader to [32, 33] for insights.

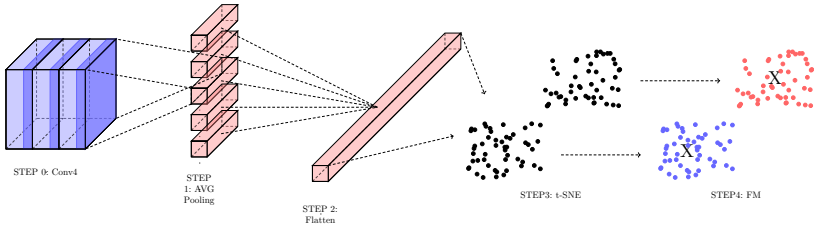


Figure 4.2: AI-pipeline for the latent space clustering extracted from the autoencoder.

The FM algorithm is an iterative fixed-point algorithm whose main purpose is to minimise an objective function given by Equations (4.4) and (4.5). The FM algorithm has two main functions shown in Algorithm 3. The FM needs two input parameters: $varepsilon_1$, which sets the maximum allowable degree of error, and $varepsilon_2$, which shows the difference between the potential minima. Once these parameters are set, the r factor is calculated for the input dataset before calculating the prototypes.

$$J_{(v)} = \sum_{x \in X} \mu_{xv} \cdot d_{xv}^2, \tag{4.4}$$

where

$$\mu_{xv} = \frac{1}{1 + r^2 \cdot d_{xv}^2}, \tag{4.5}$$

Algorithm 3 Fuzzy Minimals Algorithm pseudocode.

```
1: LoadDataSet()
2: Choose input variables ε1 and ε2.
3: r=CalculateFactorR(dataset)
4: PrototypeCalculation(dataset, r) =0
```

The r-factor is a non-linear function for measuring the degree of homogeneity and isotropy breakdown of a data set. This function is shown in Equation (4.6). Factor r takes as a partition hypothesis that clusters are created when isotropy and/or homogeneity is broken.

$$\frac{\sqrt{|C^{-1}|}}{nr^F} \sum_{x \in X} \frac{1}{1 + r^2 d_{xm}^2} = 1, \tag{4.6}$$

where $|C^{-1}|$ is the determinant of the inverse of the covariance matrix, m is the mean of the sample X, d_{xm} is the Euclidean distance between x and m, and n is the number of elements of the sample.

Algorithm 4 shows the calculation of prototypes developed by the FM approach. This procedure is based on a scoring function that has, as an argument, the previously calculated r factor. The membership function is described in equation 4.5, which measures the probability of an element x to belong to a particular cluster in which v is the prototype. In fact, these prototypes are the output of the FM that represents the most significant images of our pipeline.

4.4.4 AI-pipeline ensemble

Figure 4.2 shows the proposed AI-pipeline applicable after CAE training. The pipeline begins with the raw images that go through the auto-encoder shown in Figure 4.1. The trained auto-encoder is able to compress the main features in the conv4 layer. This layer outputs a matrix with a transformation of the input image that has been obtained by applying a set of filters. At this point, the layers behind conv4 will be removed, and this will be the new output layer of the network. Then, an iteration

Algorithm 4 Baselines of the FM *PrototypeCalculation*. n is the size of the input dataset. V is a vector with the prototypes found by FM. F is the dimensionality (2 in our case).

Initialize $V = \{\ \} \subset^F$.
for $i = 1; i < n; i++$ **do**
  $v_{(0)} = x_i$, $t = 0$, $E_{(0)} = 1$
  while $E_{(t)} \geq \varepsilon_1$ **do**
    $t = t + 1$
    $\mu_{xv} = \frac{1}{1+r^2 \cdot d_{xv}^2}$, using $v_{(t-1)}$
    $v_{(t)} = \frac{\sum_{x \in X} \left(\mu_{xv}^{(t)}\right)^2 \cdot x}{\left(\mu_{xv}^{(t)}\right)^2}$
    $E_{(t)} = \sum_{\alpha=1}^{F} \left(v_{(t)}^{\alpha} - v_{(t-1)}^{\alpha}\right)$
  end while
  if $\sum_{\alpha}^{F} (v^{\alpha} - w^{\alpha}) > \varepsilon_2$, $\forall w \in V$ **then**
    $V \equiv V + \{v\}$.
  end if
end for=0

of the whole dataset will be performed to obtain all the filters extracted from the network for each image.

Each image processed by the autoencoder is converted into a set of matrix-represented filters.The Average (AVG) polling is applied on 2×2 blocks to each of these filters in order to create a down-sampled (pooled) feature map as shown in step 2 at Figure 4.2. The *AVG polling* obtains an one-dimensional vector for each filter by calculating the mean of each block of each filter. This means that each 2×2 square of each filter is sampled downwards to the mean value of the square. These 1-D vectors will be concatenated to feed step 3 called *Flatten* in Figure 4.2, forming a single vector where all relevant CAE information will be extracted.

Once a flatten vector for each of the images is obtained, the t-SNE algorithm is applied so that all images will be embedded into a two-dimensional array grouped by the similarity detected after applying t-SNE. It is important to note that this matrix contains all the images without creating any clusters; i.e. all images are considered as homogeneous points. Then, this matrix is clustered based on image similarity. Since the optimal

number of clusters to be obtained is not known, a fuzzy logic based clustering will be applied using the FM algorithm, where the coordinates of the most representative images of the dataset (i.e. the prototypes of the cluster) will be obtained along with the percentage of belonging to each of the image clusters generated. With this extracted information, every image can be labeled for subsequent submission. Images representing the prototypes will be sent for follow-up by the first responders.

4.4.5 Solution deployment and execution flow

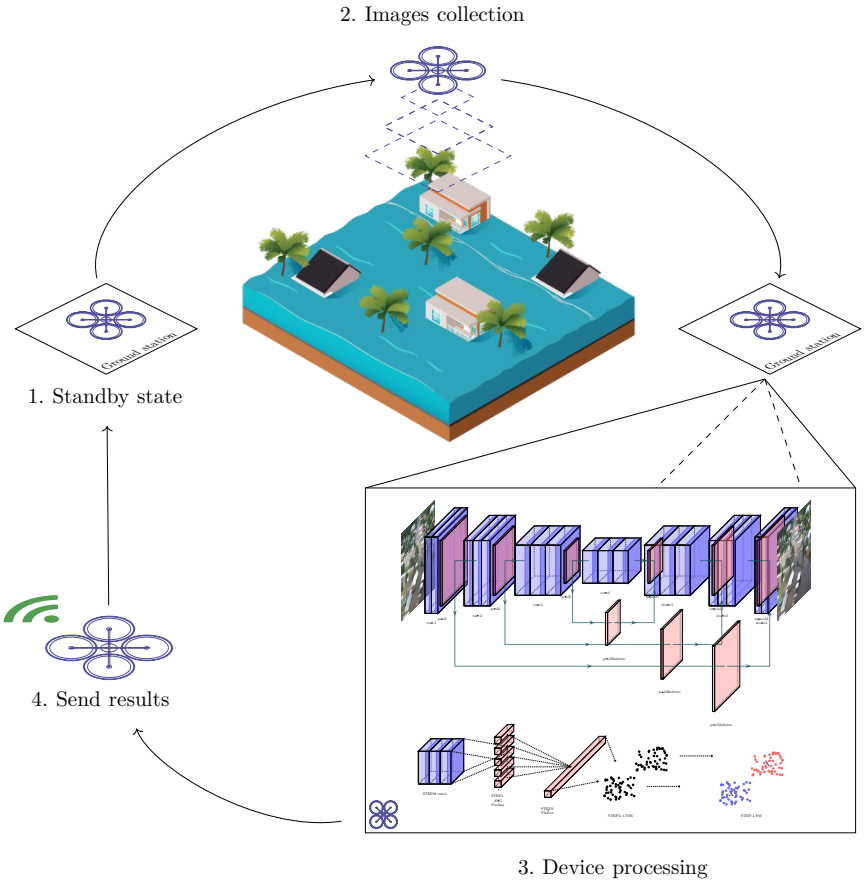


Figure 4.3: Natural disasters management overview.

Having defined the problem and the proposed AI-pipeline, Figure 4.3 shows the execution flow that should be carried out in real natural disaster management scenarios. Drones start from a standby position (step 1) in order to prepare for take-off before they begin taking images of the affected area. Once images have been taken in step 2, the proposed AI-pipeline is executed (step 3). The first step of this pipeline would be the training of the autoencoder, which is performed with the images captured by the drone during the first flight. It is important to note that this training is only performed once for each affected area, and the information learned by the autoencoder can be reused to encode the images of the following flights over the same affected area. Therefore, the training stage, the most computationally expensive, would be executed once and the inference stage would be executed as many times as surveillance missions are performed by the drone. In other words, the training stage will provide the neural network weights that will be used in the following missions in the inference stage, running only the first part of the network (i.e., Conv1-Conv4 in Figure 4.1). After the inference stage, the feature vector will be generated and reduced using t-SNE and clustered using FM. Finally, once the most significant images have been selected with the execution of the AI-pipeline, they are sent in step 4.

Regarding the execution flow, all steps of the proposed AI-pipeline could be executed at the edge. The pipeline is designed to have a low memory footprint, as it will be shown in Section 4.5. Furthermore, the pipeline only uses the images captured during the mission (step 2 in Figure 4.3). Therefore, there is no need to perform a knowledge transfer with the pre-trained network weights using other datasets.

4.5 Experimental setup

This section provides an overview of the dataset used to train and test the AI-pipeline proposed in Section 4.4. Then, the main hardware features and software details of our experimental environment are described.

4.5.1 Dataset

The AI-pipeline previously introduced requires a dataset to train the autoencoder. To the best of our knowledge there are few datasets that meet our requirements; i.e. natural disaster and aerial images. Particularly, we

use AIDER (Aerial Image Dataset for Emergency Response applications) [34, 35] for the training and testing procedures, as will be shown in Section 4.6. This dataset contains images from five different categories. Four of them are related to disaster events such as Fire/Smoke, Flood, Collapsed Building/Rubble, and Traffic Accidents, and the latter is the control case; i.e., there is not any sort of accident on it. Figure 4.4 shows all categories contained in this dataset with random examples of them to illustrate its content. It is important to note that AIDER is only composed of aerial images that were obtained by several online sources such as Google or Bing images, Youtube or news agencies websites. Particularly, authors used the keywords ”Aerial View”, ”UAV” and/or ”Drone”, along with the particular event they wanted to include in the dataset, such as "flood” or "fire". Moreover, images also have different viewpoints, resolutions, and illumination conditions. It is important to note that authors manually inspected all images to make sure that they are related to the expected disaster, and also that the event is centered at the image. The latter is to guarantee that any geometric transformation during augmentation does not remove the object of interest from the image. Finally, the dataset is not well balanced to replicate real world scenarios; i.e., it contains more images from the control class. In particular, the dataset is composed of about 500 images for each disaster class, and over 4000 images for the control class. In our case, the dataset is even more imbalanced as we have removed the images from Fire/Smoke, Collapsed Building/Rubble, and Traffic Accidents and will only use the control class and the flooding class. As shown in Figure 4.4, we will use the set made up of the normal and flood images as the framework on which we will perform the clustering tasks. Before being processed by the clustering pipeline, all images have been resized to a side size of 255 pixels, and all of them have been cropped at the center.

Table 4.1: Specification of the various GPU platforms used in our experiments.

	Pedra	**Jetson AGX Xavier**	**Jetson TX2**	**Jetson Nano**
CPU	Intel Silver 4216	NVIDIA Carmel ARM v8.2	ARMv8	ARM Cortex-A57 MPcore
2xGPU (NVIDIA)	GeForce RTX 2080 Ti	Volta	Pascal	Maxwell
Memory [Gb]	12 DDR5	32 LPDDR4x	8 LPDDR4	4 LPDDR4
Size [mm]	73.4 x 8.7 x 44.8	105 x 105	50 x 87	70 x 45
Weight [g]	17,000	280	85	61
Energy consumption [W]	80-100	10-30	7.5	3-5

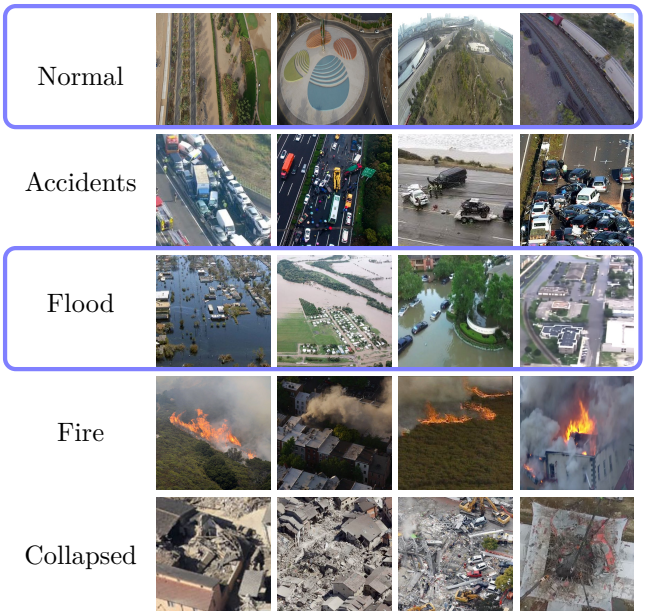


Figure 4.4: Classes within the AIDER dataset.

4.5.2 Hardware and software environment

This section introduces the hardware and software environment used to perform the experiments in Section 4.6 (see Table 5.1). We focused on four different architectures: a High Performance Computing (HPC) node called Pedra, and three low-power edge computing devices from the NVIDIA Jetson family (i.e. Jetson nano, Jetson TX2, and Jetson AGX Xavier). Although Pedra cannot be mounted directly on the UAVs (due

to its weight, size, and energy consumption), it could be used via a cloud solution when mobile Internet speed and coverage are sufficient. The main purpose of this comparison is to determine whether the AI-pipeline proposed in Section 4.4 adapted for GPUs decreases the calculation time, if it can also be performed in the edge in a reasonable time frame and, if so, which platform is the most suitable one. Table 5.1 introduces the the main features of the hardware platforms targeted.

The software environment is based on gcc v7.4.0, CUDA v10.2 with cuDNN and Python v3.6 with pytorch v1.8.0 built for edge devices, torchvision v0.9.0 built for edge devices and scikit-learn v0.24.1.

On the other hand, notice that the training stage of the neural network described above has been performed on all the devices described in Table 5.1 in order to achieve a process that is able to run on the edge from the beginning to the end. This approach differs from the usual practice of first training on a high-performance cluster, and then trying to apply techniques to adapt the trained model to low-performance devices. In our case, we designed a model with a total size of 88275 trainable parameters (the lightness of the network is evident) that is efficient enough for the target task. The main parameters of the training procedure were 30 epochs, MSELoss loss criterion, and Adam Optimizer. The batch size to feed each epoch has been adapted depending on the memory limitations of each device that was running the training process at that time.

It is worth mentioning that although this AI-pipeline is designed to train a new auto-encoder for each available dataset, if the new images are similar to those used in a previous dataset, e.g. same geographical area or same day time, with similar weather, a previously trained encoder could be reused to obtain the most relevant images for this new particular scenario. Another option could be to perform new incremental training epochs starting from a previously trained auto-encoder to which new images are added, thus reducing the convergence time with respect to performing a training process from the beginning.

4.6 Evaluation

This section shows the quality results obtained by training and validating the aerial images dataset shown in Section 4.5.1. Moreover, the per-

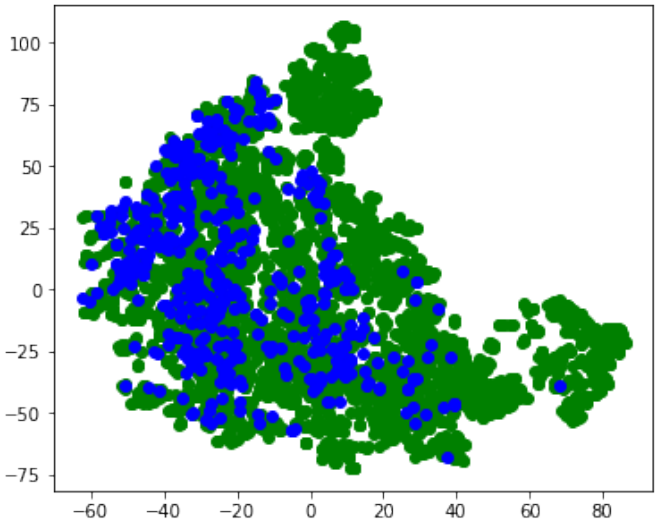
formance of several CPU-GPU based computing solutions is presented to evaluate edge computing platforms as potential infrastructure for processing AI-based workloads on drones.

