Improving classification of ballistic, non-cooperative radar targets

Master's Thesis in Applied Computer Science

Frode Tennebø

January 31, 2022
Halden, Norway

Østfold University College www.hiof.no

Abstract

The defence industry is often regarded as being conservative with long development times and stringent validation criteria for implementing new technologies. Hence, machine learning (ML) is not widely adopted in this area - at least not publicly. In this thesis, we will investigate several ML methods to classify ballistic, non-cooperating projectiles, which can be used in weapon locating radar (WLR) systems, and compare these results with Saab's original and proprietary non-ML algorithms, to improve the classification performance.

Working with the defence industry or with defence products is often an exercise in balancing the ethical issues involved against the goals one wishes to achieve. We have, as part of this work, also considered some of the ethical implications in potentially improving the performance of WLR systems and used autonomous weapons systems (AWS) as an analogue.

The data was provided by and is proprietary to Saab. The filtered dataset usable for our research consists of 2 259 unique radar tracks, just over 1 % of the total tracks available, with a total of 189 125 individual radar plots.

We have compared two proprietary Saab algorithms with the ML algorithms support vector machine (SVM), k-nearest neighbour (k-NN), artificial neural network (ANN), C5.0 and XGBoost, with a focus on the latter. We conclude that for our best XGBoost model using hyper-parameter optimised plot features to classify into six projectile classes, the accuracy increases from 0.852 to 0.984.

Keywords: Machine learning, ML, decision trees, time series classification, projectile classification, non-cooperating radar target classification, artillery.

Acknowledgements

The author is employed by Saab Technologies Norway AS, a subsidiary of Saab AB in Sweden, a global defence and security corporation, and would like to thank them for the opportunity to use their proprietary data in this research. Thanks also go to Håkan Warston, who has been the supervisor from Saab AB for this work.

Apart from providing the data and supervision, Saab has not contributed to this research and cannot be held responsible for any errors or misjudgements by the author.

Also, thanks to Kåre Stavnes, Terje Sparre Olsen and the late Andreas Mattsson for insight and advice early in the process of this master's thesis.

I would also like to thank my supervisor Marius Geitle from Østfold University College, who have pushed me and provided invaluable support and inspiration throughout this project.

Finally, special thanks go to my family for their patience.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

"Now the reason the enlightened prince and the wise general conquer the enemy whenever they move and their achievements surpass those of ordinary men is foreknowledge."

— Sun Tzu, *The Art of War*

Artillery weapons[1] have been used in armed conflicts since the invention of gunpowder. They are a category of weapons that consists of a barrel where explosives are used to fire a projectile towards a target. They come in various sizes, firepower and modes of operation. They are typically further divided into rockets, artillery guns and mortars. Common is that after the projectile has left the barrel[2] it follows a ballistic trajectory. The advantage of these types of weapons, both for offence and defence, is that they can be fired from a relatively safe location within own territory and hit targets within enemy territory, over 70 km away [1]. However, depending on the precision of intelligence and the weapons themselves, they can also cause collateral damage. Their effectiveness has in recent times been brutally illustrated in the battle of Zelenopillya in Ukraine in July 2014, where Russian artillery took out most of Ukraine's mechanised brigades in less than three minutes [2].

Hence, the detection and location of enemy artillery fire, known as weapon locating (WL), has been an important military task since the invention of artillery weapons. During World War One, apart from visually seeing the launch site, the point of origin (POO), flash detection or flash spotting of the artillery piece was used. This method required multiple forward observers spread out geographically who communicated individual results back to a command post which could then perform triangulation [3]. This technique is time-consuming and somewhat error-prone as it required and still requires that forward observers are close enough to see the flash of the weapon, possibly beyond enemy lines, putting them dangerously close to the enemy. It also requires the different observers to report on the same piece, which can be challenging in a battle situation.

Impact crater analysis [4] is another method that can also be performed where the direction of the impact crater will show the direction of the weapon. This method is potentially even more dangerous than flash spotting since soldiers performing this task will need to be quite close to the crater to perform the analysis. Around the same time as

[1] Some literature also refer to these collectively as ballistic weapons, indirect fire or just RAM (for rocket, artillery, mortar).

[2] Or when the propellant has stopped burning for rocket artillery.

flash spotting came sound ranging [5, 3] where the sound of the explosion from the artillery piece is detected from several "listeners" and triangulated. This can be performed in a more timely manner than the other methods, and has since been automated and is also used today as a passive system. The drawback of sound ranging is that propagation of sound depends on the topology and the technique is limited by the speed of sound, which in some cases means that the projectile impacts before the sound of the firing reach the detector, even though the projectile can follow a longer, ballistic path than the sound. This situation changed with the invention of the radar[3].

A radar is a system that uses radio waves[4] to detect objects in the air, potentially also in space, or on the surface of the earth. A weapon locating radar (WLR) is a dedicated and specialised radar system that can detect and track[5] artillery projectiles and estimate the POO, launch site, and point of impact (POI), impact site, while the projectile is still in the air. Based on these estimations, directions are provided for effective counter-battery fire and timely warnings are given to own troops [6].

One important part of this process is projectile classification; determining what type of ammunition is used, typically either of rocket, artillery, mortar (RAM). This information can then be used in the ballistic calculation of POO and POI and to identify the weapon type, also called weapon classification. This information is ultimately used for warning, threat evaluation of the hostile battery and tactical and strategic evaluation of the capability of the opposing force.

