

Joint Classification and Estimation of UAVs using Multi-Task Learning

By
Apostolos Pappas

in partial fulfilment of the requirements for the degree of

Master of Science

in Electrical Engineering

at the Delft University of Technology,

to be defended publicly on Monday August 28, 2023 at 10:00 AM.

Student Number:	5616700	
Supervisor:	Dr. F. Fioranelli,	TU Delft (MS3)
Thesis committee:	Dr. Ir. J. Dauwels,	TU Delft (SPS)
	Dr. Ir. J.J.M. de Wit,	TNO

This thesis is confidential and cannot be made public until August 28, 2025.

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Preface

I still cherish the moment when I received the acceptance letter for the MSc in Electrical Engineering, Wireless Communication and Sensing track at TU Delft. As I now recall, I received multiple acceptance letters almost simultaneously, a sunny April morning right after my guard duty at the final days of my military service back in Greece. I had no second thoughts when choosing TU Delft over the other options, and to this day I am glad I did. Yet, nothing could prepare me at that time for the hardships, hard work and several moments of self-doubt that would accompany my studies. Having reached the end of my MSc studies I can safely say that I am grateful for each hardship I had to go through. I explored not only the span of my abilities, but also myself. I consider this my greatest achievement so far.

The introduction of drones in our lives has brought several benefits when used for applications such as recreational or search and rescue ones. However, as recent and unfortunate developments show, drones are also changing the way wars are being fought around the globe. Nowadays they are being used both as information gathering means as well as for precision striking of enemy targets. In this context, I consider the radar as one of the most important defensive parameters in an era where information and situational awareness can draw the line between life and death. In this thesis work, an effort towards improving the drone characterization through the use of joint task training and prediction is made. I will be forever grateful to both TNO and TU Delft for giving me the opportunity to work on such a serious and exciting topic.

Apostolos Pappas

Delft, August 2023

Contents

Abstract	8
Acknowledgments.....	9
1 Introduction.....	10
1.1. Motivation.....	10
1.2. Problem Statement.....	12
1.3. Thesis contribution.....	13
1.4. Thesis structure	14
2 Literature Review	16
2.1. UAV classification among other targets.....	16
2.2. UAV parameter estimation and classification.....	17
2.3. Multi-Task Learning.....	18
2.3.1. Review of the state-of-the-art in MTL.....	20
2.3.2. MTL in radar.....	21
2.4. Conclusion.....	23
3 Radar & Neural Networks Fundamentals.....	26
3.1. Radar as a sensor	26
3.1.1. CW radar fundamentals	26
3.1.2. Micro-Doppler processing and related data formats	29
3.2. Deep Learning fundamentals	33
3.2.1. Neural Networks	33
3.2.2. The ResNet.....	37
3.2.3. Training of Neural Networks	38
3.3. Conclusion.....	40
4 Dataset.....	42
4.1. Description of the original dataset	42
4.1.1. Types of drone targets.....	43
4.1.2. Measurement Scenarios	43
4.1.3. Segmentation of the original dataset	45
4.2. Retrieving the rotation rates	45
4.2.1. Extracting representative quefrencies.....	46
4.2.2. Defining and labelling events	50

4.3.	Definition of datasets for experimentation	52
4.3.1.	Imbalance study	53
4.4.	Conclusion	57
4.5.	Acknowledgments	58
5	Proposed Methodology	60
5.1.	Defining related tasks	60
5.2.	Description of the proposed models	61
5.2.1.	The architecture of the STL models	62
5.2.2.	The architecture of the MTL models	62
5.2.3.	Handling of the joint loss and training	64
5.3.	Comparing STL and MTL networks	66
5.3.1.	Restrictions regarding the models	66
5.3.2.	Performance metrics	67
5.4.	Description of the testing configurations	70
5.4.1.	Tests on different train/test splits	71
5.4.2.	Robustness tests	72
5.5.	Summary	72
6	Results: drone structural configuration characteristics estimation	75
6.1.	Wing Type & Number of Rotors	75
6.1.1.	Testing on different train/test splits	75
6.1.2.	Robustness tests	80
6.2.	Wing Type, Number of Rotors & Multiple Drone Detector	83
6.3.	Wing Type, Number of Rotors & Payload Detection	85
6.3.1.	Testing on different train/test splits	85
6.3.2.	Robustness test	87
6.4.	Conclusion	92
7	Results: flight behaviour based estimation	95
7.1.	Mean rotor rate estimation & Payload detection	95
7.1.1.	Testing on different train/test splits	95
7.1.2.	Robustness test	97
7.2.	Rotor rate estimation & Event detector	100
7.2.1.	Testing on different train/test splits	101
7.2.2.	Robustness test	102
7.3.	Conclusion	104

8 Conclusions & Recommendations	107
8.1. Concluding remarks on the results of this thesis	107
8.2. Recommendations for further research	108
References.....	112
Appendix A	116
Appendix B	118
Appendix C	120
Appendix D	124

Abstract

Over the last decade or so, the use of drones has grown significantly and is expected to continue to grow in the coming years. This growth in use, accompanied by the ever decreasing cost in manufacturing and acquisition, caused an increase in unlawful usage of drones as well. Also in the military domain, increasing numbers of drones appear on the battlefield, for a variety of applications ranging from espionage to target identification and payload drop on enemy targets. As the latest developments have shown, apart from military-grade UAVs, commercial drones are continuously being deployed on the battlefield as means of observation or to drop grenades or other improvised explosives. Hence, the importance of developing means of characterizing drone targets within their flight environment is becoming greater and greater.

Radar as a sensor has been praised for its resilience in all weather conditions and ability to monitor vast volumes quickly, determining directly range, velocity and angular positions of many targets. The parallel development and advancements in Machine and Deep Learning (ML & DL, respectively) have additionally found their way in applications regarding both drone detection and characterization. Traditional DL approaches involving the training and testing of a single model for a single drone characterization task can be found to perform rather well in the literature. Nevertheless, there are still performance margin to be covered in order to meet all requirements in the ever-complex operational environment. A proven learning technique that has shown promise in the literature is the Multi-Task Learning (MTL), which is based on the assumption that if tasks share knowledge between them, an MTL model is easier to train and has improved generalization capabilities as compared to separately trained single-task neural networks.

In this thesis work, the MTL paradigm is investigated for the first time for the simultaneous estimation of multiple tasks based on a CW radar's Micro-Doppler spectrograms in an exploratory manner. Specifically for the radar-based monitoring of drones, these tasks include the wing type, number of rotors classification, the payload, multiple drones and event detection, as well as the mean rotor rate regression. Task groups that appear reasonable to an expert human are then formed out of the aforementioned tasks and three distinct and novel, for the field, MTL architectures are proposed. The performance of the MTL models is thoroughly assessed over two types of testing approaches and compared to corresponding single task networks. It can be concluded that the utilization of MTL led to improvements in performance in most task groups, given the available dataset of measured drone spectrograms.

Acknowledgments

Just before the beginning of this thesis work, it feels more than appropriate to give credit in the form of acknowledgements to certain people that in a way or another supported me, inspired me or even contributed to my journey as a MSc student.

First and foremost, I would like to thank my supervisors throughout these nine months, Dr. Francesco Fioranelli, Dr. Ir. Jacco de Wit and Bas Jacobs. Thank you for inspiring me to the fullest, answering questions and supporting me continuously, even when at times nothing seemed to work. I am sure I will cherish our time working together. Thank you for the trust you put in me and for bearing my oftentimes pessimistic nature.

I would also like to thank all the friends I made both in TU Delft and TNO these two years. Without them this journey would be far less pleasant. I would like especially to thank all people I had the pleasure to work and communicate with at TNO. It is a really rare phenomenon to feel so much appreciated in so little time. Leaving the “interns office” was by no means an easy task.

I would like to thank my parents, Konstantinos and Kornilia for their continuous support and patience these two years, especially during the first months of my studies. Without your support nothing written in the following pages would have taken place. I would also like to thank my friends Akis and Christos. Being where we are, doing what we do were all dreamt of in a small apartment in Volos. I am really proud of all our hard work so far. Last but certainly not least, I would like to thank Joana for being there for all my ups and downs, and there were a lot of the latter.

1 Introduction

*This chapter serves as the introduction to this thesis. In **Section 1.1**, the motivation behind the development of a system for joint classification and estimation of different aspects regarding drone targets using radar is presented. The problem statement for this thesis can be found in **Section 1.2**, alongside a brief review of the current gaps in literature. **Section 1.3** summarizes the contributions introduced by this research work. Finally, **Section 1.4** presents the structure of this thesis report.*

1.1. Motivation

In the recent years, Unmanned Aerial Vehicles (UAVs), commonly known as drones, have seen a spike in availability and, consequently, use. This has brought them into the spotlight regarding both civil and military use. Regarding their civil applications, drones are actively used by hobbyists and professionals as means towards photography or filmmaking. More recent applications of UAVs also include search and rescue missions [1], mailing and delivering, communications, meteorology, agriculture, mapping and even space exploration. An extensive review on drone applications can be found in the work of Hassanalian et al. [2].

The wide availability of drones has led to the development of malicious applications as well. Apart from espionage using various sensors (including optical, acoustic, amongst others) [3], there have been numerous reports regarding terrorist attacks using commercially available drones [4, 5]. It is important to state that both of the attacks mentioned in these reports were carried out by commercial drones carrying some type of improvised explosive device. In the recent scenes of the war in Nagorno-Karabakh in 2020 and the Russian invasion of Ukraine in 2022, the military usage of drones was showcased in numerous cases. In both wars, usage of both publicly available and military-grade built drones was not limited to reconnaissance, but expanded to direct attack on ground targets in the forms of grenade dropping or loitering munitions. Such examples are the commercially available DJI Mavic drone series (Figure 1.1a), the Iranian made Shahed-136 and the Polish built Warmate (Figure 1.1b).



a) A commercial DJI Mavic quadcopter drone.
Source: <https://www.dji.com/nl/mavic-air/info>



b) A polish made Warmate loitering munition drone.
Source: <https://wbgroupamerica.com/product/warmate/>

Figure 1.1: Two different types of drones: a rotary and a fixed wing one. Despite its small size the DJI Mavic (left) can be employed for several malicious applications. The Warmate loitering munition is a fixed wing drone explicitly built with attacking or defensive intents.

These very unfortunate events highlight the importance of developing robust systems able to detect, track and classify drone targets in populated environments. Several sensor types have been proposed in the literature so as to counter the potential threat introduced by the widespread use of drones:

- Acoustic sensors: The exploitation of acoustic sensors towards drone detection has been proposed in several studies. Although the detection and tracking can be done in a solely passive manner with relatively low cost sensors, this approach is highly dependent on environmental factors, such as background noise [1].
- Optical sensors: The high availability of cheap optical sensors of good quality, coupled with the advancements in Machine and Deep Learning (ML and DL respectively) in computer vision, have made optical sensors an attractive solution to the problem of drone detection and classification [1]. Yet, optical sensors are bounded by several constraints, most important among them being the weather conditions at hand (clouds, fog and dust are just some of the conditions that can lead to performance drops in optical sensors).
- Radio-Frequency (RF) sensors: RF based drone detection sensors are also proposed throughout the literature, such as the one in [2]. Such sensors come with a few advantages, the most important of which is the identification of the appropriate frequency domain for RF jamming purposes in the counter UAV operation context. Another advantage of the RF sensors is that they are totally passive, making them also accurate in the presence of bad weather or low visibility conditions [3]. Yet, the advancements in Software Defined Radio are expected to negatively affect the performance of such methods [6].
- Radar: What makes radar stand out as a sensor suited for drone detection is its 24 hour, all weather, non-cooperative surveillance capabilities over wide regions [9]. Making use of electromagnetic radiation emission, radar systems are able to quickly scan very large volumes. Moreover, by further processing of their output, relevant information on the drones' size and number of rotors, wing type, payload, and behaviour, can be obtained.

A very informative comparison table regarding the advantages and disadvantages of radar when compared to other sensors in the context of UAV detection and identification can be found in the work of Wellig et al. [10] where Figure 1.2 has been borrowed from.

As expected, as optimal of a sensor as it may sound, radar comes with its own, unique, disadvantages. Drones come in several shapes and sizes in the market, built by using a wide range of different materials. This in return has its implications on the radar cross section (RCS) of drone targets, making it as low as -15 dBsm to -30 dBsm. Thus, to detect such small appearing targets, highly sensitive systems are required, the side-effect of which is the detection of many other small non-drone targets such as birds or even swarms of birds, thus creating a confusing picture on a radar operator display [12], or generating a deluge of data that can overcome automatic algorithms. Furthermore, an oftentimes overlooked disadvantage of the radar as a UAV detection sensor is that it is not passive (not necessarily at least), meaning that in a hostile environment, the location of the sensor may be compromised by its emission.

Sensors	Long range	Position accuracy	Identification	Multiple targets	Low visibility conditions	Night	Passive system	Price
Visual	++	++	++++	++	-	-	++++	++
Infrared	++	++	++++	++	- (except SWIR)	++++	++++	+
Acoustic	-	-	+++	++	++++	++++	++++	++++
Radar	++++	++++	++	++++	++++	++++	-	-
Electronic support measures	++++	++++	++	++++	++++	++++	++++	+
Human surveillance	+	+	++++	-	-	-	++++	++++

Figure 1.2: Comparison between radar and other surveillance methods suited for UAV detection and identification [10].

Apart from an initial detection of drones in their flight environment, a further processing step that would lead to the increase of situational awareness is the classification or the estimation of different drone characteristics in the context of what is found in the literature as “drone-characterization”. These may include the type of UAV in the scene (i.e. rotary wing or fixed wing), the number of rotors, their rotation rates, and whether the drone in question carries certain payload among others. This is done in an effort towards robust Automatic Target Recognition, possibly leading to a ranking among drone targets regarding the threat they pose, as for instance a drone classified as carrying no payload may be seen as less of a threat than one carrying a potentially hazardous payload, depending on the context of the situation.

With the advances in the fields of ML and DL, processed radar data can be used as input to such methods providing high accuracies and inference capabilities. What has enabled such approaches to be developed efficiently is the improvement in computational capabilities with the introduction of powerful Graphic Processing Units (GPUs). Such devices are essential, since DL approaches often involve the use of complex models, high amount of input data, and complex optimization algorithm with many parameters.

In this research work, effort is made towards producing a Deep Learning approach for the joint classification and estimation of different drone parameters. The parameters in question are the:

- Wing type of the drone (i.e. fixed, helicopter, rotary wing)
- Number of rotors a drone possesses
- The existence of payload mounted on the drone
- The existence of one or more drones in a scene
- The estimation of a representative rotation rate of the rotors
- The identification of events based on the changes of the rotation rates

To achieve efficient joint training and prediction of these parameters, for the first time in this work, the Multi-Task Learning framework has been utilized, as described in the next section.

1.2. Problem Statement

The classification task of drone versus environment using Deep Learning techniques has been extensively covered in the literature, with many different model architectures and input representations explored and tested. It has been shown that neural networks are able to distinguish among drone and other targets with very high accuracies (see Chapter 2). In this work, this first classification step is omitted, assuming that the target is already classified as a UAV. The effort is then put towards drone characterization.

Based on an extensive literature review presented in Chapter 2, the following issues have been identified, mostly regarding the area of UAV parameter estimation using radar data and DL:

- Very little research has been done on UAV parameter classification and estimation, especially using DL methods and techniques compared to that of classifying UAVs against other types of targets in their environment.
- A number of these works are limited on simulated data rather than real measurement data, or a mix among these two. Training on synthetic data might have implications and lead to performance degradation when models are then applied on real measurement data, the quality and resolution of which might be different.
- Little research has been carried out in the area of rotor rate estimation and possible event detection especially using real measurement data.
- DL models can be characterized as data-hungry, meaning that in order to be trained efficiently and provide acceptable performances, often large datasets are needed. Typically though, radar data are scarce. The Multi-Task Learning (MTL) approach exploited in this work has been proposed in other Deep Learning related works as means towards a more efficient training when the available dataset is limited in size.
- To the best of the author's knowledge, there has been no other work that addresses the problem of joint classification and parameter estimation of drones in the manner presented in this thesis. No research works were found on MTL classification or regression in this context, while trying to improve training quality and efficiency over more traditional Single Task Learning approaches where multiple models predicting only one characteristic are defined.

In these terms, the goal of this thesis project and the manner in which it addresses the current gaps in literature can be summarized as follows:

The goal of this thesis is to formulate a Deep Learning framework able to jointly classify and estimate different parameters of Unmanned Aerial Vehicles using radar data. This includes the development, evaluation and verification of such a concept based on acquired radar data from real-world measurements. To counter the setbacks introduced by the lack of data, and to achieve the joint prediction of the tasks, a Multi-Task Learning approach is followed.

1.3. Thesis contribution

The novel contributions of this thesis can be summarized as follows:

- Expansion of the classical joint methods of parameter estimation of UAVs to using data-driven Deep Learning methods.
- The use of the MTL framework to jointly estimate different UAV characteristics.
- The introduction of novel DL model architectures for joint drone characterization through the use of MTL.
- A first approach on rotor rate regression and event detection during the drone's flight using supervised DL.

Part of the results of this thesis work have been accepted for presentation at the IEEE International Radar Conference, to be held in Sydney, Australia, in November 2023.

1.4. Thesis structure

The rest of this thesis report is structured as follows: Chapter 2 covers the literature review of this work including state-of-the-art methods in UAV parameter estimation, the concept and most important work in MTL and its use in radar applications. Chapter 3 provides with background necessary for this thesis while Chapter 4 introduces the reader to the various datasets created and used in this work. Chapter 5 presents the methodology behind the experimentation done for this work. Chapters 6 covers the first part of the results regarding drone structural configuration, extensively comparing between STL and MTL. Chapter 7 presents the second part of the results, mainly focusing on the flight characteristics and behaviour estimation of the drone targets. Finally, Chapter 8 concludes this thesis.

2 Literature Review

*This chapter presents the literature review performed for this thesis, mostly related to the use of Multi-Task learning for UAV parameter estimation using radar data. **Section 2.1** covers the advancements in classifying UAVs in their environment of operation. **Section 2.2** presents the current state of UAV parameter estimation in the literature. **Section 2.3** introduces the concept of Multi-Task Learning (MTL) alongside its usage in radar applications found throughout the literature, while **Section 2.4** concludes this chapter.*

2.1. UAV classification among other targets

The problem of using radar data to classify UAV targets among other, flying or not, targets in their environment of operation has been well studied throughout the literature. The research works found regarding this topic mainly revolve around the use of spectrograms as the basis for classifying drones in the presence of other targets. Fewer works additionally examine the use of other data representations like cepstrograms or Cadence Velocity Diagrams (CVDs). In all these works, a wide use of both Machine Learning and Deep Learning methods can be found.

The aforementioned data representations can be used as input to handcrafted feature extractors, like the one proposed in [12]. In their work, Molchanov et al. after filtering and aligning the micro-Doppler signature of the target to compensate for the Doppler shift inevitably caused by the target's body motion, use the eigenpairs extracted from the correlation matrix of the signature as features for classification. Testing the approach on real measurements proved the validity of the approach by providing an accuracy of the order of 95%.

Additional classification works based on handcrafted feature extractors can be found in [13-15]. The authors of [13] examine the use of mainly statistical features extracted solely from spectrograms, them being the mean, standard deviation and entropy of the whole spectrogram, the mean skewness and kurtosis of the centroid and bandwidth of the spectrogram and three coefficients of a cubic polynomial fit to the slope patterns of the drone body. In [14] the authors propose, among others, a series of features extracted from various stages of radar data processing like track age, position, velocity, acceleration, range rate, Doppler deviation, normalized amplitude, and number of Doppler components. They employ a Decision Tree classifier which performs well on the input data, classifying 93.33% of the drone targets correctly. In [15], a plethora of kinematic features alongside some statistical RCS related features are employed. The authors conclude that the mean of the acceleration, the variance of the jerk and the mean of the peak amplitudes of the range-Doppler plots are the most descriptive features for a classification task of UAV vs non-UAV tracks.

Deep Learning has a prominent role in spectrogram classification as well. As an approach it has been explored throughout the literature in many cases, such as [16]. There, Park et al. employed an FMCW radar with which they performed measurements of both UAV and human activity. Using a custom modification of the original ResNet-18 network named ResNet-SP, the authors manage to improve accuracy and reduce training time comparing to the original ResNet-18 network. The importance of the spectrogram and Micro-Doppler, two concepts closely related to each other, is also stressed in [17] where the authors address two problems: target detection, i.e. distinguishing small drones from birds, and target classification, i.e. distinguishing between different types of drones. Using a custom designed CNN network, Björklund and Wadströmer reported a very high probability of detection $P_D \approx 99.97\%$ and a much-improved accuracy compared to non DL methods for the type classification of 98%.

Although the classification of drones among their environment is not the main priority of this work, useful conclusions may be drawn from the literature regarding the challenges of classifying spectrogram input using DL. For instance, [18] performs a comparison among various state-of-the-art DL models like AlexNet[19], Squeezenet [20], two Resnet [21] variants (namely the Resnet-18 and Resnet-50), the GoogLeNet and the Inception-v3 [22].

The conclusion of the study was that shallower models such as the AlexNet can be more robust to noisy training data than their deeper counterparts. Additionally, as expected, the performance of all models drops considerably alongside Signal to Noise Ratio (SNR) degradation. A useful and important point made by the authors, oftentimes not stressed enough, is that the assessment of these models was made with fairly limited data and, in this case, using only a single drone model during the measurements.

2.2. UAV parameter estimation and classification

Parameter estimation regarding UAVs using radar has been a compelling topic throughout the literature with a variety of approaches and techniques. As in the problem of UAV classification versus other targets, the micro-Doppler signature may again be utilized in order to characterize drone targets with respect to their key parameters such as the number of rotors, their lengths, their rotation rates and others [23]. This can be seen, though, as a challenging task depending on the unpredictable variations of the rotation rate of the rotors during flight. A technique that seems to stand out in UAV parameter estimation is the extraction of the Helicopter Rotation Modulation (HERM) lines, a phenomenon analogous to the one of Jet Engine Modulation (JEM) [24, 25]. HERM lines can be used as a basis for further feature extraction, or they may provide discriminative information themselves.

In recent times, Ren et al. [26] proposed an algorithm to extract HERM line features from spectrograms of UAV targets based on peak detection at each Coherent Processing Interval (CPI). Both synthetic and real measurement data was utilized, showing the challenges induced by using the latter. The parameters estimated in this work were the main body's Doppler frequency, the number of rotors of the drone target, their length and rotational velocity. Even though the velocity characteristics of the drone body and rotors were captured at a satisfactory level, larger deviations were spotted on the length of the rotors estimation. In every case, the number of rotors was correctly identified. The authors point out the differences in continuity and overlapping of the HERM lines present in the radar signature, when produced by real measurements versus the simulated cases, a fact that makes the correct estimation on real data a far more challenging task.

Spectrograms and their Micro-Doppler properties are, as expected, not the only means of drone characterization. As shown in [27], the cepstrogram obtained by performing an additional Inverse Fast Fourier Transform (IFFT) to each spectrogram's time bin [28] can also provide valuable information in the case of determining the rotation rate of a drone target's rotors, particularly in the case of long integration times. In their study, Harmanny et al. study the additional use of cepstrogram for the number of rotor and rotation rate estimation based on their periodicity. In 2017 Fuhrmann et al. [29] expanded on the initial task of classifying drones against birds by additionally putting effort on using the cepstrogram in order to determine the drones' number of rotors, blade flash frequency, rotation rate and blade length. The conclusion on this is that the cepstograms can indeed be used towards this objective with acceptable accuracies. Yet, it is mentioned that the detection of the rotors is highly dependable on the SNR and/or the data acquisition geometry.

Deep learning has also been used for the same objective of UAV parameter estimation. In [9] Ahmad et al. used convolutional networks alongside measurements from an L band radar in an attempt to estimate the drones' rotor flash rate. They achieve this by proposing a Regression Short-Dwell CNN (RSD-CNN) model using, as its name suggests, estimates generated with as little latency as possible. The authors employ both synthetic and real measurement data, the former providing ground truth values, while for the latter optical validation is utilized. The RSD-CNN model provides a good Root Mean Square (RMS) error of 5.1 Hz while the optical validation in the real measurement case seem adequate enough.

The existence of payload mounted on a drone can be considered another parameter to be predicted. In [13], the authors employ the aforementioned features as a means towards identifying commercial drones carrying small payloads. To simulate the existence of payload, AA batteries were attached to the drone's arms. Using twelve different features extracted from the spectrograms, the quadratic SVM approach for spectrograms of five seconds duration proved to perform the best. Yet, it shall be mentioned that the metric used is the accuracy, which is not

an ideal metric for such a scenario where the false alarms and misdetections can be weighted according to the application. It shall also be noted that the classifier had a quite large number of false alarms at 31.5%. Further work on classifying unloaded versus loaded drones, this time using a multistatic radar can be found in [30, 31]. There, the authors attempt to distinguish among hovering drones carrying different payloads (namely: no payload, 200g and 500g payload) by exploiting features related to the centroid of the micro-Doppler signatures. Using two types of classification methods, the naïve Bayes and the diagonal-linear variant of the discriminant analysis, Fioranelli et al. showed that the extracted features provided with a classification accuracy consistently over 90%.

In [32] the authors deviate from the use of spectrogram and its micro-Doppler properties and propose the use of short records of the complex valued base-band temporal signal, extracting time and frequency domain features. They perform classification of number of propellers and number of blades, while performing regression tasks for estimating the blades' lengths and rotation rates respectively. The proposed classification model is a relatively simple Multi-Layer Perceptron (MLP) accompanied by a decision flow structure taking into consideration the output of other classification tasks. By training and testing on synthetic data only, the authors had the opportunity to also examine the behaviour of the classifier in different SNR scenarios. It is worth mentioning that the maximum error for the blade length was found to be 3 mm at SNR values larger than 15 dB.

It is clear from the above that DL methods have found their way in radar applications during the recent years. These applications range from drone detection (i.e. classifying drone targets among others in their environment) to estimating rotor rotation rates and detecting the presence of payload mounted on such drone targets. A common element among all the analysed research works shown previously is that all classification or regression tasks are treated by defining single, purposefully designed, DL architectures. As will be made clear in the following sections, there might be some additional performance gains when combining such tasks in a joint training manner.

2.3. Multi-Task Learning

Based on the aforementioned discursion, it is suitable in this section to introduce the concept of Multi-Task Learning (MTL) as a means of improving models' training and performance. It is seen as the norm in Machine Learning to address complex problems, or tasks, by segmenting them into smaller and preferably independent problems [33]. These smaller problems can then, consequently, be addressed by determining independent ML or DL models that are trained and optimized for each of these smaller tasks in hand. This single-task approach that ends up with combining the outputs of these individual models is referred to, and will be referred to for the rest of this work, as the Single-Task Learning (STL) approach. Figure 2.1 presents the configuration of such an approach. STL, as reasonable as it may sound, has been argued even quite early, in works such as Waibel et al. [34], to be counterproductive in some cases since it discards useful information that could be exploited by other tasks of the same general problem. In simpler words, feature representations that are produced and possibly discarded by one task might be useful in predicting another.

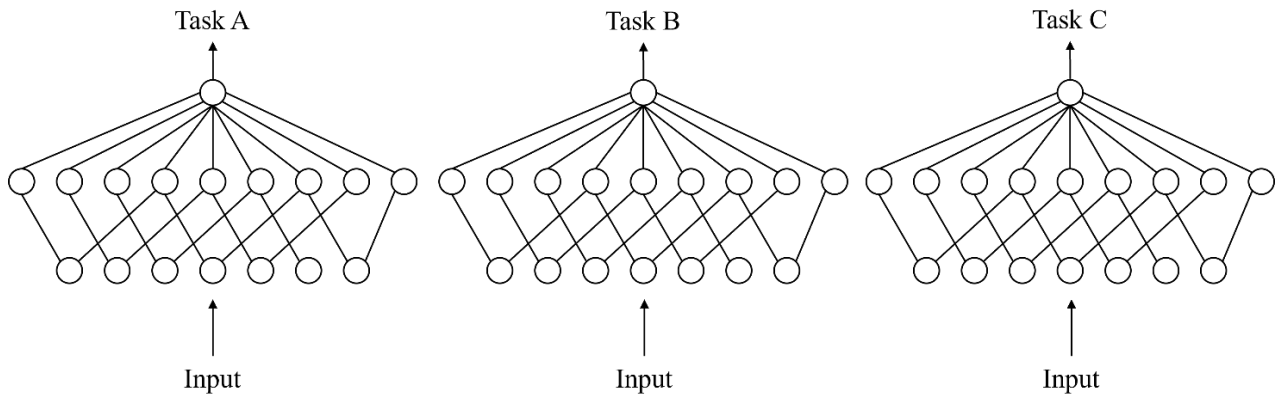


Figure 2.1: Three single task learning models, each dedicated to one task. All three models receive the same input but predict different tasks, yet possibly related to one another.

For this reason, MTL was developed on the assumption that if tasks can share knowledge among them, then the learning model may learn more easily and generalize in a better manner compared to its isolated STL counterparts.

Multi-Task Learning is not a new concept in the literature. The first, primitive versions of MTL can be traced back to 1990 in [35, 36], expressed as “hints”. In these works, the central idea of MTL was explored, sharing what is learned by different tasks while training in parallel. Rich Caruana in his seminal work on MTL [33, 37] describes it as “*an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias. It does this by learning tasks in parallel while using a shared representation; what is learned for each task can help other tasks be learned better*”. Figure 2.2 depicts in a simplified sketch the paradigm of Multi-Task Learning.

As will be discussed further in later subsections, the advancements in Deep Learning influenced MTL as well, providing many improvements over the original concept. Hence, survey and overview papers such as [38], [39] and [40] provide with useful reviews on the usage of MTL, as well as different approaches to developing such models, most of which will be reviewed in the following subsection. From this review, it is concluded that the main families of MTL models are the so-called Hard Parameter Sharing (HPS) and Soft Parameter Sharing (SPS) approaches.

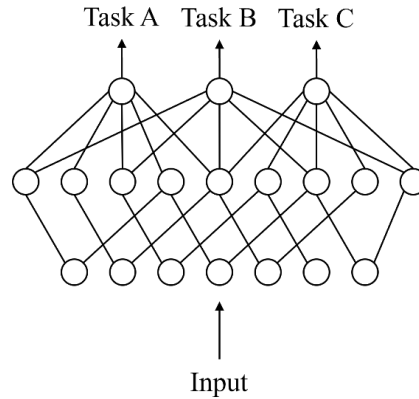


Figure 2.2: Example of usage of the Multi-Task Learning paradigm: a single neural network model predicting three tasks using a shared backbone network.

In the first case, a common feature extractor is used, usually comprising of convolutional layers. Then, individual branches depending on the number of tasks are created, as can be seen in Figure 2.3a. On the other hand, SPS approaches involve the co-existence of multiple per-task models that are trained in a joint manner by exchanging information (i.e. activation maps or weight values) among them. This configuration, in contrast to the HPS approach, can be found in Figure 2.3b.

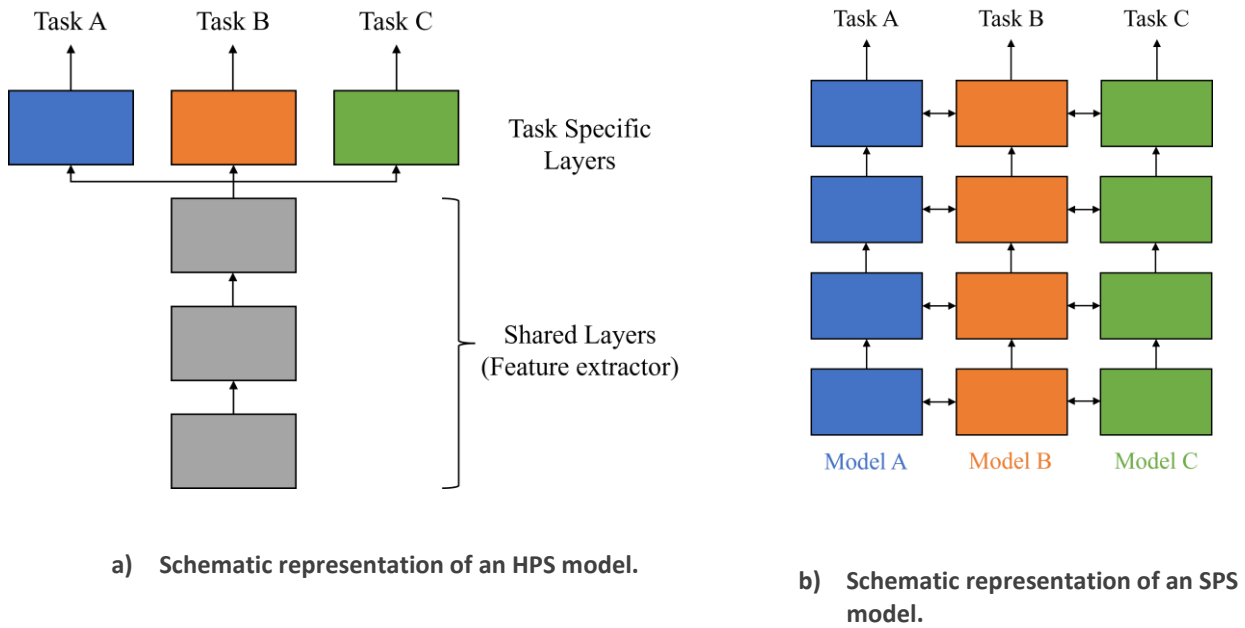


Figure 2.3: The two main families of MTL: the Hard Parameter Sharing (left) and the Soft Parameter Sharing (right) approaches. Note that in the SPS approach, three individual models are created.

2.3.1. Review of the state-of-the-art in MTL

A plethora of research works have directly adopted HPS as an approach towards performing MTL. For instance, Zhang et al. [41] proposed a Tasks-Constrained Deep Convolutional Network (TCDCN) to jointly optimize facial landmark detection with a set of related tasks. TCDCN retains, in its essence, the same configuration as seen in Figure 2.3a above. It consists of a feature extraction unit, common for every task, followed by the branching dictated by the defined tasks. The main task in this case is face landmark detection accompanied by other relevant tasks such as recognizing the pose of a person, their gender, whether they are smiling and wearing glasses. An interesting aspect of their work is that they address a common issue in MTL, that is the fact that different tasks have different learning challenges and convergence rates. To counter this, the authors propose a task-wise early stopping approach. It was then shown that the proposed TCDCN network, using the aforementioned early stopping scheme, managed to provide more robust landmark detection by training on multiple heterogenous but subtly correlated tasks.

Expanding on the notion of Hard Parameter Sharing, Long et al. [42] proposed the so-called Deep Relationship Network. There, following the shared convolutional layers are the task specific fully connected ones. The novelty of this work is that matrix priors are placed on the fully connected layers, that allow the model to learn the relationship between the tasks. Yet, as mentioned in [38], this method is still reliant on the existence of a known sharing structure and might be inadequate for novel applications such as the one addressed in this work.

One of the interesting research questions introduced by HPS is the following: Where shall one split the HPS model, and how much sharing is enough? It was this question, among others, that inspired Misra et al. [43] to develop the idea of Cross-Stitch networks. Dictated by the paradigm of SPS, in their work the authors propose the initial development of individual networks for each task. These networks are stitched together after certain layers leading to information being exchanged among them. More specifically, as can be observed in Figure 2.4, the input to layers after stitching is a linear combination of the outputs of the previous layers, and A and B define the corresponding task layers. The act of stitching is dictated by (2.1), where (i, j) point at specific parts of the activation map:

$$\begin{bmatrix} \tilde{x}_A^{ij} \\ \tilde{x}_B^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} x_A^{ij} \\ x_B^{ij} \end{bmatrix} \quad (2.1)$$

There, it is made clear that the aforementioned linear combination of the activation maps is parametrized by α (α values vary from 0 to 1), a parameter that is learned during training.

Hence, in this case, if the diagonal α is set to 1, meaning that $\alpha_{AA} = \alpha_{BB} = 1$, then the two networks discard the information of one another at that particular layer, corresponding to no joint learning. In this manner, the issue of splitting HPS networks is dealt with. The authors show that the Cross-Stitch networks perform and generalize better than both STL and MTL baseline methods.

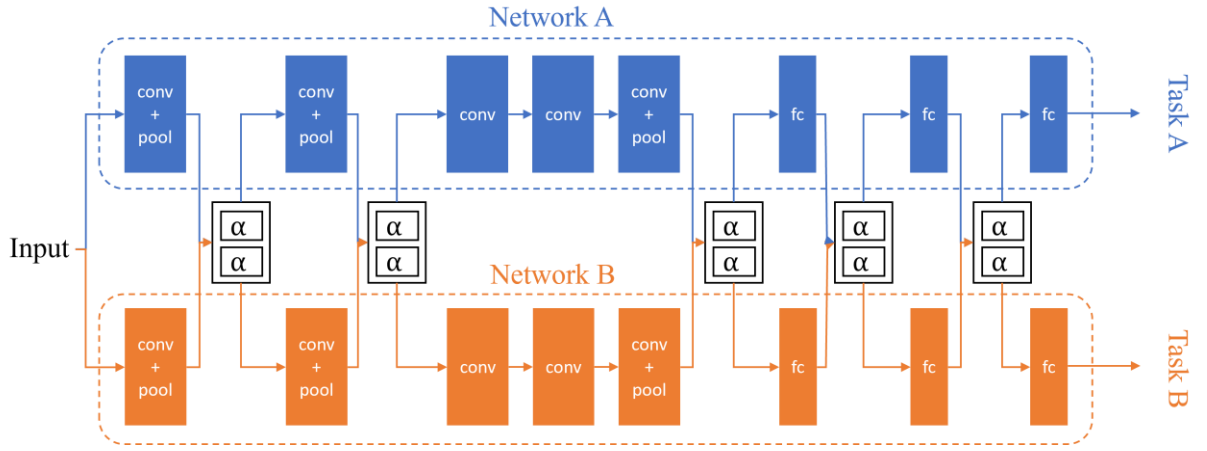


Figure 2.4: Schematic representation of the Cross-Stitch network paradigm for MTL, introduced by Misra et al. [43] (Figure adapted from the same paper).

Ruder et al. in [44] built upon the notion of Cross-Stitch networks by introducing a novel meta-architecture named “sluice networks”. This new approach encompasses a variety of approaches, such as HPS, SPS and cross-stitch units. Their approach enables learning of what layers should be shared and with what weighting, while also offering a number of skip connections at the model’s outer and final layers.

However, Soft Parameter Sharing is not restricted only in sharing activation or feature maps among the individually defined networks. The authors in [45] tackled the question of task relatedness by proposing a Dynamic Multi-Task CNN (DMT-CNN) model. This MTL model, composed of multiple per-task CNN models stitched with Task Transfer Connections (TTC) leads to the tasks in hand forming weak groups spontaneously through training progress. In the same manner as in the Cross-Stitch networks, when communicated information among the networks is ignored, each subnet works independently as an STL model.

2.3.2. MTL in radar

To the best of the author’s knowledge, very few works may be found that employ MTL as means to perform joint classification or parameter estimation in the literature. The existing works mainly seem to belong to the areas of classification or recognition regarding Synthetic Aperture Radar (SAR) data and Human Activity Recognition. While at the time of this thesis’s writing there are no documented works on improving classification of UAVs

using MTL in the context of drone characterization with radars, there are a few works where the aim is the improvement of detection performance. It is also worth noting that in all of these works, the number of tasks to be predicted is limited to two, or maximum three. It is furthermore notable that common ground in all these works is the preference towards hard parameter sharing approaches, with this being the most common MTL method among this subset of the literature. The main works in the literature are now briefly discussed.

In [46], the authors propose the use of Multi-Task Learning towards improving target recognition in targets acquired using SAR. To that end, three distinct tasks are defined, namely: separation of the target from the shadow, target's aspect angle estimation, and target recognition. Following an HPS mechanism for Multi-Task Learning, the authors consider the target recognition as the main task and the rest as auxiliaries, meant to improve the performance of the former. The authors manage to prove the benefits of Multi-Task Learning by performing comparisons on STL versus MTL in varying number of samples. In all cases, training both auxiliary tasks alongside the main one resulted in considerable improvement in target recognition accuracy.

In the same area of SAR target recognition, Chen et al. [47] combined both the concepts of meta and Multi-Task Learning and designed a Multi-Task Meta-Learning Recognition Algorithm Framework (MMRAF) alongside a variable combinatorial loss to mitigate the network overfitting problem while extracting more discriminative features for the mainline task. This work was carried out in the context of severely limited samples. Once again, N auxiliary meta-tasks are constructed, each one consisting of a main recognition and an auxiliary task. As seen in Figure 2.5, the main and the auxiliary tasks all share the same Resnet-18 backbone network, leaving only a fully connected layer as a per-tasks unique layer. The application of the proposed MMRAF framework on the publicly available MSTAR dataset containing SAR images of military targets showcased the ability of the framework to generalize better and achieve superior performance, compared to other state of the art methods.

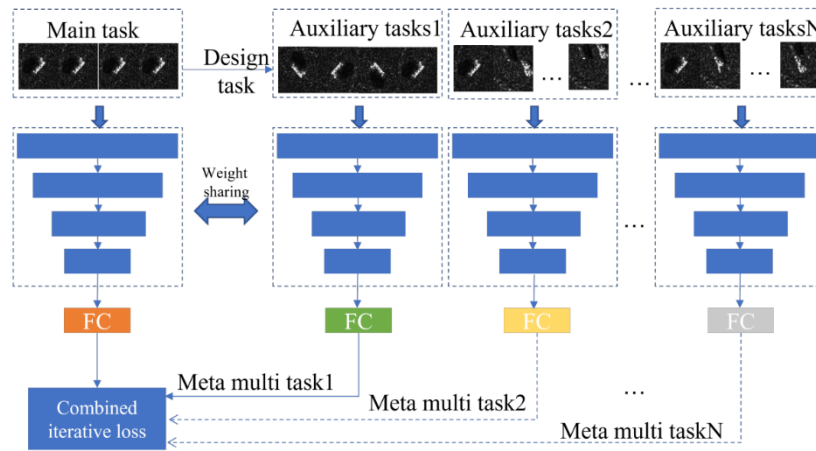


Figure 2.5: The MMRAF network presented in [47] (figure adapted from the same paper).