4.6.1 Quality evaluation and memory footprint

As explained in Section 4.4.4, all images of the target dataset are fed into the pipeline. First, the auto-encoder compresses the information into a feature vector that is reduced to two dimensions for later visualisation. Such a visualisation is shown in Figure 4.5. In Figure 4.5a the images are represented as data points where the blue points are images related to floods, and green points are the other images. Figure 4.5b shows the clustering generated by t-SNE which creates an image cloud where, for each coordinate extracted from the clustering phase, the original image corresponding to the index of that position has been drawn. It is important to note that, although the dataset is labelled so that the class of each image is known in advance, our methodology does not use this information at any point. This information is used only for evaluation purposes, and to figure out which images are actually related to the flooding class.

The cluster prototypes generated by the FM algorithm are shown in Figure 4.6a where 48 different groups have been detected for the targeted dataset. It is worth noting that the main objective of this work is to synthesise the information sent by the drone in an unsupervised manner. In this way, the drone would only send 48 out of the 5500 images captured during the mission. These 48 images are the most representative ones in the dataset. From these images, the natural disaster managers would be able to identify which images are of real interest for their work, being able to access all the images in that cluster if required. In particular, Figure 4.6b shows the prototype images that are related to flooding (highlighted in red).

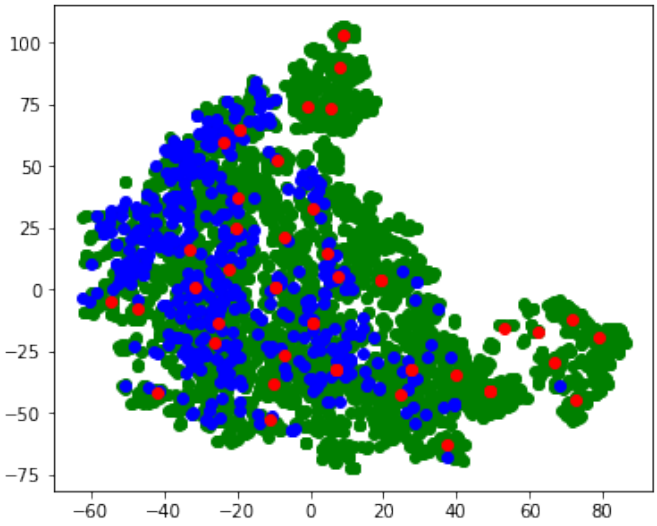
Finally, it is worth mentioning that there are deep learning models which can be used for feature extraction, and that are usually pretrained on datasets such as MobileNetV2 [36], Inception V3 [37], ResNet50 [38] or VGG16 [39]. The use of these pretrained models can be a good alternative to an autoencoder in high-performance environments where the highest possible accuracy is required. However, these models have a high memory footprint, and they are very heavy for edge computing



(a) Data points for flooding images (blue), and others (green)

(b) Images corresponding to data points.

Figure 4.5: Data points and images after running the t-SNE algorithm.



(a) Data points of flooding images (blue), others (green), and prototypes obtained by the FM algorithm (red).

(b) Images closer to the prototype found by FM.

Figure 4.6: Data points and images after running the FM algorithm.

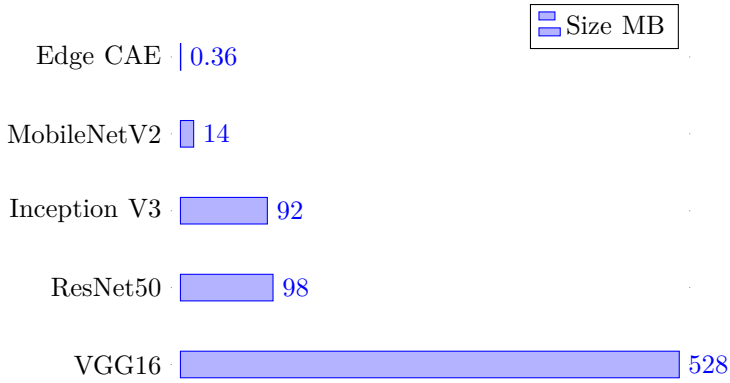


Figure 4.7: Memory footprint of similar models used for feature extraction.

devices. Figure 4.7 shows a memory footprint comparison between these models and our proposal, Edge CAE. It can be seen that our proposal is 38x lighter than the lightest proposal in the state of the art, and orders of magnitude lighter than the rest. It makes sense, as these networks have been designed for supercomputing environments, which makes them not feasible in resource-constrained IoT environments.

A further consideration is that these models have been pre-trained on classified images using a different sample than the type of images used for the clustering phase. Therefore, this proposal intends to perform a sandbox execution from start to end without relying on data other than that collected by the drone in an actual mission.

4.6.2 Performance evaluation

This section evaluates the performance (i.e. execution time) of the AI-pipeline previously presented in Section 4.4. We focused on four different architectures (see Table 5.1): a High Performance Computing (HPC) node called Pedra, and three low-power edge computing devices from the NVIDIA Jetson family (i.e. Jetson nano, Jetson TX2, and Jetson AGX Xavier). The main purpose of this comparison is to determine whether the AI-pipeline proposed adapted for GPUs reduces the calculation time, and whether it can be performed in the edge in a reasonable time frame

and, if so, which platform is the most suitable.

First of all, the training stage of the auto-enconder is evaluated in Figure 4.8. It shows the execution time for CPU and GPU versions in all targeted architectures. We ran 30 epochs, which is the actual number of epochs needed for convergence. Several conclusions can be drawn from these numbers. First of all, the use of GPUs increases the performance in all platforms, including edge computing platforms. The performance difference between CPU and GPU in the server version is up to 16.5x speed-up factor by using a single GPU. This difference increases by a factor of 2.33x when 2 GPUs are targeted on the same server (i.e. Pedra). GPU performance numbers are also interesting on the edge computing side. The use of low-power GPUs in edge devices also increases application performance substantially. The Xavier GPU delivers up to 12.7x speed-up factor compared to the sequential code, executed on the Xavier ARM-based CPU. In the same way, the TX2 GPU delivers up to 10.11x speed-up factor compared to its sequential counterpart version. The difference decreases when using the Jetson Nano GPU, with a speed-up factor of up to 4.5x when using the GPU instead of the CPU. In fact, this GPU is a very low-power, low-cost solution

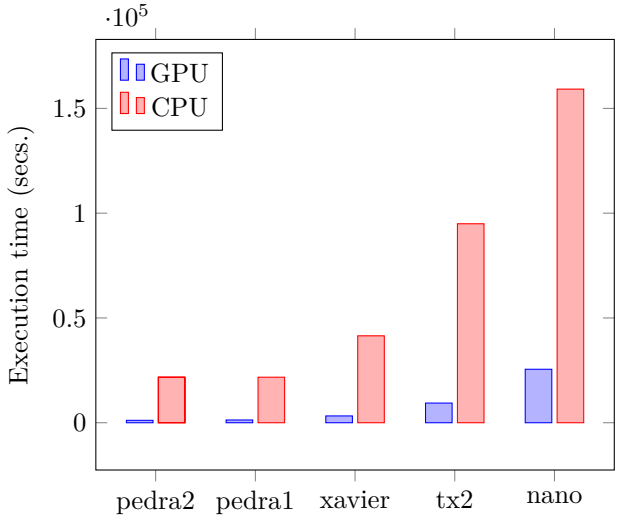


Figure 4.8: Execution time GPU/CPU for the auto-encoder's training stage.

that cannot offer a performance as high as its counterparts, but its use is definitely a good contribution in terms of efficiency.

Another relevant point is the performance difference between edge and cloud approaches. The cloud infrastructure, Pedra, defeats edge devices by a wide margin, as expected. Pedra, using two GPUs, achieves a speed-up factor of up to 2.8x compared to AGX Xavier, 8.15x compared to Jetson TX2, and 22.17x compared to Jetson Nano. It is important to note that these figures refer to training, which is not well-suited to be executed at the edge. For the inference stage, the performance differences between the computing platforms are similar (see Figure 4.9), but the overall execution time is much shorter.

Finally, the last two steps of the proposed AI-pipeline are shown in Figure 4.10. Execution times of these two steps are very low compared to the training and inference of the neural network. Therefore, these processes have been executed on CPU as their computation is hidden by the first step of the pipeline. In particular, the cross-platform differences for this algorithm are similar to those discussed above, with the pedra HPC server showing the best performance, followed by Xavier, TX2 and Jetson Nano. While it is true that the differences between Pedra and Xavier are 2.5x in speedup factor, and up to 7x between Pedra and TX2, in general execution times in the edge are reduced, with both algorithms

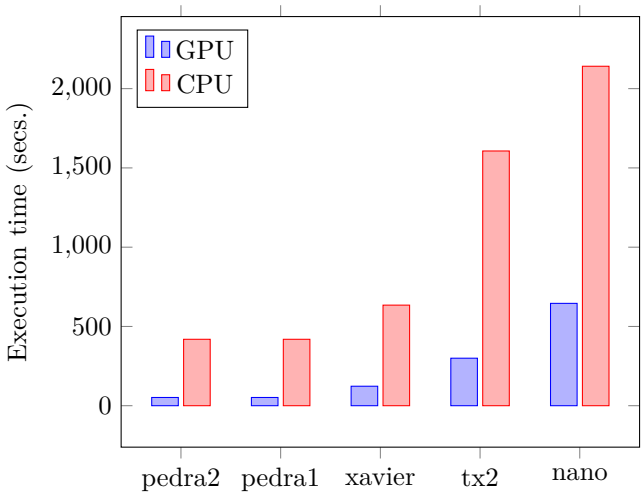
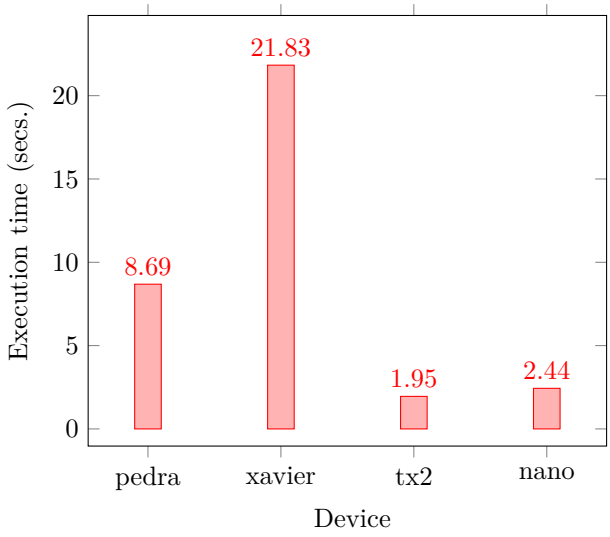
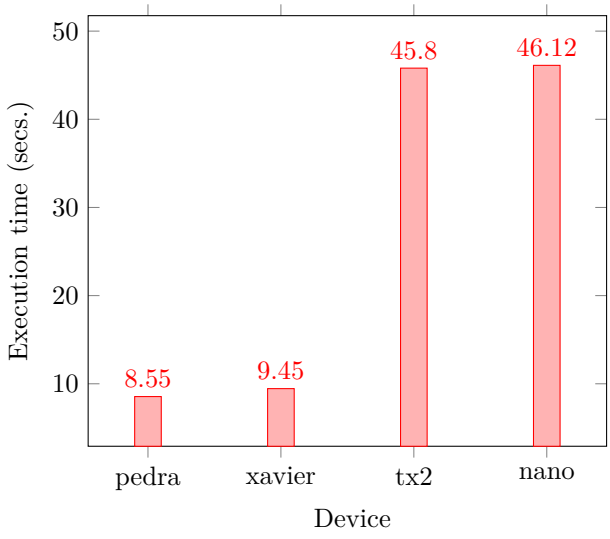


Figure 4.9: Inference comparison of GPU/CPU for the entire dataset.



(a) t-SNE algorihtm.



(b) FM clustering algorithm.

Figure 4.10: Execution time of the last two steps of the AI-pipeline.

taking less than a minute to run.

4.7 Conclusions and future work

Autonomous UAVs could play a "key role" in addressing the consequences of climate change. However, hardware and software developments are needed for these drones to really be determinant actors in these tasks. The intersection between AI and edge computing is undoubtedly the answer today, allowing to transform autonomous drones into useful tools under different emergency situations. This paper has proposed an AI-based pipeline to be run on edge computing platforms in order to enable efficient processing of drone images of natural disasters.

Our results reveal that the use of GPUs in edge computing platforms increases performance by up to a 12.7x speed-up factor, providing computational horsepower that enables the full execution of the proposed AI-pipeline. The computational differences between edge and cloud platforms are still large; in the range of 2.8x-22.17x speed-up factor, but the development of efficient platforms for the execution of specific workloads, such as those within deep learning, shows a roadmap that enables the development of applications for relevant autonomous and intelligent systems such as the one proposed here.

We definitely believe that smart autonomous drone technology can be a milestone in the fight against climate change. However, there is still a lot of work to be done from different perspectives. In terms of communication, extending the results of this article to a swarm of drones can provide a greater coverage of the area to be inspected, which is much needed in this type of natural catastrophes. The inclusion of highly energy-efficient processors in such low autonomy devices is a must in order to enable AI-based applications; tinyML is actually a good step forward in this direction. More processing steps could be added in this AI-pipeline which, after manual labeling or pseudolabeling of the clusters, the information stored in the autoencoder will be reused to categorize all the images and send the desired information in a more granular way. Finally, real-case scenarios will be approached with first-reponders to figure out new features and requirements.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science and Innovation, under grants RYC2018-025580-I, RTI2018-096384-B-I00 and RTC2019-007159-5, by the Fundación Séneca under grant E, and by the "Conselleria de Educación, Investigación, Cultura y Deporte, Direcció General de Ciéncia i Investigació, Proyectos under Grant AICO/2020/302.

References

[1] R. Dellink, E. Lanzi, and J. Chateau. "The sectoral and regional economic consequences of climate change to 2060". In: *Environmental and resource economics* 72.2 (2019), pp. 309–363 (cited on p. 36).

[2] S. K. Sood and K. S. Rawat. "A scientometric analysis of ICT-assisted disaster management". In: *Natural hazards* (2021), pp. 1–19 (cited on p. 36).

[3] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, et al. "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges". In: *IEEE Access* 7 (2019), pp. 48572–48634. DOI: 10.1109/ACCESS.2019.2909530 (cited on p. 36).

[4] M. A. R. Estrada and A. Ndoma. "The uses of unmanned aerial vehicles–UAV's-(or drones) in social logistic: Natural disasters response and humanitarian relief aid". In: *Procedia Computer Science* 149 (2019), pp. 375–383 (cited on p. 37).

[5] Y. E. Wang, G.-Y. Wei, and D. Brooks. "Benchmarking tpu, gpu, and cpu platforms for deep learning". In: *arXiv preprint arXiv:1907.10701* (2019) (cited on p. 37).

[6] M. Satyanarayanan. "The emergence of edge computing". In: *Computer* 50.1 (2017), pp. 30–39 (cited on pp. 37, 41).

[7] H. Halawa, H. A. Abdelhafez, A. Boktor, and M. Ripeanu. “NVIDIA jetson platform characterization”. In: (2017), pp. 92–105 (cited on p. 37).

[8] P. J. Hardin and R. R. Jensen. “Small-scale unmanned aerial vehicles in environmental remote sensing: Challenges and opportunities”. In: *GIScience & Remote Sensing* 48.1 (2011), pp. 99–111 (cited on p. 38).