This work will explore methods of projectile classification using machine learning (ML) and compare them with Saab's existing, proprietary algorithms. The rest of this chapter describes the motivation and research questions for this work.

1.1 Motivation

Several manufacturers of WLR systems exist. One of these is Saab, which has been developing WLR capabilities since the late 1980s, first (as Ericsson) in the dedicated WLR-radar ARTHUR, originally as a joint effort between the armed forces of Norway and Sweden [7], in several generations, and also in the GIRAFFE AMB surveillance radar family [8] and other radar systems.

ARTHUR has been delivered in multiple generations with steady improvements in performances. This also includes the projectile classification algorithm, which comes in two major generations: Generation 1, which only classifies in the main projectile types, either rocket, artillery gun or mortar, and generation 2, which also extends the classification further with light and heavy calibres[6], into each of mortar light (ML), mortar heavy (MH), gun light (GL), gun heavy (GH), rocket light (RL) and rocket heavy (RH).

To fire an artillery projectile at a target, the firing geometry of the different artillery weapons, typically firing elevation (also called quadrant elevation) and firing velocity (also called muzzle velocity), is used to direct the shot. This information is normally found in *ballistic tables*, tables of various parameters used to adjust each type of weapon based on

[3] Originally an acronym for **ra**dio **d**etection **a**nd **r**anging

[4] Electromagnetic radiation with wavelengths longer than infrared light.

[5] Tracking is the process of predicting the next potential position of an object, commanding the radar to "see" in that direction and correlating the following detection with a previous detection.

[6] Calibre is the diameter of the gun barrel and hence the diameter of the ammunition/projectile, indicating the projectile's potential impact force.

external requirements and properties like distance to shot, ammunition to use, wind and temperature. Today, some artillery systems calculate this automatically for each round based on sensor information and the required POI.

One force's firing parameters are naturally not available to the other force receiving the incoming round. Instead, they rely on some sort of WL capability, e.g. a WLR, to estimate the firing geometry and potentially other information about the projectile. These features can then be used in a classification algorithm, similar to the simplified early Saab algorithm in Listing 1.1. Here selected features, based on domain knowledge, are checked against constants derived from *a priori* knowledge of weapon characteristics, e.g. from the aforementioned shooting tables. When the type of the projectile is determined this information can be used to estimate the POO and POI based on a ballistic model of the projectile type.

Listing 1.1: Pseudo-code example of a man-made decision tree for projectile classification. Here the specific projectile type will be classified as either rocket gun mortar or unknown based on the features firing elevation firing velocity and RCS.

```
if firing_elevation <= C_FIRING_ELEVATION_LIMIT then
  if firing_velocity <= C_VELOCITY_LIMIT_UNKNOWN then
    projectile_type := UNKNOWN;
  elsif firing_velocity <= C_VELOCITY_LIMIT_ARTILLERY then
    projectile_type := ARTILLERY;
  else
    if rcs <= C_RCS_LIMIT_ROCKET then
       projectile_type := ARTILLERY;
    else
      projectile_type := ROCKET;
    end if;
  end if;
else
  if firing_velocity <= C_VELOCITY_LIMIT_MORTAR then
     projectile_type := MORTAR;
  else
     projectile_type := ARTILLERY;
  end if;
end if;
```

Saab has for many years recorded data from live firings. Each recording consists of several radar settings, tactical information, environmental information like weather data and multiple individual radar measurements, sampled several times every second, together with their respective tracking prediction for zero, one or more tracks. All information is recorded in human-readable form in regular files called *track files*. Radar measurements include measured azimuth, elevation, radial velocity, range, signal strength and signal-to-noise ratio (SNR)[7] and some features to control the measurement. Additionally, for each track, some features are estimated for the overall track and included in the recording. These synthetic features include the estimated firing elevation and velocity, coordinates for POO, POI and projectile classification, to mention a few.

These files can be from a few seconds up to several minutes long and are used for retrospective analysis of radar performance. They will also be used for our research.

[7]SNR is the ratio of the power from the returned signal relative to the current noise level as measured by the radar

However, there are several challenges with this data. Firstly, many of the track files do not have truth data associated with them or the recorded truth data is not correct. Secondly, the measurements can be inaccurate or incorrect. To use these track files as a benchmark for our research, their validity and correct weapon classification need to be determined through investigation of historic firing trial plans and evaluation reports, interviewing key people involved in the trials and finally a manual vetting of any remaining track files.

In practice, the simplified classification approach described above has not given as good results as desired. Some of the problems with this approach are:

- There is a large variety in projectiles, calibres and weapon systems. It is unrealistic to obtain data for all possible projectile types.
- Even with *a priori* knowledge, it is not unlikely that other features than the ones already used by Saab can give better results.
- The boundaries for the limits of these features may be overlapping, which means that the order of evaluation can determine the result. Therefore, a more complex algorithm could give better results.
- The firing elevation and firing velocity are estimated from the radar track based on an initial "guess" of the projectile type. The later in the trajectory of the projectile it is detected and tracked, the harder it is to get a correct "guess" and hence correct estimation which in turn could give incorrect classification.
- Measurements are influenced by the environment the radar operates in, including, but not limited to, electronic disturbances, precipitation, atmospheric temperature gradient and wind conditions. Estimated firing elevation and firing velocity might therefore be inaccurate.
- The number of combinations of geometries in three dimensions is very large. The classification needs to take this into account.