In the domain of Human Activity Recognition, Li et al. [48] propose a deep multi-task network capable of predicting human activity and also perform person identification simultaneously using the Motion Capture (MOCAP) dataset provided by Carnegie Mellon University. The authors achieve this joint classification task by introducing a custom convolutional Multi-Scale Residual Attention Network (MRA-Net). This network is composed of two major parts, the feature extractor and the Multi-Task classifier. The feature extractor part seen in Figure 2.6 utilizes two different kinds of kernels within the same blocks, one being 3×3 for fine scale-learning and the other 5×5 for coarse-scale learning. After extracting the required features from the input spectrogram, the Multi-Task classifier part of MRA-Net is used for performing the two aforementioned tasks. The number of branches in the Multi-Task classifier part is defined to be two, matching the total number of tasks. To make the

training of the model possible, the authors define the overall loss as a weighted sum of the corresponding losses of the two tasks. These weights are then defined by a greedy search approach based on the accuracy achieved by the models for each task separately. It is shown in their results that the MTL approach provided an improved accuracy score for both tasks, more prominently in the one of person identification where it surpassed the STL approach by 2.54%.

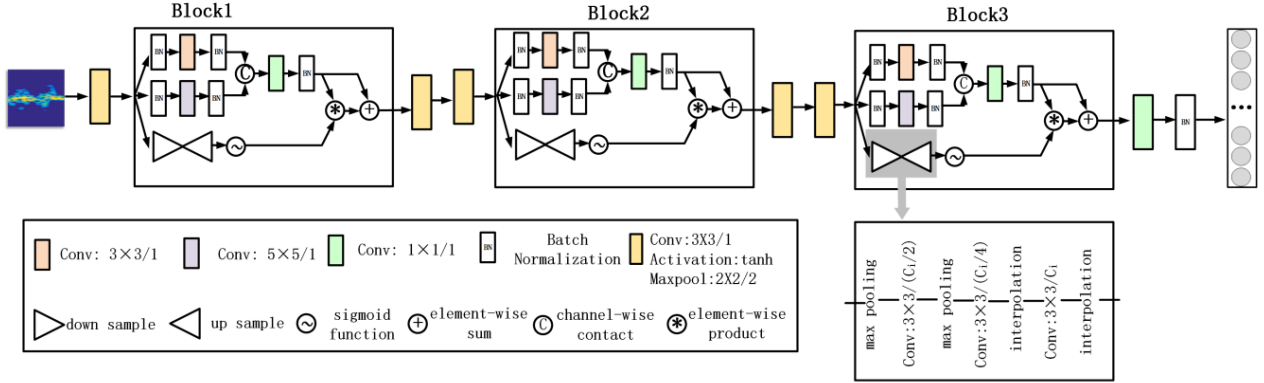


Figure 2.6: Schematic representation of the MRA-Net network's architecture, as described in [4] (figure from the same paper). Notice the three different branches used for fine and coarse scale learning, as well as the residual attention learning.

In the drone detection context, MTL has been proven successful in works such as the one by Wang et al. [49]. Inspired by the advancements of MTL in image recognition, the authors propose the use of a Multi-Task Learning model as an alternative of the widely known and studied Constant False Alarm Rate (CFAR) detector algorithm. The authors propose an MTL model predicting two tasks simultaneously with a shared backbone, leading to a HPS model with two heads: one for classifying the input range-Doppler map patch into target present or target absent, and the other for the regression of offset between the target and the patch center. The results of this work were very promising for the use of Multi-Task Learning for drone target detection. Performing experiments on both real measurements and simulations showed that the proposed method was able to locate the target more accurately and achieve a much lower false alarm rate at a comparable detection rate than a CFAR detector.

An effort towards drone characterization using MTL and Radio Frequency (RF) signals as input is shown in [50]. There, the authors employ a three-headed HPS model for three different tasks: drone detection, type identification, and activity recognition. For the drone detection task, a binary classification task is defined according to whether the target is a drone or not. Type identification refers to a task responsible of identifying the drone model, whereas the activity recognition is seen as a ten-class classification task. Compared to other state-of-the-art methods, the drone detection task was not improved over the other STL cases (almost all of whom achieved perfect classification results), while increase in performance of 1.9% and 6.59% was shown in both the type identification and activity recognition tasks respectively over the second best performing model.

2.4. Conclusion

Deep Learning has been a proven method of dealing with the issue of target classification and parameter estimation regarding drones. A plethora of different methods and models have been proposed and tested, providing good performances in a wide range of drone characterization aspects (e.g. type of drone, rotor rotation rates etc.). However, most of the works studied throughout this chapter focus on facing one task at a time, hence not showing the potential advantages in joint training and testing of different drone characterization tasks.

Regarding the joint classification or estimation of different tasks, the MTL paradigm has been proposed in the Deep Learning scene as an effective way of training Deep Learning networks jointly. It has been found to be applied in various applications providing with considerable improvements in performance over using single task models. The advancements regarding MTL, the advantages of using it, and the overall increase in the exploitation of Deep Learning methods on radar data, regardless of the application, has made MTL an attractive option for improving the performance over single task models. In Table 2.1, the works using MTL on radar applications that were discussed in this chapter are presented.

It can be concluded from the literature review presented in this chapter, that although MTL has appeared in the radar scene, it has yet not been explored and utilized towards drone characterization based on spectrograms. A question that remains to be answered in the existing literature would be whether the advantages seen in using MTL in other applications can be also present in the drone characterization problem, where the tasks could be for instance the wing type and the number of rotors of the drone among others.

Application	MTL type	Number of tasks	Type of tasks	Reference
SAR target recognition	HPS	3	One segmentation, an angle estimation and a target classification task	[46]
SAR target recognition	HPS	3	Three sets of meta tasks composed by a classification and an estimation task each	[47]
Human Activity Recognition	HPS	2	Two classification tasks, one for person recognition and one for gesture classification	[48]
Drone detection	HPS	2	Two tasks, a classification and a regression one are found in the neural network part of the algorithm	[49]
Drone characterization using RF signals	HPS	3	One binary classification, one type classification and an activity recognition task	[50]

Table 2.1: Summary of radar related works employing the MTL paradigm using Deep Learning found in the literature.

3 Radar & Neural Networks

Fundamentals

*The purpose of this chapter is to serve as a short introduction to the two main pillars of this thesis: the radar sensor, more specifically its Continuous Wave (CW) variant, and the Deep Learning (DL) tools employed in this work. In **Section 3.1**, a short introduction to radar is provided, mainly focusing on CW radar and its Doppler processing outcomes used extensively in this work. **Section 3.2** is devoted in providing a background on relevant Deep Learning techniques, by focussing on the building block of this work, namely the ResNet neural network.*

3.1. Radar as a sensor

Radio Detection and Ranging, commonly referred to as radar, is a concept that bases its operation on the transmission and reception of electromagnetic energy. In short, a radar's goal is to radiate electromagnetic energy and then detect the echo returned by reflection on objects or targets of interest [51]. During the years since its inception, different types of radar have made their appearance. Based on their working principles these are mainly categorized as:

- Pulse-Doppler radar
- Continuous Wave (CW) radar with its frequency modulated version (FMCW)

The main difference between these two types of radar can be traced in the manner through which the electromagnetic (EM) radiation is emitted. In the CW radar case, an unmodulated signal is continuously emitted typically as a single tone. On the contrary, a pulse-Doppler radar emits short pulses followed by a silent period where the transmitter is turned off. This listening period is used to perform the necessary processing to determine the range and velocity of the targets.

Since the radar used in this work is a CW one, the rest of this section is devoted to the description of the radar's operation and its derivatives. The fundamentals regarding the CW radar can be found in Subsection 3.1.1, while the signal processing operations over its output are analysed in Subsection 3.1.2.

3.1.1. CW radar fundamentals

As mentioned earlier, the CW radar is a radar variant in which an unmodulated high frequency signal is transmitted continuously. Contrary to the pulsed radar systems, where a listening period always follows a transmitting one, CW radars emit radiation at all times using mostly a dual antenna configuration, one responsible for transmitting and one for receiving the reflected signal. It shall be clear that in such a configuration, with the transmitter continuously radiating, radiation leak to the receiver is almost guaranteed. To prevent this, sufficient spatial separation among the antennas has to be achieved or a circulator of sufficient good quality employed. A much simplified, yet sufficient for the sake of this thesis, block diagram of a CW radar can be found in Figure 3.1.

At this stage, it is necessary to analyse the basic components that constitute a CW radar as included in the block diagram:

- **Signal Generator**: the signal generator, also found in literature as an RF generator is responsible of producing a continuous signal at a single, very, stable frequency.
- **Power Divider**: to mimic the functionality of a local oscillator, a power divider can be used. Power dividers are devices able to divide the power of an RF signal into two parts with the same phase shift. One part is forwarded to the transmitter and the second one is used as input to the mixer, another crucial circuit component that is described below.
- **Low Noise Amplifier (LNA)**: in radar applications, the received backscattered signal is most of the times, if not always, very weak in power terms. The goal of an amplifier circuit in general is to ultimately increase the power of the input signal, in this case the received backscattered one. Since this amplification takes place in a very early stage, it is necessary to add as little noise as possible, since this noise will be amplified further in later stages. The *low-noise* part of the term refers to the fact that LNAs amplify the very low-powered input signal without severely degrading its signal-to-noise-ratio (i.e., this component has a low noise figure), something that is crucial for later target detection.

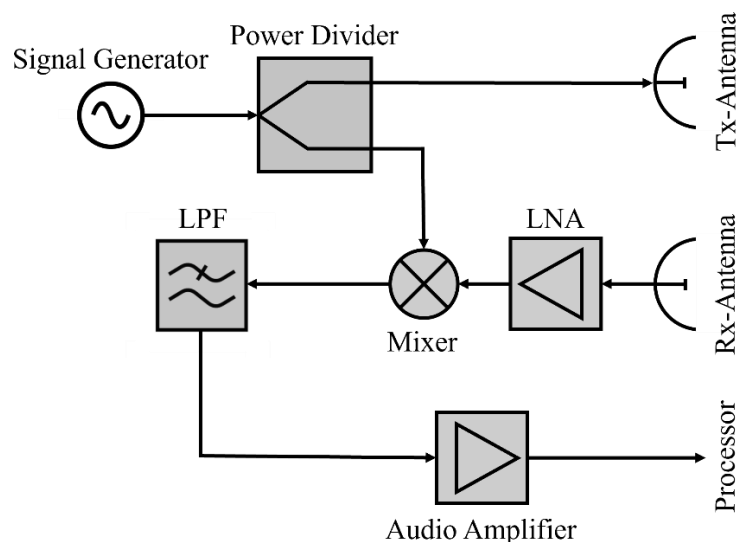


Figure 3.1. Example of a generic Continuous Wave (CW) radar block diagram.

- **Mixer**: a mixer, also referred to as a frequency mixer, is a commonly three-port passive or active electrical circuit that modifies the frequency of a signal based on a reference signal through a multiplication operation. In CW radar terms, that means the conversion of the echo signal to baseband for further processing.
- **Low Pass Filter (LPF)**: an LPF is another very common circuit used in radar systems. As its name suggests, the purpose of a low-pass filter is to discard/attenuate signals with frequencies higher than a pre-determined one.
- **Audio Amplifier**: even after all this processing, the received signal's power is still inadequate. Hence, further amplification takes place before final processing of the signal.

Among others, the block diagram of Figure 3.1 showcases one of CW radar's most important characteristics: its simplicity. Since the transmitted energy is not pulsed, it allows for several shortcuts in design, thus making the

CW radar a very simple and inexpensive option. Another important advantage is that this type of radar can maximise the time on target due to the continuous radiation from the transmitter.

Yet, CW radars come with severe limitations. One of them revolves around the continuous transmission and the maximum range of operation. CW radars may not have a maximum range of operation, yet the continuous transmission does not allow for high powered signals to be transmitted because of hardware limitations, especially the risk of leakage of power into the receiver chain.

The most serious limitation of the continuous wave radar is caused by the lack of modulation in the signal. Due to the fact that the signal that a CW radar transmits is unmodulated (in contrast to its frequency modulated counterpart, the FMCW radar) and with constant amplitude, CW radars cannot provide the user with range information, relying only on Doppler/velocity processing to distinguish among targets.

Continuous wave radars base their operations in measuring the Doppler frequency, a frequency modification to the original signal imposed by moving targets. In this manner, stationary and moving targets can be distinguished and separated. In order to do this, the CW radar evaluates the difference between the phase of the transmitted and received signals.

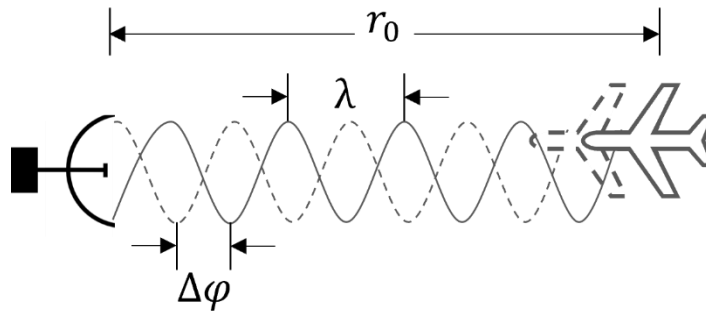


Figure 3.2. Simple sketch showing the phase difference due to the target's movement.

By definition, a target's non-zero and constant radial velocity v_r results in a time varying Doppler phase (see Figure 3.2):

$$\varphi_{\text{target}}(t) = 2\pi \frac{2r(t)}{\lambda} = 4\pi \frac{r_0 \pm v_r \cdot t}{\lambda} \quad (3.1)$$

where $r(t)$ is the progression of the distance between the target and radar with time t considering the initial radial distance r_0 and λ is the wavelength. The instantaneous angular frequency, defined as the Doppler frequency $\omega_{\text{Doppler}}(t)$ in this case, is defined as:

$$\omega_{\text{Doppler}}(t) = \frac{\partial \varphi_{\text{target}}(t)}{\partial t} = \pm 4\pi \frac{v_r}{\lambda} \quad (3.2)$$

Hence, the instantaneous Doppler frequency can be found by:

$$f_{\text{Doppler}}(t) = \frac{\omega_{\text{Doppler}}(t)}{2\pi} = \pm 2 \frac{v_r}{\lambda} \quad (3.3)$$

Considering the wavelength definition, $\lambda = \frac{c}{f_0}$, where c is defined as the speed of light and f_0 the operating frequency, the above equation reduces to:

$$f_{\text{Doppler}}(t) = \pm 2f_0 \frac{v_r}{c} \quad (3.4)$$

This Doppler frequency, also referred to as Doppler shift, is found to be additive to the transmitted frequency f_T as $f_R = f_T + f_{\text{Doppler}}$. The mixer device, as described earlier, is then responsible of essentially detecting this shift. As analysed before, the mixer receives as input two separate signals, the transmitted (after the power divider) and the received one. The two signals can be simply represented as follows:

$$S_{\text{Tx}}(t) = A_T \cos(2\pi f_T t) \quad (3.5)$$

$$S_{\text{Rx}}(t) = A_R \cos(2\pi f_R t) \quad (3.6)$$

Then, the mixer multiplies the two signals, giving the following output:

$$S_{\text{mixer}}(t) = A_T \cos(2\pi f_T t) \cdot A_R \cos(2\pi f_R t) = \quad (3.7)$$

$$\frac{A_T A_R}{2} (\cos(2\pi(f_T - f_R)t) + \cos(2\pi(f_T + f_R)t))$$

Filtering the above product using an LPF as described above provides with the desired frequency shift.

3.1.2. Micro-Doppler processing and related data formats

A key property of the targets included in this work is the Micro-Doppler effect that their moving components produce. When dealing with drone targets, such as the ones dealt with in the scope of this thesis, it is not only the main body of the structure that causes the aforementioned Doppler shift.

While UAVs operate, additional Doppler components are produced due to the rotation of their one or more rotors. These additional Doppler components are oftentimes referred to as Micro-Doppler effect. Introduced by Victor Chen in 2006 [52], the Micro-Doppler effect is defined as “*the additional Doppler modulations to the constant Doppler frequency shift, caused by the bulk motion of a radar target, which are induced by other micro-motion dynamics such as vibrations and rotations*”. It is, thus, important to describe this phenomenon in a rigorous mathematical manner.

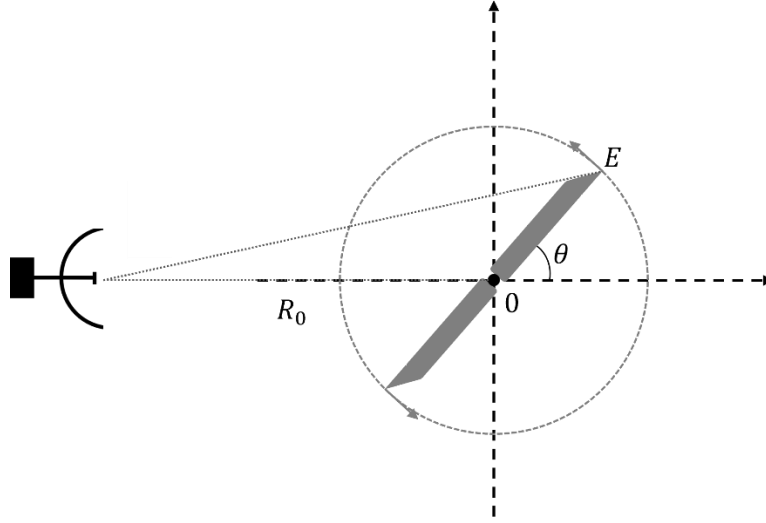


Figure 3.3 A scenario regarding the rotation of a drone/helicopter rotor to study its Micro-Doppler signature.

In a rotation scenario, such as the one depicted in Figure 3.3, considering the length of each blade as L and $R(t)$ as the radar's range from the top blade's edge marked as E , the Micro-Doppler component for a simplified point case can be calculated having as starting point the varying range of point E to the radar $R(t)$. This is calculated by exploiting the Pythagorean theorem as:

$$R(t) = \sqrt{(R_0 + L\cos(\theta))^2 + (L\sin(\theta))^2} \approx$$

$$\sqrt{R_0^2 + 2R_0L\cos(\theta)} = R_0 \sqrt{1 + \frac{2L\cos(\theta)}{R_0}} \quad (3.8)$$

Here, it is safe to assume that the distance R_0 will be always considerably larger than the length of the UAV's blade, L . In this manner, based on the approximation that $\sqrt{1+x} \approx 1 + \frac{x}{2}$, the varying range of the tip of the blade to the radar becomes:

$$R(t) = R_0 + L\cos(\theta) \quad (3.9)$$

Of course, in rotation scenarios like the one examined here, the angle θ is not constant and it varies in time according to some angular frequency ω_v , effectively making it a function of time as $\theta(t) = \omega_v \cdot t$. Borrowing the derivation of the Doppler frequency from (3.4), one can obtain the Micro-Doppler frequency component of this rotation as:

$$f_{MD} = \frac{\partial \varphi_{\text{target}}(t)}{\partial t} = -\frac{2f}{c} L \omega_v \sin(\omega_v t) \quad (3.10)$$

Now, this model that is depicted above is a very simplified version of the Micro-Doppler shift induced by, essentially, a single point scatterer's rotation. To make the model more realistic, let us consider that each of the N blades with rotor number $k = 0, 1, \dots, N - 1$ can be defined and modelled as a series of point scatterers. Additionally, consider the elevation angle β which is defined as the angle between the radar and the rotation plane of the target's rotor, an initial rotation phase of the blade φ_0 and the rotation rate of the blades, Ω . The Micro-Doppler in this case is calculated as:

$$f_{MD} = \frac{\partial \varphi_{\text{target}}(t)}{\partial t} = -\cos(\beta) \frac{L}{\lambda} 2\pi\Omega \left[\sin \left(2\pi\Omega t + \varphi_0 \frac{2k\pi}{N} \right) \right] \quad (3.11)$$

The continuous wave radar, as described in Subsection 3.1.1, cannot provide range information due to its waveform design. It was also mentioned extensively in Section 2.1 that Doppler processing is the key element in detecting and distinguishing among targets. What follows is a short description of the most important and commonly used signal processing approaches used for both Doppler and Micro-Doppler processing.

The most well-known and used method for obtaining the frequency representation of the received signal is the Fourier transform (FT). An important assumption that is oftentimes made when using FT is that the frequency content of the signal is stationary, that is that it does not change during the observation time. The issue with this assumption is that it is not applicable to many practical scenarios. Take for example the drone observation using radar. During its flight, a UAV will most likely perform a variety of manoeuvres, correctional to their flight paths or not. These manoeuvres inevitably cause a variation in both Doppler and Micro-Doppler frequency shifts, since not only the whole drone moves with different radial velocities, but its rotors are also expected to adjust their rates (i.e. rotational velocities) accordingly in support of the corresponding manoeuvre. The Fourier transform is defined for a discrete signal as:

$$X_k = \sum_{n=0}^{N-1} x_k \cdot e^{-\frac{i2k\pi n}{N}} \quad (3.12)$$

where x_k is the discrete signal in time, N is the number of samples in the analysis window and k the discrete frequency. It can be concluded by (3.12) that by having signals with non-stationary frequency content the FT will be lossy in terms of frequency representation. To this end, it would be much more suitable to apply the Fourier transform over a segmented version of the time domain signal, with the assumption that the signal can be characterized as stationary in these shorter segments. This method of applying the FT is widely referred to throughout the literature as the Short Time Fourier Transform (STFT). In simpler words, the STFT can be seen as using a sliding window $w[n]$ so as to limit the transformation into short segments of the original signal. The STFT is computed as following:

$$\text{STFT}\{x[n]\} = X(m, k) = \sum_{n=-\infty}^{\infty} x[n]w[n - m] \cdot e^{-\frac{i2k\pi n}{N}} \quad (3.13)$$

where in similar fashion as previously, $x[n]$ is the discrete time domain signal, $w[n]$ is the chosen window and m denotes the position of the window. Taking the squared magnitude of the STFT gives the spectrogram of the signal, which consequently gives the variations of the frequency content over time. Each row of a spectrogram represents how a specific value of the Doppler frequency of the signal changes over time (so-called Doppler bins). Each

column represents what Doppler frequencies or velocities are detected at that given time and are referred to as time bins.

There are three main parameters controlling the resolution and the quality of the produced spectrograms. These are: the duration of the window, the type of the window and the overlap among two consecutive windows. The duration of the window, and thus the segment length, controls different aspects of the spectrogram's resolution. Consequently, a short window provides good temporal resolution but results to poor spectral resolution in Doppler and vice versa. As extensively analysed in [27], the correct choice of window length in STFT may reveal a wide spectrum of target's properties. In the case of fast moving events, such as the rotor blades of a UAV, a short integration time may reveal specific properties of the rotor's radial movement based on individual cycles of the blades. On the contrary, a longer integration time, and thus a longer time window, contains several cycles of the blades and the resulting spectrogram is dominated by the rotation rate of the blades causing modulation peaks, making the instantaneous radial velocity mentioned earlier no longer visible. The importance of the integration time has been thoroughly discussed in the literature, since the existence or not of HERM lines, an important factor in classification and even parameter estimation regarding drones, in the produced spectrogram highly depends on it [53]. The window type can be abstractly chosen, with the most usual ones being windows tapered towards zero. The effect that the selection of the window type has been a point of discussion in the past [54, 55] as different windows provide different spectral leakage performance, sidelobe level and resolution among others. Finally, the overlap between the windows serves as a smoothing feature for the spectrogram. Small values lead to coarser-looking spectrograms, where discontinuities in frequency from one segment to the following are visible. On the other hand, using overlap values of over 50% leads to such discontinuities being smothered and a more detailed spectrogram in general, at least to the human eye.

The spectrogram is not the only approach of acquiring information on the UAVs' rotors and their velocities. An additional source of information on rotation rates, number of rotors and their angular velocity could indeed come from an extra processing step on the pre-acquired spectrograms. The rotor's angular velocity is directly related to the spectrogram's periodicity at each time bin. Introduced by Bruce Bogert in 1963 [28], the so-called cepstrum is seen as a means of unravelling information about the rate of change in the spectrum. It is calculated, for each time bin, as the inverse Fourier transform of the logarithm of the estimated signal spectrum. In the context of a previously applied STFT, this is mathematically defined [27] as:

$$C\{x[n]\}(m, k) = |\text{IFFT}\{\log(|\text{STFT}\{x[n]\}|^2)\}|^2 \quad (3.14)$$

The free variable is called quefrency and can be seen as the inverse of the distance between successive spectral lines in the spectrogram. It is measured in seconds. A spectrogram-cepstrogram comparison for a helicopter drone target can be seen in Figure 3.4 below. There, the distinction among the two different rotors, the main and tail ones, is clearly visible.

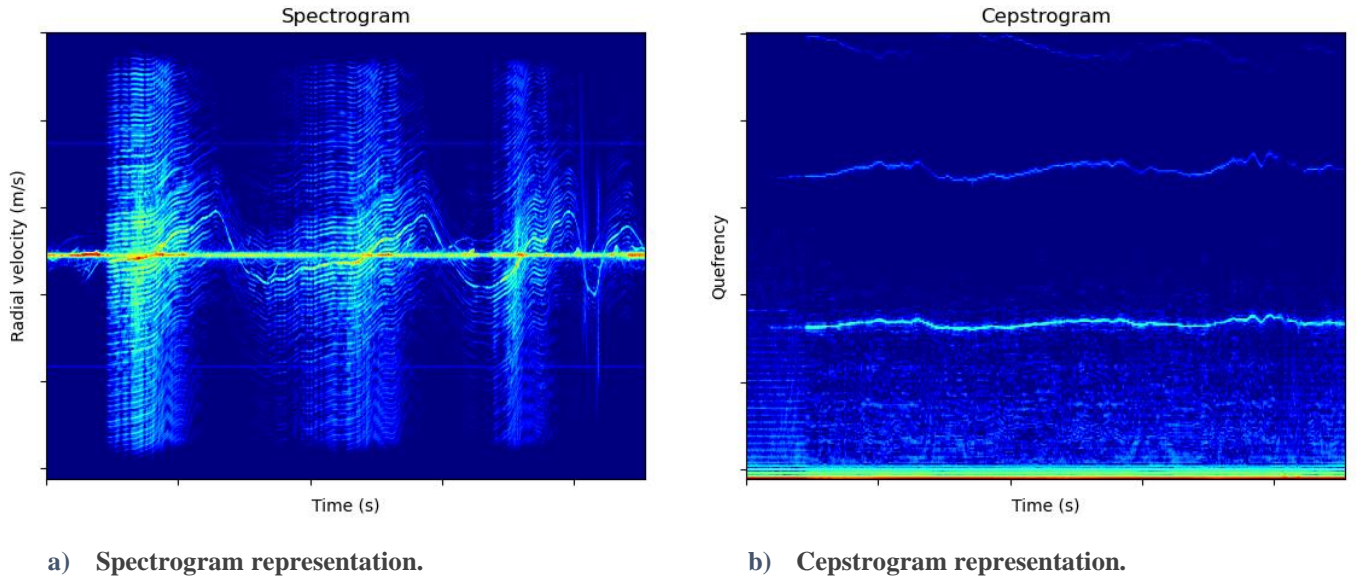


Figure 3.4. Spectrogram and cepstrogram representation comparison, both belonging to a helicopter drone present in the dataset analyzed for this thesis.

3.2. Deep Learning fundamentals

In this section, a short introduction to the Machine and Deep Learning tools used in this thesis is provided (in short, ML & DL, respectively). It should be noted that the aim of this section is not to offer a comprehensive explanation of all the topics and techniques in this vast field, but to put emphasis on the fundamentals of how neural networks are constructed and trained, in particular the ResNet network, the backbone of the methods presented in this thesis.

3.2.1. Neural Networks

The term neural networks refers to a subset of learning methods developed in a manner that resembles biological networks constituted by several neurons found in animal brains. In biology, a neuron is an electrically excitable cell that propagates electric signals in a cluster of such cells, connected into a network. In a similar manner an artificial neuron operating in a wider Artificial Neural Network (ANN) receives information from other neurons, processes it, and in its turn propagates the newly processed information to other neighbouring neurons. This process can be modelled in a fairly simple way as seen in Figure 3.5. There, $x_{i=0,\dots,N}$ can be considered as the input to the neuron, $w_{i=0,\dots,N}$ the corresponding weight to each input value and b the bias of the neuron. The neuron processes the input information in a weighted manner and propagates the output \hat{y} .

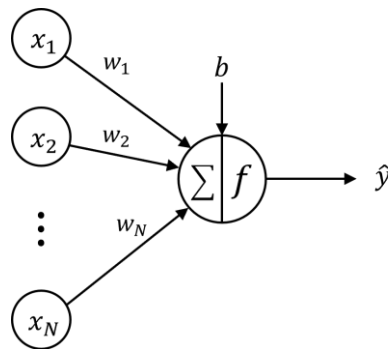


Figure 3.5. Schematic representation of a single neuron. Σ denotes the weighted summation of the input and f the activation function used to determine the output.

However, the output of the weighted sum has no use by itself. The role of an activation function, denoted as f in Figure 3.5, is to define the neuron's output in an appropriate (e.g. probability) form. These functions can be either linear or not, and some of the most widely used functions are the:

- Linear: $f(x) = x$
- Binary: $f(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$
- Sigmoid or Logistic: $f(x) = \frac{1}{1+e^{-x}}$
- Softmax: $f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}$, where K is the number of classes in a multiclass case
- Tanh or Hyperbolic Tangent: $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$
- Rectified Linear Unit: $f(x) = \max(0, x)$

Oftentimes an additional term is added to the sum of weighted inputs in the neuron. This term, named bias, can be defined as a constant which is added to the product of features and weights, inducing an offset to the result. This modification enables the networks to shift the activation function.

In view of the above, the output of a single neuron can be described mathematically as:

$$\hat{y} = f\left(b + \sum_{i=1}^N w_i x_i\right) \quad (3.15)$$

Yet, as expected, a single neuron is quite constrained in its ability to provide inference for different applications. To expand the capabilities of the single neuron, layers of stacked neurons can be introduced. Additionally, by increasing the number of layers in a multilayer perceptron [56] fashion and thus the depth of the network, better inference can be achieved since more difficult patterns may be identified. The input part of the network can be defined as the input layer, the stacked layers are commonly referred to as the hidden layers and the final layer of the network leading to the prediction is defined as the output layer.

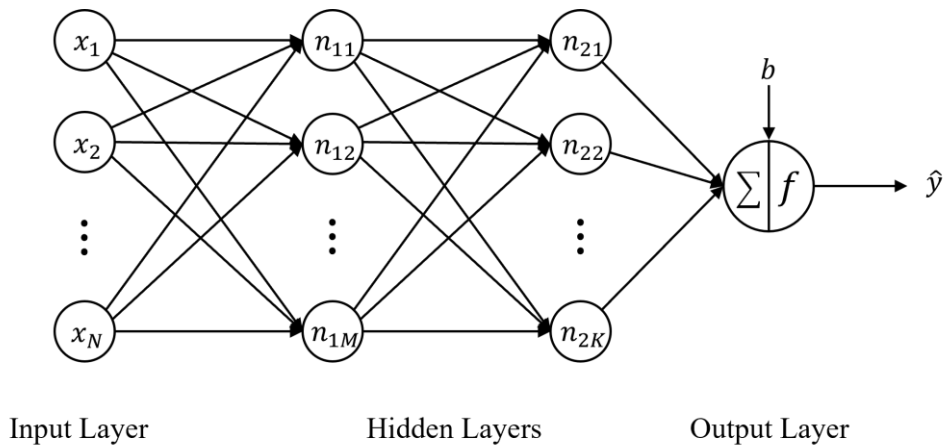


Figure 3.6. Example of a simple multi-layer ANN. This network has an input layer, two hidden layers and an output one.

Due to this expansion into multiple neurons and layers, a vector/matrix notation is preferred. Hence, the input layer is defined as \mathbf{x} , a $N \times 1$ vector. The weights of the first layer as shown in Figure 3.6 can be modelled as a $M \times N$ matrix \mathbf{W}_1 containing all weights of all the neurons of the corresponding layer ($M \times 1$ size vectors):

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_M^T \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MN} \end{bmatrix} \quad (3.16)$$

Note that each neuron contains also a bias term, this time represented also by a $M \times 1$ vector \mathbf{b}_1 . Thus the output of the first layer of stacked neurons according to Figure 3.6 is represented by:

$$\hat{\mathbf{y}}_1 = f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \quad (3.17)$$

Following the same train of thought, for the output of the second hidden layer, one gets:

$$\hat{\mathbf{y}}_2 = f(\mathbf{W}_2 \hat{\mathbf{y}}_1 + \mathbf{b}_2) \quad (3.18)$$

Finally, the output layer gives:

$$\hat{\mathbf{y}} = f(\mathbf{W}_3 f(\mathbf{W}_2 f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3) = f(\mathbf{W}_3 \hat{\mathbf{y}}_2 + \mathbf{b}_3) \quad (3.19)$$

However, useful as they are, these types of ANNs are unable to process image-like data, such as the data in this work, due to their inability to capture spatial characteristics over the input data. Consequently, a highly important component towards the classification of spectrograms in this work is the so called convolutional layer. Convolutional layers are a fundamental component of a class of Deep Learning models widely used for computer vision tasks such as image recognition and object detection, the Convolutional Neural Networks (CNNs). These layers play a crucial role in extracting meaningful features from input data, enabling the network to learn hierarchical representations.

As its name suggests, a convolutional layer is responsible of applying a convolution operation to the input data (most often 2D images, I) described mathematically by borrowing the notation of [57] for a 2D image I as:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n) K(i - m, j - n) \quad (3.20)$$

Using a small trainable filter called kernel, K , over the input in a sliding window fashion, the layer extracts local patterns or features from the input by performing element-wise multiplications and summations as seen in Figure 3.7. In this manner, the convolution layer captures the spatial relationships between neighbouring regions. Each convolutional layer consists of multiple filters each of which produces a two dimensional activation map known as feature map. These maps represent the learning patterns, or features, at different spatial locations of the input.

The number of filters in a convolutional layer determines the depth of the feature maps generated, thereby increasing the network's capacity to capture diverse features.

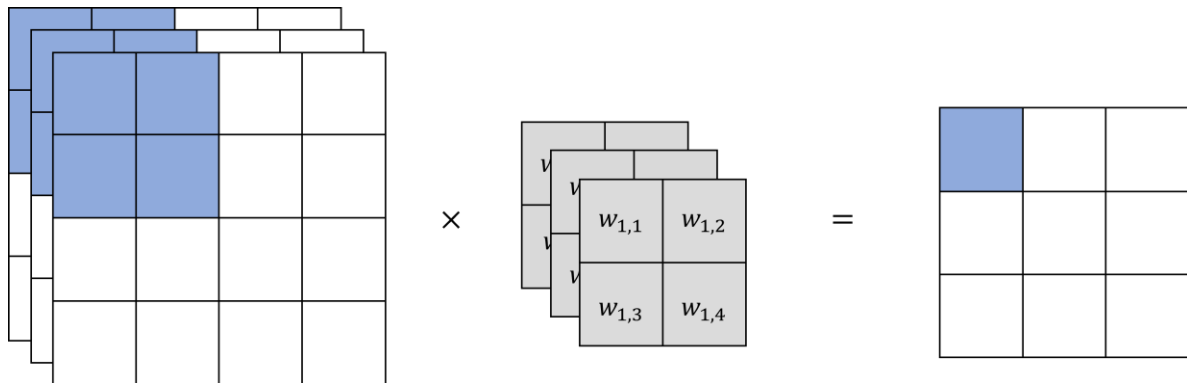


Figure 3.7. Example of convolution operation.

Even though Figure 3.7 implies the use of a square kernel, the shape of it can also be rectangular and optimized depending on the application. Further parameters that can be defined regarding the convolution operation are the number of filters to be applied, the step size of the filter (also mentioned as stride) and padding of the input among others.

An issue often identified in CNNs is the large number of parameters in the networks, something that is also regulated by the choices regarding the number of convolution filters, the depth of the networks and the types of layers used. One layer purposefully designed to reduce the spatial size of the representation, and thus the parameters of the network, is the so-called pooling layer. Defined as another filter, the pooling layer aims to reduce dimensionality by applying a predefined function over the area of the input space it is applied. The two most common pooling techniques are the max and average pooling shown in Figure 3.8.

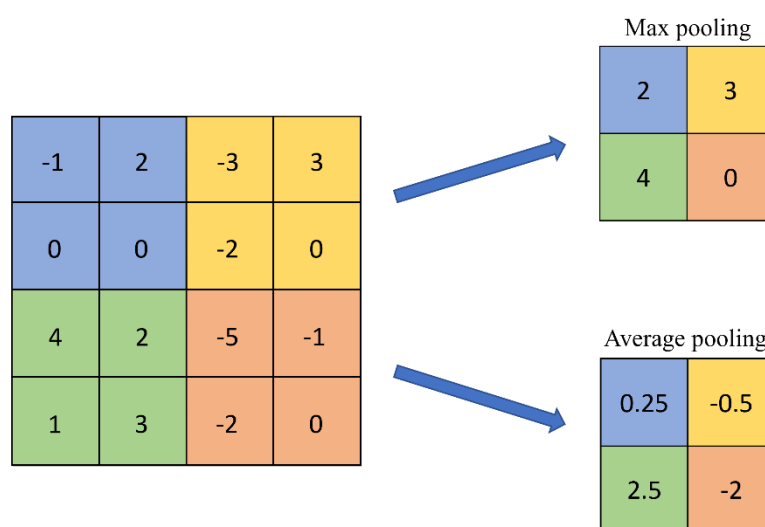


Figure 3.8: Example of max and average pooling.

In the above example, two 2×2 pooling filters of the aforementioned types are applied over a 4×4 input space with a stride of 2, leading to the 2×2 outputs on the right. As their names suggest, the max pooling filter selects the maximum value of the area that it is applied on while the average pooling filter obtains the average of the corresponding values.

3.2.2. The ResNet

As the ResNet is the building block of the MTL networks developed and tested throughout this thesis, it is deemed as important to analyse them structurally. A widespread problem in Deep Learning is the so-called degradation phenomenon. That is, increasing the depth of a network leads to a decrease in performance on both test and training data [58]. Introduced by He et al. [21], ResNet18 belongs to the wider family of Residual Networks. What differentiates ResNet models over other, plain, architectures is the presence of the so-called residual blocks, the application of which can be seen in Figure 3.9.

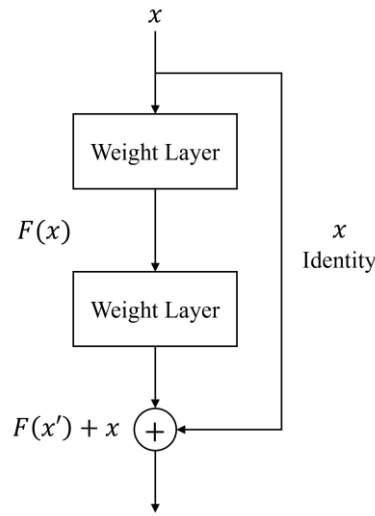


Figure 3.9: Schematic representation of a skip connection. The output of the second layer is combined with the input of the first.

In the aforementioned figure, the skip-connection is skipping two weight layers and directly passing the input, x , as the output. This is called a shortcut, or skip, connection because it does not involve any additional parameters as previous information is just forwarded to the layer. In simpler words, a residual block provides another path for data to reach latter parts of the neural network by skipping layers.

A detailed view of the Resnet18 architecture used alongside information on its trainable parameters can be found in Table A.1 of Appendix A. It shall be additionally mentioned at this stage that the aforementioned model comes with 11,689,025 trainable parameters. Moreover, through the Torchvision [59] package there is the possibility of using a pretrained version of ResNet18, by having its weights initialized such that they reproduce closely the results of the original paper.

Apart from the convolution layer, each block contains also a Batch Normalization layer, which applies Batch Normalization over a 4D input (i.e. a mini-batch of 2D inputs with additional channel dimension) [60]. The output of Batch Normalization is defined as:

$$y = \frac{x - E[x]}{\sqrt{\text{Var}[x] + \epsilon}} \cdot \gamma + \beta \quad (3.21)$$

There, the mean and standard-deviation are calculated per-dimension over the mini-batches. Additionally, the terms γ and β are trainable parameters, which introduction reassures that the transformation inserted in the network can represent the identity transform. In this manner, the output of Batch Normalization will not change what the corresponding layer represents.

As for the activation part, the Rectified Linear activation function, or ReLU, is a widely used and simple activation function. As dictated by the corresponding equation in subsection 3.2.1, ReLU will produce an output of 1, if $x > 0$ or zero if $x \leq 0$. The extensive use of ReLU lies in the fact that it is simple and does not require any heavy processing.

The reasoning behind the selection of the ResNet18 as the building block in STL models as well as the backbone for the MTL ones lies on the fact that with its residual blocks and skip connections, it has been proven as a successful model in a variety of applications, even when using radar data. Moreover, it provides ground for further experimentation and optimization over its depth as it can be easily managed by removing certain blocks of it. In a more practical view, other model types had been considered, with a good candidate being the AlexNet. Yet, as it was discovered, the pretrained AlexNet version provided by the torchvision package is implemented in such a way that full reproducibility of the results was not guaranteed among different runs. This, in combination with the limited dataset (as discussed further in the next chapter of this report) would make the convergence of the model different among different runs.

3.2.3. Training of Neural Networks

The procedure of training neural networks has as its goal the definition of suitable weights so that the network may learn patterns hidden among the provided data and increase its prediction capabilities. Hence training procedure of neural networks can be reduced to one of optimizing a certain figure of merit. In the context of optimization, this figure of merit is used to evaluate the calculated set of weights which lead to a certain outcome and is calculated with use of certain functions. This function is named loss, objective or error function depending on whether the goal is its minimization or maximization [57]. The loss function receives as input the predictions made by the network as well as the true values, or labels, corresponding to these sample instances. Depending on the nature of the task in hand, a variety of loss functions can be employed. Throughout this thesis the tasks examined are either classification or regression tasks. Hence, the three main loss functions employed are the Cross Entropy loss function, the Mean Squared Error (MSE) and the Mean Absolute Error (MAE).

The Cross Entropy loss measures the classification performance of a model whose output is a probability ranging from 0 to 1. It can be summarized mathematically as follows [57] (note that \mathbf{y} denotes the ground truth vector and $\hat{\mathbf{y}}$ the corresponding prediction vector):

$$\text{Cross Entropy Loss} = -\mathbf{y} \log(\hat{\mathbf{y}}) = - \sum_{i=1}^{N_{\text{class}}} y_i \log(\hat{y}_i) \quad (3.22)$$

The MSE is the average squared difference between the estimated values and the corresponding ground truth. Due to its nature, the MSE is more suitable for tasks dealing with continuous values as their targets. In this work, the MSE will be employed for creating regression models regarding the number of rotors estimation. In (3.23), the mathematical formulation of MSE is given. There, y_i denotes the true value for instance i , \hat{y}_i the corresponding estimate for that instance, and N the total number of data samples examined:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (3.23)$$

The MAE follows the same line of reasoning with the MSE, yet in this case the absolute value of the difference between the predicted and true value is taken.