[9] P. A. Rodriguez, W. J. Geckle, J. D. Barton, J. Samsundar, T. Gao, M. Z. Brown, and S. R. Martin. “An emergency response UAV surveillance system”. In: 2006 (2006), p. 1078 (cited on p. 39).

[10] Q. Wen, H. He, X. Wang, et al. “UAV remote sensing hazard assessment in Zhouqu debris flow disaster”. In: 8175 (2011), p. 817510 (cited on p. 39).

[11] S. Mehrdad, M. Satari, M. Safdary, and P. Moallem. “Toward real time UAVS’image mosaicking”. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 41 (2016), p. 941 (cited on p. 39).

[12] I. Bekmezci, O. K. Sahingoz, and Ş. Temel. “Flying ad-hoc networks (FANETs): A survey”. In: *Ad Hoc Networks* 11.3 (2013), pp. 1254–1270 (cited on p. 39).

[13] J. Lin and P. Singer. “China Is Making 1,000-UAV Drone Swarms Now”. In: *Popular Science* 8 (2018) (cited on p. 39).

[14] E. A. Marconato, J. A. Maxa, D. F. Pigatto, A. S. Pinto, N. Larrieu, and K. R. C. Branco. “IEEE 802.11 n vs. IEEE 802.15. 4: a study on communication QoS to provide safe FANETs”. In: (2016), pp. 184–191 (cited on p. 39).

[15] X. Zhang, H. Wu, M. Wu, and C. Wu. “Extended motion diffusion-based change detection for airport ground surveillance”. In: *IEEE*

Transactions on Image Processing 29 (2020), pp. 5677–5686 (cited on p. 39).

[16] A. Singh and K. K. Singh. "Satellite image classification using Genetic Algorithm trained radial basis function neural network, application to the detection of flooded areas". In: *Journal of Visual Communication and Image Representation* 42 (2017), pp. 173–182 (cited on p. 39).

[17] R. Nijhawan, M. Rishi, A. Tiwari, and R. Dua. "A Novel Deep Learning Framework Approach for Natural Calamities Detection". In: (2019), pp. 561–569 (cited on p. 39).

[18] A. Gebrehiwot, L. Hashemi-Beni, G. Thompson, P. Kordjamshidi, and T. E. Langan. "Deep convolutional neural network for flood extent mapping using unmanned aerial vehicles data". In: *Sensors* 19.7 (2019), p. 1486 (cited on p. 40).

[19] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. "The case for vm-based cloudlets in mobile computing". In: *IEEE pervasive Computing* 8.4 (2009), pp. 14–23 (cited on p. 41).

[20] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. "Fog computing and its role in the internet of things". In: (2012), pp. 13–16 (cited on p. 41).

[21] X. Chen, C. Wu, Z. Liu, N. Zhang, and Y. Ji. "Computation offloading in beyond 5g networks: A distributed learning framework and applications". In: *IEEE Wireless Communications* 28.2 (2021), pp. 56–62 (cited on p. 41).

[22] B. Blanco-Filgueira, D. Garcia-Lesta, M. Fernández-Sanjurjo, V. M. Brea, and P. López. "Deep learning-based multiple object visual tracking on embedded system for iot and mobile edge computing applications". In: *IEEE Internet of Things Journal* 6.3 (2019), pp. 5423–5431 (cited on p. 41).

[23] M. H. Ionica and D. Gregg. "The movidius myriad architecture's potential for scientific computing". In: *IEEE Micro* 35.1 (2015), pp. 6–14 (cited on p. 41).

[24] S. Cass. "Taking AI to the edge: Google's TPU now comes in a maker-friendly package". In: *IEEE Spectrum* 56.5 (2019), pp. 16–17 (cited on p. 41).

[25] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan. "Scalable crowd-sourcing of video from mobile devices". In: (2013), pp. 139–152 (cited on p. 41).

[26] L. Van der Maaten and G. Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008) (cited on p. 44).

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830 (cited on p. 45).

[28] A. Nagpal, A. Jatain, and D. Gaur. "Review based on data clustering algorithms". In: (2013), pp. 298–303 (cited on p. 45).

[29] Z. Cui, X. Jing, P. Zhao, W. Zhang, and J. Chen. "A New Subspace Clustering Strategy for AI-Based Data Analysis in IoT System". In: *IEEE Internet of Things Journal* (2021) (cited on p. 45).

[30] J. M. Cecilia, I. Timón, J. Soto, J. Santa, F. Pereñiguez, and A. Muñoz. "High-Throughput Infrastructure for Advanced ITS Services: A Case Study on Air Pollution Monitoring". In: *IEEE Transactions on Intelligent Transportation Systems* 19.7 (2018), pp. 2246–2257 (cited on p. 45).

[31] J. C. Bezdek, R. Ehrlich, and W. Full. "FCM: The fuzzy c-means clustering algorithm". In: *Computers & geosciences* 10.2-3 (1984), pp. 191–203 (cited on p. 46).

[32] I. Timón, J. Soto, H. Pérez-Sánchez, and J. M. Cecilia. "Parallel implementation of fuzzy minimals clustering algorithm". In: *Expert Systems with Applications* 48 (2016), pp. 35–41 (cited on p. 46).

[33] J. M. Cebrian, B. Imbernón, J. Soto, J. M. Garcıa, and J. M. Cecilia. "High-throughput fuzzy clustering on heterogeneous architectures". In: *Future Generation Computer Systems* 106 (2020), pp. 401–411 (cited on p. 46).

[34] C. Kyrkou. "AIDER (Aerial Image Dataset for Emergency Response Applications)". In: (June 2020). DOI: 10.5281/zenodo.3888300 (cited on p. 51).

[35] C. Kyrkou and T. Theocharides. "Deep-Learning-Based Aerial Image Classification for Emergency Response Applications Using Unmanned Aerial Vehicles." In: (2019), pp. 517–525 (cited on p. 51).

[36] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: (2018), pp. 4510–4520 (cited on p. 54).

[37] X. Xia, C. Xu, and B. Nan. "Inception-v3 for flower classification". In: (2017), pp. 783–787 (cited on p. 54).

[38] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: (2016), pp. 770–778 (cited on p. 54).

[39] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014) (cited on p. 54).

Chapter 5

Flood detection using real-time image segmentation from unmanned aerial vehicles on Edge-Computing platform.

Hernández, D., Cecilia, J. M., Cano, J. C., Calafate, C. T. (2022). Flood Detection Using Real-Time Image Segmentation from Unmanned Aerial Vehicles on Edge- Computing Platform. Remote Sensing, 14(1), 223. doi: 10.3390/rs14010223.

5.1 Abstract

With the proliferation of Unmanned Aerial Vehicles (UAVs) in different contexts and application areas, efforts are being made to endow these devices with enough intelligence so as to allow them to perform complex tasks with full autonomy. In particular, covering scenarios such as disaster areas may become particularly difficult due to infrastructure shortage in

some areas, often impeding a cloud-based analysis of the data in near real time. Enabling AI techniques at the edge is therefore fundamental so that UAVs themselves can both capture and process information to gain an understanding of their context, and determine the appropriate course of action in an independent manner. Towards this goal, in this paper we take determined steps towards UAV autonomy in a disaster scenario such as a flood. In particular, we use a dataset of UAV images relative to different floods taking place in Spain, and then use an AI-based approach that relies on three widely used Deep Neural Networks (DNNs) for semantic segmentation of images, to automatically determine the regions more affected by rains (flooded areas). The targeted algorithms are optimized for GPU-based edge computing platforms, so that the classification can be done on the UAVs themselves, and only the algorithm output is uploaded to the cloud for a real-time tracking of the flooded areas. This way, we are able to reduce dependency on infrastructure, and to reduce network resource consumption, making the overall process greener and more robust to connection disruptions. Experimental results using different types of hardware and different architectures show that it is feasible to perform advanced real-time processing of UAV images using sophisticated DNN-based solutions.

5.2 Introduction

Unmanned Aerial Vehicles (UAVs) are a type of aircraft that does not operate with a pilot on board and which, depending on their type, are either remotely piloted by a human, or operate autonomously with on-board computers [0]. From their invention, UAVs have been used for information gathering, surveillance, and agriculture in areas ranging from military, civilian or entertainment use [1]. As their usefulness was widely validated, these vehicles were made to face more complex tasks. Particularly, they have been extensively used for survivor detection and geolocalization in complex post-disaster environments [2].

Indeed, UAVs are particularly useful in the response phase of natural disasters. However, they require to be equipped with monitoring sensors such as ground sensors, light detection and ranging (LIDAR), cameras [3, 4], and/or hyperspectral cameras, enabling a detailed analysis of objects using unique information at different wavelengths (e.g. dryness of

leaves/soil, or health of trees) [5]. However, in such scenarios, immediacy in decision-making is crucial, which requires real-time processing of images to rapidly detect heat sources or flooding areas.

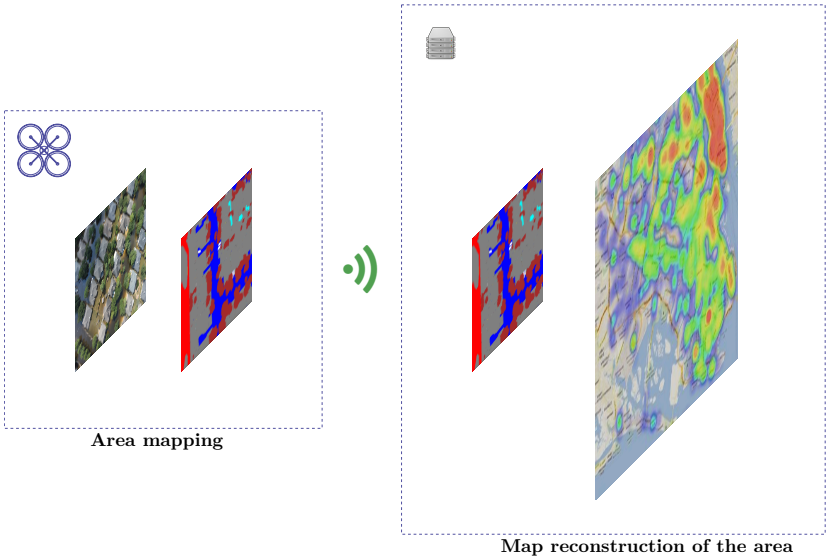


Figure 5.1: Overall view of our proposal.

Edge computing offers computational capabilities close to (or at) the source of data, i.e. sensors or mobile devices [6], opening up a new range of interactive and scalable applications [7]. We are witnessing a rapid evolution of specialized hardware (e.g., GPU, TPU) during the last decade, which enables more complex image processing workloads at the edge. Nevertheless, gathering real-time data from off-grid devices remains a challenge due to the network, energy and cost constraints. To optimize AI for the edge, compression methods for Neural Networks (NNs) have been proposed [8], together with distillation methods to learn a less complex network using an existing one [9]. Recent proposals experiment with the inclusion of accelerators in UAV devices. An example is [10], where a DNN-based real-time vehicle monitoring system is proposed to estimate the speed of vehicles using a UAV and the Xavier NX jetson device, being one of the first proposals for a monitoring system in drone platforms under real-world constraints.

Despite these advances, the real-time constraints and the addition of

AI workloads at the edge still pose a challenging problem due to the cost of these workloads [11]. Within the umbrella of natural disasters, there are Deep Learning (DL)-based image processing techniques for the detection of risks in such scenarios. For instance, Nijhawan et al., developed a Convolutional Neural Network (CNN) plus a feature extraction algorithm to classify images of natural disasters such as cyclones, avalanches, fires and tornadoes [12], using synthetic data developed by the authors. In the same way, gebrehiwot et al. proposed the use of a CNN for flood image classification using images obtained from a UAV where the classification was performed offline [13]. Another job listed is [14], where a genetic algorithm combined with a neural network was proposed to classify images from a flood; this approach was compared with three DL methods, being that the proposed algorithm was able to achieve better results. In all these works, techniques based on Deep Learning are the ones that achieve the best performance in image classification of natural disasters, but they consume a lot of computational resources, and therefore image classification is always performed offline in the cloud. In our previous work [15], we started to figure out what happens if we developed AI-pipelines at the edge for processing natural disaster images taken from drones. We showed that edge computing platforms with low-power GPUs were a compelling alternative for processing fuzzy clustering workloads, obtaining a performance loss of only 2.3x compared to its cloud counterpart version, running both the training and inference steps.

In this paper, we conducted a comprehensive evaluation of different Deep Learning modes for image segmentation on low-power GPU-based edge computing platforms. In particular, we focus on the identification of flooded areas after a natural disaster to find out whether these platforms can provide enough computing power to build autonomous drones that, in real time, assist emergency teams in assessing flooded areas for immediate decision making on the ground. This work is focused focusing on finding and obtaining all flooded areas from a series of images captured from a drone, in order to reconstruct the emergency situation on a map. Figure 5.1 shows an overview of the structure of the proposed solution. Major contributions of this paper are the following:

1. Different state of the art DNN-based Deep Learning models for image segmentation are analyzed for flooding detection in terms of

performance, accuracy and memory footprint. Particularly, PSPNet, DeepLabV3 and U-Net are under study.

2. Several encoders are also analyzed as they are the main bottleneck for the memory footprint, the disk weight and the execution time of the inference process. Particularly, we analyze ResNet152, EfficientNet and MobileNet are under study.

3. The Cartesian product of these neural networks models and encoders are trained to identify flooded areas from aerial images; i.e., 9 models in total. A semi-supervised training procedure with a pseudo-labelling strategy is designed to increase the accuracy in up to 4%.

4. An in-depth performance evaluation of different low-power GPU-based edge computing devices is provided to assess the feasibility of autonomous AI drones in detecting flooding areas.

5. An evaluation in terms of memory, performance and quality is provided to analyse which of the models and enconders for flood image segmentation is best suited to be run on drones to provide real-time feedback to decision-makers.

Section 5.3 introduces the general infrastructure the main Deep Learning models targeted and methods used to deal with aerial images of flooded areas. Section 5.4 shows the experimental setup before showing the performance and quality evaluation of our approach. A discussion of the results is then provided in Section 5.5. Finally, Section 5.6 shows conclusions and directions for future work.

5.3 Materials and Methods

This section introduces different neural network models that are widely used in detecting flooded areas. These models will be trained in order to obtain the segmentation mask of each image from the drone. The data used and the training procedures carried out are also summarized prior to the evaluation on the drone to be performed in the following section.

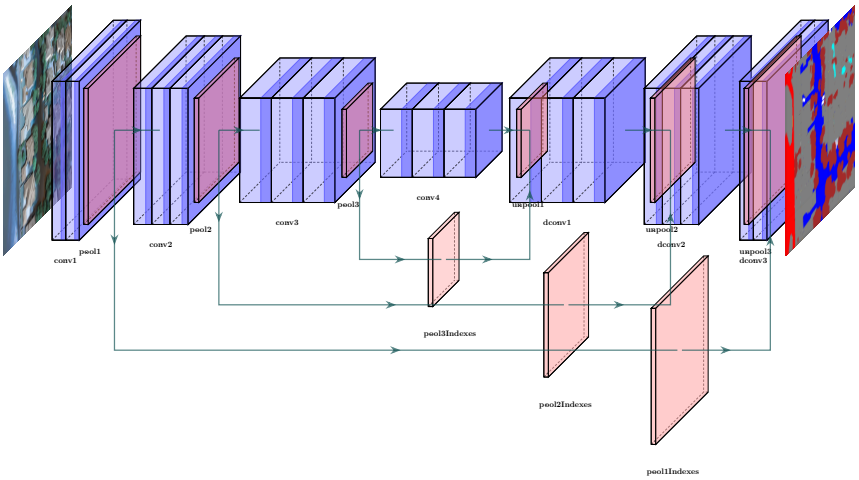


Figure 5.2: Simple deep convolutional autoencoder where you can see the two main parts of a network dedicated to image segmentation.

5.3.1 Hardware environment

The UAV device that supports the segmentation hardware consists of the following components:

- DJI F550 frame
- Pixhawk flight controller
- Processing unit
- Marshall Electronics CV503-WP Mini Full HD Camera.
- battery Tattu 9000 mAh 22.2 V 25 C 6S1P Lipo.
- 3DR GPS receiver
- Sunnysky X2212 motors - 980kv.
- APM 433 MHz telemetry.
- Taranis X9D controller + X8R receiver

Table 5.1: Hardware details of the hardware used in our experiments.