The goal of this research, by potentially utilising more features from the radar, either measured or synthetic, is to achieve improved projectile classification performance.

Determining the correct projectile type is important for several reasons:

- Accuracy of POO and POI are improved (the error ellipse is decreased). This means that for hostile fire, impact warnings can be issued earlier and more specifically, with fewer false warnings, potentially saving lives.
- The projectile type is closely linked to the weapon type, and correct determination of the weapon type can give intelligence about hostile forces and their deployment, giving a tactical advantage.
- The countermeasures towards a POO can be used more sparingly, saving time, resources and potentially minimising collateral damage such as civilian lives.
- Timely countermeasures can also reduce the effectiveness of hostile weapons, potentially also saving lives [9].
- It is conceivable that the result from the classification can be used to detect false targets, e.g. birds or planes, thereby avoiding undesired alarms and the use of countermeasures.

1.2 Research questions

Firstly, we want to see how well ML can classify our data:

RQ 1 *How good accuracy can be achieved using machine learning techniques compared to Saab's original algorithms?*

Secondly, since the original algorithm uses synthetic features, we want to compare measured features with synthetic features:

RQ 2 *How can measured features improve the classification power?*

Thirdly, we are also interested in how early we can expect good enough results from the projectile classification:

RQ 3 *How does the accuracy of the classification evolve over time for a track?*

Chapter 2

Background

"The bomb may have ended the war but radar won it."
— Louise Brown, *A Radar History of World War II: Technical and Military Imperatives*

This chapter will cover general radar theory and principles as well as the specifics of artillery weapons and WLR. In addition, an introduction to machine learning (ML), various ML techniques and theory as well as existing work related to ML in radar applications before summing up.

2.1 General radar principles

Radar is a collective term used for several technologies which use electromagnetic (EM) waves in the radio spectrum, typically 100 MHz to 100 GHz, to detect and locate objects. Depending on the objective of the radar, whether its purpose is to detect close proximity pedestrians or other vehicles in modern cars, weather patterns in the different layers of the atmosphere, planes at long range, tactical ballistic missiles at even longer ranges or sea vessels, to mention just a few applications, various technological compromises are made. The general working principle is depicted in Figure 2.1.

The various wavelengths have different physical qualities. In general, higher frequencies, which results in shorter wavelengths for a given antenna size, give better angular resolution, but will also suffer more atmospheric attenuation and hence a shorter range for a given output power. However, this higher resolution gives more accurate angle measurements which in turn translates to a longer range. Shorter wavelengths also mean that the antenna can be smaller which impacts the physical size and weight of a system, which are important attributes for mobile systems. However, a bigger antenna in relation to the wavelength, means more power and better directionality which again gives better performance. Hence, designing a radar becomes an exercise in weighing multiple compromises against each other to optimise for the objectives of the specific radar.

There are several ways of classifying radars in use today which reflects the difference in usage. Below is a non-exhaustive list of physical and functional properties that defines a radar.

- Frequency band; e.g. S-, C-, X- and Ku-band.
- Continuous-wave vs. pulse-Doppler.

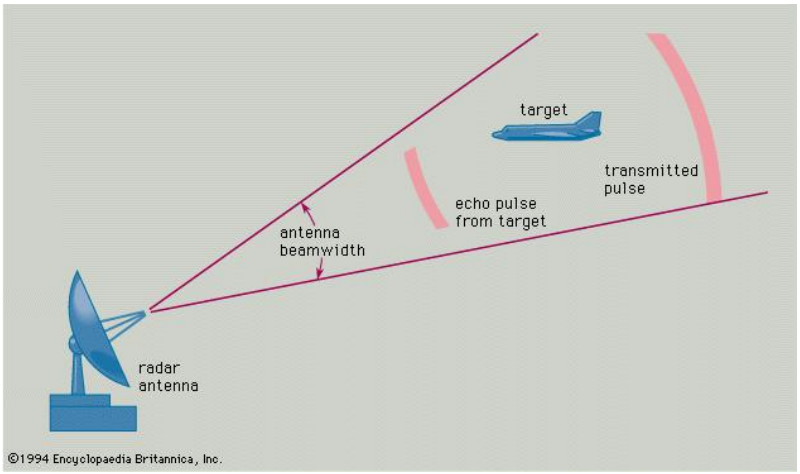


Figure 2.1: The general working principle of a radar where the radar sends an electromagnetic pulse and the target reflects an echo back to the radar. Image source: Encyclopædia Britannica.

- Fixed antenna for sector coverage vs. rotating for 360° coverage.
- Mobile vs. stationary platform.
- Passive vs. active radars, where the former does not have a transmitter, but depends on ambient EM illumination, while the latter combines a transmitter and receiver.
- Geospatial configuration; mono-static, bi-static or multi-static where the transmitter and receiver have varying degrees of co-location.
- Primary vs secondary, where the former predominantly deals with non-cooperating targets and the latter requires an active transponder in a cooperating target that is interrogated by the radar.
- Surveillance radar vs. tracking radar, where the former usually covers more volume searching for objects while the latter attempts to actively predict and follow objects.
- Passive electronically scanned, with a single, powerful transmitter, or active electronically scanned, with multiple, smaller transmitters.

In a setting where we have non-cooperative targets using a primary surveillance radar (PSR) the process of identifying a target is called non-cooperative target recognition (NCTR). We will continue this chapter by describing a typical monostatic passive electronically scanned array (PESA) radar before describing in more detail a WLR.