The diagram of Figure 3.10 graphically describes the typical training procedure of a neural network. As can be seen, the input is fed to the network and after numerous calculations a certain output is defined. The output is then compared to the ground truth by employing a loss function suited to the corresponding task. It is then the outcome of this function that will eventually define the weight update via an update algorithm such as the backpropagation one. The minimization of the loss function is the responsibility of a corresponding minimization algorithm, the most widely used one being the gradient descent. These two algorithms are fairly different in nature, one defining the weight update and the other the minimization of the loss function, yet both coexist and are vital for the network's training.

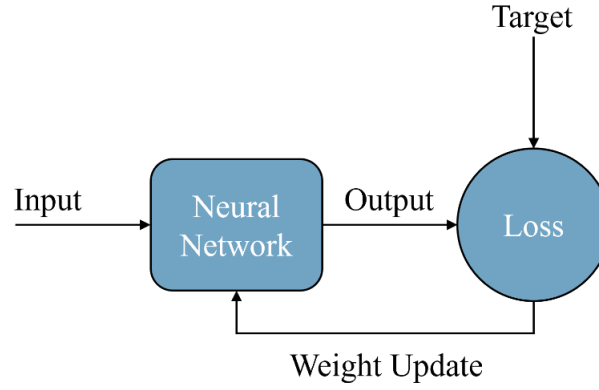


Figure 3.10. A simple block diagram representation of the training procedure of neural networks.

Gradient descent is an iterative optimization method aiming to find a local (or optimally, a global) minimum of a differentiable function. In the context of deep learning and neural networks in particular, gradient descent is used as a method of minimizing the chosen cost or loss function. The concept behind gradient descent is to iteratively update the model's parameters (i.e. weights and biases) in the direction of the steepest descent of the cost function. To that end, the gradient computed using the current parameters within the iteration is multiplied by a hyperparameter governing the pace of the descent, the learning rate, and subtracted from the current parameter values. Adapting the notation of [57], the gradient descent updates the model's weights as:

$$\mathbf{W} = \mathbf{W} - \alpha \frac{1}{m} \nabla_{\mathbf{w}} \sum_{i=1}^m L(f(\mathbf{x}_i; \mathbf{W}), \mathbf{y}_i) \quad (3.24)$$

In the above equation α is the learning rate, L the chosen loss function applied over the network's output $f(\mathbf{x}_i; \mathbf{W})$ and the truth value \mathbf{y}_i and m the size of the mini-batch over which the loss and gradients are computed. This minibatch mentioned earlier is a subset of the training set randomly drawn without replacement. It is usually much smaller in size than the original training set.

First introduced by Rumelhart et al. [61] in 1986, backpropagation is a crucial component of the model's training procedure as its goal is to iteratively update the model's weights by propagating the error from the output layer to

the input one while adjusting the weights of each corresponding layer along the way. As an algorithm it consists of two steps: the forward and the backward pass. During the forward pass, the data is fed through the network, the model generates predictions and an appropriate loss is computed. The backwards update is made possible via the chain rule and aims to decompose the overall loss or error into contributions from individual neurons. That key element of the backpropagation process can be summarized as:

$$\frac{\partial L}{\partial W_{jk}^l} \quad (3.25)$$

in which l denotes the network layer while j and k denote a corresponding neuron in the aforementioned layer.

Training neural networks, especially when they can be characterized as very deep, on a relatively small amount of data can be a highly challenging task that might prove catastrophic in terms of overfitting. A way to counter this comes from a similar to the Multi-Task Learning framework called transfer learning. The ultimate goal of transfer learning is to apply knowledge gained by learning one task to some other. As denoted by Goodfellow et al. [57], transfer learning refers to the situation in which using a network trained to predicting a certain, oftentimes widely different, task might assist to the generalization and learning of another.

Adding to this is the difficulty in acquiring radar data to be used in training and testing such models. As will be analyzed in the coming chapters, the limitations in creating diverse and large enough datasets containing radar output can make the creation of appropriate neural networks for target detection and classification a highly complex task. In an effort to artificially expand the dataset, a series of transformations can be applied to the input of the networks, making sure that the frequency component of the available spectrograms is not altered in resolution terms.

3.3. Conclusion

In this chapter, the concepts of radar and neural networks were introduced. More specifically, the main components and operation principles of the CW radar were analysed followed by an introduction to the Micro-Doppler phenomenon. There, the importance of acquiring the spectrogram and cepstrogram representations of the signal was made clear. For the rest of the thesis, the spectrogram will play a crucial role since as the input of the classification methods. The cepstrograms will be employed in a different role, that of labelling rotation rates of the drone measurements, as will be explained further in Chapter 4.

What followed the radar part in this background chapter was an introduction to Artificial Neural Networks. There, starting from the simple single neuron, the basics of neural networks and more specifically the Convolutional Neural Networks were presented. Emphasis was given in presenting the ResNet18, a well-known CNN type network aiming to reduce overfitting by employing a purposefully designed block of layers, the residual block with their skip connections. This, alongside its flexibility as a model and the possibility of controlling the randomness over different runs with the same parameters makes the ResNet18 an ideal candidate for experimentation using both the STL and MTL approaches.

4 Dataset

Oftentimes, Deep Learning and data driven approaches can perform only as good as the data that is available. A very important aspect of the whole DL pipeline is the creation of the dataset used by the corresponding algorithms. In cases such as the data dealt with in this work, which are experimental measurement data, reliable and correct labelling of samples is crucial. Hence, the purpose of this chapter is to discuss the available dataset, necessary preprocessing and labeling, as well as the steps to address imbalance issues. The outline of the chapter is as follows: **Section 4.1** aims to describe the dataset in its original form by presenting the types of targets and measurement scenarios as well as a necessary preprocessing step. **Section 4.2** describes the process of manually labeling appropriate rotation rate and event detection target values, while **Section 4.3** provides a description of the datasets created for experimentation purposes and the imbalance present among the classes of each task. Finally, **Section 4.4** concludes the chapter.

4.1. Description of the original dataset

The data considered in this work is produced by real life measurements using an X-band experimental continuous wave radar. The targets present in the dataset are all various types of drones ranging from commercially available types to homemade or customized ones. In total, 110 different measurements of varying durations, ranging from a few seconds to over three minutes, are available. Due to using a CW radar for experimentation, the output data comes in form of spectrograms created in the manner described earlier in Chapter 3. An example of such a spectrogram can be found in Figure 4.1. Since the measurements are of varying durations, inherently so are the spectrograms present in the obtained dataset.

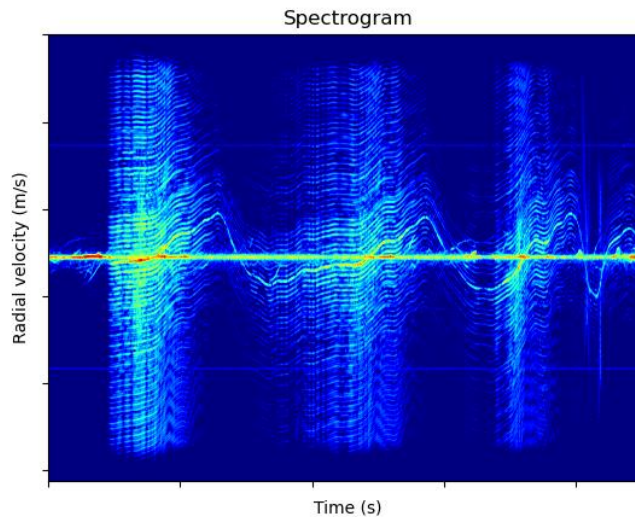


Figure 4.1 Example of a spectrogram belonging to a remote controlled helicopter drone.

Due to the inconsistency of measurements regarding the type of drones and flight scenarios, considerable class imbalance is also present throughout dataset. What follows is a description of both the types of drone targets and measurement scenarios found within the originally obtained dataset as well as the treatment of it in order to produce appropriate input for the learning methods assessed in this thesis.

4.1.1. Types of drone targets

The types of drones measured during the measurement campaigns range from commercial ones such as the DJI Mavic to amateurly and even homebuilt, customized ones. The in-depth description of the types of drones is out of the scope of the thesis and cannot be disclosed for sensitivity issues. However, it is important to describe the nature of the drone targets present in the dataset. These targets are categorized in three main categories depending on their structural and flight characteristics. These are the:

- Fixed wing drones: this type of UAVs resembles the operational principles of the well-known fixed-wing aircraft (i.e., airplanes). They possess one or more propellers in the vertical plane and their flight is controlled by flaps and by changing the propeller rotation rates. In level cruise flight, the propeller rotation rate is quasi-constant.
- Helicopter drones: resembling their larger counterparts, the helicopters, these drones have one large rotor in the horizontal axis and a secondary (and smaller) tail rotor to counteract the torque effect of the main rotor. These two rotors are thus mechanically coupled and the tail rotor spins about five to six times faster than its larger counterpart. Due to their structure, these drones are able to hover, unlike the previous ones.
- Multicopter drones: these drones possess more than three rotors in the horizontal plane which need to provide both thrust and lift. They are able to manoeuvre by adjusting the rotor rotation individually. This makes them able to both hover and be highly manoeuvrable.

The number of rotors for each type of drone present in this thesis varies as seen in Table 4.1. What can be also derived from the same table is the fact that identifying the total number of rotors of a target does not necessarily fully characterize its type. For example, it is obvious that detecting four rotors in a drone target is not a necessary and sufficient condition to further classify it as a multicopter drone.

Type of drone	Possible number of rotors
Fixed wing	1, 2, 4
Helicopter	2
Multicopter	4, 6, 8

Table 4.1: Types of drones present in the dataset and their respective number of rotors. Notice that, for example, there are rotary wing drones with 4 rotors, but there are also fixed wing drones with the same amount of rotors. This shows the inability of the number of rotors detection to fully characterize the type of the examined drone target.

4.1.2. Measurement Scenarios

The measurements included in the dataset are not limited in capturing only a portion of the flight of different drone targets. Even though this is the main measurement scenario, there are other three scenarios that create variance in the captured spectrograms and additionally provide the ground for additional classification or estimation tasks. The measurements scenarios in their entirety are listed in Table 4.2. Unfortunately, as it is often the case in acquiring radar data, apart from the lack of samples, there is a large imbalance in the measurement scenarios as well. This is showcased in the bar plot of Figure 4.2. To provide some context: the default flight measurement

scenario poses as the majority class, as 93 out of the total 110 spectrograms are of this kind, an imbalance that is translated in the same manner in terms of measurement time (in seconds) regarding the same scenarios.

Measurement Scenario	Description
Default measurement of a drone's flight	Simple monitoring of a drone target during its flight (or often a portion of it). Oftentimes the take-off and landing parts are also captured.
Drone held stationary	Measuring a drone's Micro-Doppler signature while being held stationary by a person or mounted in a purposefully constructed base.
Drone flying while a person is walking nearby	Measuring a drone's Micro-Doppler signature during flight while a person is walking underneath its flight path causing additional artefacts near the zero-Doppler region.
Two drones flying close together	During the measurement campaign, it occurred for two drones to fly in the same flight path one behind the other. These drones were both rotary wing drones: a quadcopter and an octocopter.
Drone with heavy payload	To enable the construction of a payload detector measurements of a drone carrying heavy payload were performed.

Table 4.2: Summary table regarding the measurement scenarios and their corresponding descriptions.

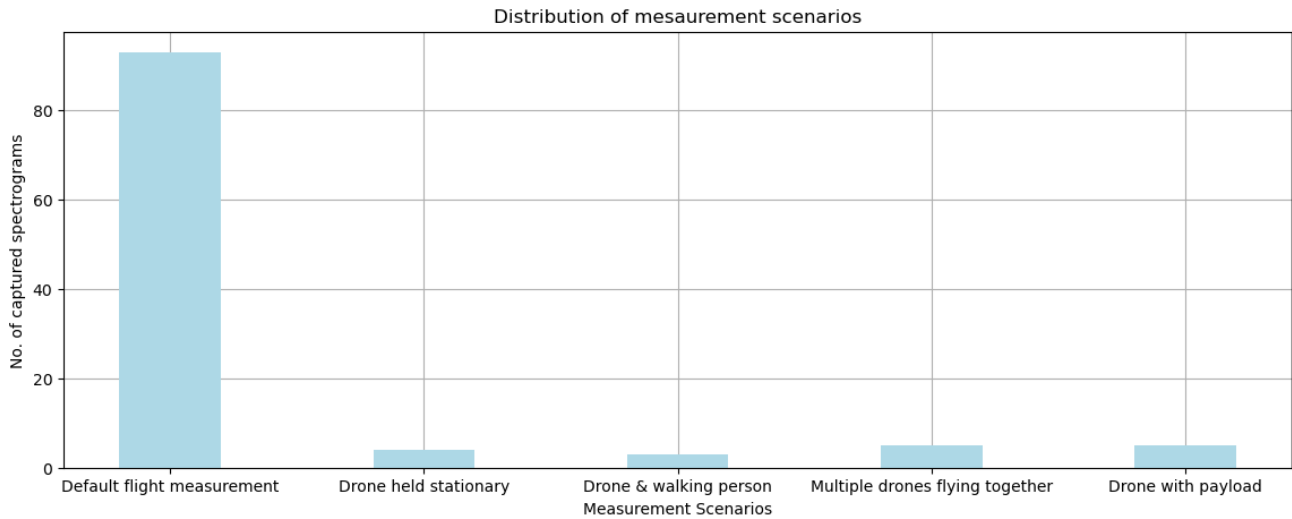


Figure 4.2: Bar plot presenting the distribution of measurement scenarios found throughout the original provided dataset used for this thesis.

4.1.3. Segmentation of the original dataset

The number of the original and unprocessed spectrograms is of course too small to be used as basis to perform any substantial training, validation and testing experiments using Deep Learning models. Moreover, due to the nature of the measurements, the discrepancies in time duration of the spectrograms leads to a dataset containing spectrograms of rather different lengths; however, it shall be noted that due to the processing parameters selected being the same across all measurements, the Doppler frequency axis is equal for all measurements. To account for this it was decided that further segmentation shall take place, producing much shorter spectrograms of the same dimensions to be used as individual samples for the subsequent classification processing.

An initial value of 0.5 s was selected as the new spectrogram duration. This selection is based on the fact that the radar system in hand is a CW one, meaning that the amount of information gained regarding the targets is severely limited. Furthermore, to limit the correlation among the segments, a gap of 0.0625 seconds was enforced among consecutive spectrogram segments. This segmentation led to a total of 8916 shortened-in-time spectrogram samples. During the segmentation, it often occurred that the last segment of each original spectrogram will be shorter than 0.5 s. These shorter spectrograms are eventually discarded prior to their use as input to the proposed networks.

4.2. Retrieving the rotation rates

Part of the work for this thesis is the estimation of the rotation rates of each target under test. The use of real measurement data imposes a great setback in completing this task: the retrieval of the ground truth regarding the rotation rate of the drone rotors is not a trivial process. The rotation rates are not provided in real time, especially since the measurements been taken also involve many amateurly built drones. Moreover, the measurements are experimental radar output and not simulated data. Hence, effort was put towards manually extracting the rotation rates of the drones' rotors.

4.2.1. Extracting representative quefrequencies

In view of the above, the cepstrogram produced by each measurement's spectrogram as described in Chapter 3 can be used towards rotor rotation rate calculations. In Figure 4.3b, a sample cepstrogram of an octocopter drone produced based on the corresponding spectrogram (Figure 4.3a) is presented.

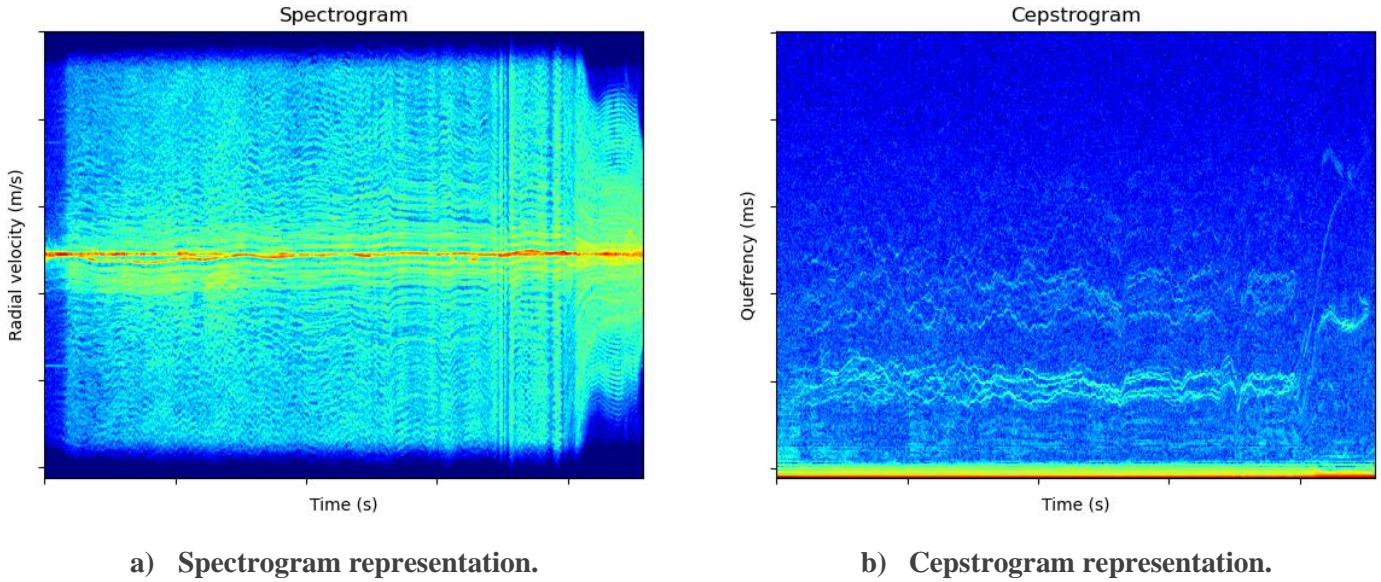


Figure 4.3 Spectrogram (left) and related cepstrogram (right) data representations belonging to an octocopter drone.

It is evident from Figure 4.3b that the extraction of true rotation rates for all eight rotors of the example is practically impossible. Specifically, the rotation rates of the corresponding rotors being close to each other as well as the resolution capabilities of the radar used are the two main factors in making the independent rotation rate extraction not feasible.

The exception to this rule, of course, are more straightforward and easily distinguishable cases such as single rotor fixed wing drones and helicopters where the rotation rates are either only one or two high powered quefrecny components for each time bin such as the case depicted in Figure 4.4. Another fact that makes the rotation rate extraction easier considering fixed wing drones is the fact that during level flight which they usually perform, the rotation rate of the single, rotor is in most cases constant, leading to its extraction being of lesser effort.

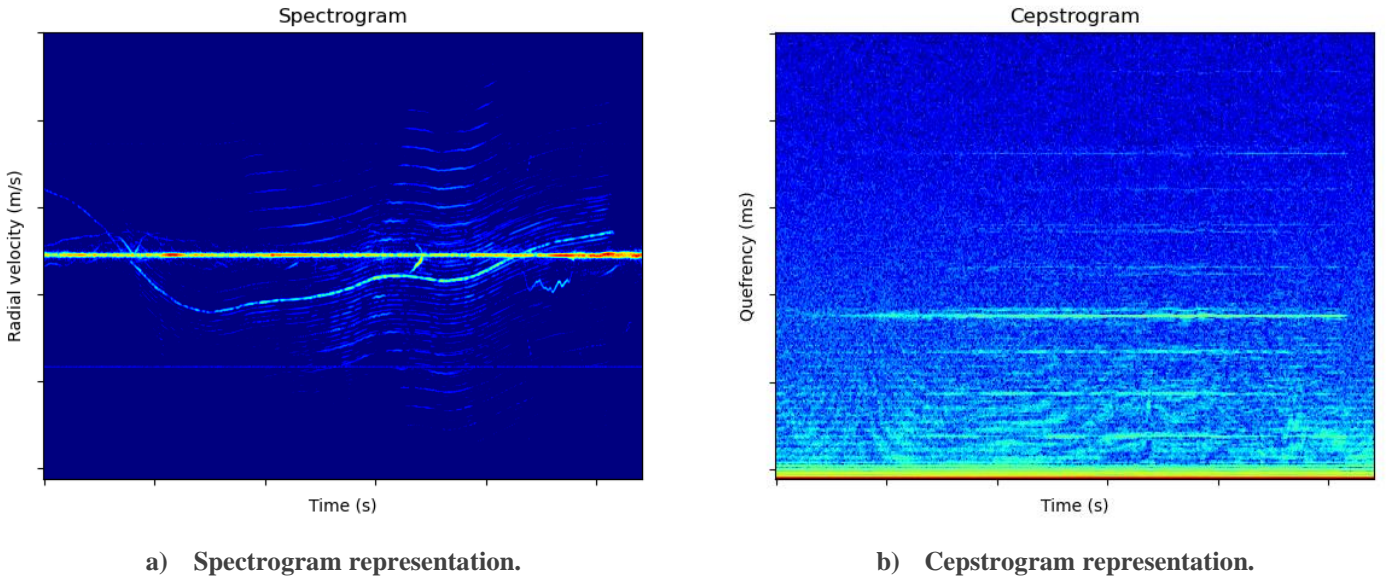


Figure 4.4: The spectrogram and related ceprogram representations of a helicopter drone target. One may notice that the extraction of the quefrency components produced by the rotors' rotation are easily distinguishable.

Due to the nature of the measurements, the various drone models, the SNR conditions and the maneuvers that each target possibly performs, a simple multiple peak detector to identify the quefrency corresponding to each rotor at every time bin would prove to be impractical. Additionally, the variety of the drone targets present in the dataset would translate to regression targets of inconsistent dimensions depending on the various number of rotors. For example, a single rotor fixed wing drone's ceprogram will in this manner produce a single vector target corresponding to the corresponding rotor. On the contrary, an octocopter will produce eight distinct target vectors. This inconsistency in target variables, assuming that individual quefrencies and rotor rates can be identified and labeled, would make the definition of appropriate regression methods highly complex if even at all possible.

To counter the challenges induced by the above, a compromise had to be made in that instead of individual rotor quefrencies a single representative one is identified. The fact that the used data come from real-world measurements as well as the variation in rotor tip velocity among the different experiments makes an automated approach of retrieving a representative quefrency for each time bin very complex. It would appear that each ceprogram poses a different case in obtaining the necessary quefrency and thus manual and individual effort for each available ceprogram was required.

The extraction of the representative quefrency is summarized in four steps as shown in Figure 4.5:

1. In the beginning, after obtaining the ceprogram using the spectrogram of the corresponding sample, an area of interest is denoted, capturing the lowest quefrency component produced by the drone target's rotors rotation. This area definition is done manually across the whole duration of the spectrogram and may vary depending on the ceprogram and target in hand.
2. After defining the aforementioned areas, the representative quefrency in each time bin is defined as the quefrency component with the highest power within the predefined boundaries. In this stage, one of the points made about the dataset and its nature mentioned earlier can be showcased: at the point of the rotors' deceleration (near end of the ceprogram where the high power quefrency component increases) the SNR seems to decrease and individual quefrencies can no longer be identified. This is also true for the rest of

the cepstrogram prior to that point as well, where even if a high powered component is visible, no individual quefrequencies can be obtained for every time bin.

3. As could be expected, the proposed method leads to a very crude approximation of the representative quefreny as can be seen from the wildly fluctuating example of second spectrogram (upper right) of Figure 4.5. In an effort to obtain a more realistic representation of the quefreny, a simple moving average filter was applied to each case individually using a number of samples (in the form of quefreny components over time bins) tailored to each cepstrogram.
4. The final form of the extracted quefreny overlayed on the corresponding cepstrogram of the octocopter drone example can be viewed in the fourth plot (lower right) of Figure 4.5. What is also showcased in the same figure, apart from its much smoother appearance, is that the extracted representative quefreny is able to match the trends and fluctuation of the whole constellation of the eight individual ones. In the cases where landings take place, or the drone has not yet taken off, the quefreny at these time bins where the rotors are static is considered as infinite.

Representative quefreny extraction

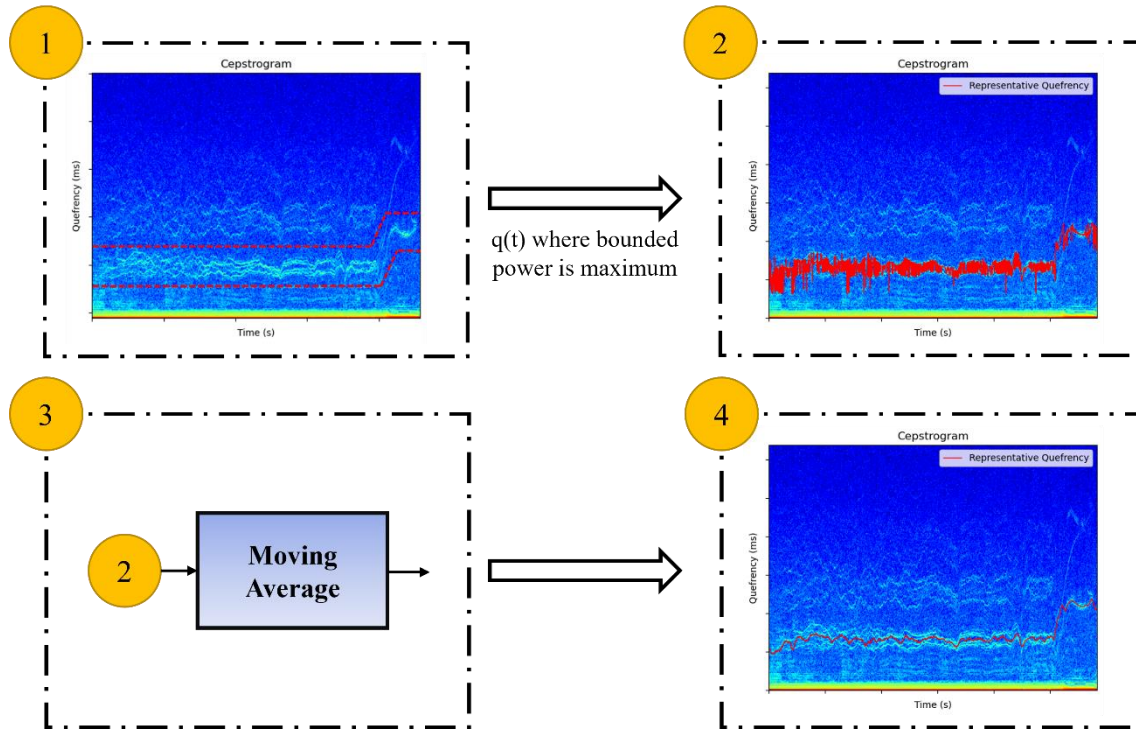
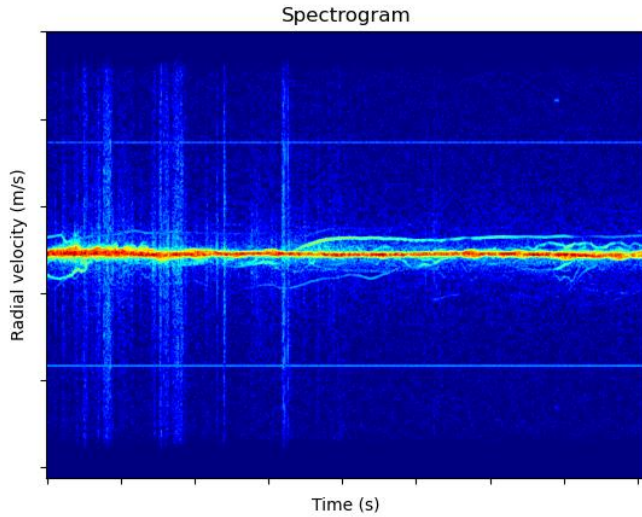


Figure 4.5: The representative quefreny extraction procedure in block diagram form.

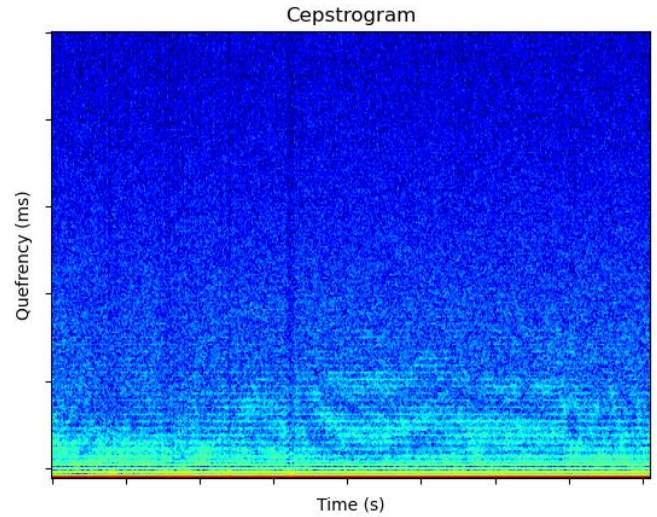
For the case of single rotor drones (and oftentimes the fixed wing drones with two rotors one behind the other), the aforementioned area of interest follows the lowest quefreny component with the highest power. In the case of helicopter drones, where at least two distinct high powered quefrenies can be defined, the one assumed to belong to the main rotor (usually corresponding to higher quefreny) is obtained. For the multicopter drones areas of interest are designated around the suspected batch of quefrenies corresponding to the different rotors the drone is equipped with. These batches are chosen to be the highest in power and almost always correspond to the ones with the lowest quefrenies.

Of course, the extraction of appropriate quefrenies is not always possible. Adding to the difficulty of defining appropriate rotation rates is the occasional (mainly due to range restrictions) inability of the radar to capture meaningful Micro-Doppler signatures produced by the targets. In these cases no quefreny components can be traced in the produced cepstrograms, making the rotor rotation rate extraction implausible. Such a case is depicted

in Figure 4.6, where no rotation rates can be seen. For measurements where the continuity of the Micro-Doppler signature in the spectrogram is questionable, the discontinuities in the detected queffrequency are interpolated manually considering the flight characteristics of the target (i.e. if the target performs level flight, a quasi-constant and level queffrequency is obtained). Such a case is depicted in Figure 4.7, where the queffrequency content of the target's cepstrogram contains gaps.

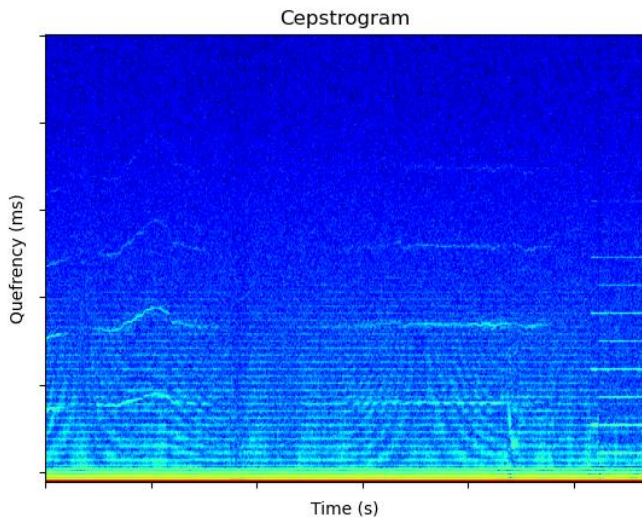


a) Sample spectrogram where no Micro-Doppler signature attributed to the rotors is captured.

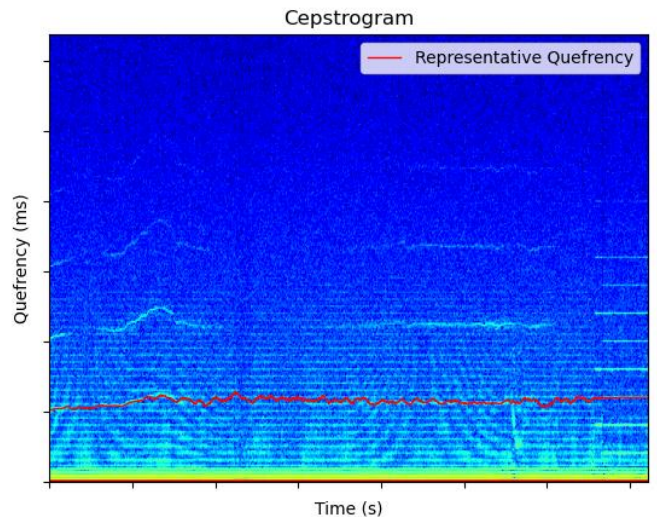


b) The result of capturing no rotor Micro-Doppler: no representative queffrequency can be traced.

Figure 4.6: An example of a spectrogram case (left) with no evident Micro-Doppler signature resulting in inability of queffrequency extraction from the corresponding cepstrogram (right).



a) The cepstrogram of a target which Micro-Doppler signature is not captured at every time instance.



b) The gaps in the extracted representative queffrequency are interpolated by assuming that the rotation rate remains constant.

Figure 4.7: Example of the treatment of cases where the queffrequency to be extracted cannot be detected in every time bin.

However, until now, no rotation rates have been calculated. The conversion to rotation rates (in Hz) is done via:

$$\text{Rotor rate} = 2 \cdot \frac{1}{\text{quefrecy} \cdot 0.001} [\text{Hz}] = 2 \cdot \text{blade flash rate} \quad (4.1)$$

multiplying with 0.001 to compensate for the fact that quefrecies come in millisecond scale. The multiplication with two takes place so that to distinguish between blade flash and the rotation rate where the latter is two times the former rate. This is based on the fact that two blade flashes are observed in a single full rotor circle in the case where the rotor is comprised of two blades, which is the case in all observed targets in the used dataset. The final product of the quefrecy extraction and its transition to rotation rates in Hz scale can be seen in the plots of Figure 4.8. Note that both plots are not in the same scale as the cepstograms presented previously. Also, note that prior to their use as target values, the rotation rate vectors corresponding to each cepstrogram are also segmented following the same manner as the spectrogram input, as described previously in Section 4.1.

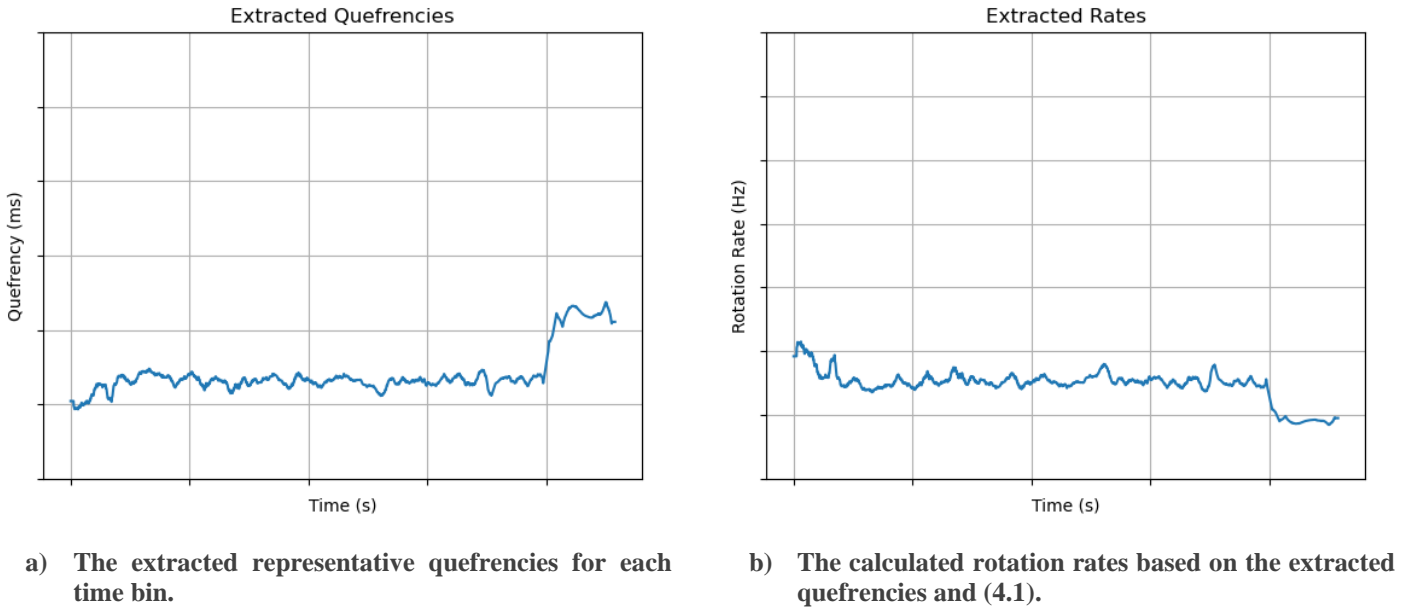


Figure 4.8: Extracted representative quefrecies (left) and rotation rates (right) based on the octocopter example depicted in Figure 4.5.

4.2.2. Defining and labelling events

The original dataset contains produced spectrograms capturing signatures of various events of the drones' flight, them mainly being their take-off and landing procedures. Additionally, sudden acceleration and deceleration of the rotors can be observed in certain cases such as the one presented in Figure 4.3a. It would be of great interest for an operational system to be able to predict or estimate not only the rotation rate but different events relevant to the target's flight as well. Such events may range from take-off and landing procedures (in the form of sudden spin or stop of the rotors) to sudden changes in rotation rates that could, for instance, imply the attempt to compensate for the drop of a drone's payload.

Having identified a representative quefrequency and, consequently, the rotation rate of the rotors for every time bin in each case possible, the next step is creating suitable labels for event detection. To suit the needs of this thesis, an event is characterized at each time bin by firstly taking the first order derivative of the rotation rate with respect to time. For practical reasons, this is computed as the absolute value of the rotor rate difference among two consecutive time bins divided by what is essentially the resolution in time for the used radar. It can be formulated as:

$$\frac{drate}{dt} = \frac{rate(t + 1) - rate(t)}{dt} \quad (4.2)$$

where with rate, the retrieved rotation rate is denoted and dt is the difference of two consecutive time bins, a constant value among the measurements. To capture the much needed events, a threshold is empirically defined. If the derivative at a time bin is larger than this threshold, then that bin is characterized as containing an event. Once again using the octocopter example of the previous sections, this leads to the event extractor capturing most of the events, both major and minor ones (such as simple turbulence or stabilization efforts) as seen in Figure 4.9.

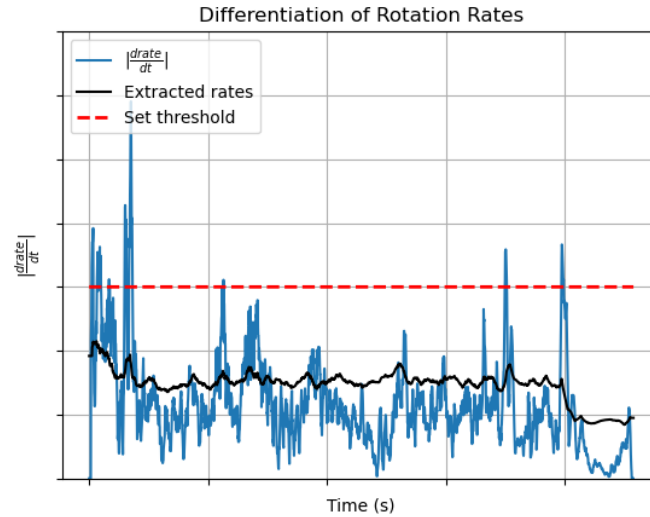


Figure 4.9: The extracted rates (in Hz) and the set threshold overlayed on the result of the differentiation over the rotation rates using (4.2).

Two main issues can be identified in using this approach: firstly, the event labeling inevitably inherits the flaws of the also manually extracted rotation rates, hence there is the possibility that a number of extracted events are result of induced fluctuation due to the rate extraction process and do not depict the situation accurately. Secondly, as expected, due to the empirical nature of the threshold value and the fact that derivative of the rotation rates is taken over a single time bin, a plethora of minor and possibly unimportant rotation rate changes will be characterized as events.

In an effort to mitigate the effects of the above, a sliding moving average filter using ten samples was applied over the rate derivative, providing a much smoother result as seen in Figure 4.10 below. The averaging is applied in the assumption that important events, such as a take-of, landing or payload drop will produce large enough spontaneous differences that will not be greatly affected by the moving average filter. On the other hand, less important events, such as turbulence, or even faults during the rate extraction will be diminished.

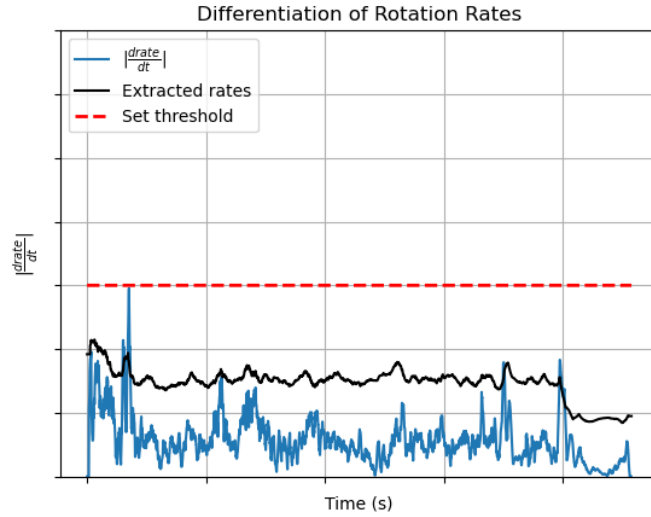


Figure 4.10: The extracted rates (in Hz) and the set threshold overlaid on the result of the differentiation over the rotation rates using (4.2).

4.3. Definition of datasets for experimentation

As discussed extensively in this chapter, the original dataset contained too few and varying in dimensionality spectrograms. To counter this, segmentation of the original spectrograms into much shorter in duration (0.5 s) ones without any overlap was carried out. This segmentation provided with 8916 unique spectrograms to be used as input to the further proposed neural networks employed for experimentation.