	Pedra	**Jetson AGX Xavier**	**Jetson TX2**	**Jetson Nano**
CPU	Intel Silver 4216	NVIDIA Carmel ARM v8.2	ARMv8	ARM Cortex-A57 MPcore
2xGPU (NVIDIA)	GeForce RTX 2080 Ti	Volta	Pascal	Maxwell
Memory [Gb]	375 DDR4	32 LPDDR4x	8 LPDDR4	4 LPDDR4
Size [mm]	73.4 x 8.7 x 44.8	105 x 105	50 x 87	70 x 45
Weight [g]	17,000	280	85	61
Energy consumption [W]	80-100	10-30	7.5	3-5

Table 5.1) shows the hardware platforms used to perform the experiments below. Particularly, we use a High Performance Computing (HPC) computer, called Pedra, as a baseline for our experiments. Then, three GPU-based edge computing devices from the NVIDIA Jetson family (i.e. Jetson nano, Jetson TX2, and Jetson AGX Xavier) are also targeted. Moreover, the software environment is based on gcc v7.4.0, CUDA v10.2 with cuDNN and Python v3.6 with pytorch v1.8.0 built for edge devices, torchvision v0.9.0 built for edge devices and scikit-learn v0.24.1.

5.3.2 Dataset

FloodNet[16] is a set of images captured in the aftermath of Hurricane Harvey witch made a landfall near Texas and Louisiana on August, 2017, as a Category 4 hurricane. In it, special emphasis is placed on areas that have been flooded. In particular, it contains UAV images captured during the response phase by emergency personnel. The authors of the dataset obtained the images using small DJI Mavic Pro quadcopters. The images were captured at a height of 60 meters with a spatial resolution of 1.5 cm. Currently, it is the only dataset created with images of floods captured from a drone after a catastrophic event, and at an optimal level of detail to train a neural network that must be operational within a UAV.

Floodnet images are labeled at the pixel level, which makes them useful for the semantic segmentation task. FloodNet attempts to do fine labeling that encompasses situations such as detecting flooded roads and buildings, and distinguishing between natural water and flooded water. Although the dataset approach is broader than image segmentation, in this particular case we will focus on it, as the training dataset consists of 51 labeled images of flooded areas and 347 labeled images of non-flooded areas and 1047 unlabeled images on which pseudolabeling tasks are to be performed labeled images includes the following instances: Building flooded (3248 instances), Building Non flooded (3427 instances), Road

flooded (495 instances), Road Non Flooded (2155 instances), Water (1374 instances), Tree (19682 instances), Vehicle (4535 instances), Pool (1141 instances), and Grass (19682 instances).

5.3.3 Semantic segmentation for flooding detection

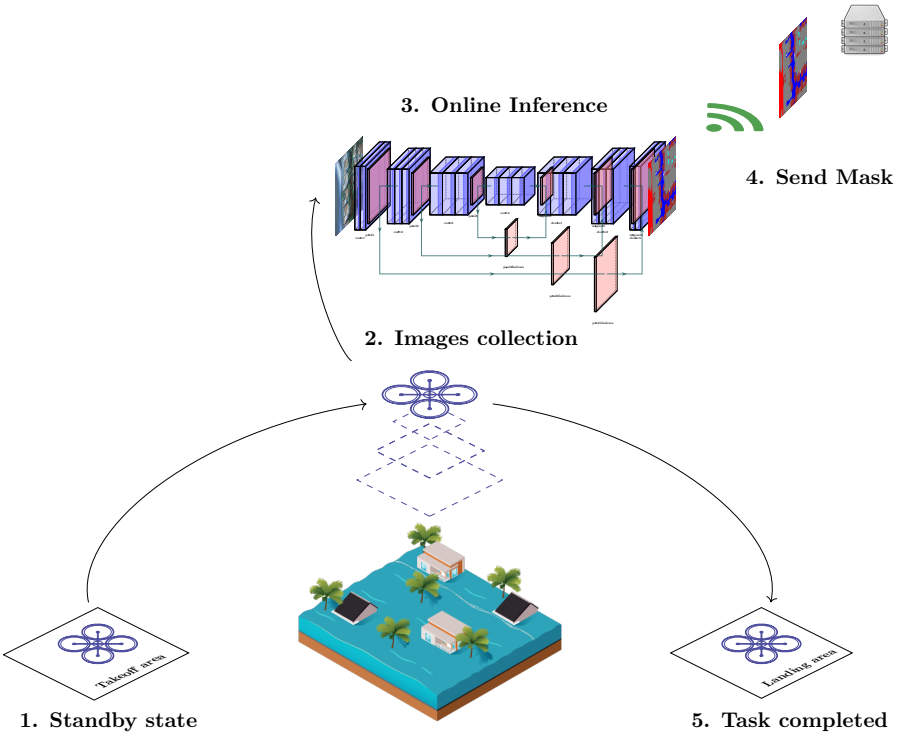


Figure 5.3: Overview of our image collection and processing solution.

In this study the experimentation is focused on obtaining the data from the drone for sending. For this task, image segmentation has been considered the technology best suited to the requirements of detecting flooded areas. Semantic segmentation is the technique of classifying parts of an image that belong to the same object class. It is a form of pixel-level prediction where pixels in an image are classified according to a category. An example of this type of network can be seen in Figure 5.2, which

shows a simple deep convolutional autoencoder where you can see the two main parts of a network dedicated to image segmentation, on the left the one that tries to capture the visual patterns of the image and on the right the one that tries to reconstruct a representation of the detected classes. Notice that image classification merely has to identify what is present in the image, whereas semantic segmentation identifies not only what is present in the image, but also where to find it by signalling all pixels that belong to each characteristic identified. Indeed, since the emergence of Deep Neural Network (DNN), segmentation has made a tremendous progress. We refer the reader to [17, 18, 19] to see a full review of Deep Learning techniques for image semantic segmentation. Among these techniques, we may highlight the following:

- **PSPNet[20]** is a model that uses a pyramid parsing module to exploit the global context information through region-based context aggregation. The combined local and global cues make the final prediction more reliable. Given an input image, PSPNet uses a pre-trained CNN with the dilated network strategy to extract the feature map. The final size of the feature map is the same as that of the input image. On top of the map, it uses the pyramid clustering module to gather context information. Using 4-level pyramid, the clustering kernels cover the whole, half and small portions of the image. They are merged as the global priority and then concatenate the prior with the original feature map in the final part of the decoder. Next, a convolution layer generates the final prediction map.

- **DeepLabV3[21]** is a semantic segmentation architecture that builds on DeepLabv2 [22] with the following modifications: in order to deal with the problem of multi-scale object segmentation, modules have been designed that use cascaded or parallel Atrous convolution to capture the multiscale context by adopting multiple Atrous rates. Another modification over the previous version is that the Atrous Spatial Pyramid Pooling module [23] is extended with image-level features that encode the global context, further increasing performance. The last change is that DenseCRF [24] post-processing has been dropped.

- **U-Net[25]** uses an encoder-decoder architecture based on a contraction route to capture the context, and a symmetric expansion route to achieve accurate location. It was originally designed for segmenting medical images, and it achieves robust results with a more uniform training set. In recent years, studies have shown that U-Net is also suitable for remote sensing images [26], and has great potential for improvement.

All the segmentation models mentioned make use of a backbone that is capable of extracting the fine-grained patterns of the image in the form of an encoder of the information. A large part of the trainable parameters of the network come from these blocks, and depending on the type of network chosen for this purpose, the memory footprint, the disk weight and the execution time of the inference process will be different. For this study, three backbones of different characteristics have been chosen in order to find the best option between inference time and accuracy obtained. These are described below:

- **ResNet152** or Residual Networks are a variant of convolutional neural network that was first introduced by Kaiming et al[27]. Their main feature is that it learns the representation functions of the residuals instead of learning the signal representation directly. ResNet introduces the hop connection (or direct access connection) to adjust the input from the previous layer to the next one without any modification of the input thus allowing a deeper network that is easy to optimize and can gain in accuracy by greatly increasing the depth. In this work, the 152-layer version has been chosen to perform the encoder task because it has obtained better accuracy while maintaining better complexity than other networks such as Visual Geometry Group (VGG), as the authors explained in their work.

- **EfficientNet[28]** is a type of DNN whose main feature is that it scales uniformly all dimensions of depth, width and resolution with a set of preset scaling coefficients. It is based on the intuitive concept that the larger the size of the input image the greater the number of layers and the greater the number of channels the network will need to capture the patterns in the image, this concept has been studied in works such as [29].

- **MobileNet**[30] is a convolutional neural network that has been specifically designed to be executed in embedded devices that require solving computer vision tasks. Its architecture is formed by depthwise separable convolutions with the objective of obtaining the lightest possible network and that allows a low latency when executed in devices that have few resources and that do not usually have a graphic accelerator. The complete MobilNetV2 model, which has been used in this work, is formed by the initial convolutional layer of 32 filters and is followed by 19 residual bottleneck layers. The complete model consists of 2 million parameters.

These encoders (i.e., ResNet152, EfficientNet, MobileNet) are trained to embed the information patterns from the aerial images into an input that can be processed by the network in charge of performing the semantic segmentation (i.e., PSPNet, DeepLabV3, U-Net). However, there are certain scenarios, such as the one we are dealing with in this study, where the training dataset for the encoders is very small, i.e. there are a small number of labelled images and therefore it is difficult to achieve sufficient accuracy to generate an accurate mask of the captured image to serve as input for the image segmentation procedure. Several works proposed the use of transfer learning techniques [31] to take advantage of the knowledge acquired when solving one problem and apply it to a different but related problem. In transfer learning, neural networks are initialized with weights from a neural network that is already trained (a.k.a, pretrained) using a large, structured and labelled dataset. This procedure has shown better performance than those trained from scratch on a small dataset [32].

In this work, we use the ImageNet database for the pre-training of the encoders. ImageNet [33] is a large visual database designed for visual object recognition. It consists of over 14 million images that have been hand annotated to indicate which objects appear in them and, in at least one million of these images, bounding boxes are also provided. ImageNet has more than 20,000 categories, each consisting of several hundred images. Imagenet has proven to be optimal for transfer learning exercises as shown in [34], where an empirical investigation is provided that studies the importance of the number of images, and the balance between the images of each class and the number of classes.

5.3.4 Image Preprocessing

1. **Resize**: Rescale an image so that the maximum side is equal to *maxsize param*, maintaining the aspect ratio of the initial image.

2. **ShiftScaleRotate**: Randomly apply affine transforms: translate, scale and rotate the input.

3. **RGBShift**: Randomly shift values for each channel of the input RGB image. In **[RGBinproceedings]** authors show that images with different wavelengths and RGB channels determine which kind of images with different color spectrum provide better information to generate a better accuracy of our DNN model.

4. **RandomBrightnessContrast**: Randomly change brightness and contrast of the input image to reduce a model's sensitivity to color.

5. **Normalize**: Normalization is achieved through formula:

$$img = \frac{(img - mean \times maxPixelValue)}{(std \times maxPixelValue)}$$

This transformation sequence is applied to each of the images after being captured, and before performing the process of obtaining the mask through the neural network. It is an operation that will be performed on each of the edge devices.

5.3.5 Inference precision

This section shows the accuracy of the combination models/encoders on the test data set. First, an evaluation of all of these combination is provided by training with the dataset described in section 5.3.2. Then, it is analysed how the accuracy of the models improves with the semi-supervised training proposed in section 5.3.6.

The mean Intersection-Over-Union (MIoU) metric has been used to measure accuracy. This metric is based on the Jaccard index, which is defined as the ratio of intersection and union area between the predicted segmentation map and the ground truth defined. It was developed by Paul Jaccard, and it is also known as the Jaccard similarity coefficient; specifically, it is a statistic used for gauging the similarity and diversity of sample sets defined as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

where B shows the predicted segmentation maps, and A represents ground truth.

Based on it, we can calculate Intersection Over Union (IoU), which is a number from 0 to 1 that specifies the amount of overlapping between the predicted and ground truth:

$$IoU = J(A,B) = \frac{tp}{\sum_{i=1}^{n} fn + \sum_{i=1}^{n} fp - fn}$$

MIoU is defined as the average value of IoU over all label classes. It is generally used to report the performance of segmentation models. It usually ranges between 0 and 1 given as:

$$MIoU = \frac{1}{k+1} \sum_{i=0}^{k} \frac{tp}{\sum_{j=0}^{k} fn + \sum_{j=0}^{k} fp - fn}$$

where k represents total classes, tp is number of true positives, fp and fn are false positive and false negatives.

5.3.6 Semi-supervised training procedure

As shown in section 5.3.2, the number of labeled instances is very small, consisting of only 398 images. With regard to the number of test images, 47, this value is considered low; therefore, to achieve a correct performance of the network, it is necessary to apply semi-supervised learning techniques on the 1047 training images that have not been previously labeled, therefore creating a dataset of 1445 images. Semi-supervised learning is a training method that mixes a small amount of labeled data with a large amount of unlabeled data in the training phase. Semi-supervised learning stands between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data). Unlabeled data is only useful if it provides information for label prediction that is either not present in the labeled data, or cannot be obtained from the labeled data alone. The procedure consists of the following steps:

1. Train the model using training labeled data.

2. Infer labels for an unlabeld data.

3. Add confident predicted test observations to our training data.

4. Train the model for a few extra epochs using also the new labeled data.

5.3.7 Solution deployment

Once the problem has been defined, and the image segmentation proposal enabling a subsequent reconstruction on the server has been proposed as a solution, we now proceed to describe the execution flow that would be carried out in a real natural disaster management scenario. Figure 5.3 shows the life cycle of an image collection and processing mission at the edge. The UAVs start from a standby position (step 1) to prepare for takeoff before starting to gather images from the affected area. While the images are being collected (step 2), the Deep Learning models (step 3) are running to perform a real-time inference of each of the images from the UAV device, resulting in image processing at the edge. In step 4 images are sent to the server. Steps 2-4 are repeated until all the area to be mapped has been imaged, and all the masks are generated. It is noteworthy that, in step 4, only masks are sent to the cloud, which means using only 4 bits per pixel, instead of the 24 bits per pixel that would be sent in case of using a standard bitmap having 24 bits. Notice that 4 bits per pixel are adopted because they conform the minimum value required to represent the 10 classes that the system is able to detect, and therefore each pixel of the image can be represented merely by a value between 0 and 9.. This significantly reduces the overall bandwidth requirements by a factor of 8, as shown in Section 5.4.3.1.

5.4 Results

This section starts by summarizing main hardware features of the hardware environment targeted for assessing the Deep Learning models for image segmentation previously presented. Then, it provides an overview of the dataset used to train and test those models before a performance assessment in terms of quality, execution time and memory footprint is provided.

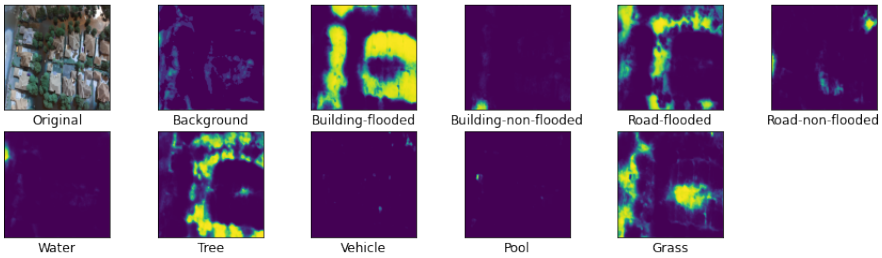
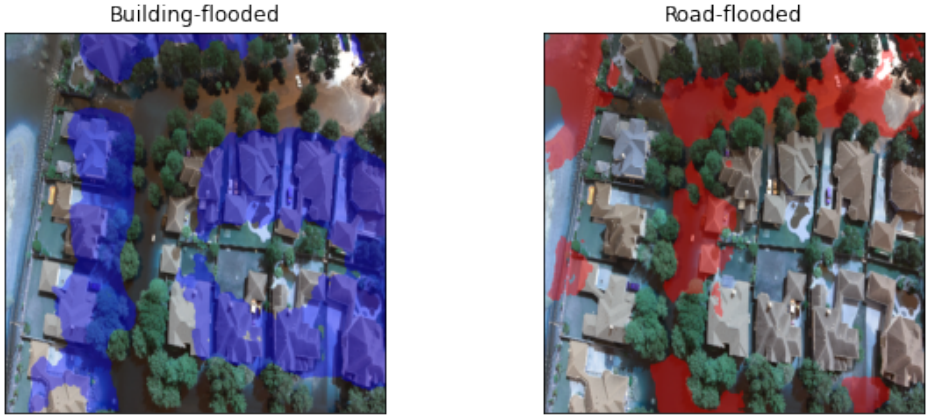


Figure 5.4: Segmentation mapping of each of the classes on the original photo.