2.1.1 Monostatic passive electronically scanned array radar

A monostatic radar consists of a few basic components: a transmitter, which also makes it an active radar, an antenna, a receiver and some processing capability. The basic working principle is illustrated in Figure 2.2. If an object is within the propagation path of the radio wave it will reflect some of this energy back to the antenna where it will be fed to the receiver. If the receiver can discern this echo from the background noise, it has a *detection* that is processed into a *plot*. Depending on the type of radar, the state of the signal and other factors, additional detections are needed for the processing capability to establish a *track*. A track, in turn, is a series of plots that the radar determines are associated with the same physical object.

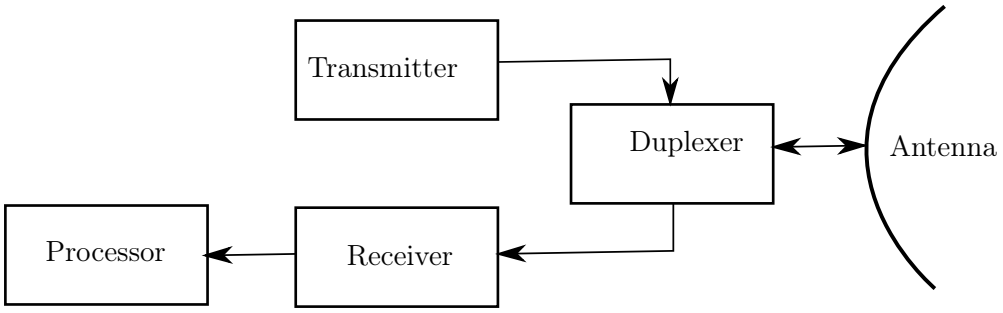


Figure 2.2: A basic radar system block diagram: The duplex unit switches between the transmitter and receiver, which handles signals to and from the antenna, while the processor works on the received data.

However, a detection is not necessarily from a target. If a signal is above the background noise and not from a target of interest for the radar, this signal is called *clutter*. Different types of clutter exist in different environments, e.g. precipitation, birds, windmills, traffic or the ground or sea. Clutter in one setting can be a target in a different setting, e.g. birds in the field of radar ornithology, or rain in a weather radar system. To avoid spending resources on clutter, most radars have *clutter maps* and other technologies to suppress or filter unwanted clutter as early in the processing chain as possible. What remains of detections from a primary radar after all filtering is performed are often referred to as non-cooperating radar targets.

The physical limitations of a radar are given by the radar equation in Equation (2.1).

$$R^4 = \frac{P_t t_p G_t G_r \lambda^2 \sigma}{(4\pi)^3 P_{min}}, \tag{2.1}$$

Where:
R = distance from the transmitter to the target
P_t = peak transmitter power
t_p = transmit pulse length
G_t = gain of the transmitting antenna
G_r = gain of the receiving antenna
λ = transmit wavelength
σ = radar cross-section, or scattering coefficient, of the target
P_{min} = minimum detectable signal

The higher the numerator or the lower the denominator, the longer the detectable range can be expected. The signal path starts with the transmitter transmitting a pulse with peak power P_t (watts) for a length of t_p (seconds) at a wavelength of λ (meters). For a monostatic PESA antenna, the transmitter and receiver gain can often be considered equal and shortened to G^2 (dBm). The amount of echo returned from a target is directly proportional to its radar cross-section (RCS), σ (m^2). The constant 4π shows that the transmitted and received energy scatter uniformly in the shape of a sphere and the third power relates to the transmitting antenna, the scatter from the object itself and the receiving antenna[1]. The sensitivity of the receiver to discern signals from the

[1]Even though the transmitting and receiving antenna is physically the same antenna, losses happen in

background is P_{min}, and since the power of a signal scales with the square of the range R in both directions the range term is R^4.

The processing unit will then take the signals from the receiver unit, separate detections and produce plots for transmission to a display system or external command and control (C2) system. Each plot typically consists of measured properties like 3D position, radial velocity and SNR, but can also include features used to control the measurement. From this, other parameters like speed, acceleration and direction are calculated. A series of correlated plots identified to belong to a single target, a track, might be produced by the processing unit - or sometimes this is done in the C2.

A radar is not a perfect instrument and hence does not measure perfectly. It is influenced by the environment it operates in and is sensitive to noise from processes around it. The actual measurement will vary with properties of the target, e.g. variations in RCS over time, and can in addition be ambiguous depending on the radar technology used. In other words: a combination of stochastic noise and systematic errors is constantly introduced to the measurements.

An air surveillance radar typically manages the entire volume of 360° horizontally and potentially up to 90° vertically while a tracking radar manages a smaller volume. Often, a surveillance radar has some tracking capability and a tracking radar has some surveillance capability, and in some specialised cases, a surveillance radar can queue a tracking radar. The available energy budget must be divided into searching this volume for new targets and possibly tracking existing targets. There are several ways to expend this energy budget and the search pattern is usually proprietary from vendor to vendor, radar to radar and task to task. The data extracted from this process is normally forwarded to a C2 system or presented to an operator for further processing or decision making.

This was a very brief introduction to general radar principles. For a more in-depth understanding of the operation of radars, we refer to dedicated books on this topic like Heimish Meikle's *Modern radar systems* [10].