Earlier in Section 4.1, it was made clear that the provided dataset contains multiple measurement scenarios, which then translates into an opportunity of assessing the performance of both STL and MTL models in a variety of different tasks. To satisfy the dataset needs of the experiments analysed in Chapter 5, three different datasets were produced in support of the classification and regression tasks in hand, as well as for the proper treatment of the defined classes and scenarios. These three datasets are named for the rest of this work as:

- Dataset A
- Dataset B
- Dataset C

The first two defined datasets (A and B) describe the structural configuration of the available targets as well as the existence of payload and multiple UAVs in the same scene. Both of them are suited to the wing type classification, number of rotors classification and payload detection. Dataset B additionally treats the multiple UAV case by labelling it as a rotary wing drone instance, while assigning 12 as the number of rotors of this case (the two drones in this scenario are one octocopter and a quadcopter). Moreover, it labels the existence or not of a second drone, providing with the possibility of performing multiple drone detection as an additional task.

Dataset C aims to describe mostly the flight characteristics of the detected target by providing, apart from the 0.5 s spectrograms, the full and mean rotor rate vector for each input instance, indicatory information on whether an event has occurred within these 0.5 s, as well as information about whether there is payload mounted on the target or not. A summary of the created datasets can be examined in Table 4.3. It shall be noted at this point that all the information depicted in Table 4.3 refers to the whole datasets and does not make any assumption regarding training and testing splits. Chapter 5 provides further information on that aspect.

Dataset	No. Samples	Tasks supported	Target Values
A	8467	<ul style="list-style-type: none"> Wing type classification Number of rotors classification Payload detection 	<ul style="list-style-type: none"> Fixed, Helicopter, Rotary 1, 2, 4, 6, 8 True, False
B	8916	<ul style="list-style-type: none"> Wing type classification Number of rotors classification Payload detection Multiple drone detection 	<ul style="list-style-type: none"> Fixed, Helicopter, Rotary 1, 2, 4, 6, 8, 12 True, False 1, 2
C	6317	<ul style="list-style-type: none"> Rotor rate regression Mean rotor rate regression Payload detection Event detection 	<ul style="list-style-type: none"> 0.5 s bins of rotation rates Single continuous variable True, False True, False

Table 4.3 Collective description of the three datasets created for experimentation purposes.

4.3.1. Imbalance study

An issue easily identified in all aforementioned datasets is the heavy imbalance among the different existing classes, an issue partially inherited by the inconsistencies in measurements, both in target type and measurement scenarios. This imbalance problem can be traced in all classification tasks considered in this work, being more prominent in the binary classification ones. In this subsection, the distribution of the classes for each classification task considered in this work will be presented. These are the:

- Wing type
- Number of rotors
- Payload detection
- Multiple UAV detection
- Event detection

This imbalance study can be performed in two parts: the first refers to datasets A and B regarding the drone targets' configuration and the second to dataset C regarding their flight characteristics (rotor rate regression, payload and event detection).

Datasets A & B

Figure 4.11 depicts the distribution of the samples regarding the wing type classes for datasets A and B. Both datasets contain the same number of fixed wing and helicopter drones, yet the added spectrograms where two targets are being captured at the same time lead to an increase of rotary wing drones. These additional to the dataset A's cases are depicted with pink in the bar plot of Figure 4.11. It can be concluded that in dataset A there is a slight imbalance among the fixed and rotary wing drones, something that is no longer an issue when considering the additional multiple drone cases. The most distinctive issue is the number of helicopter type drones in the dataset that leads to a more considerable imbalance among the three classes.

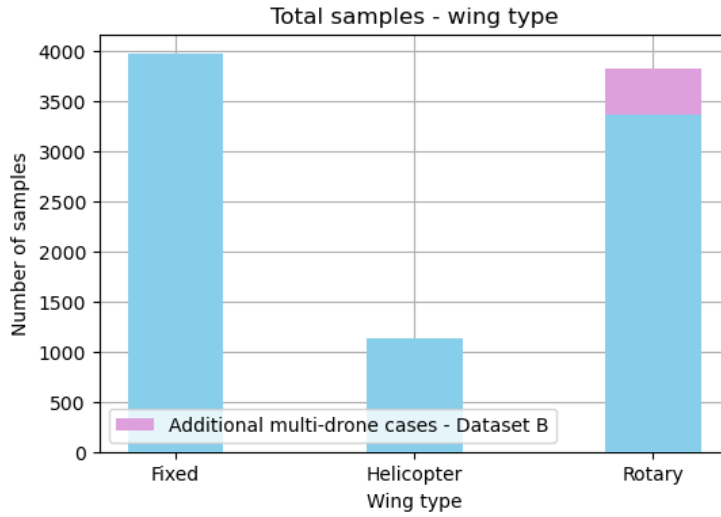


Figure 4.11: The distribution of the wing type class for datasets A and B describing drone target configuration. The pink field on the rotary class describes the added cases where two rotary wing drones are captured in the same spectrogram.

The number of samples distribution regarding the number of rotors classification task can be seen in Figure 4.12. Again, in the same manner as in the previous bar plot, the additional class of 12 alongside its number of samples are represented using the pink colored bar. The aforementioned difficulty of characterizing a drone only by its wing type is further supported by the findings of the same figure. It can be seen that not all fixed wing drones possess a single rotor, since a considerable amount of them are equipped two or even four rotors.

The division of the rotary wing drones into their rotor number defined classes induces class imbalance especially for the cases of quadcopter and hexacopter drones with the latter being the least represented number of rotor class in the two drone configuration datasets. Similarly, the 12 rotor target (multiple drone cases) spectrograms are posing equivalent levels of rarity, being only 449. This imbalance is further observed when considering the detection of multiple drone cases a separate task as in Figure 4.13 where a ratio of $\sim 1:19$ can be observed.

Another case of serious imbalance can be found in the payload detection task (dataset A) as seen in Figure 4.14. Even though the ratio is found to be slightly better ($\sim 1:15$ for the dataset that does not contain the multiple drone cases), it is still a very important issue which will be vastly affecting the performance of the proposed classifiers.

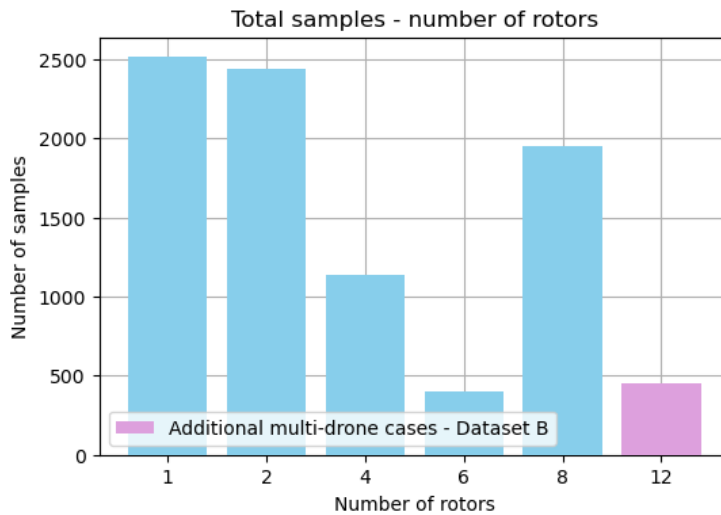


Figure 4.12: The distribution of the number of rotors classes for the first two datasets describing drone target configuration. The additional class is defined as the sum of the number of rotors present on the two drones captured in the same spectrogram.

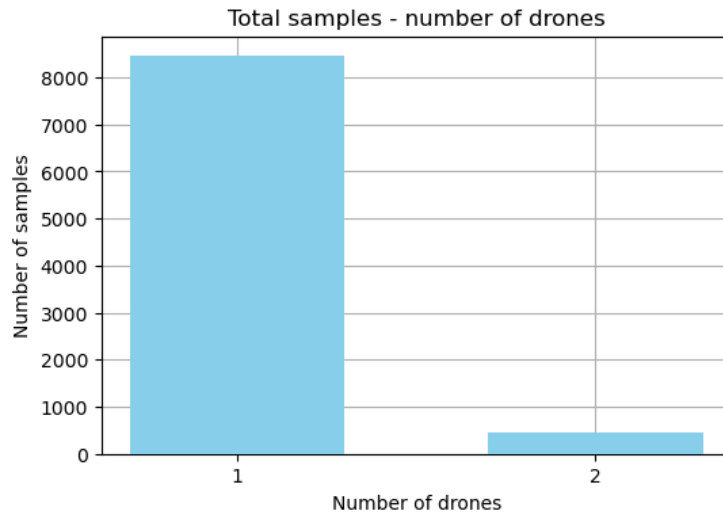


Figure 4.13: The distribution of the number of drones classes for dataset B containing spectrograms with multiple drones in the scene. Serious imbalance can be observed.

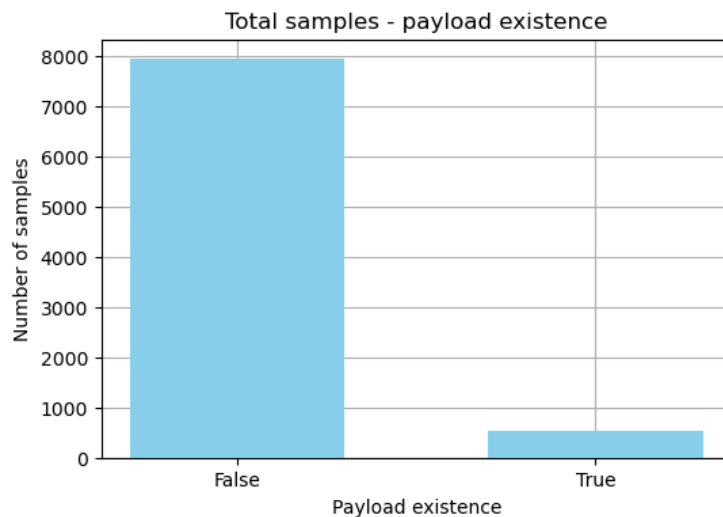


Figure 4.14: The distribution of payload existence samples for dataset A that does not contain spectrograms with multiple drones in the scene. Once again the dataset is heavily imbalanced for this task.

Dataset C

Regarding the flight characteristics dataset, an imbalance study can be conducted on two tasks, the one of payload and the one of event detection. Partially inheriting the characteristics of the above analyzed datasets, the dataset destined for rotor rate regression and payload detection also suffers from serious class imbalance as seen in Figure 4.15. Note that in this case all the available cepstrograms belonging to drone carrying payload cases were able to produce viable quefrequencies for defining appropriate rotor rate regression targets, something that is clearly not true for the majority class and hence it appears slightly trimmed of samples, making the ratio only slightly better.

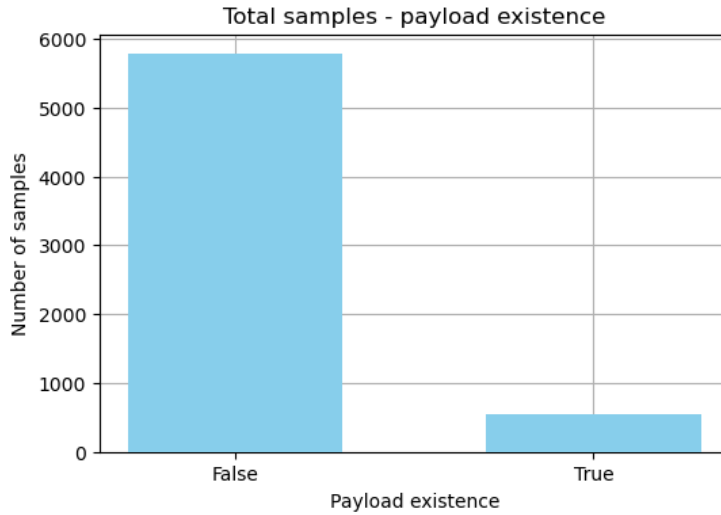


Figure 4.15: The distribution of the number of drones classes for dataset C containing spectrograms with multiple drones in the scene

For the event detection task, since the labels are based on manual extraction, the imbalance is associated to the definition of the threshold and number of samples used by the moving average filter. However, using the settings mentioned earlier provides the distribution seen in the plot of Figure 4.16.

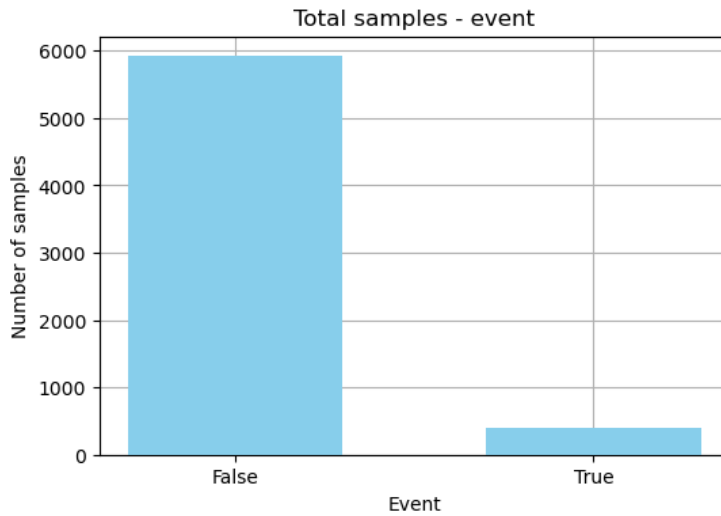


Figure 4.16: The distribution of the number of drones classes for dataset C containing spectrograms with multiple drones in the scene

It is expected that the imbalance in all tasks shown above will play a major role in the performance of the methods and classifiers examined in this work. Several methods have been proposed in the literature in order to counter the problem of class imbalance. Two of them that were considered but eventually discarded were the Synthetic Minority Oversampling Technique (SMOTE) and a modified version of the so called stratified sampling.

Introduced in 2002 by Chawla et al. [5], SMOTE works by firstly selecting a minority class sample and finding a predefined nearest minority neighbours. Choosing one of these neighbours, the algorithm generates new instances via a convex combination of the originally chosen sample and its selected neighbor. In this way, as many synthetic samples belonging to the minority class as needed can be produced, avoiding the classical oversampling approach that often leads to overfitting. It is also suggested in [5] that under-sampling, or trimming, of the majority class

alongside SMOTE (oversampling) applied on the minority class may lead to better performance than plainly under-sampling the majority class.

Stratified sampling refers to a sampling technique aiming to firstly divide the original dataset into homogenous subsets, the so-called strata, and then apply random sampling within each subset. This sampling approach is used especially when the dataset is large and unbalanced. Based on the latter, stratified sampling was also considered as a means towards creating training, validation and test sets that would contain a fair amount of samples belonging to minority classes leading to better evaluation of the classifiers' performances.

Both the SMOTE and stratified sampling could be easily applied when training STL models as no interaction between the different task classes exists. However, this is not true when considering MTL models where fixing the class imbalance in one task can induce, or even worsen the imbalance in another. A possible solution to this problem would be the definition of new fused classes based on all possible combinations of them over all the addressed tasks as seen in the example of Figure 4.17. In this manner, two tasks with three and six classes respectively would be temporarily merged into one with eight classes. Even though this approach would allow for a more fair sampling approach and would even allow for possible SMOTE application, it was deemed impractical and suboptimal especially when expanding to more than two tasks.

Hence, due to the implications of the aforementioned approaches and the complications induced by label fusion, it was decided that no action would be taken in dataset level in order to deal with the problem of class imbalance. In Chapter 5 more information regarding the handling of imbalance in the loss function and training of the classifiers is provided.


Wing type	Number of rotors		Fused labels
Fixed	1		(Fixed, 1)
Fixed	2		(Fixed, 2)
Fixed	4		(Fixed, 4)
Helicopter	2		(Helicopter, 2)
Rotary	4		(Rotary, 4)
Rotary	6		(Rotary, 6)
Rotary	8		(Rotary, 8)
Rotary	12		(Rotary, 12)

Figure 4.17: Example of the considered label fusion among all different tasks so as to provide the ground for the application of methods to eliminate class imbalance such as SMOTE. Even though such an approach would be suitable for tasks with smaller number of classes, it is seen as not optimally scalable when more classes are added to each task.

4.4. Conclusion

To summarize, this chapter offered a description of the experimental data used in this work and the process of acquiring the much needed datasets on which the classifiers presented later in this report are trained and evaluated. The labels regarding the structural characteristics of the targets (wing type, number of rotors, payload existence) and the number of drones captured in a single spectrogram were readily available. To make the rotor rate regression possible, a single representative queffreny for each time bin of each cepstrogram was extracted, paving the way for calculating the corresponding rotation rates. Each time bin was further characterized as containing an event or not based on the rotor rate derivative at that point.

Three datasets were defined out of the experimental data made available for this thesis, two of which holding structural configuration information and a third tailor-made for rotor rate regression, payload and event detection. In the imbalance study of the defined datasets, the significant imbalance issue of the binary classification tasks as well as the imbalance among the classes of the wing type and number of rotors classification tasks was discussed. A further issue identified related to the imbalance is the impracticality of applying classic methods such as SMOTE and stratified sampling in MTL cases. As discussed, due to the relationship among the classes of the different tasks, no task can be individually oversampled or downsampled without negatively affecting another and avoiding class overlapping at the same time. Hence, no action was taken to correct the imbalance in the various task presented.

4.5. Acknowledgments

Hereby we would like to thank Thales Nederland B.V. for making this thesis work possible. The radar data used for this study have been kindly made available by Thales Nederland B.V. within the framework of D-RACE: the Dutch Radar Centre of Expertise, a strategic alliance of Thales Nederland B.V. and TNO.

5 Proposed Methodology

*The exploratory nature of this study and the comparison needed among the two aforementioned paradigms (STL and MTL), as well as the challenges induced by the notion of MTL require a strictly-defined testing methodology. Therefore, this chapter aims to introduce the reader to the methodology followed throughout the experiments, along with the proposed classifiers. The chapter begins with the introduction of the task groups examined in this work, as well as the reasoning behind them in **Section 5.1**. After introducing the task groups, the proposed STL and MTL type models are described thoroughly in **Section 5.2**. The means of comparison between the two paradigms and their corresponding models alongside the appropriate performance metrics are presented in **Section 5.3**. Following that is the description of the two testing configurations used to assess the performances of the proposed classifiers in **Section 5.4**. Finally, **Section 5.5** summarizes the chapter.*

Chapter 4 introduced vital aspects of this work, such as the acquired data, the process of labelling it and the datasets created to support the prediction of certain tasks. Figure 5.1 loosely summarizes the classification pipeline followed in this thesis. To recap, after recording drone measurements using a CW radar such as the one presented in Chapter 3, the corresponding measurement spectrogram is calculated and split into segments of equal time duration. Regarding the preprocessing of the data, transformations other than normalization are applied to the spectrogram segments only during the training phase. Hence, after normalizing the available spectrograms, they are used as input to the Deep Learning model which predicts the task in question. In the case of an MTL model, several tasks are predicted simultaneously.

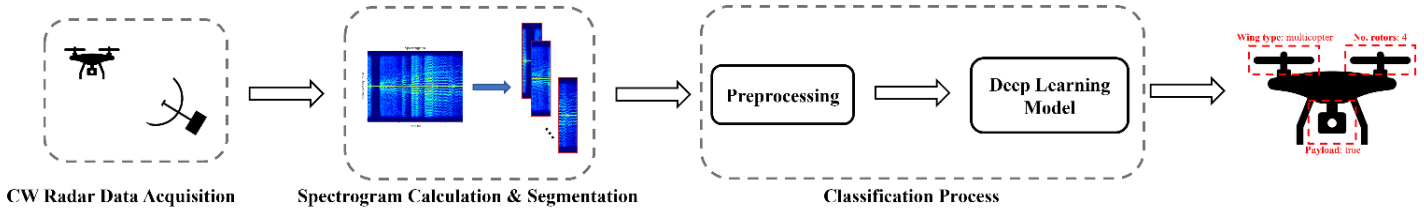


Figure 5.1: Illustration of the drone characterization pipeline investigated in this thesis.

In the context of MTL and the exploratory approach of this work, the following research questions arise:

1. How are related tasks defined and which task groups can be examined using the available (relatively limited) data?
2. What are the possible architectures for both STL and MTL approaches?
3. How can STL and MTL classifiers be compared?
4. How can one define test configurations that both eliminate biases and provide information regarding the robustness of the methods using the available, and severely limited, data?

The following sections aim to provide with answers and insights to these questions by describing the methodology followed during experimentation.

5.1. Defining related tasks

The previous chapter presented an analysis of the defined datasets, accompanied by the types of tasks that each one supports. Part of the success of an MTL model is dependent on the relation and correlation among the given

tasks. In general, two trains of thought exist regarding the choice of tasks to be jointly predicted by MTL models. On the one hand, related tasks are expected to be constructive to each other due to the network learning features that would otherwise be discarded (i.e. in an STL approach) [33]. On the other hand, training uncorrelated tasks together may mimic the addition of noise to the backpropagation and thus help improve generalization [62]. Of course, this is not always the case since unrelated tasks will mostly lead to inefficient models. This is additionally an advantage the SPS approach poses over its HPS counterpart, since uncorrelated and destructive to each other tasks will automatically be trained either in parallel (by passing a small or no amount of information among the models).

As indicated by Rich Caruana [33], the definition of relatedness among tasks is an open problem as there is no adequate definition of when two tasks are related to one another. The discovery of related tasks usually requires an empirical and heuristic approach, the results of which have to be thoroughly examined, since as mentioned above good performance on all tasks does not necessarily translate to them being actually related. However, for the needs and scope of this work, since MTL has not been employed for drone characterization using spectrograms as input in the past, a simple heuristic approach is followed for the definition of groups of tasks. The tasks described in Chapter 4 are thus grouped in a manner that seems reasonable to a human expert. Thus, the examined combinations are the following (note that the abbreviations in the parenthesis will be utilized through the rest of this work in a great extent):

- **Wing type & Number of rotors (WT & NR)**: the wing type and number of rotors are two seemingly related tasks, since theoretically these two structural aspects of the drone are highly correlated. For example, a helicopter drone is expected to have two rotors.
- **Wing type, Number of rotors & Payload detection (WT, NR & PD)**: the wing type and number of rotors a drone has may also imply the possibility of payload mounted on the drone. For instance, small helicopter drones are less likely to carry payload when compared to their larger multicopter counterparts.
- **Wing type, Number of rotors & Multi-drone detection (WT, NR & MDD)**: treating the multiple drone spectrogram cases as 12 rotor ones (basically noting the existence of two multicopters, a quadcopter and an octocopter in such data) may hint to the existence of two drones in the scene. Hence, the multiple drone detection task may benefit from the number of rotors one and vice versa.
- **Rotor rate regression & Payload detection (RRR & PD)**: jointly learning to estimate the rotors' rate (both in its mean or full form) and the existence of payload might prove advantageous since a drone carrying some sort of payload is expected to exhibit higher rotation rates in order to compensate for the additional weight to lift and carry.
- **Rotor rate regression & Event detection (RRR & ED)**: predicting sudden changes of the representative rotor rotation rate is thought to also imply the occurrence of a possible event. For example, that event might be the take-off of the drone with the rotation rate rapidly increasing from zero to a few hundred Hz in one time bin.

5.2. Description of the proposed models

It was defined previously in Chapter 3, that the architecture of both the STL and MTL models is chosen to follow the one of the ResNet18. This network type was chosen for three main reasons, two of which were its resilience in terms of overfitting and the flexibility provided by its implementation. In the span of this chapter, the ResNet18 is presented in a simplified and abstract manner, imitating its implementation in the Torchvision package.

5.2.1. The architecture of the STL models

In Figure 5.2, the entirety of ResNet18 which is used as the STL model is shown. Note that when employing the pretrained version of the model, all layers up to the fully connected one maintain the provided by Torchvision weights, while the last fully connected one had to be reinitiated to fit the classification task in hand.

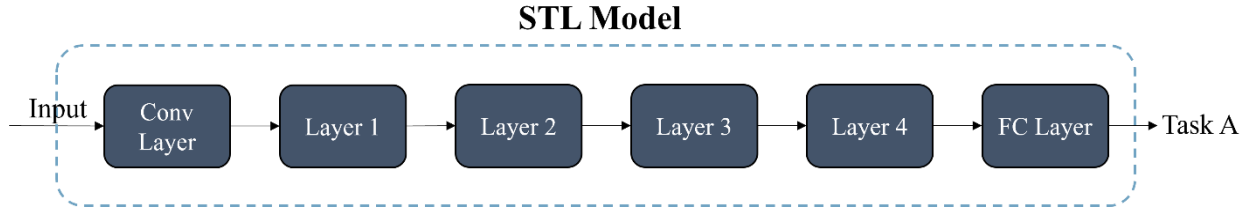


Figure 5.2: A simplified illustration of the STL model used in this thesis and inspired by the ResNet18.

In the above figure, the simplified ResNet18 architecture depiction is enclosed within the dashed blue line, which represents the portion of the backbone architecture that remains intact in each case (note that here, since the depicted model is an STL one, the ResNet18 is used in its entirety). This will be the convention in representing the models in the rest of this work.

5.2.2. The architecture of the MTL models

Three Multi-Task Learning models are defined in an effort to assess the performance of different approaches over multiple tasks. As stated earlier, all three of these models were based on the ResNet18 architecture provided by the Torchvision package and mainly followed the Hard Parameter Sharing approach as introduced in Chapter 2. These models are the:

- “Simple” HPS
- Adjusted HPS
- Info-sharing HPS

For the rest of this work, the Adjusted and Info-sharing models will be also addressed as “Adjusted MTL” and “Info-sharing MTL” for conciseness. The “Simple” HPS approach, as depicted in Figure 5.1, is solely defined by the backbone network. As was already mentioned earlier in this thesis, the HPS approach is implemented by splitting the backbone network in more than one parts depending on the number of tasks in hand. In the example of Figure 5.3 for instance, two tasks are assessed and thus the network is split into two streams. It was decided in this case that the split would take place just before the final and fully connected layer of the backbone network, providing with two identical fully connected layers among the two different tasks. In the case where the backbone network is pretrained, as was tested in this work, all layers up to and including Layer 4 kept the pretrained weights of the original backbone. Yet, as it was the case with the STL model above, this was not possible for the fully connected layers tailored to each task separately, since they needed to be reconfigured in shapes not identical to the provided network’s corresponding layers.

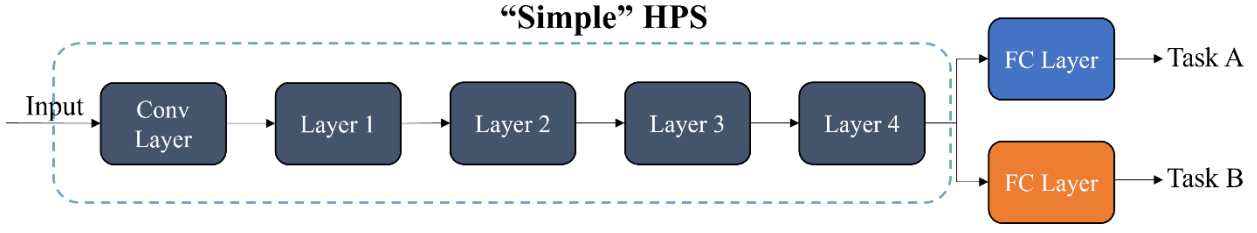


Figure 5.3: Schematic representation of the “Simple” HPS Multi-Task Learning model defined in this thesis.

It was also pointed out in Chapter 2 regarding the HPS paradigm that a frequent dilemma around HPS is the one of where one should split the backbone network into independent streams for each available task. In an effort to examine the impact of splitting, the “Adjusted” HPS model is proposed, where the split into independent task streams takes place just prior to the ResNet18’s fourth layer. In this way, higher independence is given to each task in the hope that the respective layers will be fine-tuned in a more efficient manner. The initial weights of Layer 4 are the same for both tasks and equal to the ones provided by the pretrained ResNet18 of Torchvision. This model’s architecture can be viewed in Figure 5.4.

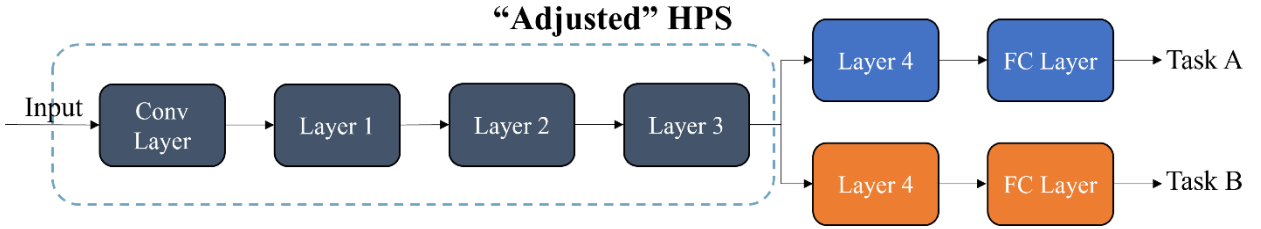


Figure 5.4: Schematic representation of the Adjusted HPS Multi-Task Learning model defined in this thesis. Notice that in this case the split into individual task streams takes place one layer earlier than in the “Simple” HPS network.

Partially inspired by the Soft Parameter Sharing paradigm and the multi-scale residual attention network of Li et al. [48] the “Info-sharing” HPS model is proposed, illustrated in Figure 5.5. The influence of SPS can be seen in the exchange of information after the last convolution layer of each independent task stream in an effort to benefit from each other’s extracted features.

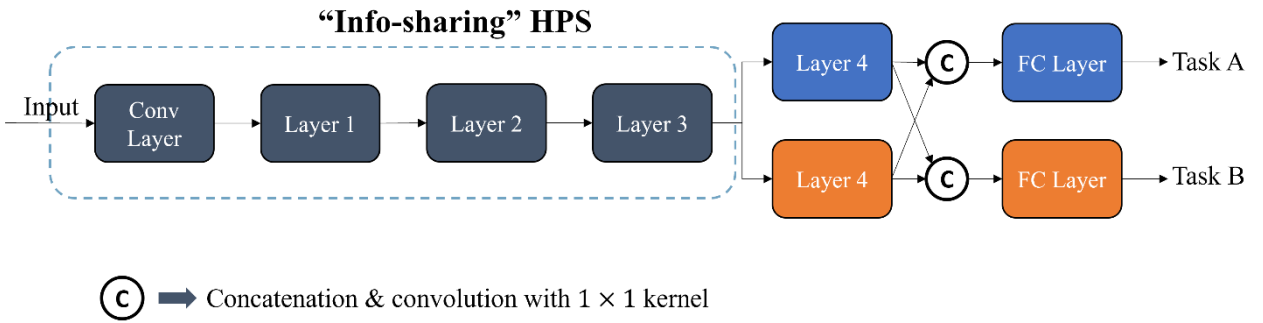


Figure 5.5: Schematic representation of the Info-Sharing HPS Multi-Task Learning model defined in this thesis. What differentiates this model from the Adjusted one is the communication of information among the task streams.

Considering the two task example of Figure 5.5, information sharing begins with a simple concatenation of the feature maps of the fourth layer’s output of each task. Considering the dimensionality of each feature map which has a (H, W, C) shape, where H is the height, W the width, and C the number of channels, the shape of the

concatenated feature maps will be $(H, W, 2C)$. In this manner, each stream possesses not only its own produced feature space, but the one provided by the other task(s) as well.

In an SPS approach, the two extracted representations would be combined together via a weighted sum, giving higher weight to one or another task's representation, such as in the case of Cross-Stitch networks analysed in Chapter 2. The architecture of info-sharing ResNet18, which still lies within the HPS realm, treats the combination of the concatenated feature maps by applying a 1×1 convolution kernel over them. Hence, the resulting representation for each task after the convolution operation can be seen as a weighted linear sum of the corresponding feature maps.

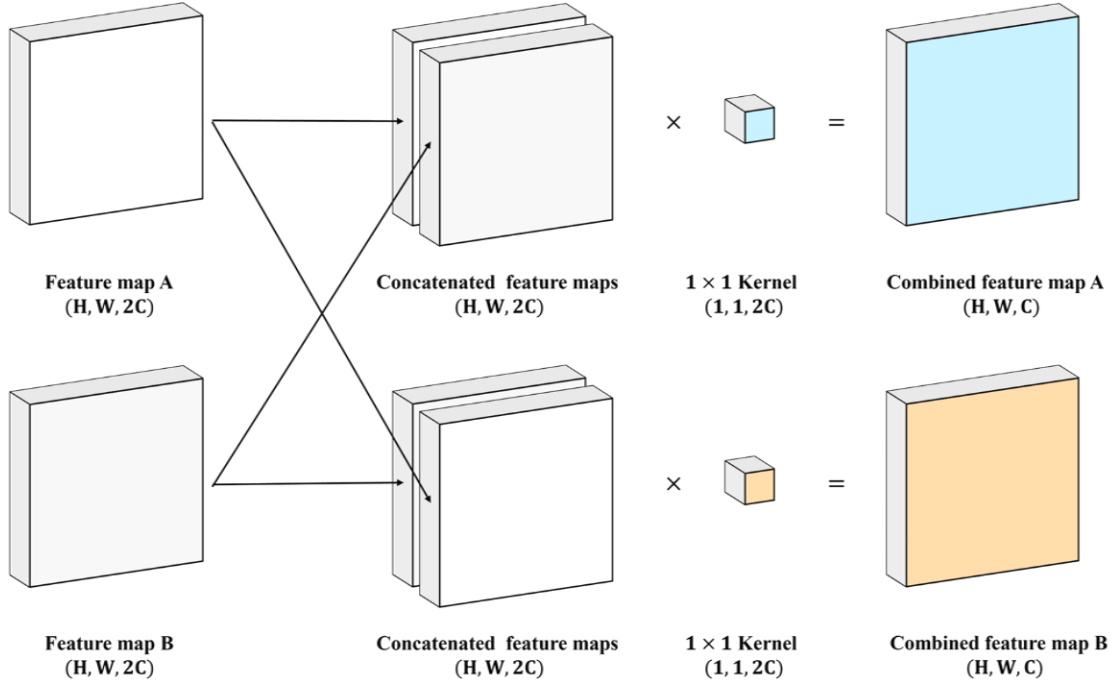


Figure 5.6: Schematic representation of the information sharing module in the Info-sharing HPS model. The output of each Layer 4 (denoted as Feature Map A and B in the figure) in each independent task stream is communicated, concatenated, and passed through a 1×1 convolution kernel. This combines the feature maps and reduces dimensionality.

This operation can be seen in Figure 5.6, where it can be deduced that the unit convolution acts not only as means of combining the two tasks output, but also as a way of reducing the dimensionality induced by the concatenation process. In the case of more than two tasks, the same approach as seen in Figure 5.6 is followed, leading to concatenated feature maps with size (H, W, nC) , where n denotes the number of tasks in the MTL model.

Note that the above MTL approaches have been illustrated using the full ResNet18 as their backbone. As it will be made clear in the next chapter, trimmed, and thus more shallow versions of the same network, proved to perform better under certain circumstances. The above MTL models using such a trimmed network as backbone can be viewed in Appendix B.

5.2.3. Handling of the joint loss and training

A point regarding MTL that has not yet been discussed in this thesis is the treatment of the loss of the model during the training phase. The process of training an STL network was described in Chapter 3. There, the importance of the loss function in the training procedure was highlighted by showing that the weight update in an STL case is

highly based on the single loss function calculated for each epoch. Considering that different tasks require the calculation of different losses, the question that arises is then the following: *how can the multiple losses that are computed based on the number and nature of the defined tasks in an MTL paradigm be combined in a single loss function, in order for the network to be properly trained?*

A simple yet naive answer to the above question is the definition of a total loss, L_{total} , as the sum of all corresponding losses. Bearing in mind the simple two-task models of the previous section, the total loss could be simply defined as:

$$L_{total} = L_{task A} + L_{task B} \quad (5.1)$$

Observing the above equation, three issues may be identified. Firstly, depending on the examined tasks, the two losses may be scaled differently, one being much larger than the other, essentially prioritizing the optimization with respect to the most “lossy” task. Secondly, being potentially fairly different in nature, each task might have different training demands, and defining global hyperparameters (i.e., defining a single learning rate for both tasks) may prove highly inefficient. Finally, the ever-present risk in MTL, the so-called negative transfer, can be a direct consequence of the difficulty in training MTL models. Negative transfer refers to the occurrence where learning multiple tasks at the same time leads to worse performance than learning the same tasks individually with STL models.

As a result, during the development of MTL, the treatment of the loss function of the model has been a point of discussion in the literature. Hence, many approaches for balancing the corresponding losses and suppressing negative transfers have been proposed. A simple and empirical solution for the balancing of the losses is assigning custom weights multiplied by the corresponding calculated losses. This approach is found to be suboptimal when considering that the ratio of the different task losses is not expected to remain constant. To counter this issue more sophisticated loss balancing approaches have been proposed in the literature. For instance, the adaptive balancing of losses of different scales is described in the work of Kendall et al. [63] where calculating the homoscedastic uncertainty of each task allows for learning weights that make the aforementioned balancing possible over consecutive epochs. In this work, the loss values of all tasks considered were found to be of the same order of magnitude and hence weighting them in this context was not required.

Regarding the negative transfer issue, an interesting approach used extensively in this work is the Loss Balanced Task Weighting (LBTW) scheme proposed by Liu et al. [64]. As described by Algorithm 5.1, LBTW follows a batch wise approach, assessing the loss for each batch within each epoch, and defining task weights based on the ratio of the current batch loss to the epoch’s initial batch, raised to the power of α . Then, the total batch loss is defined as the weighted sum of the corresponding losses. In this manner, there is the possibility of “punishing” tasks that are not well trained over each epoch (i.e., the batch loss is larger than the loss of the first batch used as proxy) by assigning larger weight values. The parameter α is defined as a real number ranging from zero to one. The selection of α is arbitrary and based on empirical trial and error. Setting $\alpha = 0$ diminishes the task weights and the whole approach reduces to the naïve sum of losses. It shall be noted that through experimentation the optimal value for α was found to be 0.5, being in line with the conclusions of Liu et al.

Algorithm 5.1: Loss Balanced Task Weighting

```
Given T tasks and parameter  $\alpha$ .
Initialize neural network weights W.
for each epoch i do
  for each batch B do
    Get the loss on each task  $L_B \in \mathbb{R}^T$ 
    Get the first batch loss  $L_{(0,i)} \in \mathbb{R}^T$ 
    for each task t do
      Set the task weight  $w_t = \left( \frac{L_{(b,i)}}{L_{(0,i,T)}} \right)^\alpha$ 
      Update weighted loss  $L_{(B,t)} = L_{(B,t)} \times w_t$ 
    end for
    Update W with respect to  $L_B$ 
  end for
end for
```

5.3. Comparing STL and MTL networks

In assessing the potential gains in performance that MTL provides when used in radar applications, two major setbacks are presented. Firstly, comparing the performance of the models presented in this work to others found in the literature (possible STL cases of the tasks examined during this thesis) is futile due to the very different data employed. Secondly, since no works that tackle the usage of MTL on radar data can be found in open literature, no direct comparisons can be drawn over other approaches. Hence, the MTL models are directly compared to their STL counterparts seen as the most correct baseline rather than the literature.

Having presented both the STL and MTL architectures examined in the scope of this work, it is natural to also provide information on the means of comparison and restrictions that apply. The rest of the section is then dedicated to describing the comparison conditions, presenting restrictions regarding the models as well as the performance metrics employed.

5.3.1. Restrictions regarding the models

One additional reason for selecting a specific backbone network, such as the described ResNet18, was posing limitations in architecture of the proposed models and the definition of a fair platform for performance comparison. The need of constraining the search space for the scope of this thesis is evident, since performing any kind of search procedure at model level would be very costly and time consuming. The use of the proposed backbone network reduces the search procedure to only optimizing the model at hand by hyperparameter fine-tuning. Undoubtedly, this approach may pose some form of setbacks to tasks for which the set backbone is not optimal for. This will be made apparent mostly in the regression tasks of this thesis.

It was mentioned in 5.2.2 that during experimentation trimmed versions of the ResNet18 were found to generalize better in certain situations. To make a fair comparison between STL and MTL models, the following restriction is applied: the layer path from input to output for each task must be identical in both STL and MTL cases. This is showcased in Figure 5.7 where, for instance, the layers regarding the first task in the MTL model are exactly the same as in the STL model (i.e., three layers of any defined type and a fully connected one leading to the prediction). What can be seen as an exception to this rule is the use of the Info-sharing MTL model, which adds a processing

step that cannot be implemented in an STL fashion without being considered as some kind of joint classification approach. Yet, the architectures of the Info-sharing and STL remain comparable.

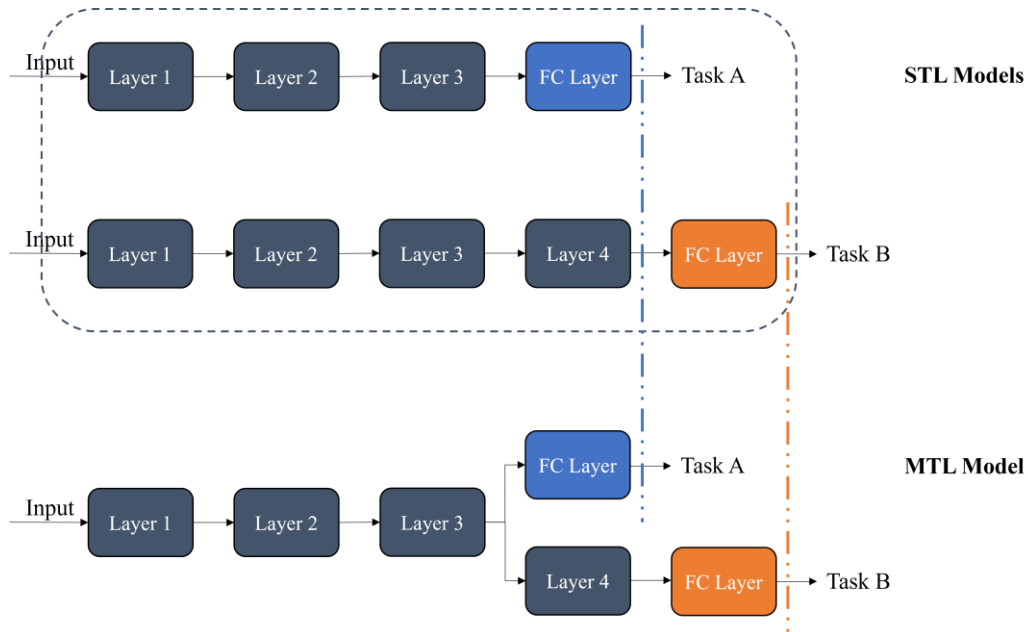


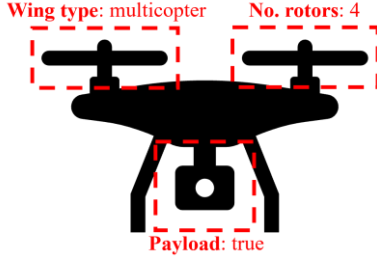
Figure 5.7: The strategy around the comparison of STL and MTL models regarding their architectures: the layer path from input to output for each task must be equal.