(a) Blue mask over flooded buildings

(b) Red mask over flooded roads

Figure 5.5: Segmentation mapping of each of the classes on the original photo.

This section evaluates the performance (i.e. execution time) and precision of the Deep Learning architectures previously presented in Section 5.3.

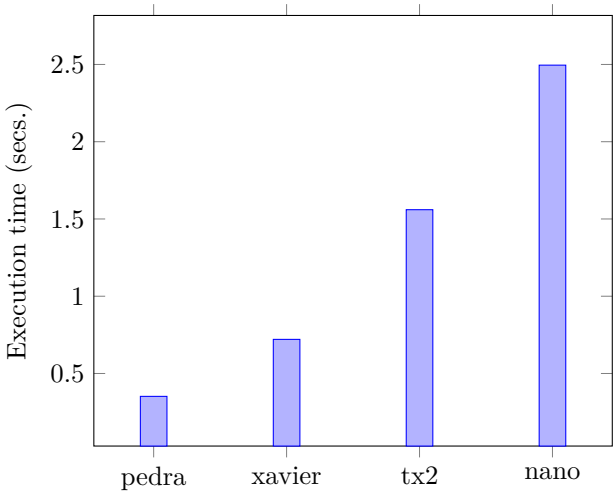


Figure 5.6: Image preprocessing execution time.

5.4.1 Image Preprocessing results

Figure 5.6 shows the difference in execution time between the different devices. This execution time is added to the mask acquisition time to define the frame rate that can be sent to the server. The HPC server will also be included in each of the time figures in order to have a reference of the inference time that would be required with a non-edge device. The figure shows that the execution times range from 0.35 seconds on the server hosted in the data center, to 2.49 seconds on the Jeson Nano device; i.e. a performance loss of up to 7.11X on the edge platform. The difference between these two devices is notorious, and for image processing during the mission, where many images have to be captured within a short time span, the compartmentalization of this step with the neural network processing will have to be taken into account, being able to execute the processing of the current image on the CPU, while obtaining of the mask of the previous image through the neural network using the GPU.

5.4.2 Inference precision results

This section shows the MIoU obtained for each model with each of the chosen encoders. Figure 5.4 shows the mask generated for each of the

10 classes of the dataset on one of the images of the dataset. For the requirements of this work only the classes that refer to floods are necessary. Two of these masks are shown in figure 5.5: flooded buildings and flooded roads. Nevertheless, in the training of the network, the IoU of each of the classes has been taken into account.

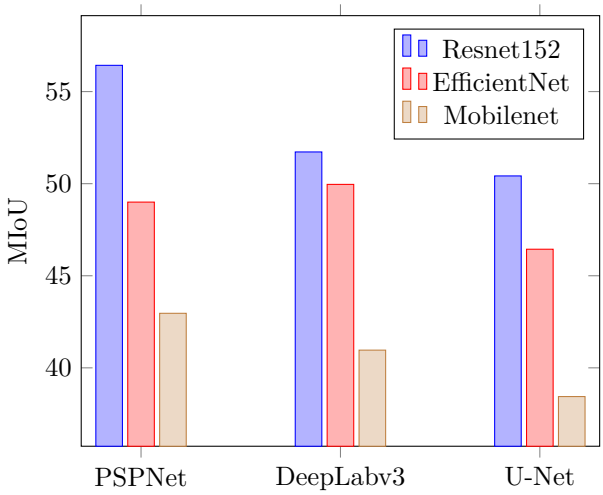


Figure 5.7: Accuracy (MIoU) of three neural networks and encoders used, having been trained with the baseline dataset.

Figure 5.7 shows the model efficiency based on the MIoU achieved for each combination of neural network model and encoder. For PSPNet training, a constant learning rate of 0.001 for 15 epochs was set; for DeepLabV3, the learning rate was of 0.01 for 10 epochs, while for U-Net it was trained for 15 epochs with a learning rate of 0.001. For the batch size during training, a value of 4 has been set for all models when using the RestNet152 encoder, a batch size of 7 for EfficientNet encoders, and 8 for MobileNet encoders. It can be seen that the RestNet152 encoder achieves the best result, with the PSPNet Network having the most significant difference, which increases by more than 5% with respect to EfficientNet. In the DeepLabv3 network, the difference between RestNet and EfficientNet is of 2%, which is not so significant. The MobileNet encoder shows the worst results of all, with a difference of up to 13% compared to the best case.

Figure 5.8 shows the accuracy of the trained models when they are

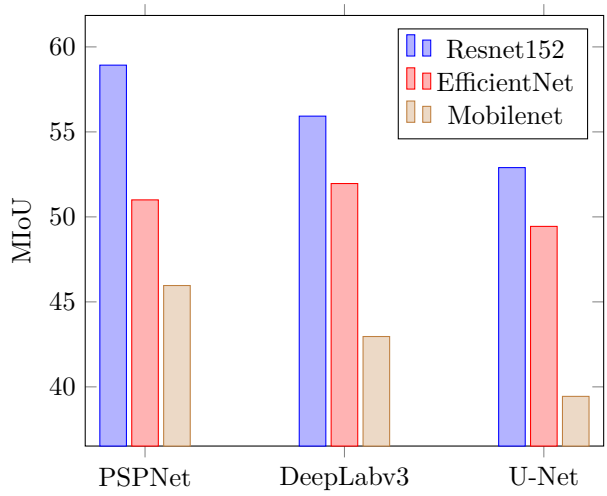


Figure 5.8: Accuracy (MIoU) of three neural networks and encoders used, having been trained with the pseudolabeling technique described in Section 5.3.6.

trained using the pseudolabeling technique described in Section 5.3.6. For this process, the same parameters of learning rate, number of epochs and batch size have been used than in the previous results shown in Figure 5.7. Figure 5.8 shows a slight increase in the MIoU of all models: 2% for PSPNet with the ResNet152 encoder, 3% for DeepLabv3; for EfficientNet, the biggest difference is found in the U-Net network with an increase of 4%. The MobileNet encoder obtains almost identical values with U-Net, with a great improvement when it comes to PSPNet. It can be seen that the network that has benefited the most from this type of training is PSPNet. This has to be taken into account when choosing which model to use for the deployment of the solution, since it is possible to add previously unlabeled data to generate soft-labels with them, and thus adapt the model to changes that may occur in the area to be mapped.

It is not possible to consider the supremacy of a model over the rest for all the use cases in which image segmentation can be applied, since with another dataset or another task the result may be that DeepLabv3 or U-Net behave better than PSPNet, for this reason it would be necessary to redo the experimentation and a reparametrization of the leraning rate, optimizer and other factors that may affect the performance of the

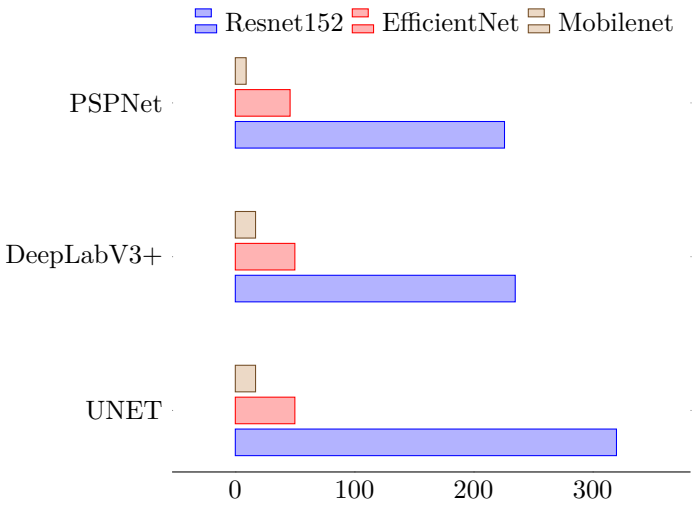


Figure 5.9: Size (in MB) of the output of every model, with every encoder.

network.

5.4.3 Model comparative

This section shows a comparison of the three models chosen with their corresponding encoders in terms of size, performance and inference time for each of the edge computing devices to be embedded in a UAV. The size of a model is represented by the number of trainable parameters of the model. After the training process, these models are usually serialized in a certain standard format to be distributed to different applications, and for different devices. The size of the exported model on disk will be smaller than the size it will occupy in memory and, therefore, these two variables must be taken into account in the study of the space footprint of a model. The size of each encoder used must also be taken into account, and a choice then made based on the size and accuracy of the encoder.

Figure 5.9 shows a comparison of the disk size (in MB) of each serialized model, and for each of the encoders used in the training phase (i.e., ResNet152, EfficientNet and MobileNet). It can be seen that, regardless of the network used, ResNet152 has a much higher weight than the rest, being the difference between DeepLabV3 and PSPNet negligible,

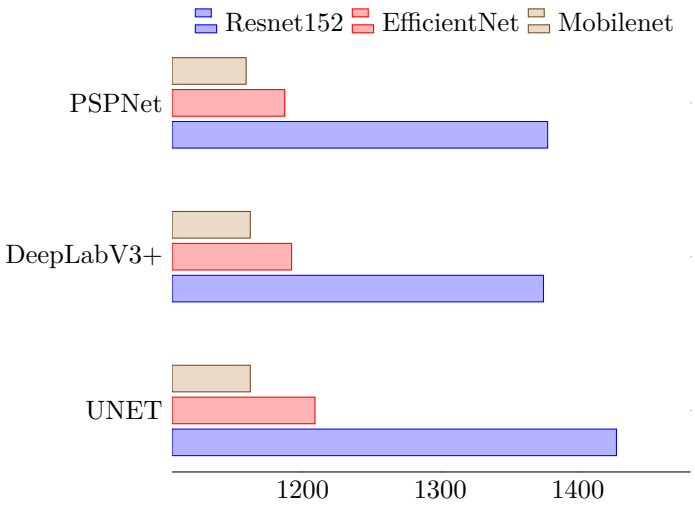


Figure 5.10: Memory footprint (in MB) for every model, and with every encoder.

of 235MB for DeepLabV3, and of 226MB for PSPNet. For EfficientNet, both Deeplab3 and PSPNet occupy a space of 50MB. UNet has a slightly lower value of 46MB. The MobileNet encoder, together with PSPNet, has the smallest footprint of all combinations, with a value of 9.1MB, followed by equal values of 17MB for the two other networks.

Figure 5.10 shows a comparison (in MB) of the footprint of each model when deserialized in memory. As in figure 5.9, the values of each of the encoders with each of the models are shown. The increase in space required with respect to the serialized model is evident, tripling in all models. These results will have to be taken into account when deploying each of the models on edge devices, especially in the TX2 and Jetson Nano platforms where the GPU memory capacity is very limited.

The inference time, together with the image pre-processing time, is one of the values to be taken into account when deploying the trained model for the inference process. Undoubtedly, a compromise has to be found between the accuracy of the models and the time it will take to obtain the result; i.e. the mask of the image captured by the drone. An very elongate inference time to generate the mask would prevent the real-time processing we are looking for to help first-reponders. Therefore,

the ideal scenario would be to process the image and send its mask before the next image is captured. For this reason, a performance evaluation for each model+enconder targeted on each edge computing platform that can be embedded in a UAV has been carried out. CPU and GPU processing times have been measured to study in which cases it is necessary to install the latter, and in which cases a single CPU would be enough, thus saving both weight and energy, which would reduce both the weight and the power consumption of the UAV. The HPC server will also be included in each of the time figures in order to have a reference of the inference time that would be required with a non-edge device.

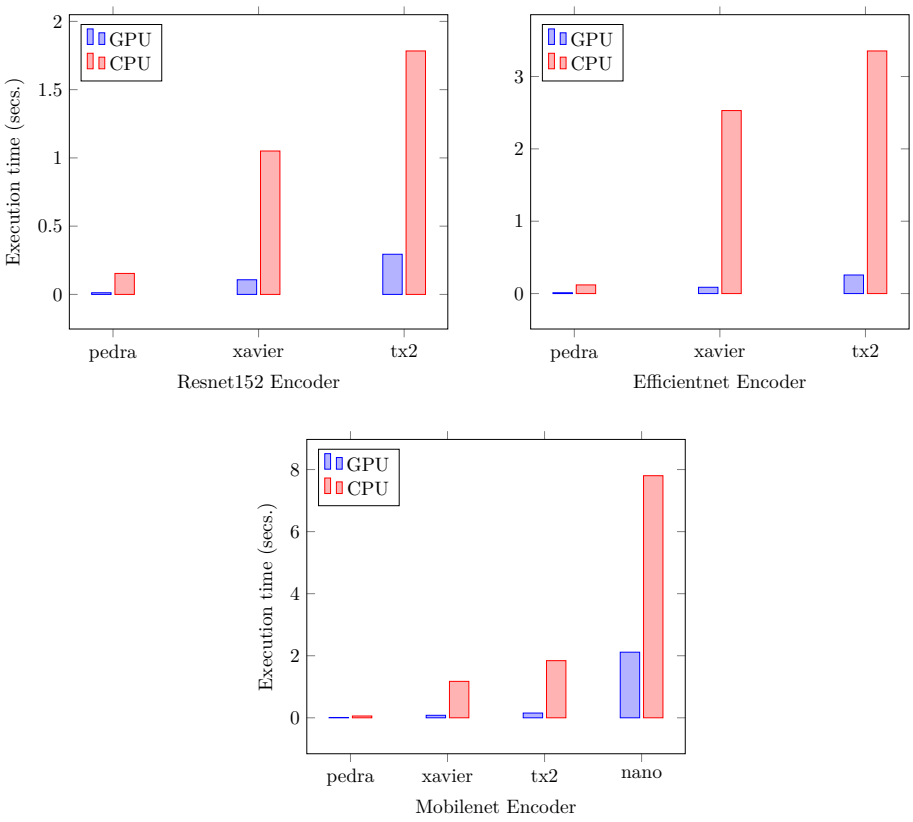


Figure 5.11: PSPNet architecture inference time comparative with all encoders.

Figure 5.11 shows three bar charts, all corresponding to the PSPNet

network with each of the given encoders. For each of the bar charts, the execution time on GPU and CPU is shown. The ResNet encoder could only be executed on the Xavier and TX2 devices, the latter doubling the Xavier platform in both GPU and CPU time. The Jetson Nano device could not finish the inference process, and therefore some simplifications would have to be done on it in order to run it. PSPNet, together with EfficientNet, could not be run on the Jetson Nano platform neither, and both the AGX Xavier and TX2 show similar inference times. PSPNet-MobileNet could be executed on the Jetson Nano device, this being the only encoder out of the three with which we have experimented that could be hosted without problems, although the execution time is several orders of magnitude higher than that of TX2 and Xavier which, as with EfficientNet, have very similar times, to the detriment of TX2.

Figure 5.12 follows the same structure as Figure 5.11 but showing the UNET execution times with all encoders. This network is the one that shows the most limitations in terms of execution, where only the MobileNet encoder has been able to be executed in all the devices, and where the execution times are higher than in the other two models. In the first graph, with ResNet, the execution in the Jetson Nano platform has not been possible, and in TX2 only when using the GPU, and with a time overhead close to the second. In the EfficientNet encoder, it has only been possible to perform the execution in the Xavier platform, with a remarkable CPU time of 14 seconds per image. MobileNet was executed on all devices using GPU, while on the Jetson Nano device it could not be executed on CPU.