2.2 Weapon locating radar

For a brief history of early weapon locating radar history with a discussion of some of the issues involved when designing a WLR, *Firefinder, a Radar Forty Years in the Making,* by William Fishbein [9] is recommended. It is mostly based on US history, but notes that the English were first to detect projectiles in combat using radar in February 1942. Since then, many countries have developed domestic WLR capabilities, both for own use and export. Besides USA and Norway/Sweden, these countries include Poland [11], India [12], a joint effort between Germany and France [13] and Israel [14] among others.

The purpose of a WLR is to find hostile artillery weapons and warn for incoming artillery impacts. These weapons fires projectiles that follow a ballistic trajectory. Therefore it is possible to estimate the projectile's POO and POI from measurements of a section of the ballistic trajectory, using ballistic models. Since RAM projectiles are typically fired from a position on the surface of the earth, either land or sea, the projectiles will first be visible for the WLR when they emerge somewhere over the horizon as seen by the radar. Hence a WLR has a predictive and defined search pattern where only a limited sector above the horizon needs to be searched. There is therefore no

---

both directions.

need to search 90° vertically nor is there usually a need to search 360° horizontally since the area of interest will be in a specific direction past the enemy front line. This is illustrated in Figure 2.3.

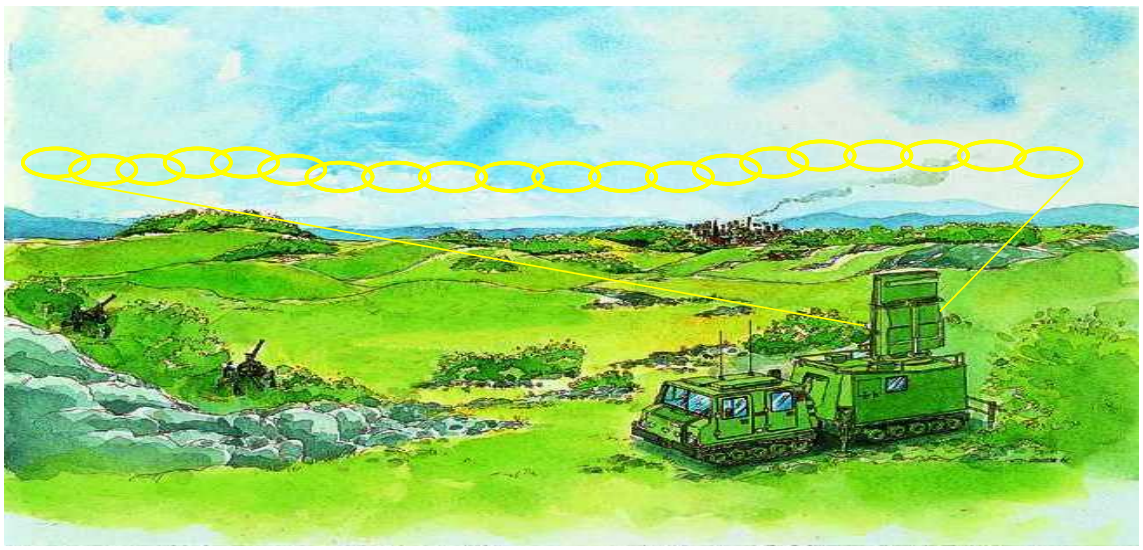


Figure 2.3: The search pattern of a WLR (in yellow) shows that a sector of the horizon is searched. This typically happens several times every second to detect projectiles emerging over the horizon. To avoid ground clutter, the search is performed slightly above the actual horizon. Image source: Saab.

When it has *acquired* a potential projectile, *tracking* is performed for as long as necessary or for as long as the projectile is visible for the radar. This is illustrated in Figure 2.4.

If or when enough points have been gathered and they are of good enough quality, a numerical parameter estimation, typically using a statistical inference method like a relatively simple loss function like least squares, or more complex methods like maximum likelihood estimation (MLE), is performed to fit the radar measurements to a ballistic model. The model can include various degrees of freedom, but as a minimum firing velocity, position and velocity vector, as well as a drag estimate at some point in the trajectory, must be determined. This information can then be used to calculate POO and POI, as illustrated in Figure 2.5.

For a mass-point model for trajectory calculation, the differential equation to solve can look like the one given in Equation (2.2). One component, a_{drag}, is dependent on the projectile type since the drag through the air is strongly dependent on the size and shape of the projectile, mode of stabilisation in flight, as well as the current velocity. But at the time of detection, this information is unknown to the radar system, hence it is important to identify the correct projectile type to select the best model to get a good estimation of POO and POI [15].