What is not constrained is the optimization of each STL or MTL model. The optimization and training process is unique for each model based on the notion that training STL models should be not influenced in any way by the training of MTL models, in order to provide with a fair comparison approach.

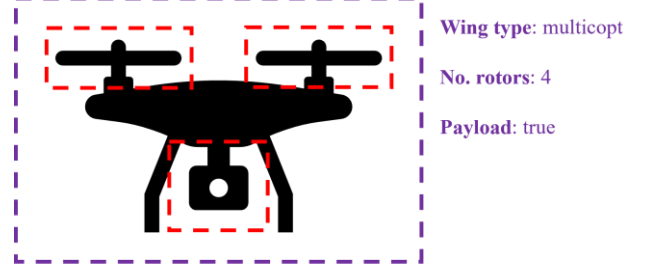
5.3.2. Performance metrics

Before introducing the performance metrics used for comparing the trained classifiers, it is necessary to introduce the ways in which they can be calculated so as to simultaneously assess multiple tasks. When possible, the comparison of the models is done in two ways: task-wise and jointly.

It is often the case in the literature that MTL models are compared to their STL counterparts by comparing their performances in a task-wise manner (i.e. the tasks in [48]). That is, comparisons are made for each task separately, as if they are independent from each other. For example, considering the tasks depicted in Figure 5.8a below, comparisons on three fronts defined by the three tasks (i.e. wing type, number of rotors classification and payload detection) can be carried out. This approach enables the use of different types of performance metrics, thus allowing for a more in-depth analysis and comparison. Another advantage of this approach is that all the tasks examined can be assessed.



a) The task-wise approach framework. Each task is thought of as independent.



b) The joint approach framework. All tasks must be classified correctly simultaneously.

Figure 5.8: The two approaches regarding predictions followed in this work: task-wise and joint.

Since the classification tasks considered in this work are all highly imbalanced, the plain accuracy was deemed as insufficient as a metric to assess the performance of the proposed models. To offer a more concrete view of each model's performance, it was decided that the F_1 score was the most suitable metric. As its name suggests, the F_1 score belongs to a family of metrics called the F-Measures. F-Measures are quite popular metrics for imbalanced classification problems. These, on their side, rely on the calculation of two additional metrics, the so-called precision and recall. In binary classification terms, such as the case in Figure 5.9, precision summarizes the fraction of examples assigned the positive class that belong to the same class, $\frac{TP}{TP+FP} = \frac{TP}{P}$, and can be viewed as the probability of detection P_D . Recall summarizes how well the positive class was predicted and is calculated as the ratio of the correctly classified positive class instances over the total number of predicted as positive samples, $\frac{TP}{TP+FN} = \frac{TP}{PP}$.

		Predicted values	
		PP	PN
True values	P	TP	FN
	N	FP	TN

Figure 5.9: A sample confusion matrix corresponding to a binary classification task. *TP* denotes the true positives, samples that belong to the positive class and were classified correctly as such. *TN* similarly represent samples that both belong and were classified as negative class samples. *FP* and *FN* denote negative samples classified as positive and positive samples classified as negative respectively.

Both precision and recall can be used as metrics, yet they only tell part the story. It is more appropriate then to combine them into a single score that seeks to balance both concerns in a harmonic mean manner, called the F_1 score:

$$F_1 = \frac{2}{\text{precision}^{-1} + \text{recall}^{-1}} = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2TP}{2TP + FP + FN} \quad (5.1)$$

When upscaling to multiclass classification tasks, such as the wing type and number of rotors classification, the F_1 score is still a valid and fair performance metric, but it must now be adapted to the multiple classes present. The Scikit-learn package [65] available in Python provides with two ways of averaging the F_1 score over multiple classes: a weighted approach and a macro-average one. Both ways of averaging begin with calculating the respective scores for each class separately. The weights in the first approach are calculated based on the number of samples each class is represented with, leading to more populated classes receiving higher weights. The macro-average approach assumes equal contribution for all classes, and since from an operational point of view no distinctions are made among the structural characteristics of the drone targets, is selected as the preferred averaging method. An example of the differences in calculating the average F_1 score using the aforementioned methods can be found in Figure 5.10. The reason behind the selection of the macro-averaging, which does not consider the class imbalance, is that all classes are considered of equal importance and the effect of misclassifying a considerable amount of samples belonging to the minority classes is more apparent.

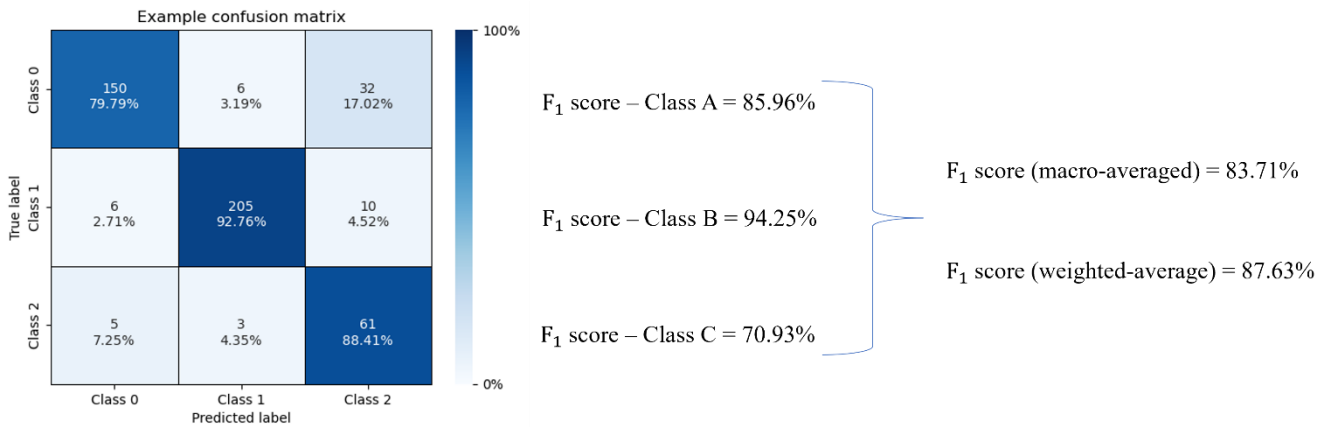


Figure 5.10: Comparison between the macro and weighted-average F_1 scores. Notice that the macro-averaged score is lower due to it being a simple mean of the corresponding scores of each class. In this manner, minority classes contribute equally to the final score.

For the binary classification tasks, such as the payload detector one, the F_1 score remains a fair and appropriate performance metric for comparison, yet it is calculated considering a certain threshold of 0.5 for the class assignment at each network's output. On the contrary, the so-called Precision-Recall (PR) curve, shown in Figure 5.11, illustrates the trade-off between the precision and recall metrics when different threshold values are applied. For instance, in the plot of Figure 5.11, one may notice that with certain thresholding, the classifier can achieve 100% precision and 60% recall scores at the same time. The point of comparison between classifiers when considering the PR curve can be the area under the curve (AUC) calculated by integrating the PR curve. A high AUC indicates possibly high recall and precision, while a low AUC value indicates the opposite. Due to design, the PR curve is highly sensitive when applied to imbalanced problems which makes it ideal for the purposes of this work.

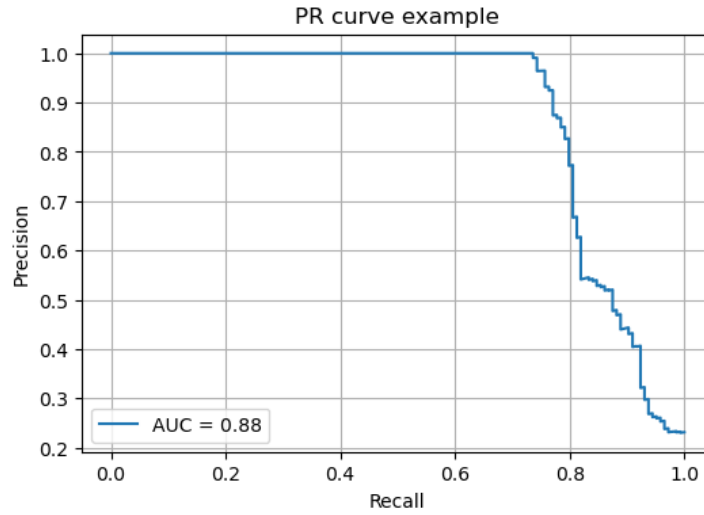


Figure 5.11: Example of a PR curve and the corresponding AUC.

The classification tasks supported by the drone configuration datasets additionally allow for a joint performance assessment over the examined tasks. Illustrated in Figure 5.8b in this joint approach, the predictions of all STL models and all streams in the MTL case are combined and treated as a single one. Hence, in order for a prediction instance to be considered correct, all tasks' predictions shall be correct. This approach may additionally highlight any advantages of jointly training tasks, especially in cases where the task-wise performances are similar. On the downside, the only analysis that can be carried out in this case is reliant solely on the accuracy metric, calculating the number of correct instances over the total number of samples in the corresponding test set.

For the mean rotor rate regression task, the MSE and MAE scores as described in Section 3.2.3 are used as metrics for comparison between STL and MTL models.

5.4. Description of the testing configurations

The diversity and the limited size of the available dataset as described in Chapter 4 makes the testing of both STL and MTL models a fairly challenging task. A question that arises in this case is the following: *how does one create a bias-free training and testing environment that can also highlight the robustness of the classifiers?*

The answer to this question is not straightforward, since one has to take into account the varying SNR, target and measurement scenarios distribution within and among different measurement campaigns. That is:

- Different measurement campaigns often contain measurements of different targets and scenarios.
- It occurred that certain targets and scenarios were captured in a single track measurement.
- The SNR conditions throughout the measurement campaigns vary, being more favourable for the classification of certain targets than others.

Therefore, it is obvious that a single train/test split approach is not sufficient for completely assessing the performance of the classifiers. Two different in nature testing configurations are proposed:

- An iterative train/test split approach imitating the well-known cross-validation process [66].
- The use of handmade train/test sets, tailor made for each task group, so as to assess the robustness of the classifiers across targets (whether previously unseen or not) in various measurement scenarios, campaigns etc.

5.4.1. Tests on different train/test splits

The size and variability of the used data makes a simple train/test split susceptible to external biases when manually, or even randomly, dividing the data into train and test set: it might occur that the test set comprises of the hardest targets to classify and vice versa. Thus, the results can be either too optimistic or too underwhelming, depending on the types of targets and the SNR regimes found in the test set.

To mitigate this issue, a recursive train/test split method, heavily inspired by the well-known cross-validation approach is proposed. As dictated by Figure 5.12, the dataset containing 0.5 s spectrograms is first shuffled. After shuffling, ten unique and non-overlapping subsets are defined. Then, ten models are trained from scratch and tested on the corresponding subset acting as test set.

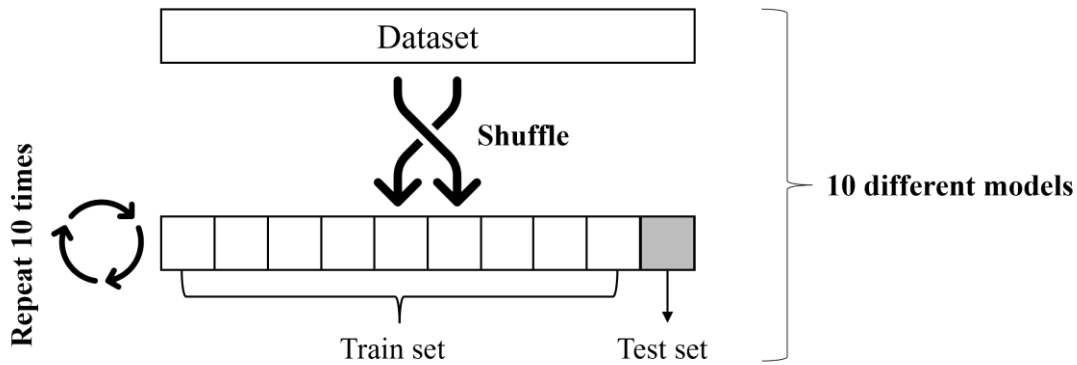


Figure 5.12: Schematic representation of the iterative testing approach. The provided dataset is first shuffled and then split in ten distinct and unique subsets. Each subset is used as test set, while the nine others as training set. In the end, ten different models are trained and tested through this method that resembles the well-known cross-validation method.

This method uses whole datasets as its basis for assessing the performance of STL and MTL models. Having introduced the three dataset used for experimentation (see Chapter 4), each one of them is used to examine the performance of the classifiers over a number of tasks. Table 5.1 that follows summarizes the tasks examined using each created dataset. Each task combination is assessed by the corresponding number of STL model and three MTL ones. Hence, for example, when examining the wing type and number of rotor classification tasks, two STL models are created, one for each task, as well as the three types of MTL models described earlier in this chapter. Finally, the statistics of the performance metrics over the ten splits are calculated and serve as means of comparison among the models.

Dataset	Task groups
Drone configuration dataset (A)	<ul style="list-style-type: none"> WT & NR WT, NR & PD
Drone configuration dataset + multiple drone cases (B)	<ul style="list-style-type: none"> WT & NR (multiple UAV cases treated solely as a separate class in number of rotors task) WT, NR & MDD
Drone flight characteristics dataset (C)	<ul style="list-style-type: none"> RRR & PD RRR & ED

Table 5.1: The task groups subjected to the iterative train/test approach and the corresponding datasets that support them. The different datasets were defined in Section 4.3.1 and the task groups earlier in Section 5.1.

5.4.2. Robustness tests

The testing configuration presented in 5.4.1 manages to assess the performance of the classifiers in different train/test split scenarios, minimizing the effect of bias by performing multiple unique splits, training, testing and finally obtaining statistics over the models' performances. However, this approach comes with a serious flaw: since the train/test split of each fold is performed on the segmented spectrograms level, spectrograms belonging to the same track can be found in both the training and test set. This results in the models learning and adapting to different SNR conditions for the same targets, leading to possibly overly optimistic outcomes.

In an effort to assess the performance of all models in every task combination scenario possible, the datasets presented in Chapter 4 are manually split in train and test sets in order to assess generalization over unseen measurement scenarios or even targets. Not all tasks can be subjected to a robustness test. For instance, since all the spectrograms of the original dataset containing multiple drones come from essentially the same measurement, no robustness test can be defined for this task. Table 5.2 presents a description of the train/test scenarios and the corresponding targets present in each test set. More information about the number of samples per target in these test sets can be found in each of the dedicated subsections of the experiments in Chapter 6.

5.5. Summary

The lack of available literature on radar-based drone characterization using MTL as well as the uniqueness of the available experimental dataset, require the definition and creation of a well-suited methodology for testing the proposed classifiers. The first step in forming a testing framework for MTL in the context of drone characterization with radar is the definition of the task groups in a manner that would appear reasonable to an expert human and consider possible practical/operational usefulness for situational awareness.

In this context, five task groups were defined:

1. Wing type and number of rotors classification.
2. Wing type, number of rotors classification and payload detection.
3. Wing type, number of rotors classification and multiple drone detection.
4. Rotor rate regression and payload detection.
5. Rotor rate regression and event detection.

Due to its resilience in overfitting and in order to limit the optimization search space, the ResNet18 and its possible custom-made derived architectures were selected as the backbone for the MTL models and the baseline STL model for each task. Three MTL models were introduced in this chapter in order to assess different MTL architectures, namely the "Simple" HPS, the Info-sharing, and the Adjusted MTL models.

The issue of dealing with multiple losses and their different needs in the MTL context is addressed with the use of a batch-wise task weighting method, named LBTW, which aims to reduce the detrimental phenomenon of negative transfer for MTL.

To account for the inevitable imbalance in the available datasets, the plain accuracy was abandoned as a performance metric, and in its place the F_1 score and the AUC from PR curves were selected instead. In order for fair and unbiased comparisons to be made, two types of tests are proposed: one inspired from the well-known cross-validation method, and one assessing the robustness of the classifiers in each task group possible by creating tailor-made train/test splits. In the upcoming chapter, the results of the aforementioned experiments will be presented.

Task Group	Train/Test scenario	Test set targets description
WT & NR	<u>Test set</u> : a whole measurement campaign containing commercially available drones. <u>Training set</u> : the rest of the measurement campaigns – dataset A.	<ul style="list-style-type: none"> • Single rotor fixed wing drone • A helicopter drone • A quadcopter drone
	<u>Test set</u> : mostly non commercially available, drone targets were entirely removed and placed in the test set. Each one possesses as certain distinct structural or scenario characteristic. <u>Training set</u> : the rest of dataset A.	<ul style="list-style-type: none"> • A single rotor fixed wing drone • A fixed wing drone with two rotors one behind the other • A flying wing drone with four rotors • A custom made quadcopter • A three-arm hexacopter • An octocopter with a person walking under its flight path
WT & NR & PD	<u>Test set</u> : an octocopter drone carrying some sort of payload, a fixed wing and a quadcopter without payload. <u>Training set</u> : The rest of dataset A.	<ul style="list-style-type: none"> • A fixed wing drone without payload • A quadcopter without payload • An octocopter with heavy payload
	<u>Test set</u> : the same octocopter case as previously accompanied by the “mainstream” targets of the first robustness test. <u>Training set</u> : the rest of dataset A.	<ul style="list-style-type: none"> • Single rotor fixed wing drone • A helicopter drone • A quadcopter drone • An octocopter with heavy payload
WT & NR & MDD	Not applicable	Not applicable
RRR & PD	The same targets as in the second WT & NR & PD task group robustness test were selected and used as test set.	<ul style="list-style-type: none"> • A fixed wing drone without payload • A quadcopter without payload • An octocopter with heavy payload
RRR & ED	<u>Test set</u> : a small and random combination of measurements. <u>Training set</u> : the rest of dataset C.	<ul style="list-style-type: none"> • A custom-built quadcopter • A flying wing with four propellers • A custom VTOL drone with four propellers • A fixed wing drone • An octocopter with heavy payload

Table 5.2: Description of the splitting process leading to the defined robustness tests. The types of targets present in each test set are also described. More information on the number of samples of each target in the test set is provided further in Chapter 6.

6 Results: drone structural configuration characteristics estimation

*The methodology introduced in Chapter 5 is utilized in this chapter to assess the performance of the proposed classifiers in a variety of different task groups. The task groups are formed by combining the wing type classification, number of rotors classification, multiple drone, and payload detection tasks. The proposed classifiers are assessed through two different testing methods, so as to fully characterize the possible gains of utilizing the MTL approach. In this manner, **Section 6.1** analyses the performance of the two approaches through their respective classifiers in the wing type and number of rotors classification task group. In **Section 6.2** a third task is added to the previous two, and the two paradigms are assessed over the wing type, number of rotors classification and multiple drone detection tasks. **Section 6.3** sees the addition of payload detection as a third task instead of the multiple drone detection one. Finally, **Section 6.4** concludes the chapter.*

6.1. Wing Type & Number of Rotors

The task combination first presented in this chapter is comprised of the wing type and number of rotors classification tasks. To assess these two tasks, the two different datasets (datasets A and B as defined in Chapter 4) containing information regarding mainly the structural characteristics of the targets are utilized. As described in Chapter 5, for each test type two STL (one for each task) and three MTL models (“Simple” HPS, Info-sharing MTL and Adjusted MTL) are defined and compared.

The iterative training and testing approach provides some initial insights on the classification capabilities of the proposed STL and MTL models. Both dataset A and B are utilized, the difference among them being the spectrogram cases with multiple drones being present in the latter. Then, two robustness tests take place, each one assessing the classification performance of each model in different target and measurement scenarios. For both of these robustness tests, only dataset A was used.

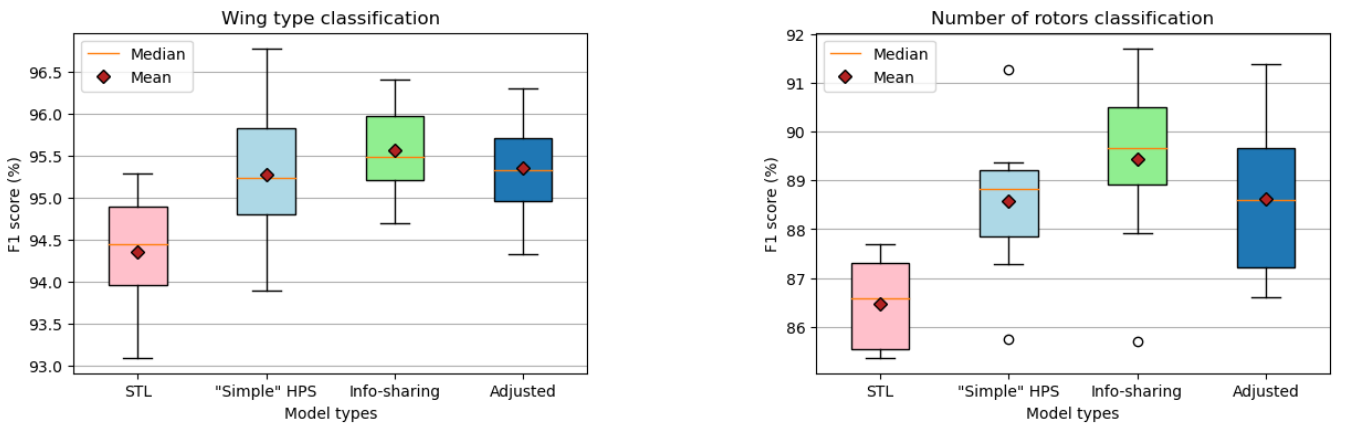
6.1.1. Testing on different train/test splits

Test on Dataset A

The cross-validation based testing method can provide some initial insight regarding the classification capabilities of the proposed classifiers. The boxplots of Figures 6.1a and 6.1b present the statistics of the F_1 score over the ten splits for the wing type and number of rotors classification tasks, as introduced in Chapter 5. By examining the aforementioned plots, it is evident that the use of the MTL paradigm for these two classification tasks can be profitable for both. This appears to be true when comparing task-wise as well as jointly, as dictated by the joint accuracy comparison plot of Figure 6.2.

Initially, the classifiers are assessed on Dataset A, a drone configuration dataset that does not include multiple drone targets (i.e., each spectrogram contains measurements corresponding to a single drone). Examining the wing type classification task, the STL model performs considerably well as it achieves an average F_1 score of 94.36%. Even though the average performance of the single task ResNet18 is satisfactory by itself, further improvement is made on it by using the MTL networks. More specifically, a statistically significant increase of 1.2% can be observed when using the Info-sharing HPS model. Additional, but somewhat less significant improvements of 0.91% and 0.99% can be observed when using the “Simple” and Adjusted HPS models.

Improvements in classification performance when employing the MTL paradigm can be further traced in the second task examined in this Section, the number of rotors classification. In this task the improvement achieved by the use of Multi Task Learning is more evident, observing an F_1 score increase of up to 2.98% when using the Info-sharing model, which achieves a score of 89.45%. At the same time, the ResNet18 model acting as the STL baseline in this case falls short by achieving a respective score of 86.47%. It is worth mentioning that once again, all MTL models performed better, as suggested by the boxplot of Figure 6.1b.



a) Wing type classification task F_1 score boxplot comparison.

b) Number of rotors classification task F_1 score boxplot comparison

Figure 6.1: The iterative train/test performance statistics of the proposed classifiers applied on Dataset A for the wing type and number of rotors classification tasks.

As previously mentioned, the STL and MTL approaches are additionally compared in a joint classification manner. In this task group, the joint accuracy is defined as the number of samples for which both the wing type and number of rotors aspects have been classified correctly, divided by the total number of samples. Hence, it is expected that the maximum possible joint accuracy in each fold will be, in the best case scenario, equal to the accuracy of the least well performing task (in this case the number of rotors).

As expected, since the performance of all model types in the wing type classification task is comparable, the joint accuracy follows the performance pattern of the second task. The joint STL predictions see a fall in accuracy, as the two models combine for an average accuracy score of 83.41% as seen in Figure 6.2. As expected, all MTL model variants achieve superior joint accuracy, with the highest one belonging to the same model as seen previously at 87.94%, i.e., a 4.53% increase over its STL counterpart. It is also worth mentioning that the improvement in performance that MTL achieves when applied to this task group is consistent throughout all performance metrics in their macro-averaged form as seen in Table C.1 of Appendix C. The same consistency in performance was also observed among the different folds, where the MTL models achieved better performances in the majority of the splits.

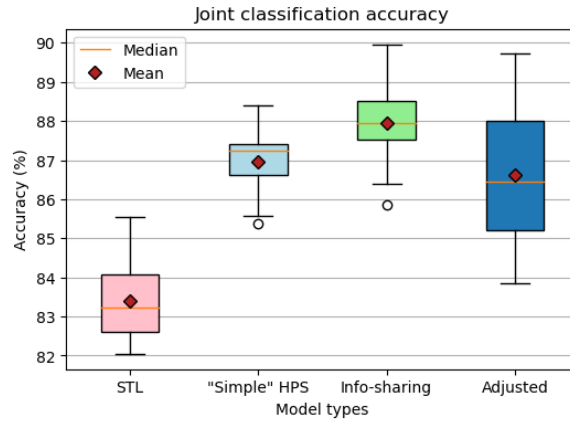
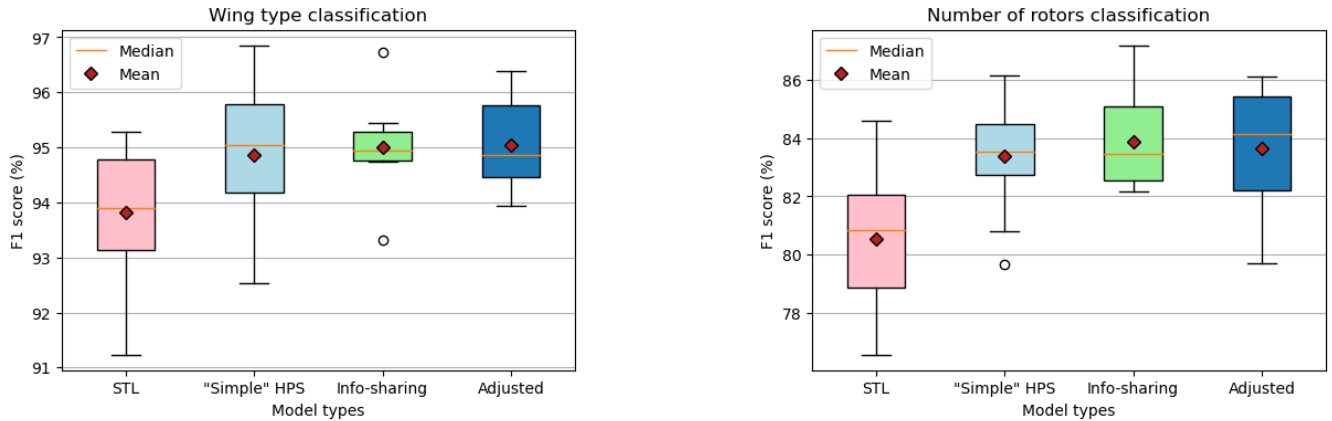


Figure 6.2: The joint classification accuracy statistics over the same ten folds as in the boxplots of Figure 6.1. STL and three different proposed MTL approaches are compared.

Test on Dataset B

As mentioned in the description of the available data in Chapter 4, there are instances where two multicopter targets, a quadcopter and an octocopter, are captured simultaneously in the same spectrogram. As discussed, this enables experimentation regarding the detection of the co-existence of multiple drone targets. As a first step, in this Section the multiple drones are represented as follows: they are characterized as multicopter wing type drones and the number of rotors is set to the sum of the corresponding number of each drone. Therefore, in this case, the number of rotors for these samples is set to twelve, creating a separate class in the number of rotors classification task. As it will be made clear, the introduction of these special cases in this manner involves the risk of inducing a significant amount of class overlap.

Observing the plots of Figure 6.3, the effects of adding this separate class are evident. A degradation in performance for both STL and MTL models is seen in both tasks, with the number of rotors being the one affected the most.



a) Wing type classification task F_1 score boxplot comparison.

b) Number of rotors classification task F_1 score boxplot comparison.

Figure 6.3 : The iterative train/test performance statistics of the proposed classifiers applied on Dataset B for the wing type and number of rotors classification tasks.

Even though the performance of all models in F_1 score terms suffers a considerable drop, the MTL models still manage to perform better than their single task counterparts in both tasks. The Adjusted HPS model achieves the highest average score in wing type classification, it being 95.04%, bringing a 1.23% improvement over the STL model. The Info-sharing model, which was the best performing model in the dataset where multiple drone cases

are absent, also performs considerably well with its performance being almost equal to the Adjusted model's one. Overall, on average all MTL models provided equivalent scores, albeit introducing some variance in certain instances such as the simple HPS approach.

Some variance is also introduced in the number of rotors classification task alongside the drop in F_1 score, where for instance a drop of 5.92% is observed in the STL case when comparing to the previous case which had no multiple drone samples. An increase in performance comparable to the previous case is also observed even with the addition of a new and possibly overlapping class. More specifically, the Info-sharing model provides an increase of 3.35% by achieving an F_1 score of 83.9% (a drop of 5.55% comparing to the single target case presented previously). The rest of the MTL models also achieve superior scores, almost equivalent to the Info-sharing's one.

The reason behind this drop in performance that is observed in this approach can be attributed to the aforementioned class overlap that is induced to the dataset alongside the introduction of the multiple drone cases as a separate class. This issue can be easily identified by observing the sample confusion matrix belonging to a representative fold that was derived from the predictions of the STL model case. In the confusion matrix of Figure 6.4, the effects of the aforementioned class overlap can be viewed as the main cause of misclassifications regarding the newly added class with respect to the one of the octocopter.

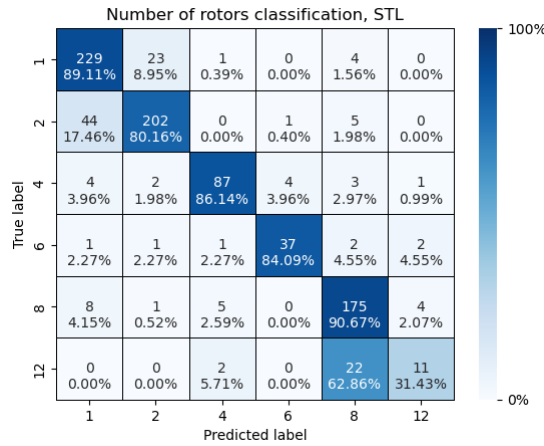


Figure 6.4: Sample number of rotors classification confusion matrix for the iterative train/test testing approach over Dataset B using an STL model. The confusion induced by class overlap due to the additional twelve rotor class is evident. Most twelve rotor samples are eventually classified as octocopters.

Considering the physical nature of the measurement, with one drone leading and the other following closely, the possible reasons behind the above misclassifications become more clear: the octocopter, having larger rotors due to its design, possibly produces more intense backscattering which is captured by the radar and translated into the spectrogram's Micro-Doppler signature. The second reason behind the possible "domination" of the octocopter in signature terms is once again associated to the relative size of the two drones. The octocopter, being considerably large and leading the formation, may be able to cause an electromagnetics related phenomenon called shading, potentially preventing any returns from the smaller quadcopter. A comparison between a single octocopter and a spectrogram containing the two preceding targets can be viewed in the spectrograms of Figure 6.5, where both look very similar, at least to the human eye. It should be noted in any case that not all segments of the multiple drone measurement are dominated by the octocopter signature. It is very likely that a number of them contain additional Micro-Doppler components produced by the quadcopter's rotors, making the classification of these cases as a spectrogram representing twelve rotors feasible.

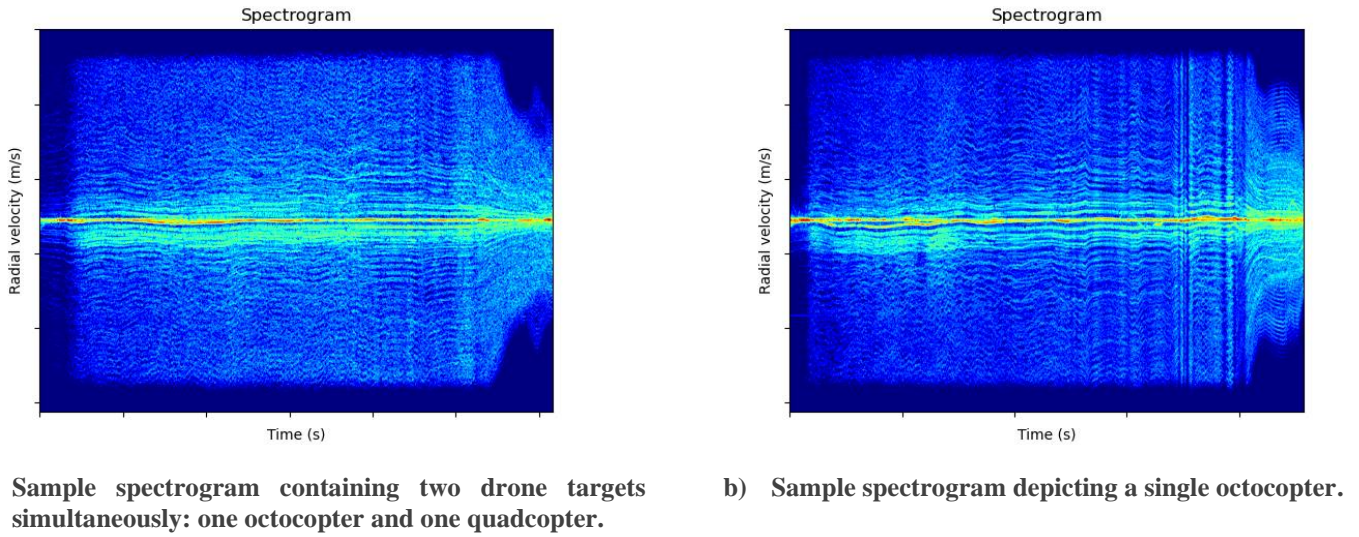


Figure 6.5: Comparison between a two-target spectrogram (left) and one containing a single octocopter (right). Both cases would appear very similar, at least to the human eye.

As done previously, the joint classification accuracy is also assessed in this scenario. Using the same reasoning as before, it can be foreseen that the joint accuracy will follow the performance of the worst performing task. Hence, it is made clear by the boxplot of Figure 6.6 that MTL models provide a consistent and statistically significant improvement in joint task performance over their STL counterparts. The best performing model is found to be the Info-sharing one which manages to achieve an accuracy of 84.45%, providing an improvement of 4.27% over the baseline models.

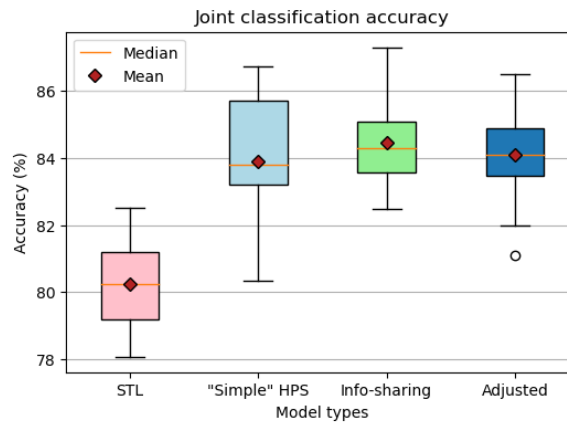


Figure 6.6: The joint classification accuracy statistics over the same ten folds as in the boxplots of Figure 6.3. Note the drop in performance compared to the case of Figure 6.2. STL and three different proposed MTL approaches are compared.

A more in-depth view of the performance of all classifiers for this case is found in Table C.2 of Appendix C. This fold-based assessment approach provides initial insights regarding the possible benefits of using MTL models over defining multiple STL ones.

However, it was stated in Chapter 5 that this method provides potentially highly optimistic results, since segments of the same measurements may be found in both the training and test sets. Hence, what follows is the robustness tests defined for assessing the proposed models in the wing type and number of rotors tasks.

6.1.2. Robustness tests

Testing on commercially available targets

To better assess the performance of the proposed classifiers in a more realistic manner, two robustness tests were defined, each one covering specific aspects of the highly varying dataset. The first test assesses the performance of the models on commercially available, “mainstream”, drone targets that belong to a whole measurement campaign treated as the test set. The purpose of this test is to assess the generalization capabilities of the classifiers when facing potentially seen-before types of targets, but in different SNR and flight regimes. A summary of the targets present in the test set can be found in Table 6.1.

Description of target	Wing type	Number of rotors	Number of samples
Single rotor fixed wing drone	Fixed wing	1	188
Helicopter drone	Helicopter	2	221
Quadcopter drone	Multicopter	4	69

Table 6.1: The content of the test set of the Robustness test on targets seen before but in different conditions. All targets in the test set belong to a separate measurement campaign, making their SNR conditions fairly unique.

To assess the difference in generalization capabilities, in the two robustness tests of this task group a cropped version of the ResNet18 was also utilized. As described in Chapter 5, the original ResNet18 architecture is cropped and the convolutional layers after Layer 2 are omitted. Observing the joint accuracy comparison plot of Figure 6.7, two points may be made: the use of the cropped ResNet18 provided with the best performing MTL model, yet it did not prove particularly beneficial in the STL case. Furthermore, the MTL models were found to perform consistently better, or at least as well as the STL baseline. An increase in joint classification accuracy of 5.22% when using the “Simple” HPS model with the cropped version of the ResNet18 as its backbone.

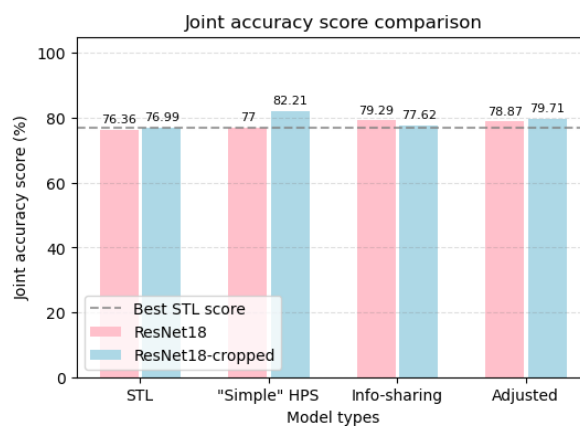


Figure 6.7: The joint accuracy performance comparison between the MTL models and their STL counterparts. The MTL networks perform consistently better, or at least as well as the baseline. STL and three different proposed MTL approaches are compared in this robustness test.

Testing on “exotic” and unseen targets

The second robustness test of this task group aims to assess the performance of the proposed models on not only previously unseen targets, but targets that possess some structural or scenario distinctiveness as well, hence they are nicknamed to be ‘exotic’. The targets that were drawn from the dataset and treated as test set in this case are shown in Table 6.2.

Description of target	Wing type	Number of rotors	Number of samples
Single pusher propeller drone	Fixed wing	1	323
Fixed wing drone with two rotors one behind the other	Fixed wing	2	49
Custom built VTOL flying wing drone	Fixed wing	4	121
Hexacopter with paired rotors	Multicopter	6	168
Octocopter with person walking underneath	Multicopter	8	558

Table 6.2: The content of the test set of the robustness test on targets never seen before, and “exotic” targets with either a distinctive structural or measurement scenario characteristic. These targets are excluded from the training set in their entirety, and each comes from all available measurement campaigns.

Once again, the networks assessed are optimized via manual search (more details may be found in Table C.5 of Appendix C in terms of their hyperparameters) and both the full ResNet18 and its cropped version are used. Here, the advantages that come with the use of a shallower network are evident. Using the cropped version of the backbone network proved crucial since it provided a rapid increase in performance of up to almost 12% for the STL case. Comparing the two paradigms, the MTL approach seems to be constantly on top of its STL baseline, with the Adjusted MTL model based on the cropped version of the ResNet18 backbone achieving the highest joint accuracy score among all models, as seen in Figure 6.8.

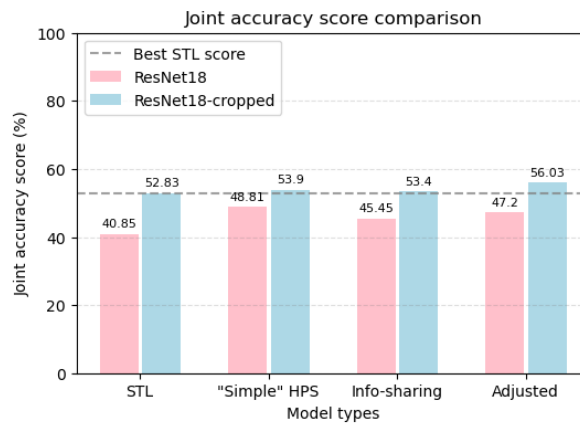
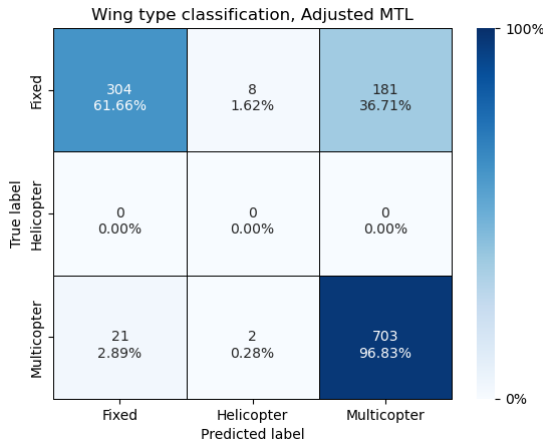


Figure 6.8: Joint accuracy score comparison plot. The Adjusted MTL model based on the cropped version of the ResNet18 backbone achieves the highest accuracy among all models. STL and three different proposed MTL approaches are compared in this robustness test.