Figure 5.13 follows the same structure as in Figure 5.11 but showing the DeeplabV3 execution times with all encoders. The big difference is that this model could not be run on the Jetson Nano device with any encoder. In ResNet152 it can be seen how, in the TX2 platform, the CPU the time is very high with respect to Xavier, achieving a GPU time of less than 1 second. In EfficientNet, the execution times are significantly higher than in ResNet152, and it has not been possible to perform the execution of the TX2 platform in CPU due to lack of memory. Using MobileNet, the fastest execution times are achieved within the TX2 device, but not for Xavier.

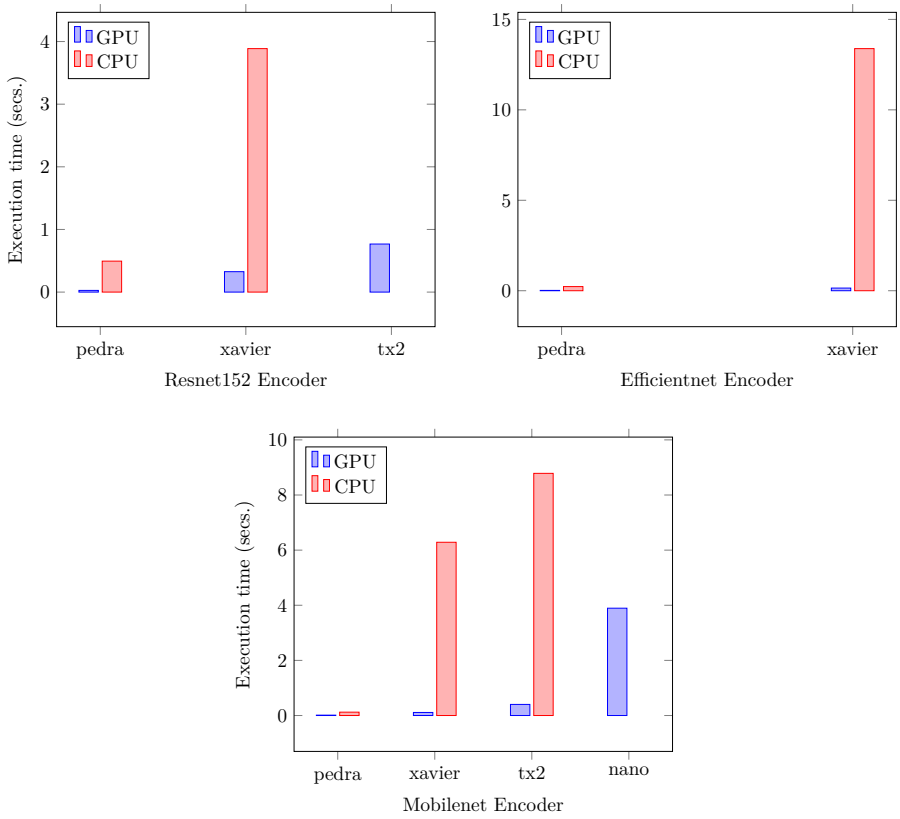


Figure 5.12: Unet architecture inference time comparative with all encoders.

5.4.3.1 Data compression

Following the image classification procedure described before, the next step consists of sending an alert to the control station located in the cloud with the frame, and the meta-information that details the coordinates marked as positive for the emergency team to evaluate them. We show a comparison between sending the raw information and the mask resulting from the network processing. Notice that by just sending the mask we are able to save bandwidth.

Figure 5.14 shows the information compression achieved by sending the mask to the server from the UAV. If the complete image was used, the values represented over three channels would be sent with values

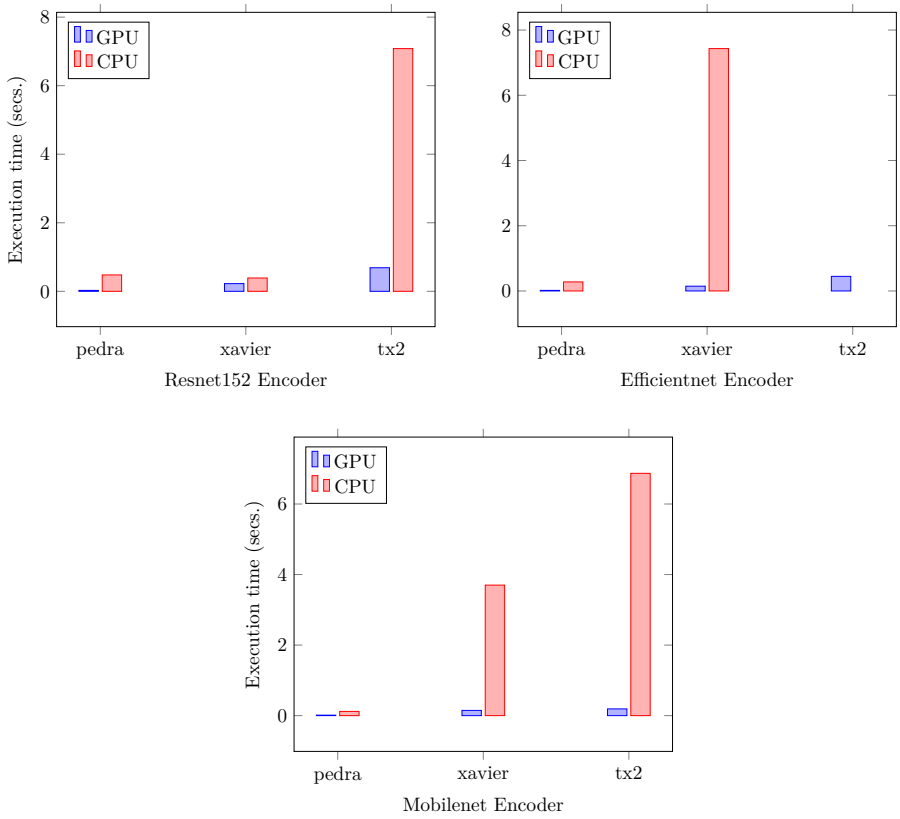
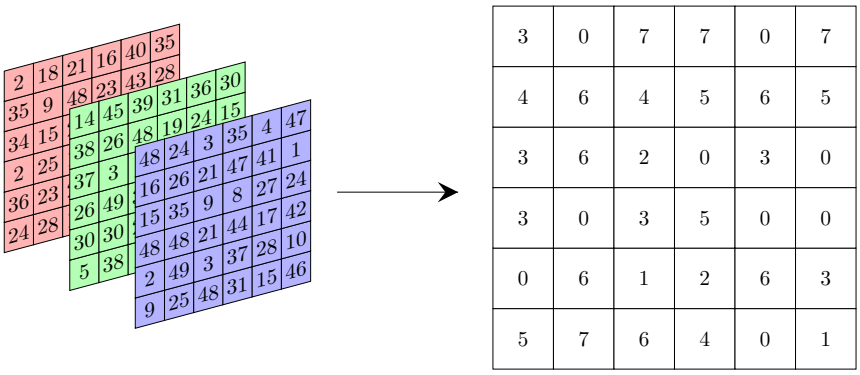


Figure 5.13: DeeplabV3 architecture inference time comparative with all encoders.

ranging from 0-255. However, to store the segmentation information found according to the target classes, only one channel and one value for each of the classes is needed. In this case, the dataset has a maximum of 10 classes, and so only 4 bits would be needed for each pixel represented, in contrast to the 8 bits per pixel required when sending the original image.

This compression represents a saving in the bandwidth to be transmitted, which would affect both the consumption and the speed of sending information to the server. In particular, the original image sized 4000×3000, which occupies 5.488MB, would in turn generate a mask of 910KB, which would represent a saving of 83.42% of the information

to be transmitted. This would not be enough if we were looking for a detailed representation of the information. Nevertheless, in this work we are merely aiming at sending of enough information to be able to reconstruct the scene on which the tracking operation is being performed, this being the information collected by the semantic classification of the entities of the captured image.



3	0	7	7	0	7
4	6	4	5	6	5
3	6	2	0	3	0
3	0	3	5	0	0
0	6	1	2	6	3
5	7	6	4	0	1

Figure 5.14: Compression comparison between an RGB image and its segmentation mask.

5.5 Discussion

In the experimentation carried out in the previous section, we have extracted the data to be taken into account when implementing an area monitoring system using semantic segmentation perform from a resource-constrained device such as a drone. In this section, and based on the results obtained, we discuss the most solid proposal to be deployed.

This study has used an input-ready dataset, so far the best suited to the task at hand, using semi-supervised learning techniques, which is an obvious disadvantage compared to a rich dataset with each instance correctly labeled in all photos; this reveals that more work needs to be done on collecting and labeling flooding datasets deriving from UAVs. A positive observation regarding the results obtained using such a small dataset is that, using semi-supervised learning techniques, it would be possible to deploy the solution in areas where the geography and architecture are different from the training dataset, and a process of image

collection and labeling is required. By obtaining a small sample, and with a training process of less than 10 epochs, a functional model to run on the UAV would be available.

In terms of accuracy based on the Floodnet validation dataset, we can estimate that the PSPNet network, together with a Resnet152 encoder pre-trained on ImageNet, and further trained in a semi-supervised way using soft labels, has been the combination that will allow a more accurate server reconstruction of the mapped area, achieving 56% of MIoU. This is a valid result to solve the problem of obtaining a mask of flooded areas on roads, buildings and places near river or sea beds, as it would allow differentiating natural water masses from those that have flooded areas, and where water is not expected. Figures 5.4 and 5.5 show an example of the pixel-level segmentation of each of the classes detected. This information is the main component in the server reconstruction of the monitored area. The main issue of using this proposal is its memory size limitation, since it is the second largest model in terms of memory footprint, and hence cannot run on a Jetson Nano device, which is the most suitable device to be installed on a UAV due to its weight and dimensions. Another advantage of selecting these parameters is their speed, as they would be among the fastest of all those studied; if we wanted to use them, we would obtain a speed of less than half a second on a TX2 device using the GPU. On the opposite side, UNet has been the most problematic model in terms of execution and adaptation to the different edge devices, and some of its parameterizations, for example using Efficientnet, do not allow its execution neither in TX2 nor in Nano, the two lightest devices, and therefore more susceptible to be assembled in a UAV.

If the weight and power limitations are sufficient to install a Xavier device, PSP-Resnet152 would be the best option. Otherwise, and if we were looking for the most efficient solution in these two parameters, we would have to use a Jetson Nano, and our software combination would be limited to the PSPNet-MobileNet and Unet-MobileNet models (GPU restricted), with an MIoU of 45.9 and 39.4, respectively, and a GPU execution time of 2 seconds and 8 seconds CPU in PSPNet, against 4 seconds in Unet GPU. Therefore, PSP-MobileNet would be the best choice for the Jetson Nano platform. In case a weight margin could be obtained in the processor, and the weight of the UAV is no excessive, it would be convenient to install a TX2 device, since for 24 grams and a

little more power consumption, the best combination would be achieved, which would mean an improvement of 13% in accuracy, and a reduction of execution time of 1.7 seconds if the GPU is used, and of 6 seconds if CPU power is used instead. Therefore, a good configuration taking offering an adequate tradeoff between accuracy, execution time, weight and power of the drone could be the TX2 platform as an inference hardware device where a PSPNet network would be deployed with a ResNet152 encoder pre-trained with ImageNet.

The frame sending interval (process that starts when a new image is obtained, and that ends with the sending of the mask obtained to the server, without taking into account the network speed and latency of the same), and that could be achieved with the best solution, would be of 1.5 seconds of image preprocessing time, plus 0.29 seconds of GPU inference, so it would be 2 seconds using GPU (0.5 Hz) and 3.3 seconds in case of using only the CPU (0.3 Hz). In addition, as shown in the compression section, the bandwidth of sending the information from the drone will be significantly reduced when generating the mask on the edge, and that can save network resources and lower battery consumption.

5.6 Conclusions

UAVs have the potential to play a "key role" at mitigating the consequences of climate change. However, both hardware and software solutions are needed to make these devices truly crucial players in these tasks. AI and edge computing are undoubtedly a winning combination by enabling to transform autonomous drones into useful tools in various emergency situations. In this work, an AI-based pipeline has been proposed for execution on edge computing platforms to enable efficient processing of natural disaster images captured by UAVs.

Our results reveal that the use of neural networks designed for real-time image segmentation from drones can be a viable solution as long as the drone is equipped with an edge computing device endowed with GPUs. The benefit of merely sending the result mask instead of the raw image, which is made possible by performing image processing at the same location where the image is captured, reduces the required network traffic by several orders of magnitude. It is worth mentioning that the computational load differences between edge and cloud platforms

remain large, with speedup factors in the range of 2.8x-22.17x. Yet, the development of efficient platforms for the execution of specific workloads, such as Deep Learning, shows a roadmap that enables the development of applications for relevant autonomous and intelligent systems, such as the one proposed here.

We are certain that autonomous UAV technology can be an important factor in the fight against climate change. This study has shown how with a small dataset correctly labeled and the right model, a real-time segmentation system can be embedded in a UAV, bringing the main computational work closer to the device in charge of processing the information to make an offline inference and send the digested data. However, there is still a lot of work to be done from several points of view. In terms of communication, extending the results of this article to a swarm of drones can provide a greater coverage of the area to be surveyed, something very necessary in this type of natural catastrophes. Additionally, it becomes necessary to increase the performance of IA models by providing new datasets, as well as studying in more depth the representation part of the semantic segmentation result obtained by the network on a map of the area where images have been captured, so as to provide greater insights.

References

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, et al. "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges". In: *Ieee Access* 7 (2019), pp. 48572–48634 (cited on p. 68).

[2] B. P. Rohman, M. B. Andra, H. F. Putra, D. H. Fandiantoro, and M. Nishimoto. "Multisensory surveillance drone for survivor detection and geolocalization in complex post-disaster environment". In: (2019), pp. 9368–9371 (cited on p. 68).

[3] N. A. Hagen and M. W. Kudenov. "Review of snapshot spectral imaging technologies". In: *Optical Engineering* 52.9 (2013), pp. 090–901 (cited on p. 68).

[4] Y. W. Wang, N. P. Reder, S. Kang, A. K. Glaser, and J. T. Liu. "Multiplexed optical imaging of tumor-directed nanoparticles: a review of imaging systems and approaches". In: *Nanotheranostics* 1.4 (2017), p. 369 (cited on p. 68).

[5] B. Geelen, N. Tack, and A. Lambrechts. "A snapshot multispectral imager with integrated tiled filters and optical duplication". In: 8613 (2013), pp. 861–314 (cited on p. 69).

[6] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed. "Edge computing: A survey". In: *Future Generation Computer Systems* 97 (2019), pp. 219–235 (cited on p. 69).

[7] D. Guo, S. Gu, J. Xie, L. Luo, X. Luo, and Y. Chen. "A Mobile-assisted Edge Computing Framework for Emerging IoT Applications". In: *ACM Transactions on Sensor Networks (TOSN)* 17.4 (2021), pp. 1–24 (cited on p. 69).

[8] D. Balemans, W. Casteels, S. Vanneste, J. de Hoog, S. Mercelis, and P. Hellinckx. "Resource efficient sensor fusion by knowledge-based network pruning". In: *Internet of Things* 11 (2020), p. 100231 (cited on p. 69).

[9] S. Zhou, Y. Wang, D. Chen, J. Chen, X. Wang, C. Wang, and J. Bu. "Distilling Holistic Knowledge with Graph Neural Networks". In: (2021), pp. 10387–10396 (cited on p. 69).

[10] N. Balamuralidhar, S. Tilon, and F. Nex. "MultEYE: Monitoring System for Real-Time Vehicle Detection, Tracking and Speed Estimation from UAV Imagery on Edge-Computing Platforms". In: *Remote Sensing* 13.4 (2021) (cited on p. 69).

[11] F. Tang, W. Gao, J. Zhan, et al. "AIBench training: balanced industry-standard AI training benchmarking". In: (2021), pp. 24–35 (cited on p. 70).

[12] R. Nijhawan, M. Rishi, A. Tiwari, and R. Dua. "A Novel Deep Learning Framework Approach for Natural Calamities Detection". In: (2019), pp. 561–569 (cited on p. 70).