$$\frac{dv}{dt} = a_{gravity} + a_{Coriolis} + a_{drag}(v), \tag{2.2}$$

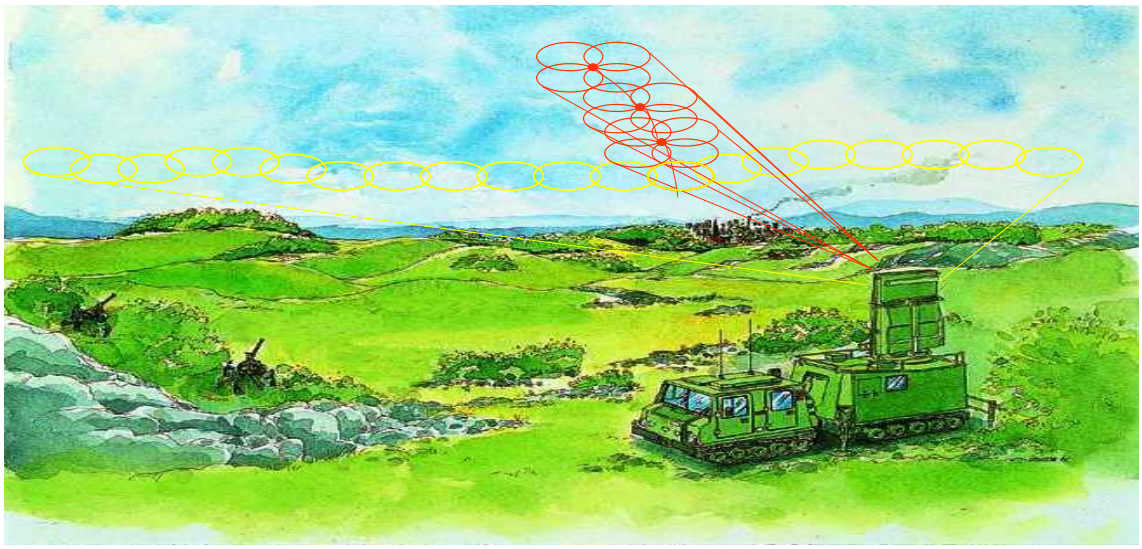


Figure 2.4: The tracking (in red) of a single projectile after it has been detected. A WLR can typically track several projectiles simultaneously while continuing the search procedure described in Figure 2.3. Image source: Saab.

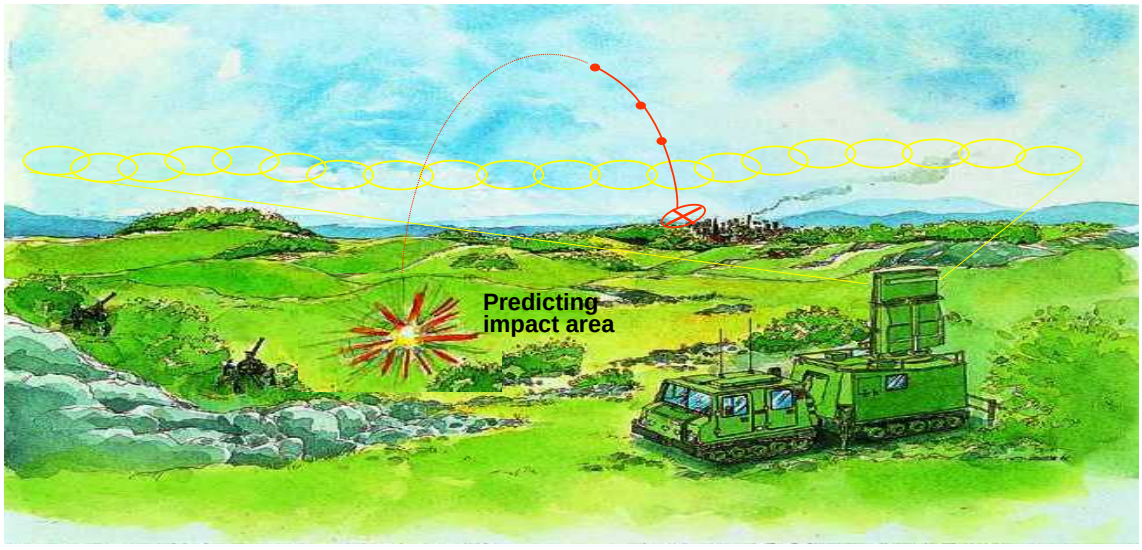


Figure 2.5: The calculation of the POO (far) and POI (near) is performed using the radar plot information of the track. Image source: Saab.

where:
t = time
v = velocity
$a_{gravity}$ = standard gravity (9.80665 m/s^2)
$a_{Coriolis}$ = Coriolis acceleration component
a_{drag} = projectile type dependent drag acceleration component

In principle, the drag component can be measured by the radar given high enough accuracy of the measurements and that a sufficient portion of the trajectory has been measured. Since this is often not possible to obtain directly, we still need to determine this differently. One mechanism for doing this is to first guess the projectile type and then estimate an initial firing elevation and firing velocity based on the drag model. Based on this, a prediction of the projectile type is done, see Listing 1.1 for a simplified algorithm, and the estimation of firing elevation and firing velocity is performed once more based on the predicted projectile type if it differs. Typical values for elevation and velocity limits are presented in Table 2.1.

Table 2.1: Example of typical limits of firing elevation (vertical) and firing velocity (horizontal) for different projectile types. Note that for low firing elevation and high firing velocity, there are some ambiguities as to which projectile type it can be.

Firing elevation					
	$> 45°$	Mortar	Mortar	Gun	Gun
	$> 0°$	Unknown	Gun	Gun	Rocket (gun)
		> 0 m/s	**> 175 m/s**	**> 350 m/s**	**> 600 m/s**
		Firing velocity			

However, if we can classify the projectile type before the model fitting is performed, this second estimation is superfluous.