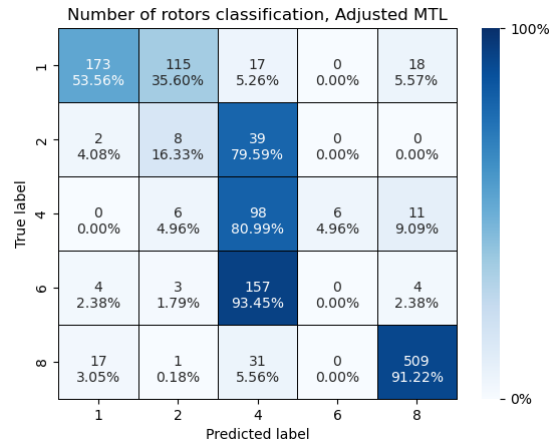
At first glance, it would appear that the use of either paradigm does not yield accurate results. Yet, some considerations shall be made, since the presented joint accuracy does not convey the task-wise advantages and the

per class classification performance of each approach. For a deeper analysis of this test, consider the confusion matrices for both tasks belonging to the Adjusted model with the cropped ResNet18 as backbone, which was shown to be the best performing one. Inspecting these confusion matrices in Figure 6.9, the following can be observed:

1. Only 16.33% of the fixed wing drone with twin rotors case is classified correctly, most of the misclassifications being attributed to the target being classified as a quadcopter in number of rotors terms.
2. The number of rotors of the four-rotor flying wing drone are mostly classified correctly, yet 117 out of the total 168 (69.64%) of such samples are classified as multicopter. This fact can be attributed to the similarities found in the spectrograms of this flying wing drone, and the ones belonging to quadcopters due to the number of rotors and the Micro-Doppler signatures they produce. This, alongside the next observation, were the two recurring themes across all models.
3. No instances of the three-armed hexacopter were classified correctly. Moreover, almost all of the hexacopter cases are classified as having four rotors, something that may be attributed to the structural characteristics of the drone. This was the case for every classifier.
4. The existence of a secondary target (in this case a person walking under the octocopter while flying) does not seem to affect the classification capabilities of the number of rotors task part of the network, with the classifier classifying 91.22% of the octocopter samples in the dataset correctly.



a) Wing type classification confusion matrix. The second row contains only zeros due to the test set containing no helicopter drones.



b) Number of rotors classification confusion matrix. The Adjusted MTL model misclassified 93.45% of all three-arm hexacopter drone samples.

Figure 6.9: Confusion matrices for both WT and NR tasks regarding the “exotic” target classification belonging to the Adjusted MTL model, the best performing one on the Robustness test in figure 6.8.

Task wise, the Adjusted model achieves a wing type accuracy of 82.61% and a number of rotors accuracy of 64.64%, compared to the respective accuracies of 82.85% and 61.94% of the STL model. Regarding the failure to classify the hexacopter, the spectrograms presented in Figure 6.10 may provide some insight in the reasons behind it. It is clear by comparing the spectral lines of all three spectrograms that the ones belonging to the three-armed hexacopter (Figure 6.10a) and the quadcopter (Figure 6.10c) are fairly similar. On contrary, the spectral lines of the hexacopters found in the training set, such as the one of Figure 6.10b, are denser, potentially due to the six rotors having larger variance in their rotation rates. The three-armed hexacopter may have different motors on each arm, yet due to its design they might either rotate at the same rate or the need of adjusting individual rotor rates might be diminished.

Furthermore, it can be observed that the increase in performance of the MTL cases is based on the better classification performance on the more “mainstream” and commercial targets such as the single rotor fixed wing drone and the octocopter.

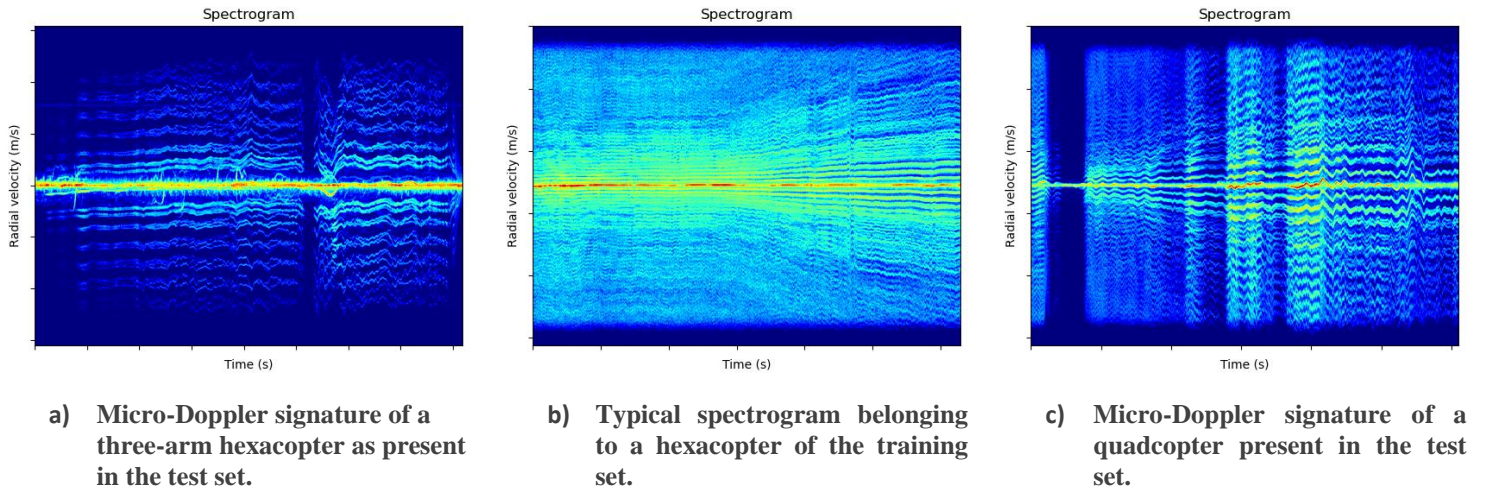


Figure 6.10: The possible root of confusion between the three-arm hexacopter (left) and the quadcopters (right) in the test set. The former, due to its structural characteristics produced Micro-Doppler signatures resembling more a quadcopter than the hexacopters present in the training set (center).

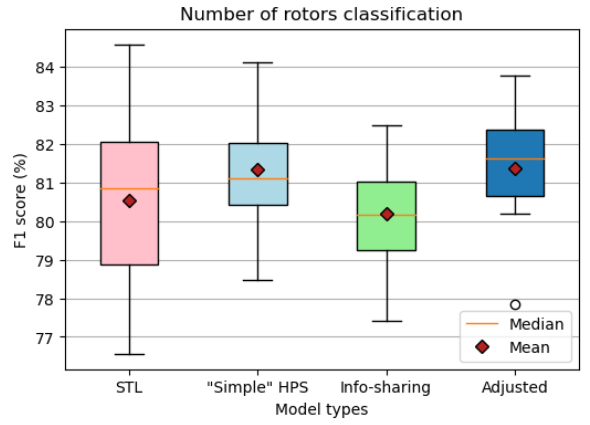
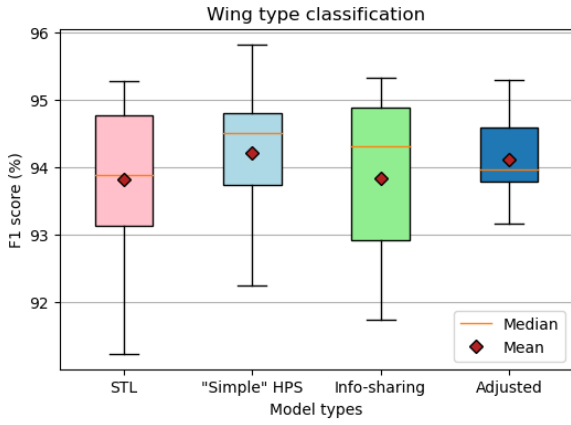
6.2. Wing Type, Number of Rotors & Multiple Drone Detector

The previous section dealt with the wing type and number of rotors tasks. It was discussed in Chapter 4 that within the provided dataset, there are measurements (and thus spectrograms) that contain more than one targets. It was additionally mentioned in Chapter 5 that treating these cases as 12 rotor rotary wing drones could possibly provide useful hints to a multiple drone detector when trained jointly with the aforementioned tasks.

In this section, these three tasks (wing type classification, number of rotors classification and multiple drone detection) are assessed only through the iterative train/test split scheme due to inability of creating a suitable split for robustness testing. Dataset B was utilized in order to perform the iterative test. In this case, no retraining and testing of the STL models for wing type and number of rotors classification was needed since both are readily available from the previous experiment (the iterative train/test on Dataset B of Section 6.1). Hence, for the needs of this test, in every iteration, one STL model and three MTL models are trained and tested.

The boxplots of Figures 6.11a and 6.11b showcase the effects of possible negative transfer induced by the new task included with the other two that were previously examined. It would appear that for the wing type and number of rotors classification tasks, the addition of the multiple drone detection task proves to be detrimental, since the advantages of using the MTL models are diminished.

One may find MTL models that perform better on average in these two tasks, but the increase in F_1 score is not significant. Regarding the number of rotors task, the misclassifications attributed to class ambiguity, which was identified as a major issue previously, were increased. This fact would appear to be counter-intuitive since the third task, the multiple drone detector, seems to be related to the 12 rotor class of the number of rotors classification. The effects of this are present through all MTL models, especially in the Info-sharing one that obtains eventually equivalent performance in the wing type classification task, and even drops below the baseline in the number of rotors one by 0.59%.



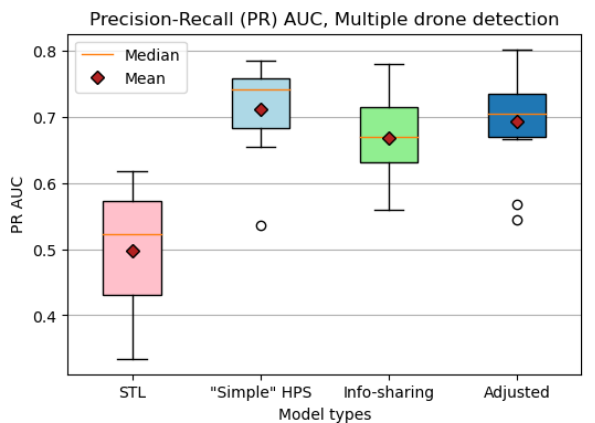
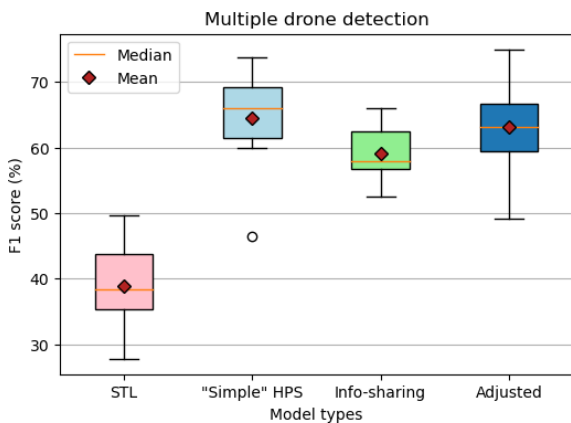
a) Wing type classification task F_1 score boxplot comparison.

b) Number of rotors classification task F_1 score boxplot comparison.

Figure 6.11: The iterative train/test performance statistics of the proposed classifiers applied on Dataset B for the wing type and number of rotors classification tasks in this three-task group. The drop in performance when comparing to Figure 6.3 regarding the MTL models is notable. STL and three different MTL models are compared.

Contrary to the number of rotors task, the multiple drone detection one seems to be highly benefitting by the joint training with the rest of the tasks in this task group. The use of a single task dedicated model, the STL ResNet18 model in this case, led to a very large number of false alarms. This is translated to an extremely poor and prohibitively low F1 score as seen in Figure 6.12a. Compared to this, every MTL model proved to be better in multiple drone detection. More specifically, the “Simple” HPS model achieved an average score of 64.57%, a major 25.65% improvement over the STL model’s performance of 38.92%.

The performance in F_1 score terms may still remain low, yet it is much improved over the one observed when using the STL approach. In settings like this, one has to take into consideration also the serious imbalance between the two classes as shown in Chapter 4, and the size of the available dataset which leads to misclassifications seriously impacting the recall and precision of the classifiers. The advantages of employing the MTL paradigm also extend to the Precision-Recall (PR) AUC seen in Figure 6.12d. There, a difference of 34.71%, from 0.5 for the STL model to 0.71 when using the “Simple” HPS model, can be viewed. This further establishes the MTL as a preferred method when carrying out the task of multiple drone detection. Table C.3 of the corresponding Appendix offers a more in-depth view of the proposed classifiers’ performance across various metrics.



a) Multiple drone detection task F_1 score boxplot comparison.

b) Multiple drone detection task AUC score boxplot comparison.

Figure 6.12: The iterative train/test performance statistics of the proposed classifiers applied on Dataset B for the multiple drone detection task of the three-task group presented in this Section. All MTL models provide with substantial increase in performance in both F_1 and PR AUC metrics. STL and three different proposed MTL approaches are compared.

Even though the performance of the MTL models in the wing type and number of rotors classification tasks saw a drop in this three-task problem, the improvement in performance in the additional task is translated into a further improvement in joint accuracy. The “Simple” HPS model, being the overall best performing MTL model, classifies on average 81.42% of the test samples correctly along all three tasks. At the same time, the huge number of false alarms that the STL model had for multiple drone detection led to an also low joint accuracy score of 71.36%.

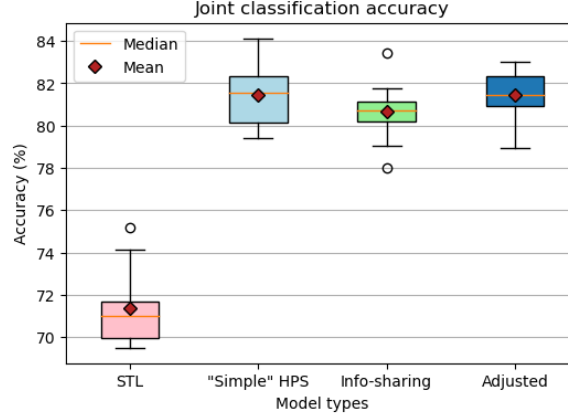


Figure 6.13: The joint classification accuracy statistics over the same ten folds as in the boxplots of Figure 6.11 and 6.12a. The superior performance of the MTL networks in the multiple drone detection task of the three-task group translated to a higher joint accuracy. STL and three different proposed MTL approaches are compared.

6.3. Wing Type, Number of Rotors & Payload Detection

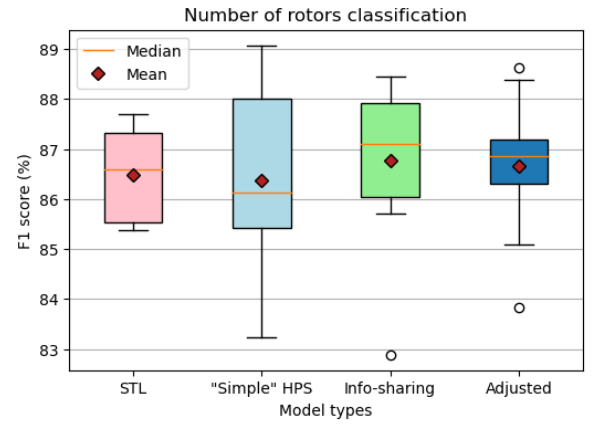
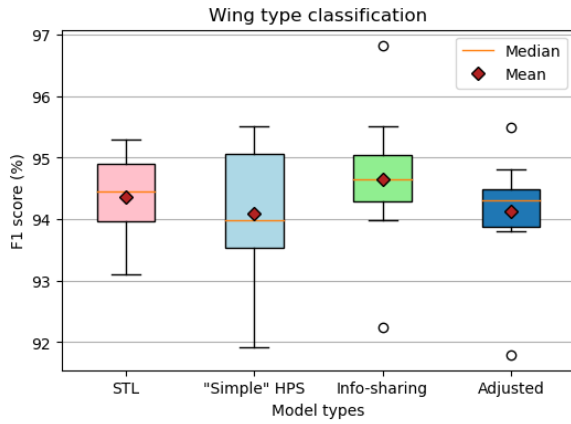
The possible association of the structural characteristics of a drone target and the possibility of it being equipped with some kind of payload was introduced in Chapter 5. Considering this possibility, in this section the results of performing joint prediction of three tasks, the wing type, number of rotors classification, and payload detection (treated as a binary classification task) is presented. The two testing approaches described in Chapter 5 are followed and discussed.

6.3.1. Testing on different train/test splits

The first testing type towards assessing the performance of the proposed classifiers regarding this task group is the one based on iteratively assigning training and test sets in a cross-validation manner. For the needs of this particular group of tasks, the testing process took place utilizing Dataset A, which as described in Chapter 4 does not contain any instance of multiple drones per spectrogram. Hence, the performance of the two STL architectures concerning the wing type and number of rotors classification tasks is identical to this case as well.

The negative impact of predicting three tasks at the same time, as also noticed previously, can be found in this task group as well. A drop in performance in both wing type and number of rotors classification tasks is apparent in the boxplots of Figure 6.14 compared to the cases of only two tasks considered. More specifically, the F_1 score performances of all MTL architectures were negatively affected in the wing type and number of rotors tasks. The “Simple” HPS and Adjusted models suffer the most from such negative transfer and performed, on average, worse than their STL counterpart (1.18% and 1.23% respectively, when compared to the corresponding two tasks without the payload detection). The best performing model in this case, regarding WT and NR tasks, is found to be the Info-sharing one, as it achieves an F_1 score of 94.66%. Yet, this improvement is not significant as can be seen in Figure 6.14a. A similar, non-significant improvement in performance was achieved by using the same model in the number of rotors task, as Figure 6.14b suggests. There, the gain in F_1 score is 0.3%, from 86.47% with the STL model, at a value of 86.77% with the Info-sharing one. A more in-depth view of the performances of all

classifier types regarding this task group can be found in Table C.4 of Appendix C. What can be further noticed in the plots of Figure 6.14 is the overall increase in variance when employing MTL models.

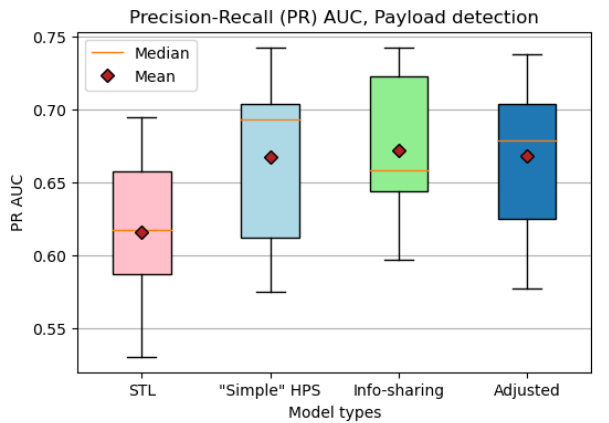
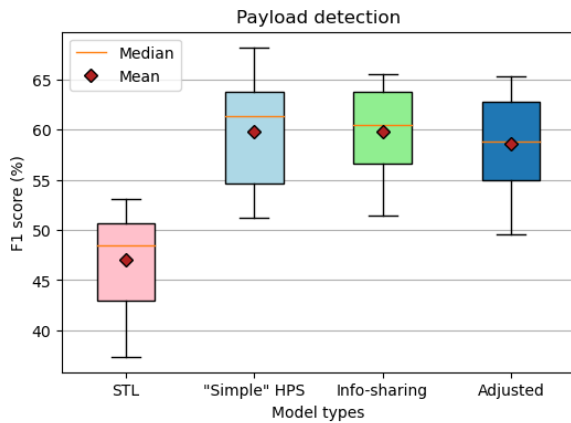


a) Wing type classification task F_1 score boxplot comparison.

b) Number of rotors classification task F_1 score boxplot comparison.

Figure 6.14: The iterative train/test performance statistics of the proposed classifiers applied on Dataset A for the wing type and number of rotors classification tasks. The drop in performance when comparing to Figure 6.1 regarding the MTL models is even more notable than the one presented previously in the plots of Figure 6.11. STL and three different proposed MTL approaches are compared.

Similarly to the multiple drone detection, the payload detection task was also largely benefitting from the utilization of MTL. Even though in this case the number of false alarms using the STL was lower, all MTL models managed to outperform their STL counterpart in this cross-validation test approach. In this setting, the “Simple” HPS model managed to achieve 59.86%, a 12.82% increase over the STL payload class F_1 score as reported in the boxplot of Figure 6.15a. A seemingly less significant increase is seen in the PR AUC metric, demonstrated in Figure 6.15b. It is important to mention that as in the F_1 score case, the PR AUC performance of all MTL models is equivalent, with the highest AUC achieved by using the Info-sharing model. An AUC increase of 8.7%, from 0.616 with the STL model to 0.672 when utilizing the Info-sharing MTL one, can be spotted.



a) Payload detection task F_1 score boxplot comparison.

b) Payload detection task PR AUC score boxplot comparison.

Figure 6.15: The iterative train/test performance statistics of the proposed classifiers applied on Dataset A for the payload detection task of the three-task group presented in this Section. Similarly to the previous section’s task group, all MTL models provide with substantial increase in performance in both F_1 and PR AUC metrics.

The performance gains of utilizing the MTL paradigm are also translated into higher joint accuracy. Here, as in Section 6.2, all three task predictions must be correct in order for a given sample to be counted as a correctly

classified one. In Figure 6.17 the additional advantages introduced with MTL in joint accuracy performance are made clear, as every MTL model surpassed the performance of all three STL networks combined together to form the joint three-task predictions. The Info-Sharing model is found to be the best performing one, achieving an average joint accuracy of 80.18%, a 7.3% improvement over the three STL networks which achieved an accuracy of 72.88%.

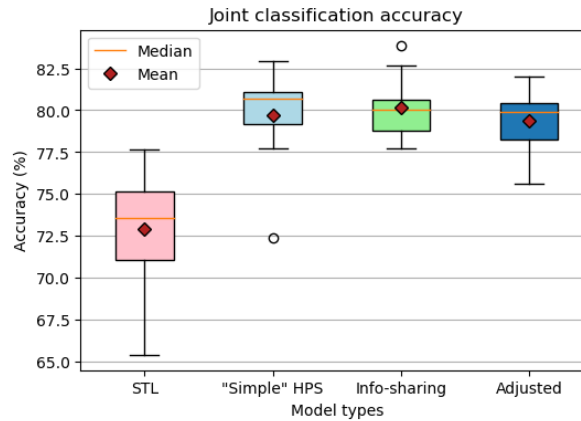


Figure 6.16: The joint classification accuracy statistics over the same ten folds as in the boxplots of Figure 6.14 and 6.15a for the three tasks of this group. An increase of 7.3% can be spotted in favor of the Info-sharing MTL model over the baseline.

6.3.2. Robustness test

Testing using a balanced test set

In this robustness test, further importance was put in the third and newly added task, the payload detection one. The test set is comprised of 144 samples representing an octocopter drone carrying some type of payload and 138 samples of cases not containing payload. More specifically, measurements of a commercial fixed wing drone and a custom quadcopter constitute the non-payload carrying targets of the test set. This makes the test set small and balanced in terms of payload existence aspect. A more in-depth analysis of the test set can be found in Table 6.3.

Description of target	Wing type	Number of rotors	Payload	Number of samples
Single pusher propeller drone	Fixed wing	1	False	91
Quadcopter drone	Multicopter	4	False	47
Octocopter drone	Multicopter	8	True	144

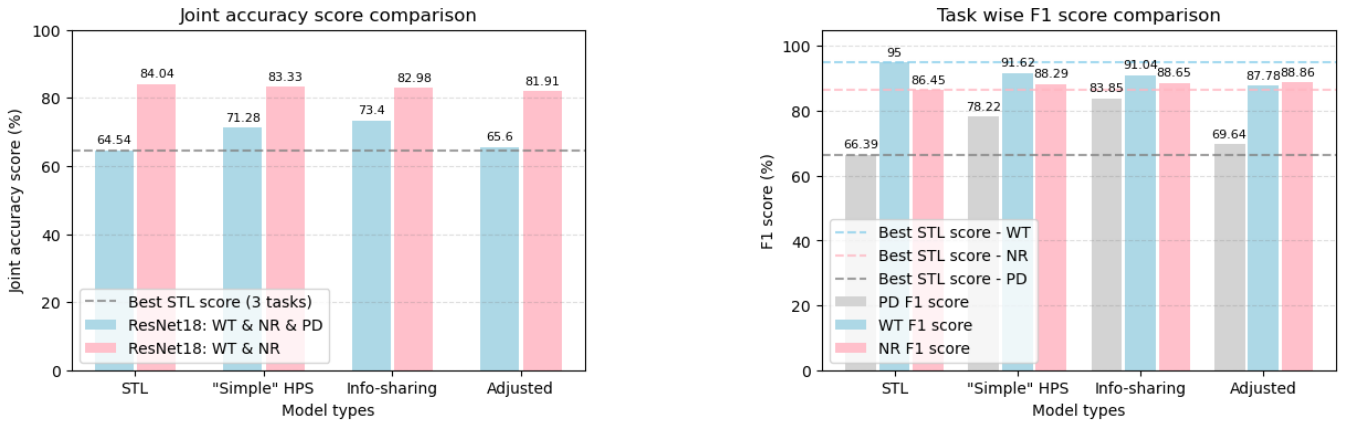
Table 6.3: The content of the test set of the robustness test for the wing type, number of rotors classification, and payload detection task group. The test set in this case is almost perfectly balanced.

In this instance, to enable comparison among the STL and MTL approaches three STL models are defined, each serving one of the tasks consisting this task group. On the other hand, a single MTL model would be sufficient to predict all three. Moreover, to assess the joint performance over all three tasks, the predictions for all tasks must be correct simultaneously for a test sample to be considered as correctly classified. Training and testing the classifiers provides the joint accuracy scores present in Figure 6.17.

It is clear from the part of the plot regarding the simultaneous classification of all three tasks that MTL provides with the highest accuracy, at 73.4% over the 64.54% one achieved by using three different task-optimized STL networks. Hence, the Info-sharing MTL model which performs the best, increases the joint accuracy by 8.86%. For reference, the joint accuracy of the wing type and number of rotors task is also presented in the same plot. The performance of all models is higher, yet in similar range, than the performance of the corresponding classifiers on the mainstream targets presented earlier in Section 6.1.2. What differentiates the two cases is the performance of the STL model on the wing type task, which sees a huge improvement, as it achieves an accuracy of 95%. This is the cause of the STL models having a slightly higher joint accuracy when not considering the payload detection task.

For a more in-depth analysis of the per-task performance, let us consider the F_1 score plot of Figure 6.17b. There, the negative transfer introduced by the additional payload detection task is evident. In the wing type classification task, all MTL models seem inferior to their STL counterpart, although this was not the case in number of rotors classification where the opposite happened. An improvement of 2.41% was achieved when using the Adjusted MTL model. The task where the use of MTL proved to be largely advantageous was the payload detection.

The ability of the “Simple” HPS and Info-sharing models to correctly distinguish among drones mounting some kind of payload and others that did not, as well as the much smaller number of false alarms, is made clear in the payload detection’s F_1 score. The Info-sharing model, in particular, managed to achieve a score of 83.85%. This high score can be explained when considering the very small number of false alarms produced in the test set, having misclassified less than ten samples as payload-carrying drones. The PR curves of Figure 6.18 further establishes the Info-sharing MTL model as the best performing model for this particular test set.



a) Joint accuracy score comparison between MTL and STL models. b) Task-wise F_1 score comparison between MTL and STL models.

Figure 6.17: Joint classification accuracy and task-wise F_1 score comparison plots for the case where the test set is balanced in this task group consisting of three tasks (WT, NR & PD). All MTL models manage to outperform their STL counterparts in all cases with the exception of the wing type classification task.

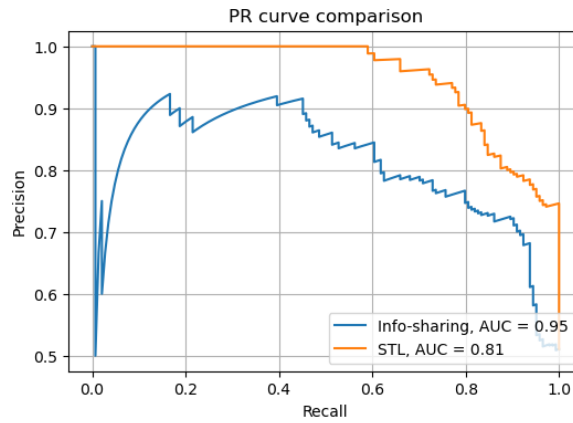


Figure 6.18: PR curve comparison between the best performing MTL model (Info-sharing) and the corresponding STL one predicting the existence of payload when applied to a balanced test set.

Testing using a fairly imbalanced test set

The previous test gave some insight about the possible robustness of the algorithms when they are trained with an efficient amount of data. Yet, the test set in that particular case was balanced, something that is very rarely the case in real-life scenarios. To account for this, a second robustness test was carried out, this time consisting of all “mainstream” targets of the respective test in Section 6.1.2 and the octocopter drone equipped with heavy payload. This particular split produced 478 samples of no payload targets and the 144 samples of the octocopter target with heavy payload. Table 6.4 summarizes the targets present in this test set.

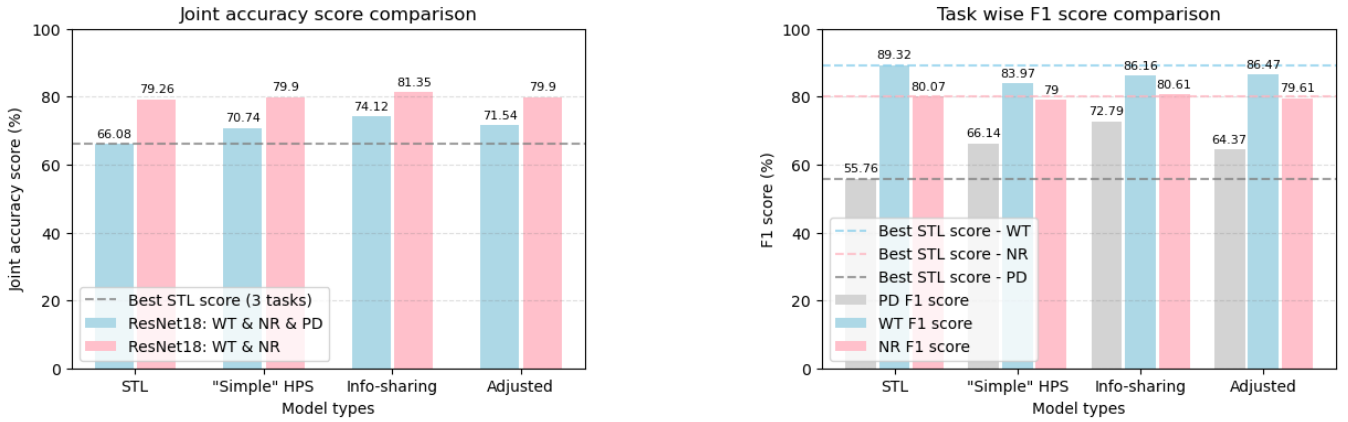
Description of target	Wing type	Number of rotors	Payload	Number of samples
Single rotor fixed wing drone	Fixed wing	1	False	188
Helicopter drone	Helicopter	2	False	221
Quadcopter drone	Multicopter	4	False	69
Octocopter drone	Multicopter	8	True	144

Table 6.4: The content of the test set of the second robustness test on the wing type, number of rotors classification and payload detection task group. The test set in this case is fairly imbalanced and diverse in target types.

In similar fashion as in the previous test, three different task-specific STL models are trained, tested, and then compared to the three MTL models also created to predict the three tasks of this group in a joint manner. Then, the predictions of all tasks are combined, and the joint accuracy is shown in the bar plot of Figure 6.19a. There, it is made obvious that utilizing the MTL paradigm to jointly train these three tasks leads to superior performance over its STL counterpart. Increase in joint accuracy up to 8.04% can be seen when using the Info-sharing MTL model, interestingly following the conclusions of the previous test. The advantages of using MTL in this test are also evident when considering only the wing type and number of rotors classification tasks. There, all three MTL models achieve higher accuracies when compared to the baseline STL models, with the most significant increase belonging once again to the Info-Sharing model.

What was evident in the iterative test of this section was the possibly negative transfer induced from the newly added payload detection to the other two tasks of the group. This can be also observed in the F1 score comparison

plot of Figure 6.19b, where the MTL models would appear inferior to the STL ones, with the exception of the Info-sharing model's performance in the number of rotors task where it performed marginally better than its STL counterpart. The payload detection task, on the other hand, was much improved under the MTL paradigm, seeing a 17.03% increase in F_1 score (from 55.76% to 72.79%) when utilizing the Info-sharing MTL model.



a) Joint accuracy score comparison between MTL and STL models.

b) Task-wise F_1 score comparison between MTL and STL models.

Figure 6.19: Joint classification accuracy and task-wise F_1 score comparison plots for the case where the test set is fairly imbalanced. The STL models for the wing type and number of rotors classification tasks slightly outperform the corresponding MTL ones. Yet, the later introduce a very important increase in performance on the payload detection task. STL and three different proposed MTL approaches are compared.

The superiority of jointly training and predicting the payload detection task alongside the other two is also depicted in the PR curve of Figure 6.20. The curve belonging to the Info-sharing model, the best overall performing MTL model, almost fully encapsulates the one of the baseline STL model. It is also obvious from the curves that different thresholding of the output can provide for instance better precision by sacrificing less recall for the Info sharing model when compared to the STL one. The AUC of these two curves quantify the total predicting capabilities of each of the two classifiers. In this context, the better classification capacity of the Info-sharing model is translated into a higher AUC of 0.802, a 28.49% difference from the baseline model's 0.602.

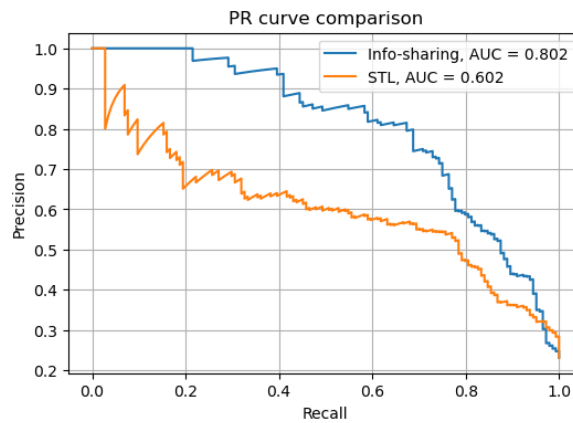


Figure 6.20: PR curve comparison between the best performing MTL model (Info-sharing) and the corresponding STL one predicting the existence of payload when applied to a fairly imbalanced test set such as the one dealt with in this robustness test. The increase in AUC is substantial.

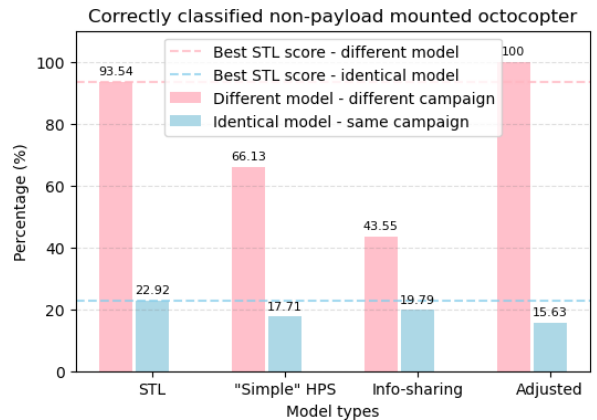
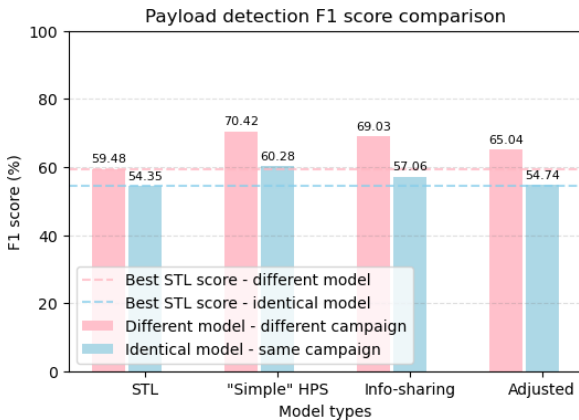
Can the models actually distinguish between an octocopter with payload and one without?

As seen in the two robustness tests of this section, the only drone equipped with payload in each test set is a specific octocopter model. A key and reasonable question that can be made at this point is therefore the following: *Do the classifiers really distinguish the existence of payload mounted on a drone, or do they learn that the certain octocopter model comes equipped with heavy payload?*

To prove the ability of the classifiers in this context, extra experimentation was carried out. For this, the test set of the second robustness test of this section was modified by adding further octocopter measurements. Thus, two new test sets were defined following the additional octocopter measurement introduced. The added measurement in the first case is of an octocopter of a different model captured during a different measurement campaign, and is thus governed by substantially different SNR conditions. In the second case, an identical octocopter model captured during the same campaign as the one carrying payload was added to the test set of the second robustness test of the previous Section. Figure 6.21a presents the F_1 score performance of all proposed classifiers on both test sets.

By observing the aforementioned plot, it is evident that MTL models perform better in class F_1 score terms. In the case where an additional octocopter of different model is present in the test set, the false alarms in all cases are increased as expected, yet utilizing MTL provided with an increase of 10.94%. In the second, and more difficult case, where measurement of the same drone model this time with no payload equipped, the false alarms increased even more. The “Simple” HPS model managed to outperform its STL counterpart by imposing a F_1 score increase of 5.93%.

Even though the comparison of the proposed classifiers in F_1 score levels is fair and provides with good insight regarding their classification capabilities, it still does not provide with any information on the actual percentage of these newly added octocopter cases that were misclassified as carrying payload. This is a very serious and valid question one may have in the operational context of this study. Figure 6.21b demonstrates the disadvantages of MTL in these terms. With the exception of the Adjusted MTL model, the rest of the MTL models performed poorly in differentiating between octocopters with and without payload. Nevertheless, even though the STL models were able to perform this differentiation better, they induced a considerable number of false alarms caused by samples of entirely different drone types. A possible explanation of this advantage of the STL model over its MTL counterparts is the fact that since it is purposefully trained and optimized for this task, it might be able to learn more features associated with the RCS of the targets with payload.



a) Joint accuracy score comparison between MTL and STL models.

b) Percentage of correctly classified octocopter without payload spectrograms.

Figure 6.21: The payload detection performance of STL and MTL classifiers when added octocopter drones without payload are transferred from the training to the test set of the second and fairly imbalanced robustness test. The Adjusted MTL approach was superior in distinguishing among octocopters with and without payload when they were of different models.

6.4. Conclusion

In general, the utilization of the Multi-Task Learning paradigm was proven to be beneficial for the prediction of all task groups, whether improving the prediction capabilities for all or some of the corresponding tasks. To summarize, Table 6.5 provides an overview of all task groups assessed in this chapter, and the best performing paradigm and architecture for each of the tests.

Regarding the iterative train/test split type of tests presented in this chapter, in the wing type and number of rotors classification tasks, MTL models provided statistically significant improvements in every comparison metric employed. Similar conclusions were drawn from the two corresponding robustness tests tailor-made for assessing two aspects of the proposed classifiers: their ability to classify targets seen before but in different measurement campaigns, and thus in different SNR scenarios, as well as their generalization capabilities when classifying never seen before distinctive target cases. In both these robustness tests, MTL models managed to outperform the STL baseline models, achieving 5.22% and 3.2% increase in joint classification accuracy respectively.

In the three-task group of wing type, number of rotors classification and multiple drone detection, the advantages in performance that MTL achieves were mostly apparent in the latter task. A 25.65% increase is seen in the multiple drone classification task when applying the “Simple” HPS model, while considerable drops in performance in the other two tasks were noticed during the iterative train/test approach. As expected, the joint classification accuracy followed the example set by the multiple drone detection task, i.e., the most challenging one. Hence, all MTL models performed much better than their three STL counterparts. The fact that all multiple drone measurements took place on the same day and in the same scenario made a robustness test split impossible with the available dataset.

The use of MTL models initially proved advantageous when attempting to detect the presence of payload mounted on drones. In both the iterative train/test and robustness tests conducted so as to assess their performance, all MTL classifiers proved to be better in joint classification of the targets. Yet, further experimentation showed a major weakness of MTL when applied to payload detection while jointly trained with the wing type and number of rotors classification tasks: even though their F_1 score performance was superior to their STL counterpart, all MTL classifiers struggled with differentiating between equipped and non-equipped drones of the same model. This disadvantage of MTL can be attributed to the possibly wider area of focus so as to capture more information for all tasks. On the contrary, the STL model may be able to focus more on features that help differentiate between the two cases for the same drone model. Finally, it would be of great importance to explore possible performance gains when training and predicting the payload detection task alongside the mean rotor rate regression one.

Task group	Test type	Task	Best performing paradigm	Architecture
WT & NR	Iterative train/test (Dataset A)	WT	MTL	Info-sharing HPS
		NR	MTL	Info-sharing HPS
		Joint	MTL	Info-sharing HPS
	Iterative train/test (Dataset B)	WT	MTL	Info-sharing HPS
		NR	MTL	Info-sharing HPS
		Joint	MTL	Info-sharing HPS
	Robustness test – Mainstream targets	Joint	MTL	“Simple” HPS
	Robustness test – “Exotic” targets	Joint	MTL	Adjusted HPS
WT & NR & MDD	Iterative train/test (Dataset B)	WT	MTL	“Simple” HPS
		NR	MTL	Adjusted HPS
		MDD	MTL	“Simple” HPS
		Joint	MTL	“Simple” HPS
WT & NR & PD	Iterative train/test (Dataset A)	WT	MTL	Info-sharing HPS
		NR	MTL	Info-sharing HPS
		PD	MTL	“Simple” HPS
		Joint	MTL	Info-sharing HPS
	Robustness test – Balanced test set	WT	STL	ResNet 18
		NR	MTL	Adjusted HPS
		PD	MTL	Info-sharing HPS
		Joint	MTL	Info-sharing HPS
	Robustness test – Imbalanced test set	WT	STL	ResNet 18
		NR	MTL	Info-sharing HPS
		PD	MTL	Info-sharing HPS
		Joint	MTL	Info-sharing HPS

Table 6.5: Summary of the task groups assessed in this chapter and the best performing approach and model in each one. Color green denotes that an MTL model was found to perform better in the corresponding task. With orange, the cases where STL was found to perform better are highlighted.