[13] A. Gebrehiwot, L. Hashemi-Beni, G. Thompson, P. Kordjamshidi, and T. E. Langan. "Deep convolutional neural network for flood extent mapping using unmanned aerial vehicles data". In: *Sensors* 19.7 (2019), p. 1486 (cited on p. 70).

[14] A. Singh and K. K. Singh. "Satellite image classification using Genetic Algorithm trained radial basis function neural network, application to the detection of flooded areas". In: *Journal of Visual Communication and Image Representation* 42 (2017), pp. 173–182 (cited on p. 70).

[15] D. Hernández, J.-C. Cano, F. Silla, C. T. Calafate, and J. M. Cecilia. "AI-enabled autonomous drones for fast climate change crisis assessment". In: *IEEE Internet of Things Journal* (2021), pp. 1–1. DOI: 10.1109/JIOT.2021.3098379 (cited on p. 70).

[16] M. Rahnemoonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari, and R. R. Murphy. "FloodNet: A High Resolution Aerial Imagery Dataset for Post Flood Scene Understanding". In: *IEEE Access* 9 (2021), pp. 89644–89654. DOI: 10.1109/ACCESS.2021.3090981 (cited on p. 73).

[17] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez. "A survey on deep learning techniques for image and video semantic segmentation". In: *Applied Soft Computing* 70 (2018), pp. 41–65 (cited on p. 75).

[18] F. Lateef and Y. Ruichek. "Survey on semantic segmentation using deep learning techniques". In: *Neurocomputing* 338 (2019), pp. 321–348 (cited on p. 75).

[19] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. "Image segmentation using deep learning: A

survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021) (cited on p. 75).

[20] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. "Pyramid Scene Parsing Network". In: (July 2017) (cited on p. 75).

[21] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. "Rethinking atrous convolution for semantic image segmentation". In: *arXiv preprint arXiv:1706.05587* (2017) (cited on p. 75).

[22] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848 (cited on p. 75).

[23] K. He, X. Zhang, S. Ren, and J. Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916 (cited on p. 75).

[24] F. Liu, G. Lin, and C. Shen. "CRF learning with CNN features for image segmentation". In: *Pattern Recognition* 48.10 (2015), pp. 2983–2992 (cited on p. 75).

[25] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: (2015), pp. 234–241 (cited on p. 76).

[26] B. Huang, K. Lu, N. Audebert, et al. "Large-scale semantic classification: outcome of the first year of Inria aerial image labeling benchmark". In: (2018), pp. 1–4 (cited on p. 76).

[27] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: (2016), pp. 770–778 (cited on p. 76).

[28] M. Tan and Q. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: (2019), pp. 6105–6114 (cited on p. 76).

[29] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. "The expressive power of neural networks: A view from the width". In: (2017), pp. 6232–6240 (cited on p. 76).

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: (2018), pp. 4510–4520 (cited on p. 77).

[31] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. "A survey on deep transfer learning". In: (2018), pp. 270–279 (cited on p. 77).

[32] A. Van Opbroek, H. C. Achterberg, M. W. Vernooij, and M. De Bruijne. "Transfer learning for image segmentation by combining image weighting and kernel learning". In: *IEEE transactions on medical imaging* 38.1 (2018), pp. 213–224 (cited on p. 77).

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: (2009), pp. 248–255 (cited on p. 77).

[34] M. Huh, P. Agrawal, and A. A. Efros. "What makes ImageNet good for transfer learning?" In: *arXiv preprint arXiv:1608.08614* (2016) (cited on p. 77).

Chapter 6

Discussion

6.1 General overview of contributions

Chapters 3, 4, and 5 cover the leading publications in this thesis showing main research results obtained on the different problems encountered when trying to implement a system for the surveillance of flooded areas based on drone swarm. In general, they all focus on the same primary issue: the need to run computationally heavy workloads, such as machine learning and deep learning algorithms on low-power devices that were not initially designed for this purpose. The main algorithm proposed in this thesis included: (i) algorithms working with tabular information to coordinate the drone's take-off and cluster the information using fuzzy logic, or (ii) execution and adaptation of deep learning algorithms to provide the UAV with image processing capabilities. Each of those algorithms plays a particular role that includes the following:

- Vertical takeoff of drone swarms through the Kunh Munkres algorithm.
- Unsupervised AI processing pipeline for unknown zone scanning.
- Acceleration of CNN-based computer vision algorithms at the edge.

- Benefits obtained from processing in the UAV rather than in the cloud.

6.2 Vertical takeoff of drone swarms through the Kunh Munkres algorithm

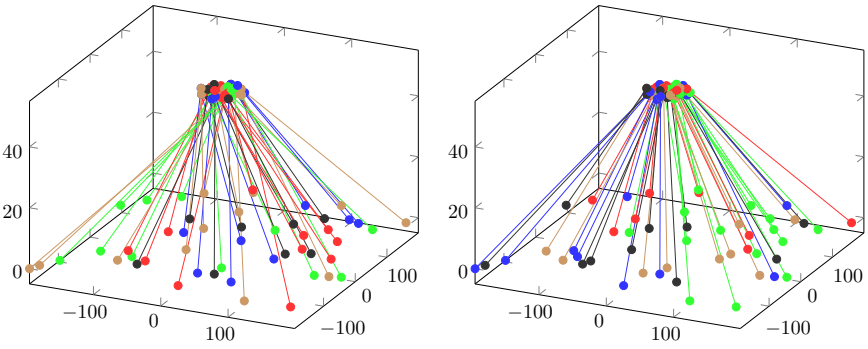


Figure 6.1: Matrix formation Path assignation graph.

In this proposal, the use of the Kunh Munkres or Hungarian algorithm has been explored since, although it is not a method that reports a result that requires automatic learning, thanks to it, a perfect assignment in terms of the total distance of all the devices located on the ground with the target positions in the air after takeoff has been achieved. The use of this solution has its limitations and implementation issues to consider. One such consideration is that the number of UAVs could escalate to the order of thousands of drones, and the calculation time of the perfect allocation based purely on the Munkres algorithm may require more time than desired, especially if it is to be implemented in real-time flight tools. This scenario is reflected in the experimentation carried out as it can be concluded that, while handling the coordinated take-off of a small number of devices, the difficulties that arise as more drones are added do not evolve in a linear fashion, i.e., both the total distance traveled and the computation time to perform an optimal allocation, and therefore, dealing with future drone swarms will require solutions that try to reduce these differences.

After the good results obtained in the implementation of the Kunh Munkres algorithm for position allocation in drone take-off, the implementation of this algorithm in GPUs was tested in both: low-power and high performance computing devices. Figure 6.3 shows a substantial improvement in large formations of more than thousands drones, and only this number of devices would justify the addition of a GPU to reduce the time of position allocation. Therefore, establishing modifications to the method for these situations could be interesting, including approaches such as the optimization of the distance matrix using genetic algorithms, an alternative already mentioned in section 3, or a probabilistic approach in case there are obstacles or possible restricted routes within the drone ground layout and the aerial target formation.

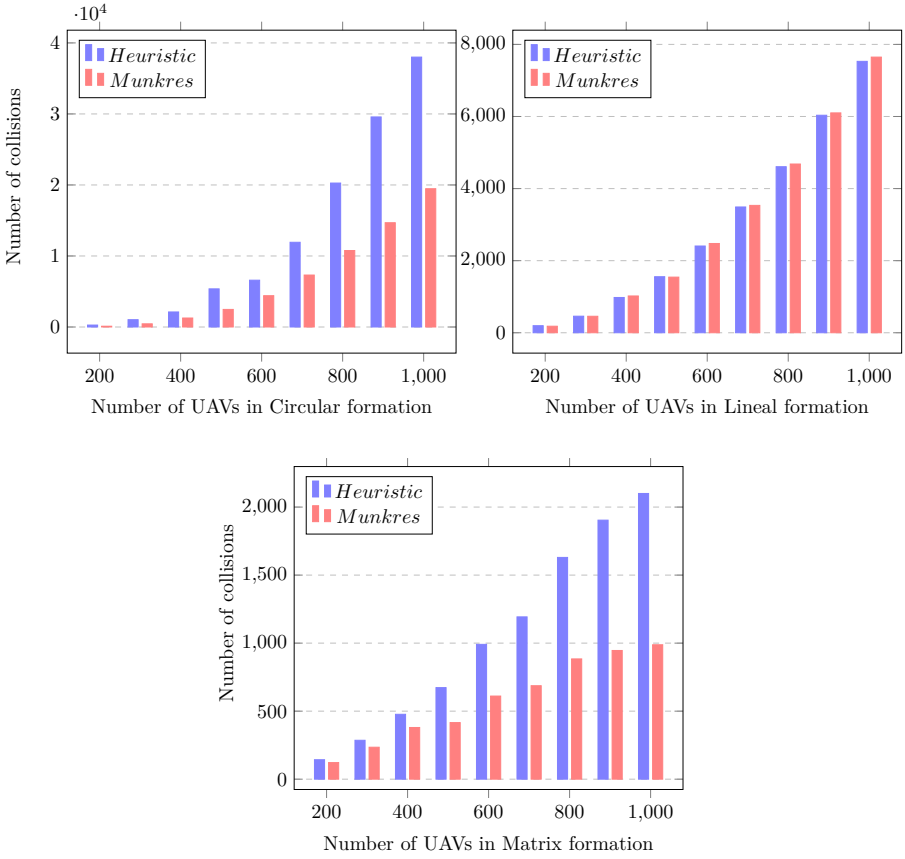


Figure 6.2: Potential collisions between all the nodes of the formation.

As future work this thesis will optimize the takeoff time taking into account the possible crossing routes that may exist, penalizing the assignment algorithms when these occur and try to reduce the assignment time required by the Kuhn-Munkres algorithm by providing a parallel implementation of it. It will also address the challenges of implementing this algorithm by taking into account additional conditions that occur in real environments, such as the coordination of drones, respecting synchronization problems between them, or taking into account obstacles that may arise both at the planning time, and during the execution.

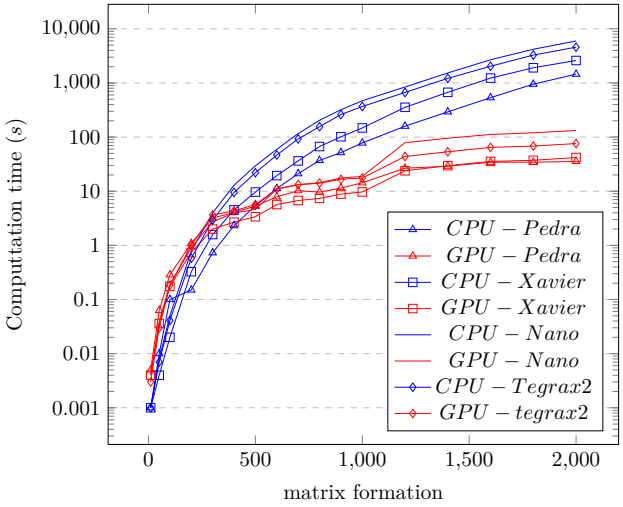


Figure 6.3: Computation time of each algorithm based on the number of UAVs.

6.3 Unsupervised AI processing pipeline for unknown zone scanning

One of the main proposals of this thesis has been to develop an unsupervised surveillance system using UAV devices under the premise that the whole system should operate autonomously without relying on any external hardware for the processing of the images captured from the camera. In order to reduce the number of images to be processed by the first responders in the natural disaster. The proposal is made up of three stages:

1. A lightweight auto-encoder based on deep learning.
2. Dimensionality reduction using the t-SNE algorithm.
3. A fuzzy clustering procedure.

Each section is a stand-alone element of the final model with particular functionality. The autoencoder seeks to store the most characteristic information of all the images in a reduced dimensionality space that

abstractly stores the most characteristic elements of the dataset. The dimensionality reduction using t-SNE seeks to simplify the execution time of the following section. It is, in this case, the primary facilitator of the pipeline execution at the edge since the fuzzy clustering algorithm, which will finally return the cluster leaders that represent the images that could summarize the entire dataset, would have a much longer processing time than expected. If run on a server with higher acceleration capabilities and for a longer time, t-SNE would not be necessary.

T-SNE can compress relevant information without the need for an autoencoder. This method is more than established in the industry [28] to group images by their similarity in a visual way and for feature visualization because, by design, t-SNE creates a probability distribution that captures these mutual distance relationships among points in the initial high-dimensional space. Next, the algorithm creates a low-dimensional space with similar relationships across points. It obtains figures similar to the ones shown in 4.5. The main difference between the two methods of compressing information by similarity to facilitate its processing is that an autoencoder tries to minimize the reconstruction error, while t-SNE tries to find a lower dimensional space and, at the same time, tries to preserve the neighborhood distances. As a result of this attribute, t-SNE is usually preferred for graphics and visualizations. Furthermore, autoencoder, as a relatively recent technique, is starting to be used to abstractly store image datasets, which would allow us to modify it using a variant of it called variational autoencoder [29] or, in this case, extract it for tabular processing of the information it contains and a post clustering.

One more issue that can be presented is to find the motivation which led to implementing the fuzzy minimals (FM) algorithm in prevalence over other methods, and the reason was the lack of prior knowledge of the number of clusters to be searched since this is an essential feature of clustering algorithms is the number of clusters to be generated; k-means and FCM require that the number of clusters to be developed is defined. In contrast, FM does not require this parameter to be set beforehand, with no need for the clusters to be well separated. This feature allows for unsupervised clustering that does not condition the sets of images obtained from each drone mission. Each of the drone missions, therefore, is what motivated us to introduce FM in the final stage of the pipeline proposed in chapter 4.

6.4 Acceleration of CNN-based computer vision algorithms at the edge.

Chapters 4 and 5 shows execution times achieved using different types of convolutional neural networks focused on compressing information in the form of Autoencoder or obtaining the segmentation mask of an image using convolutional networks based on that purpose. It has been seen the great difference between the execution of these algorithms on CPU or using different GPUs that are available on the devices. It is important to note that the use of GPUs is justified in all use cases. The main problem found in this case is the power consumption associated with using the extra hardware that allows us to obtain the results in a shorter time but equipped in a UAV, where the battery is reduced results in a lower potential flight time of the drone.

Initially, it was decided to use these techniques to make more efficient the execution of the model on edge, but it was discarded because the implementation of these techniques always resulted in a loss in the accuracy of the model, and this would be acceptable if it has a higher performance (i.e., MIoU), but because the results obtained by measuring the surface of the flooded area are close to acceptable values and a loss of accuracy would not be acceptable for this model, therefore it was decided to run the model in raw and without any extra adaptation. This is subject to obtaining more data to increase overall performance since, for this study, only 57 images were available, and several attempts were required to use transfer learning from pre-trained models on Imagenet before performing the training.

In the 4 section, it can be seen how the inference time when processing the whole dataset was, in the best case, three times higher using CPU than GPU being this difference more noticeable in devices with higher GPU. In these situations, it is shown how the use of GPU to accelerate the model is interesting. All data obtained are considered based on the PyTorch and Sklearn-based implementation of the different image preprocessing and DL algorithms, and alternative implementations such as quantization of the model to reduce the neural network model weights from floating point to integer will result in smaller models and faster execution speed. Concerning the execution of deep learning algorithms at the edge, the model could have been further optimized by using field-programmable

gate array (FPGA) devices, which are a matrix of programmable logic blocks and a hierarchy of reconfigurable interconnects that allow the blocks to be connected. The logic blocks can be programmed to perform complex combinational functions or act as simple logic gates such as AND and XOR. In most FPGAs, the logic blocks can also include pieces of memory, which can be simple flip-flops or more complete memory modules.

6.5 Benefits obtained from processing in the UAV rather than in the cloud

The deep learning algorithm proposed in section 4 does not require any dataset on which it has been previously trained, and an understanding of the area traversed is achieved in an unsupervised way that allows us to navigate through the space embedded in the central layer of the model to be then able to execute the rest of the pipeline. This training is designed to be performed the first time you pass through an area, and you do not have any information, and its training, which will last several minutes, will be performed on the ground. Afterward, the model will be stored in the device and will allow for evaluating changes in the area in future surveillance missions. In other words, in the first case, it would allow us to obtain the most relevant images of the dataset obtained in a few minutes, but then online and in real-time, it would allow us to identify changes in the same area concerning previous executions of the surveillance mission.