2.3 Artillery weapons

Mortar, also called infantry weapon, is a highly mobile and lightweight weapon using projectiles (also called *shells*) with fins, of relative low calibre. Mortar projectiles are usually fired at >45° elevation angle at sub-sonic speeds. A gun, also called Howitzer, field or conventional artillery or just cannon, uses spin stabilised projectiles and is usually of a higher calibre than mortar. It uses a wide variety of firing geometries which can give short to medium range. Rocket artillery is normally fin stabilised, and usually of the same or higher calibre than gun artillery, but are usually longer in size as the back of the rocked is filled with propellant. Rockets are also usually fired at <45° elevation angle with a short, accelerating burn phase reaching a high maximum speed after around two seconds. This means that they have a longer range, but also means that they are not ballistic until the burn phase ends. All of these three main projectile types can be grouped into two calibre groups each; light and heavy. For simplicity, they are referred to as ML, MH, GL, GH, RL and RH for mortar light and heavy, gun light and heavy and rocket light and heavy respectively.

The flight time can be anywhere from a few seconds for some mortars weapons to several minutes for some rockets and guns. It is therefore imperative to get good measurements

as early as possible to get good accuracy at POO and thus good enough accuracy to warn at POI.

Examples of ranges for some potential classification parameters can be seen in Table 2.2. Some of these must be inferred from measurements, like RCS[2], others can be calculated, like firing velocity, while yet others, like calibre, can only be inferred by a combination of other parameters. As can be seen, the ranges are overlapping and hence the classification problem is more complex than what Table 2.1 suggests.

Table 2.2: Examples of some typical ranges of potential parameters for various projectile types. However, plenty of variation exists. RCS values are not disclosed here for proprietary reasons.

Parameter	**Mortar**		**Gun**		**Rocket**	
	Light	**Heavy**	**Light**	**Heavy**	**Light**	**Heavy**
Calibre (mm)	35-81	81-120	50-122	122-220	60-200	200-950
Firing velocity ($\frac{m}{s}$)	150-300	150-350	300-750	350-850	650-1000	750-1050
Firing elevation (°)	55-80	55-80	15-40	15-40	15-40	15-40
RCS (m^2)	WND	WND	WND	WND	WND	WND

In some cases, a track is not associated with real weapons fire. This is called a *false target* and can also be produced by clutter, even with a clutter map that attempts to cancel this as early as possible in the processing chain. However, we can use the confidence level of the projectile classification to flag a track as a false target and hence remove it later in the chain.

2.4 Performance metrics

To evaluate the relative performance of our experiments we need to establish suitable metrics.

Accuracy is one of the most used ranking metrics for classification performance. It is defined as the ratio between correct predictions and all predictions [16], see Equation (2.3).

$$accuracy = \frac{correct\ prediction}{total\ number\ of\ predictions}. \tag{2.3}$$

Specifically, for a binary classification problem the equation becomes:

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ positives + false\ negatives}. \tag{2.4}$$

Often, the complement, the misclassification rate or error rate, is used:

$$error\ rate = \frac{incorrect\ prediction}{total\ number\ of\ predictions}, \tag{2.5}$$

which gives:

$$accuracy = 1 - error\ rate. \tag{2.6}$$

[2]RCS is the ratio of power per unit area reflected toward a receiver divided by the power per unit area that was incident on the target. A radar receiver does not directly measure RCS but can infer its value from measurements of signal strength and range applying the radar equation.

More formally:

$$accuracy = 1 - \overline{err}, \text{ where } \overline{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)), \tag{2.7}$$

where N is the number of data points, L is some loss function, e.g. squared error loss for a regression problem, x_i and y_i are samples from our data, $(\mathbf{X}, \mathbf{Y})$, and f is the prediction function.

Another term which is used together with the performance metric is the standard deviation (SD), often also just denoted σ:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(L(y_i, f(x_i)) - \overline{err})^2}. \tag{2.8}$$

We can then calculate the *confidence interval*:

$$CI = \overline{x} \pm z\frac{\sigma}{\sqrt{n}}, \tag{2.9}$$

where $\overline{x}$ is the n mean value, z is the confidence level constant and n is the number of samples.

However, accuracy can also easily mischaracterise the predictive performance, particularly when used with unbalanced datasets [17].

Area Under the Curve of the Receiver Operating Characteristics (AUC ROC), or simply AUC, is the metric that is derived from calculating the ROC space and describes the miscalculation cost. The ROC space is in turn defined by the false positive rate vs. true positive rate:

$$ROC = \frac{1 - specificity}{sensitivity}, \tag{2.10}$$

while varying the decision threshold. Hence AUC has been suggested as a better measure than accuracy [18].

Specificity and sensitivity are for a binary classification defined as

$$specificity = \frac{true\ negatives}{true\ negatives + false\ positives}, \tag{2.11}$$

and

$$sensitivity = \frac{true\ positives}{true\ positives + false\ negatives}. \tag{2.12}$$

In other words, the ratio of correct predictions of "no signal" and "signal" respectively. However, more recent research has suggested that AUC is not a good classification measure for small datasets [19] and that it has other significant problems in model comparison [20] [21].

2.5 Machine learning

Machine learning is a subfield of computer science and information processing that "aims to give computers the ability to learn without being explicitly programmed" [22]. The start of this field can be traced back to Alan Turing's paper *Computing machinery and intelligence* [23] where he addressed the problem of "thinking machines" and also proposed an experiment which today is known as the *Turing test*. He proposed that this machine should not attempt to simulate the adult mind, but start by simulating a child and then teaching and evolving it to an adult, which is very close to how most ML algorithms work today.