7 Results: flight behaviour based estimation

*This chapter presents the second part of the results of this thesis. The tasks dealt with in this chapter are the mean rotor rate estimation, payload and event detection. Contrary to the previous chapter, where structural characteristics of the drone targets were estimated, here the payload and event detection tasks are considered, in parallel with the estimation of the mean rotation rate of the drone's rotors, extracted according to the procedure outlined in Chapter 4. **Section 7.1** presents and analyses the performance of the two paradigms through the classifiers defined for each one for the mean rotor rate regression and payload detection tasks. Analysis and comparison among STL and MTL regarding the mean rotor rate regression and event detection task group takes place in **Section 7.2**. **Section 7.3** summarizes this results chapter.*

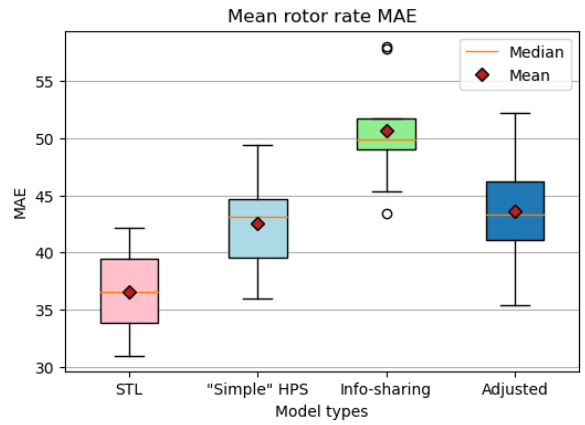
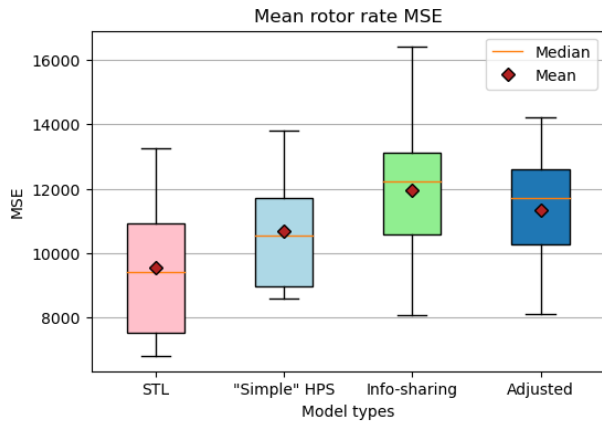
7.1. Mean rotor rate estimation & Payload detection

The first task group assessed in the flight characteristics part of the results for this thesis is the joint mean rotor rate estimation (i.e., estimating a single value for every 0.5 s spectrogram) and payload detection (which is treated as a binary classification problem). Following the methodology introduced in Chapter 5, two STL models are trained and tested, one regression ResNet18 and a binary classification one. The three MTL models proposed in this work are also assessed, each one having two output streams for the two tasks of this task group. In the same manner as the previous task groups of Chapter 6, the classifiers are first assessed over an iterative testing scheme based on the notion of cross validation. Then, a robustness test is carried out on a custom, purposefully created test set. All of the presented experiments were carried out using Dataset C as it was described in Chapter 4.

7.1.1. Testing on different train/test splits

Starting with the estimation of the mean rotor rate derived from the whole duration of each spectrogram segment, a degradation of the regression performance can be observed in the MSE and MAE box plots of Figure 7.1 when using MTL. On average, the “Simple” HPS model, which is the best performing MTL model, increases the MSE by 11.16% while its MAE performance is also degraded by 15%. It would appear that at least the mean blade rate regression is not benefitted by the parallel prediction of the payload detection task. Although the MSE degradation of the “Simple” HPS model cannot be characterized as strictly statistically significant (see Figure 7.1a), the drop in performance in MAE terms would be more important.

The admittedly large Mean Squared Error can be attributed to two factors. The blade flash frequency values are varying considerably and may range from just above 100 Hz to 800 Hz. As a result, these discrepancies between the predicted and true values are significantly amplified by the square component of the error. The second factor attributing to this high value is the existence of sudden events within the data, such as landings and take offs which lead to a rather high true mean blade flash rate. In cases where these events are missed by the STL regressor, or the regressor head of each MTL network, they are the cause of a significant increase in MSE. The same events affect the MAE in a more subtle manner, as it can be seen in Figure 7.1b. There, all MTL models failed to provide superior performance when compared to their STL counterpart.

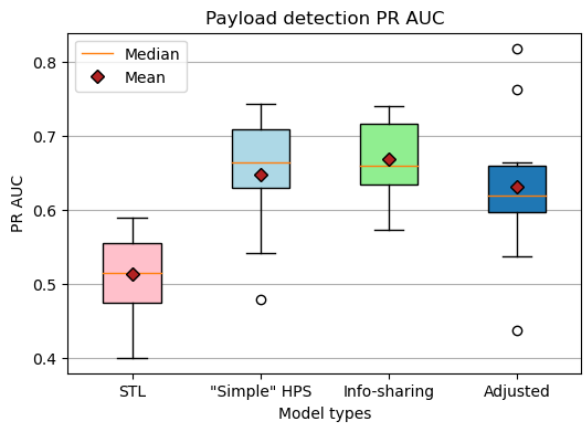
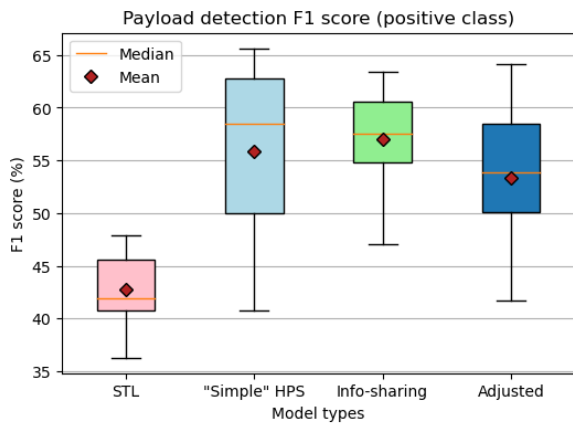


a) Mean rotor rate MSE boxplot comparison.

b) Mean rotor rate MAE boxplot comparison.

Figure 7.1: The iterative train/test performance statistics of the proposed classifiers applied on Dataset C for the mean rotor rate regression task of the two-task group presented in this Section. STL and three different proposed MTL approaches are compared. Both plots indicate that the use of MTL comes with a drop in performance regarding the mean rotor rate regression task. The circular artifacts in the Info-sharing case represent outliers in its MAE performance over the ten splits.

Similarly to the corresponding test in the cases with the additional multiple drone and payload detection, all classifiers struggle in achieving adequate performances in false alarm rates. This is translated in lower levels of F_1 score, something that is depicted thoroughly in the boxplot of Figure 7.2a. Even though both STL and MTL models do not achieve satisfactory scores, the former managed to provide an increase of 14.23% when comparing the best performing MTL model, the Info-sharing one to the STL baseline. An important point is that the STL model was unable in any case to surpass the 50% F_1 score threshold. The PR AUC follows the performance pattern of the classifiers on F_1 score terms. On average, all MTL classifiers achieve PT AUC over 0.6, a small value overall but an ultimately significant improvement over the baseline STL model. The PR AUC values constituting the boxplot of Figure 7.2b infer a possible higher classification capacity for all proposed classifiers in the payload detection task. The aforementioned PR AUC values imply that there is a certain threshold that will lead to a more adequate performance than the one depicted in this test.



a) Payload detection task F_1 score boxplot comparison.

b) Payload detection task PR AUC score boxplot comparison.

Figure 7.2: The iterative train/test performance statistics of the proposed classifiers applied on Dataset C for the payload detection task of the two-task group presented in this Section. STL and three different proposed MTL approaches are compared. Similarly to the previous chapter's task group involving the same task, all MTL models provide with substantial increase in performance in both F_1 and PR AUC metrics. Outliers in PR AUC performance are denoted with the circular artifacts as seen in the "Simple" HPS and Adjusted MTL models.

A more in-depth view of the results presented in the corresponding boxplots can be viewed in Table D.1 of Appendix D. Comparing to the payload detection capabilities of all proposed classifiers as assessed in Chapter 5, one can spot a slight decrease in performance when predicting it in parallel with mean rotation rate estimation.

7.1.2. Robustness test

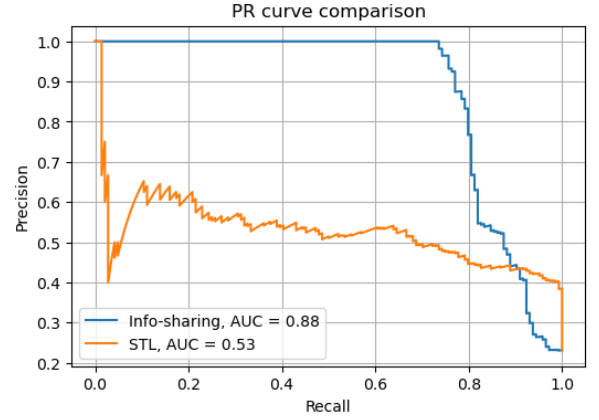
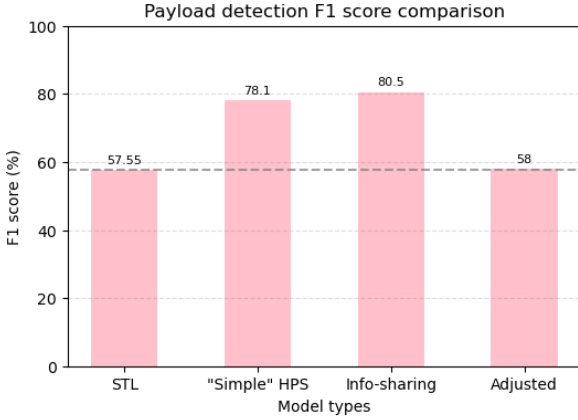
One of the key differences between the robustness tests involving the payload detection task of this section and the ones presented previously in Section 6.3 is the difference in number of samples in the formed train sets (for each one of the two robustness tests). As was extensively covered in Chapter 4, not all available spectrograms produced cepstograms that would allow for reliable rotor rotation rate extraction. This severely impacted the training set part of both robustness tests. The result was that the training set of the first robustness test scenario of Section 6.3, even if it is the largest of the two, produced classifiers that could not predict the payload task sufficiently. It is possible that the missing training samples provided some sort of class separation that benefited the learning of the models. Hence, although both STL and MTL models performed quite well in rotor rotation rate estimation terms, all struggled with the payload detection task.

On the contrary, classifiers trained on the corresponding set of the second robustness test proved able to distinguish among cases with and without payload. In view of this, as also mentioned in Chapter 5, only the second robustness test of Section 6.3.2 is utilized and presented in this subsection. It is important to mention that the test set was not affected in any way and remained intact since all corresponding cepstograms allowed for rate extraction. The content of the test set for this robustness test can be found in Table 7.1.

Description of target	Wing type	Number of rotors	Payload	Number of samples
Single rotor fixed wing drone	Fixed wing	1	False	188
Helicopter drone	Helicopter	2	False	221
Quadcopter drone	Multicopter	4	False	69
Octocopter drone	Multicopter	8	True	144

Table 7.1: The content of the test set of the second robustness test on the wing type, number of rotors classification and payload detection task group. The test set in this case is rather imbalanced and diverse in target types.

The benefits of employing the MTL paradigm in this setting is made clear when observing the F_1 comparison plot of Figure 7.3a. There, an impressive increase of 22.95% can be found when utilizing the Info-sharing MTL model, which achieved an F_1 score of 80.5% versus a much poorer 57.55% achieved by the STL version of the ResNet18. In Figure 7.3b, the PR curves for the Info-sharing model and the STL one are showcased, once again proving the superiority of the MTL model. The relatively high F_1 score of the Info-sharing model is almost solely attributed to the extremely high precision of the classifier, having no false alarms. The “Simple” HPS model on the other hand offered a more balanced, but yet much improved performance over its STL counterpart on the test set.



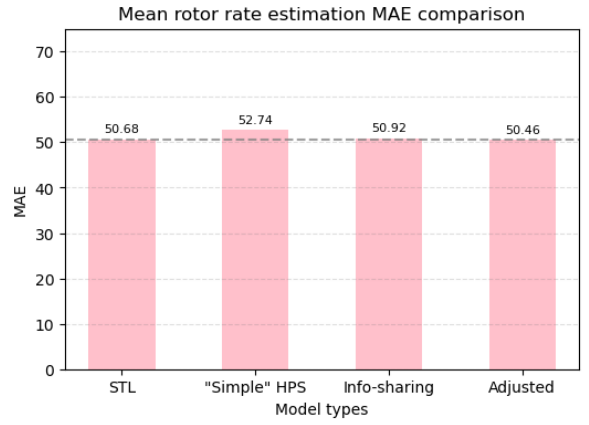
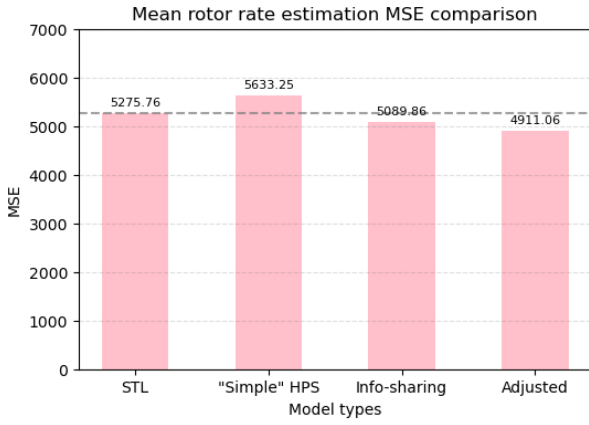
a) Payload detection F_1 score comparison between STL and the three MTL defined models.

b) PR curve comparison among the best performing MTL model (Info-sharing) and its STL counterpart.

Figure 7.3: The performance of the proposed classifiers on the imbalanced robustness test regarding the payload detection task as described in Chapter 4.

It would appear that in this particular train/test split, the mean rotor rate estimation capabilities of the Adjusted and Info-sharing models are roughly equivalent to their STL counterpart, while the “Simple” HPS model appears inferior as seen in the MSE and MAE plots of Figure 7.4. Even though the Adjusted and Info-sharing models provide a decrease in MSE of 3.59% and 7.15% respectively, this improvement should be treated with some scepticism, especially when considering the sensitivity of the metric to outliers in prediction. This is made more apparent when observing the plot of the mean rotation rates of all targets simultaneously in Figure 7.5a, alongside with the absolute error for each instance in Figure 7.5b. There it can be seen that a relatively small amount of large errors can dominate over the metric even though it is an average over all given samples. To quantify this, consider that an instantaneous absolute error of 350 Hz, which is of course a very large one, when squared gives 122,500. Hence, since no take-off or landing events take place in any of the given test samples, which would explain large discrepancies like the aforementioned one, the improvement that the MTL models introduce in MSE terms would point to a smaller amount of such errors (or errors of lower magnitude) by the two proposed models.

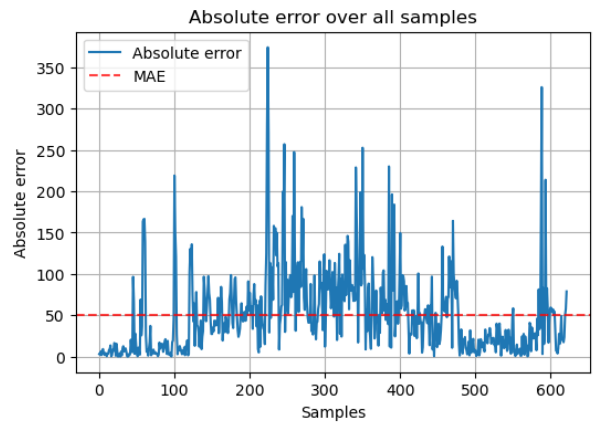
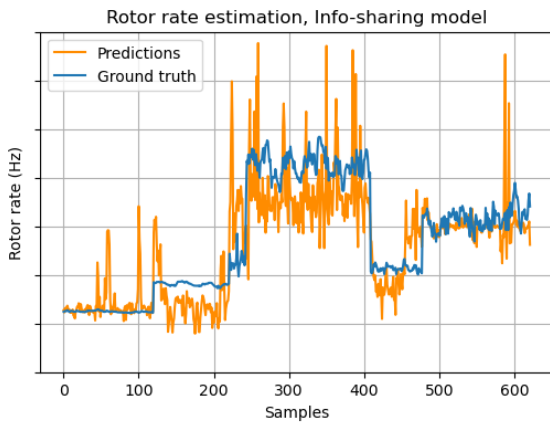
The MAE comparison plot of Figure 7.4b proves this point, since it can be seen that the estimators are all almost equal in mean absolute error terms. The constant misestimations seen in certain cases in Figure 7.5a can be attributed to the manner in which the representative rotor rate for each case is calculated. As described in Chapter 4, the rotor rate extraction to be used as ground truth was carried out by imposing specific restrictions that were not imposed to the regressor models (i.e. the helicopter cases have two distinct harmonics, yet only one contributes to the rotation rate extraction).



a) Mean rotor rate estimation MSE comparison between an STL and the three defined MTL models.

b) Mean rotor rate estimation MAE comparison between an STL and the three defined MTL models.

Figure 7.4: Rotor rate regression performance comparison for both the STL and MTL paradigms. Even though the MSE comparison barplot (left) shows a 7.15% improvement when using the Adjusted MTL model, the same network's performance in MAE terms shows that both models are probably equivalent in performance.



a) Rotor rate predictions and ground truth values for the best performing model in the payload detection task, the Info-sharing MTL model.

b) Absolute error over all samples across all measurements present in the test set. The MAE is also plotted (dashed red line).

Figure 7.5: The mean rotor rate predictions (left) and absolute error (right) produced by the best performing classifier in the payload detection task. Note that for simplicity reasons, all mean rotation rates are presented at once at these plots. Thus, each sample present in each of the two plots represents one rotation rate calculated from the corresponding 0.5 s cepstrogram as described in Chapter 4.

Even though the conclusions of the iterative train/test approach presented in the previous subsection should not be directly compared to the ones of the robustness test, since both lead to vastly different training procedures, it is true that in both cases, the MTL models appeared more capable of classifying payload cases than their STL counterparts. Another common conclusion is that the models produced following the MTL approach were, in the best case scenario, comparable to the STL ones in terms of mean rotor rate regression capabilities.

Interestingly enough, when compared to the corresponding robustness test of Section 6.3.2, the performance of the MTL models here appear inferior to their MTL counterparts of Section 6.3.2. It would, at first glance, appear that predicting the mean rotor rate regression task alongside the payload detection one instead of the structural characteristics of the target, lowers the performance of detecting targets with payload.

Can the models actually distinguish between an octocopter with payload and one without?

The same question asked in Section 6.3.2 can be asked here as well. It would be also interesting to compare the effect that predicting a different task, both in type and nature, would have on the differentiation between two structurally equivalent, or identical, drone targets.

Interesting conclusions may be drawn from the bar plots of Figure 7.6 about the differentiation capabilities of the proposed classifiers. Firstly, it is clear from the plot of Figure 7.6a that employing the MTL paradigm provides better performances in F_1 score terms for either of the two types of octocopters introduced to the test set. More specifically, the “Simple” HPS model managed to achieve an F_1 score improvement of 6.46% over the best possible corresponding score achieved by an STL network. This improvement is attributed to the much higher misdetections of payload cases. The STL model in this case managed to achieve only 21 false alarms, yet it missed 51.39% of the payload instances.

A higher improvement in F_1 score performance is seen when a measurement of the same drone model is transferred from the train to the test set. As expected, the presence of this particular target in the test set induces a high amount of false alarms for both the STL and MTL approaches, leading to drops in performance for all classifiers. Nevertheless, all proposed MTL classifiers’ performances exceeded the one achieved by their STL counterpart. The highest improvement is once again observed when employing the “Simple” HPS classifier, achieving an F_1 score equal to 65.28%, a 12.49% increase over the STL model’s one.

Although in general all MTL models performed considerably better than the STL baseline, the question of how efficient they are in distinguishing among drones of the same type with and without payload still holds. To answer this, consider the comparison plot regarding the percentage of correctly classified non-payload carrying octocopters of Figure 7.6b. There, it can be observed that both the “Simple” HPS and Info-sharing MTL models achieve 100% separation between octocopters with and without payload when the latter is of different model. This comes as a 14.52% improvement over the one achieved by ResNet18 trained and tested in an STL fashion. Hence, for these two models, the false alarms are induced solely by misclassifying other drone types.

On the contrary, all classifiers achieve a very low separation when it comes to a certain octocopter model with payload versus the same model without. Yet, the STL model was found to be considerably better when compared to the MTL ones. Other thresholds than the traditional 0.5 one for class assignment were proven to be beneficial for distinguishing among the two aforementioned classes, yet this benefit came in the expense of a major drop in F_1 score performance. For instance, setting the threshold for the Info-sharing model to 0.54 gives a percentage of correctly classified non-payload mounted octocopter class of 40.63% (a 19.8% increase of the percentage over the one calculated with a 0.5 threshold), while at the same time causing the F_1 score to drop to 50%.

7.2. Rotor rate estimation & Event detector

The last of the task combinations explored in this thesis is the mean rotor rate estimation alongside the event detection task as introduced in Chapter 4. Following the methodology introduced in Chapter 5, as well as the vast majority of the task groups in this chapter, the proposed classifiers are assessed over two types of tests: an iterative and a robustness one. A difference between the iterative train/test approach followed in this task group when compared to the previous one is the fact that due to the nature of the classification task, the iterative approach in this section does not have the design flaw explained in Chapter 5. Since spectrograms of the same track do not imply anything about the possibility of an event taking place, they can be shuffled and mixed among the training and test sets with no serious consequences. Dataset C, as described in Chapter 4, supports both of both testing methods.

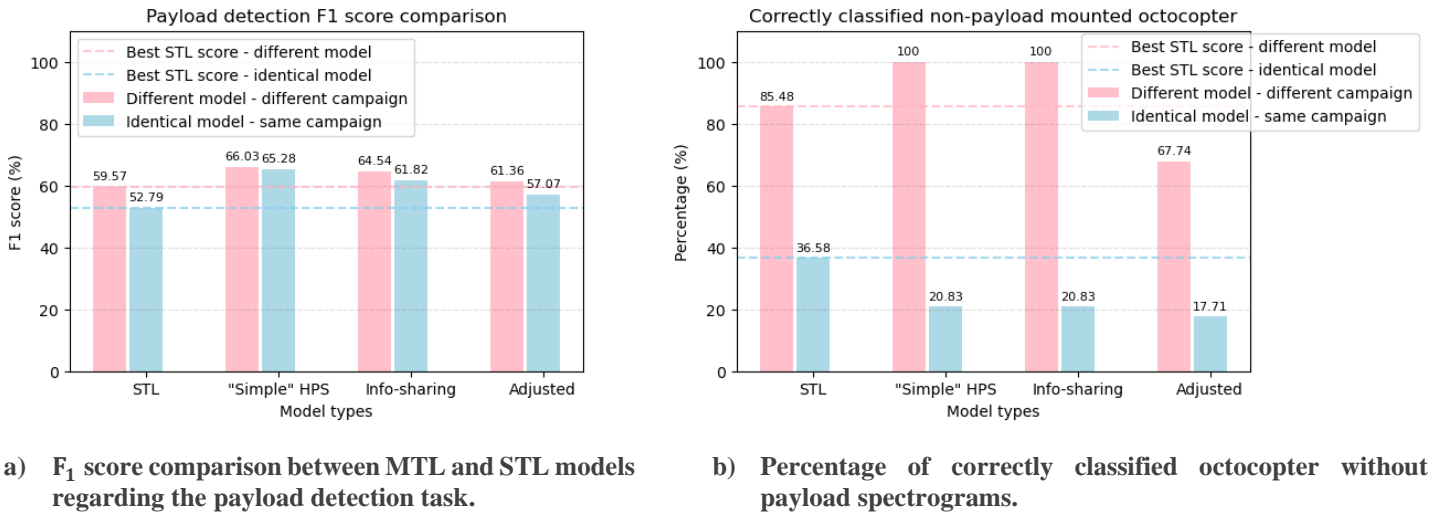
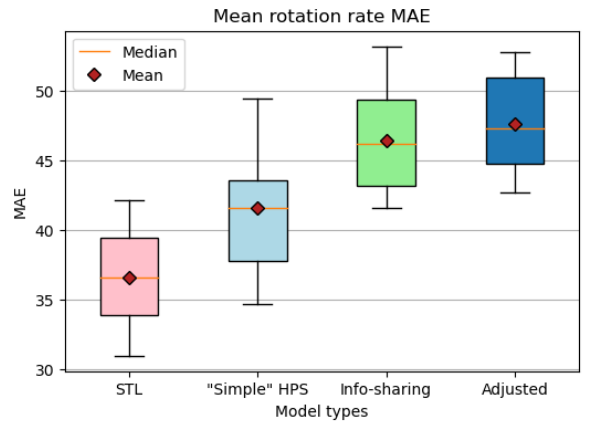
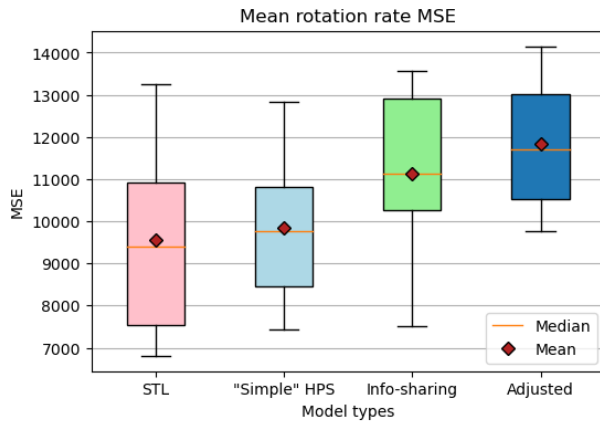


Figure 7.6: The payload detection performance of STL and MTL classifiers when samples of octocopter drones without payload are transferred from the training to the test set of the robustness test. Two MTL models, the “Simple” HPS and Info-sharing MTL models, managed to differentiate among all instances of octocopters with and without payload when the non-payload equipped octocopter is of a different model. This is not the case when the added non-payload octocopter is of the same model as the payload equipped one. In this case, as seen in Section 6.3.2, the STL model exceeds its MTL counterparts’ performance in differentiating among the two cases. STL and three different proposed MTL approaches are compared.

7.2.1. Testing on different train/test splits

Starting with the iterative train/test approach, similar degradation in mean rotor rate prediction, yet in much smaller extent is also noticed in this task group as well as seen in the boxplots of Figure 7.7. In both MSE and MAE terms, the STL ResNet18 network managed to outperform its MTL counterparts by a fairly large margin, with the exception being the “Simple” HPS model, the performance of which is comparable. On average, the “Simple” HPS network provides an increase in MSE of 3.1% (from 9545.13 to 9846.8) and a corresponding increase in MAE of 12.84 (from 36.59 to 41.61). For the same reasons analysed in Section 7.1.1, the increase in MAE is a far more concerning fact (other than it being statistically significant). The Info-sharing MTL model, the second best performing MTL model in terms of mean rotor rate estimation, presented an increase in MSE and MAE of 15.42% and 23.73% respectively.

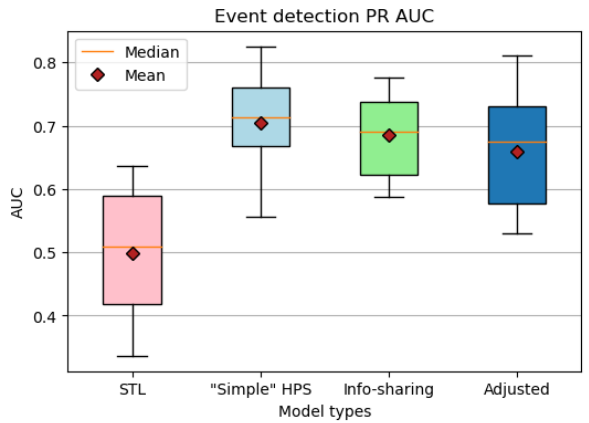
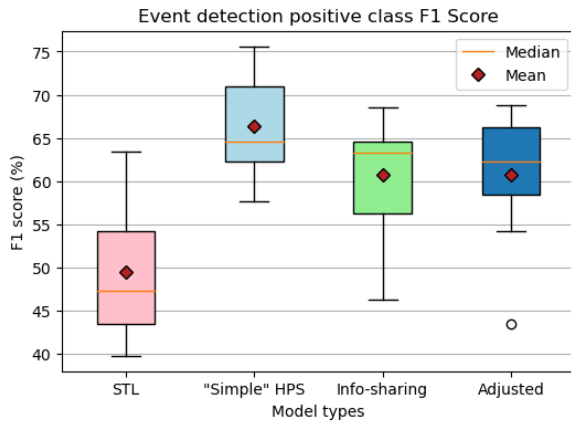
The performance of the proposed classifiers in the event detection task follows a radically different pattern. It would seem in this case that the joint task training leads to MTL classifiers that manage to identify the happening of events more efficiently. This is made obvious in the boxplots regarding both the F_1 score and PR AUC metrics shown in the two plots of Figure 7.8. There, a very much considerable and statistically significant increase in both aforementioned metrics is observed when utilizing any of the proposed MTL models. More specifically, the “Simple” HPS model achieved an F_1 score of 66.37, a major 16.93% increase over the STL’s one. The same pattern can be seen in the PR AUC case in Figure 7.8b, where a 33.33% increase of the AUC is spotted. This further establishes the MTL paradigm as a more suitable one for the event detection task, as all of its classifiers appear to have better classification capability than their STL counterpart. The reason behind this impressive increase in the performance metrics assessing the classification task is found to be the very large number of false alarms produced by the STL model. Table D.2 of Appendix D provides a more detailed view on the performance of all models in this task group.



a) Wing type classification task F_1 score boxplot comparison.

b) Number of rotors classification task F_1 score boxplot comparison

Figure 7.7: The iterative train/test performance statistics of the proposed classifiers applied on Dataset C for the mean rotor rate regression task of the two-task group presented in this Section. As it was with the previous section's task group, both plots indicate that the use of MTL comes with a drop in performance regarding the mean rotor rate regression task.



a) Wing type classification task F_1 score boxplot comparison.

b) Number of rotors classification task F_1 score boxplot comparison

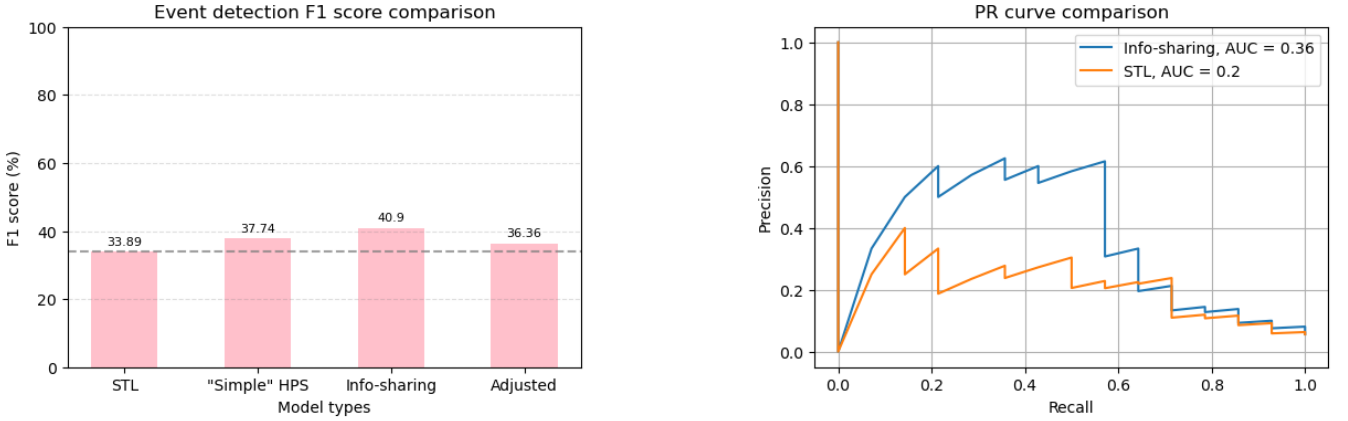
Figure 7.8: The iterative train/test performance statistics of the proposed classifiers applied on Dataset C for the event detection task of the two-task group presented in this Section. All MTL models provide with substantial increase in performance in both F_1 and PR AUC metrics. STL and three different proposed MTL approaches are compared.

7.2.2. Robustness test

The test of the previous subsection can be used as a point of comparison between MTL and STL. Yet, it is still useful and interesting to also assess the performance of the proposed methods on entire measurements' set using a robustness test in which random target measurements were placed in a test set. In the formed robustness test set, only 14 out of the total 249 spectrograms contained some kind of event.

Starting with the event detection performance of both the STL and MTL approaches, neither performed well in the test set as seen in Figure 7.9a. The issue identified with all classifiers was the high number of false alarms. All classifiers managed to identify most of the events correctly, most of them detecting more than 9 of the 14 total events; yet, this would come in the expense of a very high number of false alarms. Additionally, even in this train/test split that does not seem to be beneficial for any of the classifiers, all MTL models managed to improve

the F_1 score by as much as 7.01% when employing the Info-sharing MTL model. The PR curve of Figure 7.9b implies that more suitable thresholding in class assignment regarding the MTL classification part can provide better precision performance, with the sacrifice of a certain percentage of recall. The same cannot be said for the STL ResNet18 identifying events.

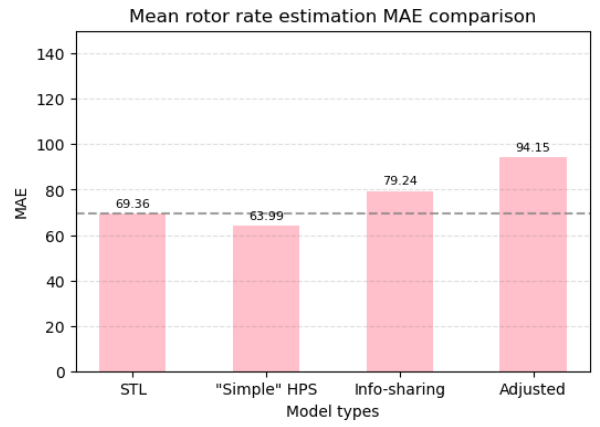
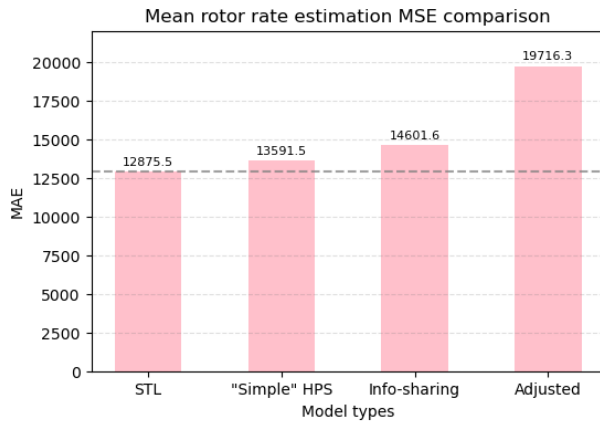


a) Wing type classification task F_1 score boxplot comparison.

b) Number of rotors classification task F_1 score boxplot comparison

Figure 7.9: The performance of the proposed classifiers on the robustness test regarding the event detection task as described in Chapter 4. All proposed classifiers, both STL and MTL, suffer from low F_1 scores due to a high amount of false alarms. MTL models managed to lower the amount of false alarms, yet in each case they were more than the number of correctly classified events.

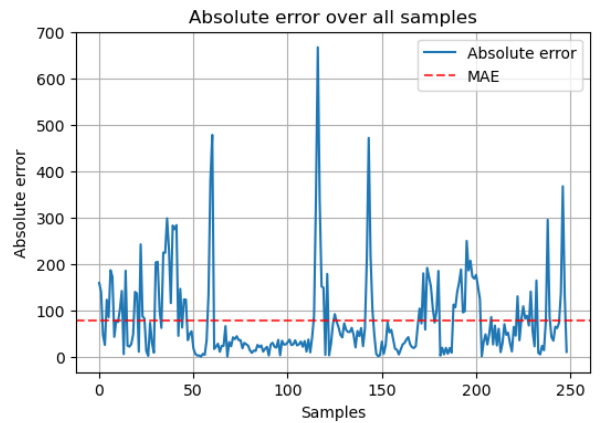
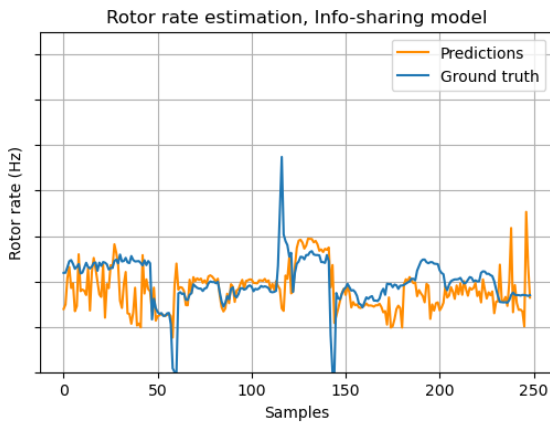
Following the performance patterns of the mean rotor rate regression of Section 7.1.2, the MSE error bar plot of Figure 7.10a shows a large increase when the MTL paradigm is employed. Even the best performing MTL model, the “Simple” HPS one, induces a 5.41% increase in MSE (from 12875.5 to 13591.5). Increase of MSE up to 41.98% can be seen when utilizing the Adjusted MTL model. As discussed also in Section 7.1.2, the MSE does not always convey the whole truth in this task. Sudden and violent changes in mean rotor rate as seen in three distinct cases of the ground truth part of the plot in Figure 7.11a can contribute to very high errors, which are then amplified by the square component of the metric. This seems to be the case in this robustness test, where a few “serious” misestimations, alongside a fairly small test set to average over, can lead to very high MSE values. Observing the bar plot of Figure 7.10b, one may observe that the “Simple” HPS model performs considerably better in MAE terms, as it reduced it by 8.05%. Thus, in general, the estimations of the “Simple” HPS model would appear closer to the truth, yet having larger errors when certain events take place when compared to its STL counterpart.



a) Mean rotor rate estimation MSE comparison between an STL and the three defined MTL models.

b) Mean rotor rate estimation MAE comparison between an STL and the three defined MTL models.

Figure 7.10: Mean rotor rate regression performance comparison for both the STL and MTL paradigms. Contrary to the previous task group, here the “Simple” HPS model performed 8.05% better in MAE terms when compared to the corresponding STL model.



a) Rotor rate predictions and ground truth values for the best performing model in the event detection task, the Info-sharing MTL model.

b) Absolute error over all samples across all measurements present in the test set. The MAE is also plotted (dashed red line). Note the large magnitudes of error in the three distinct event cases.

Figure 7.11: The mean rotor rate predictions (left) and absolute error (right) produced by the best performing classifier in the event detection task (hence, the presented plots belong to the Info-sharing model and not the “Simple” HPS one that provided the best MAE performance). Note again that for simplicity reasons, all mean rotation rates are presented at once at these plots. Thus, each sample present in each of the two plots represents one rotation rate calculated from the corresponding 0.5 s cepstrogram as described in Chapter 4.

7.3. Conclusion

In this Chapter, the performance of the proposed MTL paradigm, with the three types of models defined in Chapter 5, was assessed and compared to baseline performances from STL ResNet18 models purposefully built for each of the three tasks: the mean rotor rate regression, payload and event detection. A collective summary of all tests carried out for obtaining the results of this chapter can be found in Table 7.2.

In general, the use of MTL proved beneficial in almost all the three aforementioned tasks. Regarding the task combination of mean rotor rotation rate regression and payload detection, MTL proved highly beneficial for the

latter. In both the iterative and robustness tests, the payload class F_1 score saw substantial increase of 14.23% and 22.95% respectively. Regarding the differentiation capabilities between octocopters with and without payload, the “Simple” HPS and Info-sharing MTL models managed to achieve 100% separation between the two classes, a 14.52% improvement over their STL counterpart. On the contrary, following the conclusions of Chapter 6, the MTL models were found to be actually worse in correctly classifying octocopter cases without payload when the octocopter model was the same as the one equipped with payload. The regression capabilities of MTL models were shown to be worse in the iterative train/test testing approach, yet there were some investigated models that managed to marginally improve the MAE and MSE performance when trained alongside the payload detection task in the robustness test.

A much more substantial improvement of 8.05% in MAE regarding the rotor rotation rate regression was spotted when employing the “Simple” HPS model. Yet, its MSE being higher than its STL counterpart implies that it either had large misestimations or it completely failed to predict the three obvious events in an accurate manner. Utilizing MTL also improved the F_1 score in the event detection task, managing to reduce the number of false alarms. Even so, the performance of all models was found to be subpar. The iterative train/test approach, which can be used as a legitimate means of comparison among the classifiers by itself, showcased the detection capabilities of the MTL approaches. In this test, a 16.93% increase in F_1 score was spotted when utilizing the “Simple” HPS model.

Nevertheless, in the two particular task groups dealt with in this chapter, one has to bear in mind the importance of each task from an operational point of view. In this context, one is far more interested in whether a detected drone is equipped with payload or not or whether a possible take-off, landing or even payload drop has taken place than knowing the exact rotation rate of the rotors. Hence, improvements in payload and event detection introduced by MTL are of greater importance than the possible degradation in mean rotor rate regression that they are responsible for.

Task group	Test type	Task	Best performing paradigm	Architecture
RRR & PD	Iterative train/test (Dataset C)	RRR	STL	ResNet18
		PD	MTL	Info-sharing HPS
	Robustness Test – Imbalanced test set	RRR	MTL	Adjusted HPS
		PD	MTL	Info-sharing HPS
RRR & ED	Iterative train/test (Dataset C)	RRR	STL	ResNet18
		ED	MTL	“Simple” HPS
	Robustness Test – Random measurements test set	RRR	MTL	“Simple” HPS
		ED	MTL	Info-sharing HPS

Table 7.2: Summary of the task groups assessed in this chapter and the best performing approach and model in each one. The cases where the proposed MTL approach was found to introduce some performance improvement are marked with green. With orange, the cases where STL performed better are designated.

8 Conclusions & Recommendations

The purpose of this thesis was to investigate the novel use of Multi-Task Learning in the field of drone characterization using radar. For this, out of the available experimental dataset, a total of six distinct tasks were defined, namely the wing type, number of rotors classification, the mean rotor rate regression, the payload and event detection.

To the author's best knowledge, for the first time in the known literature, the MTL paradigm is applied to such drone characterization tasks. Three purposefully defined MTL architectures, one of them completely novel, were defined and compared with their STL counterparts in various groups consisting of the aforementioned tasks and assembled based on reasonable operational assumptions made by experts.

The use of MTL led to an overall increase in the performance across most of the tasks in the defined task groups. This Chapter aims to provide with conclusions and recommendations for future research.