In the case of semantic segmentation, collateral beneficial results are obtained for the system that was originally intended only to send an abstraction of the image obtained to help the system user to locate flooded areas since, as shown in the article, the bandwidth of sending the drone information will be significantly reduced by generating the mask at the edge, and that can save network resources and decrease battery consumption as well as protect the privacy of those affected by the area being monitored since the information is not sent in detail and could send only the location of the pixels referring to a flooded area and not a photograph that would include people, vehicles, buildings in detail and other existing entities in urban areas.

6.6 Summary

This thesis describes a series of works that aim to enable the efficient analysis of large amounts of data and images in IoT environments for the monitoring of areas affected by natural disasters. In addition, a performance study of the different solutions is carried out, showing a comparison of their execution in high-performance systems or embedded in a UAV.

The coordinated vertical take-off performance of a vehicular network of UAVs was drastically improved by introducing a solution based on the Kun Munkres algorithm, which is able to improve on previously used methods, achieving a compromise between computational time and minimisation of position allocation error. Furthermore, it is worth noting that the deployment capabilities of drone swarms are fully exploited, eliminating the limitation of the number of drones in surveillance missions.

The proposal of an unsupervised model for surveillance of areas for which no data is previously available represents a breakthrough for autonomous exploration using drones and makes it possible to obtain an overall view of a specific area without conducting a detailed study of the area.

The semantic segmentation model of flooded areas deployed for image processing in the device will allow collecting flood data in real time while respecting privacy and will allow performing virtual fidelity reconstructions of the event being monitored.

Finally, the research conducted and the proposed models create the perfect tests to check the performance of the new tools, allowing accessible evaluations for future research. Thus, the path for future development remains solid and open.

Chapter 7

Conclusions

7.1 Concluding remarks

The previous chapters have addressed techniques and systems to facilitate the surveillance and monitoring of areas affected by natural disasters using UAVs and computer vision techniques. This research different disciplines within computer science, such as image processing by developing ML/DL algorithms for image segmentation and clustering and edge computing by implementing these computationally heavy workloads in hardware with the limited computational capacity.

Given that natural disasters such as floods affect large areas and make it difficult to monitor them by first responders, a reliable autonomous drone-based surveillance solution can be very useful for those scenarios, allowing authorities to have a global view of the event in both; known areas and of an unknown areas; i.e. areas which no data is previously available. Some proposals have addressed some of these issues separately, but none of them present a comprehensive approach that addresses all aspects of this problem and makes a proposal from model conception to deployment in the autonomous device.

This thesis has addressed the development of artificial intelligence solutions at the edge, taking into account the terms of system accuracy,

performance, and reliability, to present a better overall solution that improves existing techniques and provides an alternative for future environmental crisis monitoring systems. A new proposal for the vertical take-off planning of drone swarms scaling from a dozen existing drones to thousands of drones with the minimum possible margin of error is developed. In addition, a system is proposed to obtain the most relevant images for surveillance of a previously unknown area and to facilitate surveillance in future missions. Finally, it is proposed a way to obtain flooded areas using one or several UAV devices with a fully autonomous system that does not require cloud processing of the images obtained.

7.2 Publications

Within the framework of this thesis, several papers have been published. They describe the proposed solutions to the different problems that have been exposed during the research phase.

Journals

- [30] Hernández, D., Cecilia, J. M., Cano, J. C., Calafate, C. T. (2022). Flood Detection Using Real-Time Image Segmentation from Unmanned Aerial Vehicles on Edge- Computing Platform. Remote Sensing, 14(1), 223. doi: 10.3390/rs14010223. JCR: 1er cuartil. Impact Factor: 4.848

- [31] Hernández, D., Cano, J. C., Silla, F., Calafate, C. T., Cecilia, J. M. (2021). Al- enabled autonomous drones for fast climate change crisis assessment. IEEE Internet of Things Journal. doi: 10.1109/JIOT.2021.3098379. JCR: 1er decil. Impact Factor: 9.936

International Conferences

- [32] Hernández, D., Cecilia, J. M., Calafate, C. T., Cano, J. C., Manzoni, P. (2021, April). The Kuhn-Munkres algorithm for efficient vertical takeoff of AV swarms. In 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring) (pp. 1-5). IEEE. doi: 10.1109/VTC2021-Spring51267.2021.9448873. Core A

Publications related with this thesis

Moreover, some collaboration have been carried out with other members of our research group where the PhD candidate has contributed by developing deep learning algorithms or ML solutions.

- [33] Nakamura, K., Hernández, D., Cecilia, J. M., Manzoni, P., Zennaro, M., Cano, J. C., Calafate, C. T. (2021). LADEA: A Software Infrastructure for Audio Delivery and Analytics. Mobile Networks and Applications, 26(5), 2048-2054.

- [34] Hernández, D., Arcas-Túnez, F., Muñoz, A., Cecilia, J. M. (2018, August). Bauspace: a scalable infrastructure for soft sensors development. In Proceedings of the 47th International Conference on Parallel Processing Companion (pp. 1-4).

- A real-time UAV surveillance system for natural, HiPEAC Conference Budapest 2022, HiPEAC Student Challenge: IoT for Everyone!, award-winning disaster management

- Assignment And Take-off Approaches For Large-scale Autonomous UAV Swarms. Publication pending

Projects involved

Finally, the research here presented has been developed in the context of GLOBALoT project; a private-public research project that is defining the next generation of IoT systems to deal with climate change issues. The projects that this PhD candidate has been involved are the following:

- Planificación y gestión de recursos hídricos a partir de análisis de datos de IoT (WATERoT). Funded by Spanish Government, Ministerio de Economía y Competitividad (Retos-Colaboración) from October 2018 until March 2022. Role: Principal Coordinator. Budget: 251.140€ (for the UCAM subproject). Number of Researches: 9. Total Budget: 1.121.131,83 €.

- Desarrollo de Infraestructuras IoT de Altas Prestaciones contra el Cambio Climático basadas en Inteligencia Artifical (GLOBALoT). Funded by Ministerio de Ciencia e Innovación (RTC2019-007159-5), from January 2020 until December 2023. Role: Researcher. Budget: 208,217.5€. Number of Researches: 8.

7.3 Future work

In the coming years, the use of autonomous drone swarms for surveillance and data collection is expected to grow. Among the applications where drones may be involved include fire surveillance, earthquake zones, and all kinds of situations that require an early response from the competent authority. This will lead to an increase in the number of devices used and the tasks to which they are assigned, which will pose new technical challenges.

With this thesis, the basis is laid to support these upcoming changes while providing some proposals to improve the coordinated take-off and provide intelligence to these devices. Services such as fire detection, flooded areas, and people in danger will be examples of the usefulness that can be obtained. Using our research, it will be possible to develop services that enable the coordination of large drone arrays and allow image processing to be performed without the need for extra devices. This flexibility makes our approach a future-proof bet, and provides a development path for anyone interested in deploying a drone-based surveillance system. As part of future work will seek to optimize the takeoff time taking into account the possible crossing routes that may exist, and we will try to reduce the allocation time required by the Kuhn-Munkres algorithm by performing a parallel implementation of the algorithm.

Regarding the development of an autonomous drone model for rapid assessment of climate crises, there is still much work to be done from different perspectives. In terms of communication, it is intended to extend the results of this article from a single device to the use of a swarm of drones that can provide more excellent coverage of the area to be surveyed, something essential when an area affected by natural disasters is pervasive. Including energy-efficient processors in such low-autonomy devices is necessary to enable AI-based applications; tinyML is a good step in this direction. More processing steps could be added in this AI pipeline that, after labeling the clusters, would reuse the information stored in the autoencoder to categorize all images and send the desired information more condense. Finally, it would be desirable to experiment with users interested in the solution in real-case scenarios to learn about new real needs.

To conclude, the work of Flood Detection Using Real-Time Image

Segmentation has demonstrated how, with a small set of correctly labeled data and the right model, a real-time segmentation system can be embedded in a UAV, bringing the main computational work closer to the device in charge of processing the information to make an offline inference and send the digested data. However, main results of this work towards a tool with a visual interface that displays the information and data of the flooded area on a map in real-time is what further work could be extended. In addition, it is necessary to increase the performance of the AI models by providing new datasets, as well as to deepen the representation of the semantic segmentation result obtained by the network on a map of the area where the images have been captured, to provide more knowledge.

Acrónimos

0

5G	5rd Generation of Wireless Mobile Telecommunications Technology

A

AI	Artificial Intelligence
AGVS	Airport Ground Video Surveillance
AVG	Average
AIDER	Aerial Image Dataset for Emergency Response applications
ARM	Advanced RISC Machines

C

CNN	Convolutional Neural Network

CAE	Convolutional Autoencoder
CV	Computer Vision
CWS	Compact Well-Separated
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CUDNN	NVIDIA CUDA® Deep Neural Network library

D

DBSCAN	Density-based Spatial Clustering of Noisy Applications
DNN	Deep Neural Network
DL	Deep Learning

E

EC	Edge Computing
EMD	Extended Motion Diffusion-based

F

FPGA	Field-Programmable Gate Array
FC	Fuzzy Clustering
FCM	Fuzzy C-means

G

GPU	Graphics Processing Unit
GA	Genetic Algorithm

GPS	Global Positioning System

H

HPC	High-performance computing

I

IoT	Internet of Things
LSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection Over Union

K

KNN	K-nearest Neighbours
KL	Kullback-Leibler

L

LIDAR	Light Detection and Ranging

M

ML	Machine Learning
MEC	Multi-access Edge Computing

MIoU	Intersection-Over-Union

N

NN	Neural Network

P

PhD	Doctor of Philosophy
PCA	Principal Component Analysis

R

RF	Random Forest

S

SVM	Support Vector Machines
SoC	System On Chip

T

t-SNE	t-distributed stochastic neighbor embedding
TPU	Tensor Processing Unit

U

UAV	Unmanned Aerial Vehicle

V

VTOL	Vertical Takeoff and Landing
VGG	Vertical Visual Geometry Group

References

[1] J. B. Roerdink and A. Meijster. "The watershed transform: Definitions, algorithms and parallelization strategies". In: *Fundamenta informaticae* 41.1-2 (2000), pp. 187–228 (cited on pp. 1, 11).

[2] A. Pathan, M. A. P. Kulkarni, M. N. L. Gaikwad, M. P. M. Powar, and A. R. Surve. "An IoT and AI based Flood Monitoring and Rescue System". In: *Int. J. Eng. Tech. Res.* 9.9 (2020), pp. 564–567 (cited on p. 2).

[3] S. Li, L. D. Xu, and S. Zhao. "The internet of things: a survey". In: *Information systems frontiers* 17.2 (2015), pp. 243–259 (cited on p. 5).

[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660 (cited on p. 5).

[5] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim. "Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios". In: *Ieee Access* 8 (2020), pp. 23022–23040 (cited on p. 6).

[6] Z. Wang, R. Liu, Q. Liu, J. S. Thompson, and M. Kadoch. "Energy-efficient data collection and device positioning in UAV-assisted IoT". In: *IEEE Internet of Things Journal* 7.2 (2019), pp. 1122–1139 (cited on p. 6).

[7] K. Nihei, N. Kai, Y. Maruyama, et al. "Forest Fire Surveillance using Live Video Streaming from UAV via Multiple LTE Networks". In: (2022), pp. 465–468 (cited on p. 6).

[8] P. A. Rodriguez, W. J. Geckle, J. D. Barton, J. Samsundar, T. Gao, M. Z. Brown, and S. R. Martin. "An emergency response UAV surveillance system". In: 2006 (2006), p. 1078 (cited on p. 6).

[9] Q. Wen, H. He, X. Wang, et al. "UAV remote sensing hazard assessment in Zhouqu debris flow disaster". In: 8175 (2011), p. 817510 (cited on p. 6).

[10] S. Mehrdad, M. Satari, M. Safdary, and P. Moallem. "Toward real time UAVS'image mosaicking". In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 41 (2016), p. 941 (cited on p. 6).

[11] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang. "A survey on the edge computing for the Internet of Things". In: *IEEE access* 6 (2017), pp. 6900–6919 (cited on p. 7).

[12] G. Premsankar, M. Di Francesco, and T. Taleb. "Edge computing for the Internet of Things: A case study". In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 1275–1284 (cited on p. 7).

[13] M. De Donno, K. Tange, and N. Dragoni. "Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog". In: *Ieee Access* 7 (2019), pp. 150936–150948 (cited on p. 7).

[14] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran. "The role of edge computing in internet of things". In: *IEEE communications magazine* 56.11 (2018), pp. 110–115 (cited on p. 7).

[15] C.-C. Lin and J.-W. Yang. "Cost-efficient deployment of fog computing systems at logistics centers in industry 4.0". In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4603–4611 (cited on p. 7).

[16] Y. Zhou, B. Rao, and W. Wang. "UAV swarm intelligence: Recent advances and future trends". In: *IEEE Access* 8 (2020), pp. 183856–183878 (cited on p. 8).

[17] F. Fabra, J. Wubben, C. T. Calafate, J. C. Cano, and P. Manzoni. "Efficient and coordinated vertical takeoff of UAV swarms". In: (2020), pp. 1–5 (cited on p. 10).

[18] F. Fabra, C. T. Calafate, J. C. Cano, and P. Manzoni. "ArduSim: Accurate and real-time multicopter simulation". In: *Simulation Modelling Practice and Theory* 87 (2018), pp. 170–190 (cited on p. 10).

[19] Z. Kato and T.-C. Pong. "A Markov random field image segmentation model for color textured images". In: *Image and Vision Computing* 24.10 (2006), pp. 1103–1114 (cited on p. 11).

[20] A. Smith. "Image segmentation scale parameter optimization and land cover classification using the Random Forest algorithm". In: *Journal of Spatial Science* 55.1 (2010), pp. 69–79 (cited on p. 11).

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cited on p. 11).

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90 (cited on p. 12).

[23] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014) (cited on p. 13).

[24] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: (2015), pp. 234–241 (cited on p. 14).

[25] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. "An efficient k-means clustering algorithm: Analysis and implementation". In: *IEEE transactions on pattern*

analysis and machine intelligence 24.7 (2002), pp. 881–892 (cited on p. 16).

[26] Y. Rani[1] and H. Rohil. "A study of hierarchical clustering algorithm". In: 2 (2013), p. 113 (cited on p. 16).

[27] A. Ram, A. Sharma, A. S. Jalal, A. Agrawal, and R. Singh. "An enhanced density based spatial clustering of applications with noise". In: (2009), pp. 1475–1478 (cited on p. 16).

[28] L. Van der Maaten and G. Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008) (cited on p. 104).

[29] X. Hou, L. Shen, K. Sun, and G. Qiu. "Deep feature consistent variational autoencoder". In: (2017), pp. 1133–1141 (cited on p. 104).

[30] D. Hernández, J. M. Cecilia, J.-C. Cano, and C. T. Calafate. "Flood detection using real-time image segmentation from unmanned aerial vehicles on edge-computing platform". In: *Remote Sensing* 14.1 (2022), p. 223 (cited on p. 110).

[31] D. Hernández, J.-C. Cano, F. Silla, C. T. Calafate, and J. M. Cecilia. "AI-enabled autonomous drones for fast climate change crisis assessment". In: *IEEE Internet of Things Journal* 9.10 (2021), pp. 7286–7297 (cited on p. 110).

[32] D. Hernández, J. M. Cecılia, C. T. Calafate, J.-C. Cano, and P. Manzoni. "The Kuhn-Munkres algorithm for efficient vertical takeoff of UAV swarms". In: (2021), pp. 1–5 (cited on p. 110).

[33] K. Nakamura, D. Hernández, J. M. Cecilia, P. Manzoni, M. Zennaro, J.-C. Cano, and C. T. Calafate. "LADEA: A Software Infrastructure for Audio Delivery and Analytics". In: *Mobile Networks and Applications* 26.5 (2021), pp. 2048–2054 (cited on p. 111).

[34] D. Hernández, F. Arcas-Túnez, A. Muñoz, and J. M. Cecilia. “Bauspace: a scalable infrastructure for soft sensors development”. In: (2018), pp. 1–4 (cited on p. 111).