8.1. Concluding remarks on the results of this thesis

Throughout the experiments shown in the span of this thesis, several conclusion remarks may be drawn. In an effort to provide as much closure as possible considering the large amount of tests performed in this work, the conclusions are presented in a task group wise manner as follows:

- Wing type and number of rotors classification: In this first task group, the most consistent and robust increase in performance when using MTL was noticed. In both types of tests, MTL models managed to outperform their STL counterparts in an consistent basis.
- Wing type, number of rotors classification and multiple drone detection: The iterative train/test testing approach showed two interesting facts: adding a third task to the previous two-task group accounted for some negative transfer, leading to a drop in performance of the MTL models on the wing type and number of rotors although the best performing model was again an MTL one. Where MTL proved highly beneficial was the multiple drone detection task, where all such models introduced a very substantial increase in performance.
- Wing type, number of rotors classification and payload detection: Jointly training and predicting this task group led to rather interesting conclusions. Firstly, a drop in the first two tasks was obvious and more severe than the one seen in the previously mentioned task group. Yet, all MTL models assessed were found to provide with better overall performance in the payload detection task. Yet, the existence of drone targets of the same model as the one carrying payload in the test set led to a serious increase in false alarms. What was far more concerning in this case though was the fact that the MTL models would correctly classify a smaller percent of these non-payload carrying targets than their STL baseline. This might be showing that the MTL models would associate the certain drone type as always carrying payload.
- Rotor rate regression and payload detection: This two-task group also provided with interesting conclusions with high operational interest. Even though MTL models managed to introduce some improvement in mean rotor rate regression, only rarely it was substantial. Most of the times the proposed MTL networks were found to underperform with respect to the STL regressor used for the aforementioned task. In payload detection on the other hand, once again MTL proved highly advantageous, introducing important increase in detection performance. The same conclusions as in the previous bullet about the

discrimination capabilities of the MTL models can be made here as well, indicating a rather common issue of the type.

- **Rotor rate regression and event detection:** The last task group assessed in this thesis was the rotor rate regression and event detection. Here, even though the performance of the MTL models in predicting the mean rotor rate from 0.5 s spectrograms was slightly improved, it is still questionable whether this improvement in performance is indeed substantial. The iterative cross-validation like assessment approach of this case showed a clear advantage of the MTL models in detecting possible events present in the spectrogram in hand.

It is suitable in this conclusion chapter to finally provide an answer to the following question that was first asked during this work was the following: *Does MTL as a paradigm fit the tasks present in the drone characterization with radar field?*

The immediate answer to that would be that *yes*, MTL can certainly be applied to drone characterization and offer overall substantial and in many cases robust improvement in performance across various different in nature tasks. Regarding the overall performance indices being low in certain tasks, one has to bear in mind the exploratory nature of this work, aiming to set a basis for comparisons between MTL and STL and not purposefully provide highly optimized networks. Hence, it is possible that MTL models can provide with much better performances across the discussed tasks.

Another fact that one has to keep in mind when employing the MTL paradigm in general is whether the goal set is to improve all examined tasks, a subgroup or even only one of them. A misconception around the use of MTL is that every task used will be improved. This is not always true, should not be taken for granted, and was thoroughly demonstrated in a few of the task groups of this work. All in all, depending on the prioritization of the tasks of a certain task group one or more tasks constituting the group may be seen merely as auxiliary tasks (like the case in [47]), existing only to benefit the most important one(s) through better training (i.e. the payload detection is a far more important task than knowing the rotation rate of the rotors accurately).

Finally, a rather concerning issue of DL in general, amplified by the nature of MTL at least when implemented through the HPS approach, is the lack of explainability. Explainability in the DL context refers to the effort of explaining the internal process of the DL network in hand and understanding the otherwise black box operation of the network. By adding multiple outputs and defining more sophisticated loss functions in order to train the network only impose a further complexity level that makes explainability of the model in hand even more challenging. This was made apparent in the inability to thoroughly explain the drop in performance when adding a third task (i.e. the payload detection task) and the worse percentage of correctly classified non-payload carrying octocopters. Some high level assumptions can be made for both issues, yet further research is advised in this area.

8.2. Recommendations for further research

Although the results of this thesis showed that performance in almost every task was improved when employing the MTL paradigm, this research work was purely exploratory, and based on this fact it would be suitable to include some recommendations for further research in the field of drone characterization using radar data and Multi-Task Learning. Recommendations can be made in two levels: radar related and model based recommendations.

Recommendations regarding further research based on different radar infrastructure and/or processing

As was described in Chapter 4, the radar used in this work was an X-band CW radar. This fact by itself imposes limitations in the performance of a number of tasks assessed in this thesis. Due to the radar design and its available data, both the STL baseline models and their MTL counterparts for predicting each task group were based solely

on using spectrograms as input for classification and regression purposes. This is because a CW radar can only provide Doppler-related information on targets. Hence, the following recommendations can be made regarding the radar aspect of drone characterization using MTL:

- Using spectrograms of shorter durations: The benefits of employing the MTL were demonstrated thoroughly when using spectrograms of 0.5 s as input. Yet, from an operational point of view this duration can be thought of as too long and impractical in some circumstances. Tests on wing type and number of rotors classification were ran for spectrograms of 0.25 s showing promising results, yet they were not included in this work for conciseness. It would be then interesting to explore the possible advantages, improvements in performance or even disadvantages of MTL when the duration of the spectrograms becomes smaller and smaller.
- Multiple data representations input: Using multiple data representations simultaneously has been proven to be beneficial for classification purposes in the past [67]. In the context of CW radar, the well-known by now spectrogram can be used alongside the described in Chapter 4 cepstrogram and a further data representation computed in the same manner as the later, yet frequency bin wise called the Cadence Velocity Diagram. The latter has also been used towards detection of multiple micro-drones [68], a task that was also examined in this thesis. Hence, it would be of great interest to assess how both STL and MTL approaches could handle such multimodal input.
- Using a different type of radar: As mentioned above, even though it can be characterized as a good starting point for an exploratory analysis such as this work, the CW radar is unable by design to provide, for instance, range information. On the contrary, using the frequency modulated version of the CW radar can provide extra features that could be crucial for some of the proposed tasks. It would be of great interest to analyse how MTL could handle extra features crafted from the additional radar information provided by an FMCW radar and the improvements that they might introduce in certain tasks, like the payload or the multiple drone detection ones.

Recommendations regarding further research based on the development of more sophisticated architectures

The work of this thesis assessed the potential benefits of MTL over all possible task groups allowed by the provided dataset. Further research can be still carried out on the neural network level, both in terms of further MTL architectures and in terms of drone characterization in general:

- Use of more sophisticated MTL architectures: To keep the scope of this thesis manageable, the MTL models proposed in this work were simple in nature, all following the HPS approach. This approach is both easy to grasp and implement, yet it is not necessarily optimal since possibly unrelated tasks may lead to underperforming networks for some or all tasks when trained together. As described in the literature review part of the thesis, approaches following the SPS (Soft Parameter Sharing) paradigm can mitigate these effects by self-controlling the spread of information along the corresponding network. This may lead to even higher performance gains over the STL models, and hence further research in this field would be appropriate. Another often overlooked aspect that an SPS and more sophisticated approach might be able to counter is the issue of explainability: *what is to blame when something goes wrong, i.e. when a network fails in some task?*
- Use of a more elaborate loss function scheme: It was thoroughly discussed in Chapter 5 that the selection of an appropriate loss function is fundamental for the training and thus performance of the MTL models. The LBTW scheme employed in this work was beneficial in reducing the amount of the negative transfer among tasks, yet it would not weight the loss of each task according to their magnitudes. Hence, possible performance gains related to the better training of the MTL models can still be investigated.
- Classification/Regression over tracks: Both the classification and regression tasks of this work, for both the STL and MTL approaches, were based on single spectrograms drawn out of a tracked target. Even though the performance of some of the tasks, such as the wing type and number of rotors classification were adequate enough, further performance gains can be obtained from treating the problem as a track-

based classification or regression one. Spectrogram specific predictions could be combined to characterize whole tracks.

Last but, certainly, not least is the dire need of obtaining more experimental data. As radar data is scarce and hard to obtain, the need of performing more and more real life measurements cannot be stressed enough. The development, assessment and exploration of MTL approaches on radar applications, either drone related or not, is (as was also concluded by the results of this thesis) highly dependent on the available datasets.

References

- [1] S. Waharte and N. Trigoni, "Supporting Search and Rescue Operations with UAVs," 2010 International Conference on Emerging Security Technologies, Canterbury, UK, 2010, pp. 142-147, doi: 10.1109/EST.2010.31.
- [2] M. Hassanalain and A. Abdelkef, "Classifications, applications, and design challenges of drones: A review.," *Progress in Aerospace Sciences*, no. 91, pp. 99-131, 2017.
- [3] E. Bumiller and T. Shanker, "War Evolves With Drones, Some Tiny as Bugs," 19 June 2011. [Online]. Available: <https://www.nytimes.com/2011/06/20/world/20drones.html>. [Accessed 23 January 2023].
- [4] BBC, "Venezuela President Maduro survives 'drone assassination attempt'," 5 August 2018. [Online]. Available: <https://www.bbc.com/news/world-latin-america-45073385>. [Accessed 24 January 2023].
- [5] Al-Jazeera, "Several wounded in a drone attack on Saudi airport: Coalition," 31 August 2021. [Online]. Available: <https://www.aljazeera.com/news/2021/8/31/several-wounded-drone-attack-saudi-airport-houthis-yemen>. [Accessed 24 January 2023].
- [6] I. Guvenc, F. Koohifar, S. Singh, M. L. Sichitiu and D. Matolak, "Detection, Tracking, and Interdiction for Amateur Drones," in *IEEE Communications Magazine*, vol. 56, no. 4, pp. 75-81, April 2018, doi: 10.1109/MCOM.2018.1700455.
- [7] P. Flak, "Drone Detection Sensor With Continuous 2.4 GHz ISM Band Coverage Based on Cost-Effective SDR Platform," *IEEE Access*, vol. 9, pp. 114574-114586, 2021.
- [8] S. Basak and B. Scheers, "Passive radio system for real-time drone detection and DoA estimation," *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, Warsaw, Poland, 2018, pp. 1-6, doi: 10.1109/ICMCIS.2018.8398721.
- [9] B. I. Ahmad, J. Grey, M. Newman and S. Harman, "Low-latency convolution neural network for estimating drone physical parameters with radar," *International Conference on Radar Systems (RADAR 2022)*, Hybrid Conference, Edinburgh, UK, 2022, pp. 1-6, doi: 10.1049/icp.2022.2282.
- [10] P. Wellig *et al.*, "Radar Systems and Challenges for C-UAV," *2018 19th International Radar Symposium (IRS)*, Bonn, Germany, 2018, pp. 1-8, doi: 10.23919/IRS.2018.8448071.
- [11] C. Clemente, F. Fioranelli, F. Colone and G. Li, *Radar Countermeasures for Unmanned Aerial Vehicles*, London: IET, 2021.
- [12] P. Molchanov, R. I. A. Harmanny, J. J. M. de Wit, K. Egiazarian, and J. Astola, "Classification of small UAVs and birds by micro-Doppler signatures," *International Journal of Microwave and Wireless Technologies*, vol. 6, no. 3-4, pp. 435-444, 2014.
- [13] Y. Zhao, X. Zhang and F. Fioranelli, "Initial results of Radar-based classification of commercial drone carrying small payloads," *2019 International Radar Conference (RADAR)*, Toulon, France, 2019, pp. 1-4, doi: 10.1109/RADAR41533.2019.171305.
- [14] H. Dale, C. Baker, M. Antoniou and M. Jahangir, "An Initial Investigation into Using Convolutional Neural Networks for Classification of Drones," *2020 IEEE International Radar Conference (RADAR)*, Washington, DC, USA, 2020, pp. 618-623, doi: 10.1109/RADAR42522.2020.9114745.
- [15] N. Mohajerin, J. Histon, R. Dizaji and S. L. Waslander, "Feature extraction and radar track classification for detecting UAVs in civilian airspace," *2014 IEEE Radar Conference*, Cincinnati, OH, USA, 2014, pp. 0674-0679, doi: 10.1109/RADAR.2014.6875676.
- [16] D. Park, S. Lee, S. Park, and N. Kwak, "Radar-Spectrogram-Based UAV Classification Using Convolutional Neural Networks," *Sensors*, vol. 21, no. 1, p. 210, Dec. 2020, doi: 10.3390/s21010210.
- [17] S. Björklund and N. Wadströmer, "Target Detection and Classification of Small Drones by Deep Learning on Radar Micro-Doppler," *2019 International Radar Conference (RADAR)*, Toulon, France, 2019, pp. 1-6, doi: 10.1109/RADAR41533.2019.171294.
- [18] H. Dale, C. Baker, M. Antoniou, M. Jahangir and G. Atkinson, "A Comparison of Convolutional Neural Networks for Low SNR Radar Classification of Drones," *2021 IEEE Radar Conference (RadarConf21)*, Atlanta, GA, USA, 2021, pp. 1-5, doi: 10.1109/RadarConf2147009.2021.9455181.
- [19] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017.

- [20] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.
- [21] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
- [22] C. Szegedy *et al.*, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.
- [23] J. J. M. de Wit, R. I. A. Harmanny and G. Prémel-Cabic, "Micro-Doppler analysis of small UAVs," *2012 9th European Radar Conference*, Amsterdam, Netherlands, 2012, pp. 210-213.
- [24] M. R. Bell and R. A. Grubbs, "JEM modeling and measurement for radar target identification," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 1, pp. 73-87, Jan. 1993, doi: 10.1109/7.249114.
- [25] J. Martin and B. Mulgrew, "Analysis of the theoretical radar return signal from aircraft propeller blades," *IEEE International Conference on Radar*, Arlington, VA, USA, 1990, pp. 569-572, doi: 10.1109/RADAR.1990.201091.
- [26] X. Ren, M. Jahangir, D. White, G. Atkinson, C. Baker, and M. Antoniou, "Estimating Physical Parameters from Multi-Rotor Drone Spectrograms," in *International Conference on Radar Systems (RADAR 2022)*, 2022, pp. 20-25.
- [27] R. I. A. Harmanny, J. J. M. de Wit and G. P. Cabic, "Radar micro-Doppler feature extraction using the spectrogram and the cepstrogram," *2014 11th European Radar Conference*, Rome, Italy, 2014, pp. 165-168, doi: 10.1109/EuRAD.2014.6991233.
- [28] B. P. Bogert, M. J. R. Healy and J. W. Tukey (1963): "The Quefrency Analysis of Time Series for Echoes: Cepstrum, Pseudo Autocovariance, Cross-Cepstrum and Saphe Cracking", in: M. Rosenblatt (ed.): *Proceedings of the Symposium on Time Series Analysis*, Wiley, New York, Chapter 15, p. 209-243.
- [29] L. Fuhrmann, O. Biallawons, J. Klare, R. Panhuber, R. Klenke and J. Ender, "Micro-Doppler analysis and classification of UAVs at Ka band," *2017 18th International Radar Symposium (IRS)*, Prague, Czech Republic, 2017, pp. 1-9, doi: 10.23919/IRS.2017.8008142.
- [30] M. Ritchie, F. Fioranelli, H. Borrión, & H. Griffiths, "Multistatic micro-doppler radar feature extraction for classification of unloaded/loaded micro-drones", *IET Radar, Sonar & Navigation*, vol. 11, no. 1, p. 116-124, 2017. <https://doi.org/10.1049/iet-rsn.2016.0063>
- [31] M. Ritchie, F. Fioranelli, H. Borrión, & H. Griffiths, "Multistatic micro-doppler radar feature extraction for classification of unloaded/loaded micro-drones", *IET Radar, Sonar & Navigation*, vol. 11, no. 1, p. 116-124, 2017. <https://doi.org/10.1049/iet-rsn.2016.0063>
- [32] N. Regev, I. Yoffe and D. Wulich, "Classification of single and multi propelled miniature drones using multilayer perceptron artificial neural network," *International Conference on Radar Systems (Radar 2017)*, Belfast, 2017, pp. 1-5, doi: 10.1049/cp.2017.0378.
- [33] Caruana, R. Multitask Learning. *Machine Learning* 28, 41–75 (1997). <https://doi.org/10.1023/A:1007379606734>
- [34] A. Waibel, H. Sawai and K. Shikano, "Modularity and scaling in large phonemic neural networks," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, pp. 1888-1898, Dec. 1989, doi: 10.1109/29.45535.
- [35] S. C. Suddarth and Y. L. Kergosien. "Rule-injection hints as a means of improving network performance and learning time". In: *Neural Networks*. Ed. by L. B. Almeida and C. J. Wellekens. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 120–129.
- [36] Y. S. Abu-Mostafa, "Learning from hints in neural networks," *J. Complex.*, vol. 6, no. 2, pp. 192–198, 1990..
- [37] R. Caruana, S. Baluja, and T. Mitchell. Using the future to sort out the present: Rankprop and multitask learning for medical risk evaluation. In *Advances in neural information processing systems 8 (NIPS)*. 1996.
- [38] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.
- [39] K.-H. Thung and C.-Y. Wee, "A brief review on multi-task learning," *Multimedia Tools Appl.*, vol. 77, no. 22, pp. 29705–29725, Nov. 2018.
- [40] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020.
- [41] Z. Zhang, P. Luo, C.-C. Loy, and X. Tang. Facial Landmark Detection by Deep Multi-task Learning. *ECCV*, 2014
- [42] M. Long, Z. Cao, J. Wang, and S. Y. Philip, "Learning multiple tasks with multilinear relationship networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1594–1603.

- [43] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3994–4003.
- [44] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, "Latent Multi-Task Architecture Learning", *AAAI*, vol. 33, no. 01, pp. 4822-4829, Jul. 2019.
- [45] Y. Fang, Z. Ma, Z. Zhang, X.-Y. Zhang, and X. Bai. Dynamic multi-task learning with convolutional neural network. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1668–1674, 2017.
- [46] W. Du, F. Zhang, F. Ma, Q. Yin and Y. Zhou, "Improving SAR Target Recognition with Multi-Task Learning," *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, Waikoloa, HI, USA, 2020, pp. 284-287, doi: 10.1109/IGARSS39084.2020.9324210.
- [47] Z. Chen, Y. Lin, L. Zhuang and H. Yu, "Meta multi-task learning for small sample SAR target recognition," *International Conference on Radar Systems (RADAR 2022)*, Hybrid Conference, Edinburgh, UK, 2022, pp. 668-672, doi: 10.1049/icp.2023.1270.
- [48] X. Li, Y. He and X. Jing, "A Deep Multi-task Network for Activity Classification and Person Identification with Micro-Doppler Signatures," *2019 International Radar Conference (RADAR)*, Toulon, France, 2019, pp. 1-5, doi: 10.1109/RADAR41533.2019.171263.
- [49] C. Wang, J. Tian, J. Cao and X. Wang, "Deep Learning-Based UAV Detection in Pulse-Doppler Radar," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1-12, 2022, Art no. 5105612, doi: 10.1109/TGRS.2021.3104907
- [50] R. Akter, V. -S. Doan, A. Zainudin and D. -S. Kim, "An Explainable Multi-Task Learning Approach for RF-based UAV Surveillance Systems," *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, Barcelona, Spain, 2022, pp. 145-149, doi: 10.1109/ICUFN55119.2022.9829629.
- [51] M. I. Skolnik, *Radar Handbook*, 3rd ed. New York, NY, USA: McGraw-Hill, 2008.
- [52] V. C. Chen, F. Li, S. -S. Ho and H. Wechsler, "Micro-Doppler effect in radar: phenomenon, model, and simulation study," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 2-21, Jan. 2006, doi: 10.1109/TAES.2006.1603402.
- [53] J. Markow and A. Balleri, "Examination of Drone Micro-Doppler and JEM/HERM Signatures," *2020 IEEE Radar Conference (RadarConf20)*, Florence, Italy, 2020, pp. 1-6, doi: 10.1109/RadarConf2043947.2020.9266342.
- [54] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," in *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51-83, Jan. 1978, doi: 10.1109/PROC.1978.10837.
- [55] S. Rapuano and F. J. Harris, "An introduction to FFT and time domain windows," in *IEEE Instrumentation & Measurement Magazine*, vol. 10, no. 6, pp. 32-44, December 2007, doi: 10.1109/MIM.2007.4428580.
- [56] S. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 683–697, Sep. 1992.
- [57] I. Goodfellow, Y. Bengio and A. Courville, "Deep learning," MIT press, 2016.
- [58] R. Monti, S. Tootoonian, and R. Cao, "Avoiding degradation in deep feed-forward networks by phasing out skip-connections," in *Proc. 27th Int. Conf. Artif. Neural Netw.*, Rhodes, Greece, Oct. 2018, pp. 447–456.
- [59] "Torchvision 0.15 documentation," PyTorch, [Online]. Available: <https://pytorch.org/vision/stable/index.html>. [Accessed 04 June 2023].
- [60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
- [61] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
- [62] L. Holmstrom and P. Koistinen, "Using additive noise in back-propagation training," in *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 24-38, Jan. 1992, doi: 10.1109/72.105415.
- [63] R. Cipolla, Y. Gal and A. Kendall, "Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 7482-7491, doi: 10.1109/CVPR.2018.00781.
- [64] S. Liu, Y. Liang and A. Gitter, "Loss-balanced task weighting to reduce negative transfer in multi-task learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2019, pp. 9977-9978.

- [65] "scikit-learn," [Online]. Available: <https://scikit-learn.org/stable/>. [Accessed 01 August 2023].
- [66] C. Schaffer, "Selecting a classification method by cross-validation," *Machine learning*, vol. 13, pp. 135-143, 1993.
- [67] B. Jokanovic, M. Amin and B. Erol, "Multiple joint-variable domains recognition of human motion," *2017 IEEE Radar Conference (RadarConf)*, Seattle, WA, USA, 2017, pp. 0948-0952, doi: 10.1109/RADAR.2017.7944340.
- [68] W. Zhang and H. Griffiths, "Detection of multiple micro-drones via cadence velocity diagram analysis", *Electronics Letters*, vol. 54, no. 7, p. 441-443, 2018. <https://doi.org/10.1049/el.2017.4317>

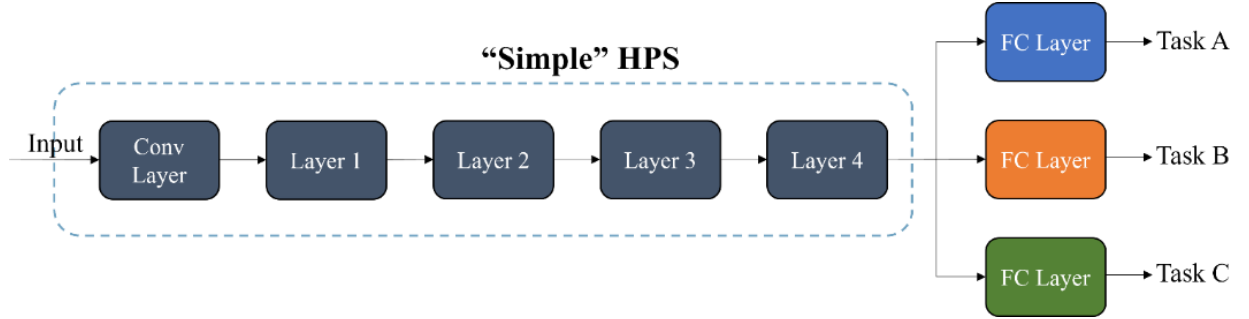
Appendix A

Layer name	Layer type	Number of parameters
Input Layer	Conv2d	9,408
	BatchNorm2d	128
	ReLU	--
	MaxPool2d	--
Layer 1	Conv2d	36,864
	BatchNorm2d	128
	ReLU	--
	Conv2d	36,864
	BatchNorm2d	128
	ReLU	--
	Conv2d	36,864
	BatchNorm2d	128
	ReLU	--
	Conv2d	36,864
	BatchNorm2d	128
	ReLU	--
Layer 2	Conv2d	73,728
	BatchNorm2d	256
	ReLU	--
	Conv2d	147,456
	BatchNorm2d	256
	Sequential	8,448
	ReLU	--
	Conv2d	147,456
	BatchNorm2d	256
	ReLU	--
	Conv2d	147,456
	BatchNorm2d	256
	ReLU	--
	ReLU	--

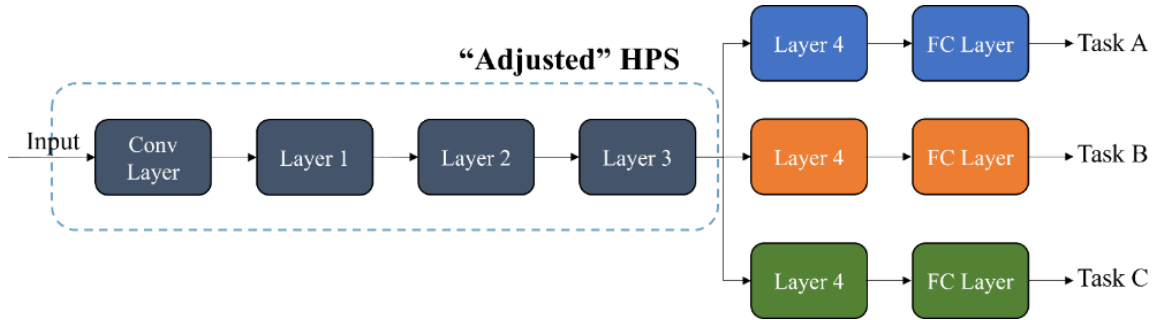
Layer 3	Conv2d	294,912
	BatchNorm2d	512
	ReLU	--
	Conv2d	589,824
	BatchNorm2d	512
	Sequential	33,280
	ReLU	--
	Conv2d	589,824
	BatchNorm2d	512
	ReLU	--
Layer 4	Conv2d	1,179,648
	BatchNorm2d	1024
	ReLU	--
	Conv2d	2,359,296
	BatchNorm2d	1024
	Sequential	132,096
	ReLU	--
	Conv2d	2,359,296
	BatchNorm2d	1024
	ReLU	--
Average pooling	AdaptiveAvgPool2d	--
Linear	Linear	513

Table A.1: Details of the ResNet18 architecture used in this thesis. The Layers 1-4 are the ones discussed also in Chapter 5.

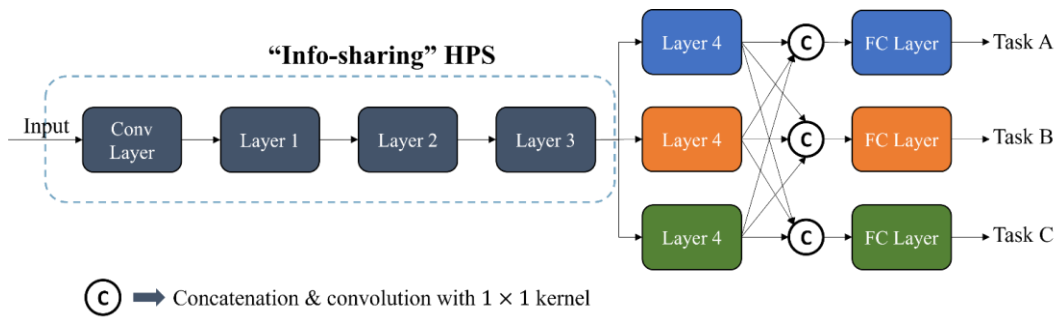
Appendix B



- a) The "Simple" HPS model with a three-task output. The same approach as the one of subsection 5.2.2 is followed.



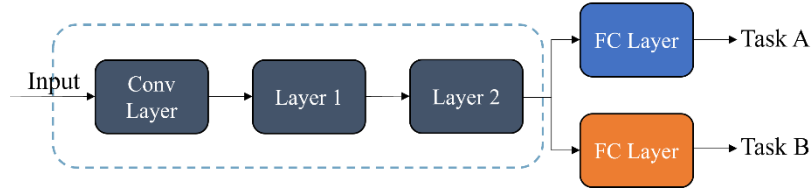
- b) The Adjusted HPS approach. Once again, three distinct task streams are defined, all starting with the fourth ResNet18 layer as the split takes place earlier.



- c) The Info-sharing HPS model. Here, information is communicated among all three tasks. In the same manner as shown in subsection 5.2.2, the feature maps are concatenated and then passed through a 1×1 convolution kernel.

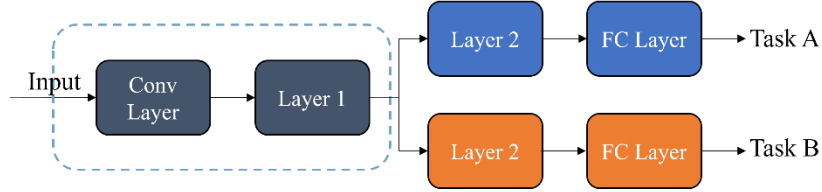
Figure B.1: The proposed MTL architectures adapted to predicting more than two tasks simultaneously.

“Simple” HPS – Cropped ResNet18



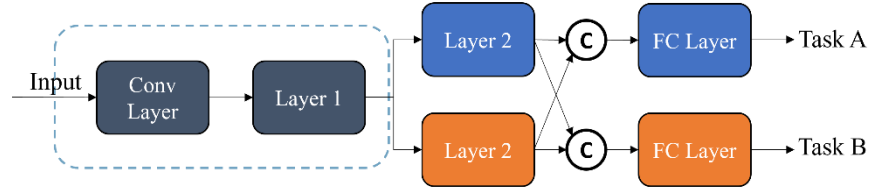
- a) The “Simple” HPS model. The split takes place after the second ResNet 18 layer and each task stream is assigned with a fully connected layer.

“Adjusted” HPS – Cropped ResNet18



- b) The Adjusted HPS model. Here, the split takes place right after Layer 1.

“Info-sharing” HPS – Cropped ResNet18



Ⓢ ➡ Concatenation & convolution with 1×1 kernel

- c) The Info-sharing HPS model. The output of each task’s Layer 2 is passed to the other, concatenated and filtered.

Figure B.2: The proposed MTL architectures adapted to predicting more than two tasks simultaneously.

Appendix C

This appendix contains supplementary information to the results of Chapter 6. It contains analytical comparisons among STL and MTL methods across a variety of metrics for all task groups examined in the chapter. Additionally, it provides information regarding the hyperparameters and the models assessed in the aforementioned chapter.

Wing Type Classification				Number of Rotors Classification		
Model	Metrics (macro averaged over 10 splits)					
	F ₁	Recall	Precision	F ₁	Recall	Precision
ResNet18-STL	94.36 ± 0.5	93.97 ± 0.4	94.78 ± 0.67	86.47 ± 0.66	86.33 ± 0.98	86.91 ± 0.69
ResNet18-HPS	95.27 ± 0.59	94.91 ± 0.48	95.69 ± 0.79	88.58 ± 1.05	88.21 ± 1.06	89.17 ± 1.09
ResNet18-Info-Sharing	95.56 ± 0.41	95.07 ± 0.38	96.11 ± 0.6	89.45 ± 1.22	89.16 ± 1.43	89.96 ± 1.04
(% increase)	(+1.2%)	(+1.1%)	(+1.33%)	(+2.98%)	(+2.83%)	(+3.05%)
ResNet18-Adjusted	95.35 ± 0.45	94.89 ± 0.49	95.86 ± 0.46	88.32 ± 1.16	88.53 ± 1.39	88.83 ± 1.06

Table C.1: Analytical comparison of STL and MTL performance in terms of F₁ score, recall and precision for the wing type and number of rotors task group when applied to Dataset A.

Wing Type Classification			Number of Rotors Classification			
Model	Metrics (macro averaged over 10 splits)					
	F ₁	Recall	Precision	F ₁	Recall	Precision
ResNet18-STL	93.81 ± 0.91	93.32 ± 1.08	94.36 ± 0.77	80.55 ± 1.95	80.18 ± 1.86	81.96 ± 1.93
ResNet18-HPS	94.87 ± 0.99	94.48 ± 1.04	95.32 ± 0.99	83.4 ± 1.45	82.67 ± 1.64	84.7 ± 1.45
(% increase)						(+2.74%)
ResNet18-Info-Sharing	95.01 ± 0.6	94.8 ± 0.6	95.29 ± 0.76	83.9 ± 1.23	83.79 ± 1.16	84.35 ± 1.43
(% increase)		(+1.48%)		(+3.35%)	(+3.61%)	
ResNet18-Adjusted	95.04 ± 0.62	94.74 ± 0.66	95.42 ± 0.62	83.66 ± 1.53	83.35 ± 1.86	84.53 ± 1.28
(% increase)	(+1.23%)		(+1.06%)			

Table C.2: Analytical comparison of STL and MTL performance in terms of F₁ score, recall and precision for the wing type and number of rotors task group when applied to Dataset B.

Wing Type Classification			No. Rotors Classification			Multiple Drone Detector		
Model	Metrics (macro averaged over 10 splits)							
	F ₁	Recall	Precision	F ₁	Recall	Precision	F ₁ - positive class	PR AUC
ResNet18-STL	93.81 ± 0.91	93.32 ± 1.08	94.36 ± 0.77	80.55 ± 1.95	80.18 ± 1.86	81.96 ± 1.93	38.92 ± 4.83	0.5 ± 0.07
ResNet18-HPS	94.21 ± 0.85	93.83 ± 0.99	94.68 ± 0.85	81.79 ± 1.34	81.34 ± 1.64	82.74 ± 1.44	64.57 ± 5.52	0.71 ± 0.05
(% increase)	(+ 0.4%)	(+ 0.51%)	(+ 0.32%)	(+ 1.24%)	(+ 1.16%)		(+ 25.65%)	(+ 34.71%)
ResNet18-Info-Sharing	93.84 ± 0.91	93.54 ± 0.71	94.25 ± 1.18	80.2 ± 1.13	79.35 ± 1.18	82.44 ± 2.06	59.04 ± 3.23	0.67 ± 0.05
ResNet18-Adjusted	94.1 ± 0.5	93.79 ± 0.51	94.49 ± 0.54	81.4 ± 1.2	80.8 ± 1.43	83.32 ± 1.8	63.14 ± 5.13	0.69 ± 0.06
(% increase)						(+ 1,36%)		

Table C.3: Analytical comparison of STL and MTL performance in terms of F₁ score, recall and precision for the wing type, number of rotors classification and multiple drone detection task group applied to Dataset B.

Wing Type Classification			No. Rotors Classification			Payload Detection		
Model	Metrics (macro averaged over 10 splits)							
	F ₁	Recall	Precision	F ₁	Recall	Precision	F ₁ - positive class	PR AUC
ResNet18-STL	94.36 ± 0.5	93.97 ± 0.4	94.78 ± 0.67	86.47 ± 0.66	86.33 ± 0.98	86.91 ± 0.69	47.04 ±3.71	0.616 ± 0.07
ResNet18-HPS (% increase)	94.09 ± 0.84	94.02 ± 0.62	94.23 ± 1.19	86.37 ± 1.44	86.23 ± 1.29	86.8 ± 1.85	59.86 ± 4.09 (+ 12.82%)	0.667 ± 0.04
ResNet18-Info-Sharing (% increase)	94.66 ± 0.83 (+ 0.3%)	94.31± 0.86 (+ 0.34%)	95.08 ± 0.9 (+ 0.3%)	86.77 ± 1.19 (+ 0.3%)	86.81 ± 1.36 (+ 0.48%)	86.99 ± 1.08 (+ 0.08%)	59.79 ± 3.59	0.672 ± 0.03
ResNet18-Adjusted	94.12 ± 0.69	93.96 ± 0.55	94.36 ± 0.94	86.66 ± 1.01	86.23 ± 0.99	87.42 ± 1.38	58.57 ± 3.9	0.668 ± 0.04

Table C.4: Analytical comparison of STL and MTL performance in terms of F₁ score, recall and precision for the wing type, number of rotors classification and payload detection task group applied to Dataset A.

Task group - test type	Task	Classifier type	Backbone	Optimizer	Learning rate	Weight decay
WT & NR (Dataset A)	WT	STL	ResNet18	Adam	0.00005	0.001
	NR	STL	ResNet18	SGD	0.001	0.001
	WT & NR	“Simple” HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001
WT & NR (Dataset A)	WT	STL	ResNet18	Adam	0.00005	0.001
	NR	STL	ResNet18	SGD	0.001	0.001
	WT & NR	“Simple” HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001
Robustness test on mainstream targets	WT	STL	ResNet18 - cropped	Adam	0.00005	0.001
	NR	STL	ResNet18 – cropped	SGD	0.001	0.001
	WT & NR	“Simple” HPS	ResNet18 – cropped	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18 – cropped	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18 – cropped	SGD	0.001	0.001
WT & NR (Dataset A)	WT	STL	ResNet18	Adam	0.00005	0.001
	NR	STL	ResNet18	SGD	0.001	0.001
	WT & NR	“Simple” HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001
Robustness test on “exotic” targets	WT	STL	ResNet18 - cropped	Adam	0.00005	0.001
	NR	STL	ResNet18 – cropped	SGD	0.001	0.001
	WT & NR	“Simple” HPS	ResNet18 – cropped	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18 – cropped	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18 – cropped	SGD	0.001	0.001
WT & NR (Dataset B)	WT	STL	ResNet18	Adam	0.00005	0.001
	NR	STL	ResNet18	SGD	0.001	0.001
	WT & NR	“Simple” HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001

	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001
WT & NR & MDD (Dataset B)	WT	STL	ResNet18	Adam	0.00005	0.001
	NR	STL	ResNet18	SGD	0.001	0.001
	MDD	STL	ResNet18	SGD	0.001	0.001
	WT & NR & MDD	“Simple” HPS	ResNet18	SGD	0.001	0.001
	WT & NR& MDD	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR& MDD	Info-sharing HPS	ResNet18	SGD	0.001	0.001
WT & NR & PD (Dataset A)	WT	STL	ResNet18	Adam	0.00005	0.001
	NR	STL	ResNet18	SGD	0.001	0.001
	PD	STL	ResNet18	SGD	0.001	0.001
	WT & NR	“Simple” HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001
WT & NR & PD (Dataset A)	WT	STL	ResNet18	Adam	0.00005	0.001
	NR	STL	ResNet18	SGD	0.001	0.001
	PD	STL	ResNet18	Adam	0.00005	0.001
	WT & NR	“Simple” HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001
Robustness test - balanced	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001
WT & NR & PD (Dataset A)	WT	STL	ResNet18	Adam	0.00005	0.001
	NR	STL	ResNet18	SGD	0.001	0.001
	PD	STL	ResNet18	SGD	0.001	0.001
	WT & NR	“Simple” HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001
Robustness test - imbalanced	WT & NR	Adjusted HPS	ResNet18	SGD	0.001	0.001
	WT & NR	Info-sharing HPS	ResNet18	SGD	0.001	0.001

Table C.5: Analytical summary of hyperparameters used for training each classifier presented in Chapter 6. Manual search was utilized towards optimization. The presented hyperparameters led to the best performance.

Appendix D

This appendix contains supplementary information to the results of Chapter 7. As with Appendix C, it contains analytical comparisons among STL and MTL methods across a variety of metrics for all task groups examined in the chapter. Additionally, it provides information regarding the hyperparameters and the models assessed in the aforementioned chapter.

Mean Rotor Rate Estimation			Payload Detection	
Model	Metrics (averaged over the 10 splits)			
	MSE	MAE	F ₁ (%) – positive class	PR AUC
ResNet18-STL	9545.13 ± 1666.47	36.59 ± 2.6	42.74 ± 2.49	0.51 ± 0.04
ResNet18-HPS	10673.16 ± 1395.9	42.52 ± 2.95	55.89 ± 6.03	0.65 ± 0.06
ResNet18-Info-Sharing	11945.24 ± 1706.5	50.61 ± 3.33	56.97 ± 3.61 (+ 14.23)	0.67 ± 0.04 (+ 27.12)
ResNet18-Adjusted	11322.53 ± 1377.53	43.56 ± 3.53	53.35 ± 5.32	0.63 ± 0.08

Table D.1: Analytical comparison of STL and MTL performance in MSE, MAE, F₁ score and PR AUC for the mean rotor rate regression and payload detection task group applied to Dataset C.

Mean Rotor Rate Estimation			Event Detection	
Model	Metrics (averaged over the 10 splits)			
	MSE	MAE	F ₁ (%) – positive class	AUC (%)
ResNet18-STL	9545.13 ± 1666.47	36.59 ± 2.6	49.44 ± 5.73	0.5 ± 0.08
ResNet18-HPS (% increase)	9846.8 ± 1345.27	41.61 ± 3.42	66.37 ± 4.38 (+ 16.93%)	0.7 ± 0.06
ResNet18-Info-Sharing	11139.82 ± 1429.92	46.44 ± 2.85	60.68 ± 5	0.68 ± 0.05
ResNet18-Adjusted	11833.1 ± 1153.17	47.62 ± 2.54	60.68 ± 5.4	0.66 ± 0.07

Table D.2: Analytical comparison of STL and MTL performance in MSE, MAE, F₁ score and PR AUC for the mean rotor rate regression and event detection task group applied to Dataset C.

Task group - test type	Task	Classifier type	Backbone	Optimizer	Learning rate	Weight decay
RRR & PD (Dataset C)	RRR	STL	ResNet18	Adam	0.00005	0.001
	PD	STL	ResNet18	Adam	0.00005	0.001
	RRR & PD	“Simple” HPS	ResNet18	Adam	0.00005	0.001
	RRR & PD	Adjusted HPS	ResNet18	SGD	0.001	0.001
	RRR & PD	Info-sharing HPS	ResNet18	Adam	0.00005	0.001
Robustness test	RRR	STL	ResNet18	Adam	0.00005	0.001
	PD	STL	ResNet18	SGD	0.001	0.1
	RRR & PD	“Simple” HPS	ResNet18	SGD	0.001	0.1
	RRR & PD	Adjusted HPS	ResNet18	SGD	0.001	0.1
	RRR & PD	Info-sharing HPS	ResNet18	SGD	0.001	0.1
RRR & ED (Dataset C)	RRR	STL	ResNet18	Adam	0.00005	0.001
	ED	STL	ResNet18	Adam	0.00005	0.001
	RRR & ED	“Simple” HPS	ResNet18	Adam	0.00005	0.001
	RRR & ED	Adjusted HPS	ResNet18	Adam	0.00005	0.001
	RRR & ED	Info-sharing HPS	ResNet18	Adam	0.00005	0.001
Robustness test	RRR	STL	ResNet18	Adam	0.00005	0.001
	ED	STL	ResNet18	SGD	0.001	0.001
	RRR & ED	“Simple” HPS	ResNet18	Adam	0.00005	0.1
	RRR & ED	Adjusted HPS	ResNet18	SGD	0.001	0.001
	RRR & ED	Info-sharing HPS	ResNet18	Adam	0.00005	0.1

Table D.3: Analytical summary of hyperparameters used for training each classifier presented in Chapter 7. Manual search was utilized towards optimization. The presented hyperparameters led to the best performance.