ML algorithms are often divided into groups based on common characteristics. Marsland divides them into four groups based on their learning paradigms [24]:

- **Supervised learning:** Based on a training set of examples where the correct responses are provided, the algorithm generalises to respond mostly correct to all possible inputs.
- **Unsupervised learning:** Correct responses are not provided, but instead, the algorithm tries to identify similarities between the inputs so that inputs that have something in common are categorised together.
- **Reinforced learning:** The algorithm gets told when the answer is wrong, but does not get told how to correct it. It has to explore and try out different possibilities until it works out how to get the answer right.
- **Evolutionary learning:** Computer models are made to evolve, using the idea of fitness, similar to how biological organisms adapt and evolve to improve their survival rates in their environment.

However, evolutionary learning is often seen as a group of optimisation methods instead [25]. Goodfellow et al. [16] focus on how the algorithms experience the training data during the learning process and as such do not include evolutionary learning as one of the main groups.

The purpose of most ML algorithms is to be able to predict the future of a process based on historical data of the same process, discover patterns in data or maximise the performance of a process. Multiple algorithms have emerged over time depending on the specific problem to solve, the amount and type of the data, the state of the research at the time of a specific algorithm's introduction, to name a few factors.

Both regression and classification problems can be solved by ML algorithms. They range from simple linear models, like linear regression, more complex non-linear models like artificial neural network (ANN) and support vector machines, and tree and rule-based models like random forest, again to name just a few algorithms [26].

Instead of using just one algorithm for the entire problem, a common technique is to combine several models in various ways to obtain better predictive performance, which is called *ensemble methods*. Some common types of ensembles are *bagging*, *boosting* and *stacking*. They work by adjusting the dataset or combining learners [16].

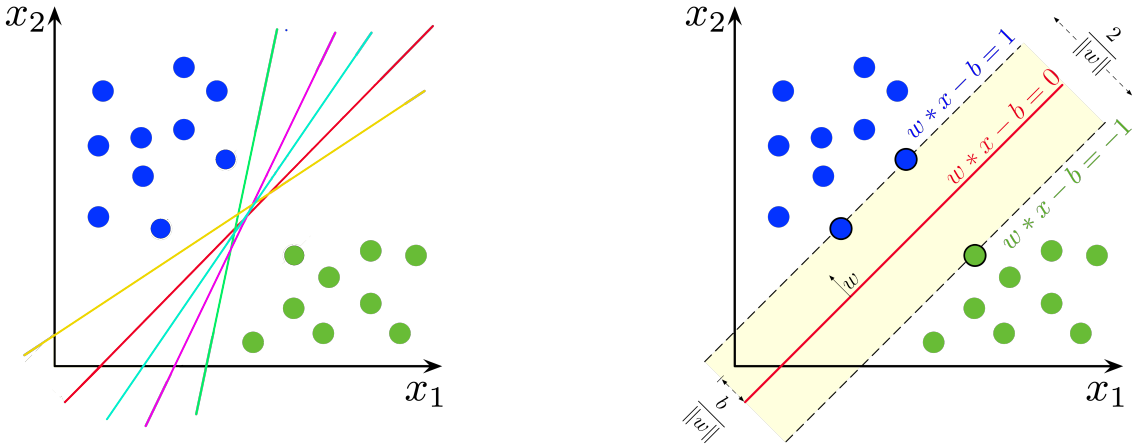
Our problem is a classification problem involving time-series data that have truth data, hence we will use supervised learning algorithms like the non-linear models support vector machine (SVM), k-nearest neighbour (k-NN) and ANN and tree-based models like C5.0 and XGBoost. Temporal sensitive algorithms like long short-term memory (LSTM) [27] and gated recurrent unit (GRU) [28] will also be considered during our research.

2.5.1 Support vector machines

In a traditional classification problem, any classification algorithm attempts to select a separation based on the accuracy of the classifier. This is illustrated in Figure 2.6 (a) where there is an infinite number of linear partitions to perfectly classify the data.

Support vector machine (SVM) attempts to define the *best* partition by defining a metric called *margin*, which describes the distance between the partition and the closest training set point, and then maximises this metric [26]. This is illustrated in Figure 2.6 (b).

This example is a linear problem. Most problems will be more complex and hence require higher dimensionality of the partition function, a *hyperplane*. SVM belongs to a group of methods called *kernel methods*, and by using an appropriate kernel, SVM can map the data onto the desired space. Non-linear kernels will produce non-linear classifiers.



(a) Infinite linear boundaries

(b) SVM maximum margin boundary

Figure 2.6: Simplified exampled of infinite partitions of a completely separable dataset (a) and on an SVM trained on the two classes with a linear optimal hyperplane and maximum margins (b). Samples on margins are called support vectors. (a) was adapted from (b). Source of (b): Larhmam[3].

2.5.2 k-nearest neighbours

Instead of a hyperplane dividing the feature space, like in SVM, k-NN looks at the k neighbour vectors which have the lowest distance metric from the training set to a new sample. The predicted class of the new sample is the class with the highest probability based on those neighbours [26], illustrated in Figure 2.7.

There are many distance metrics possible. One of the most used is Euclidean distance $d(s, x_j)$, the geometric, straight line distance between two samples, the new data point s and an existing sample x_j, for all n attributes, see Equation (2.13),

$$d(s, x_j) = \sqrt{\sum_{i=1}^{n} (s_i - x_{ji})^2}, \tag{2.13}$$

[3]From: `https://en.wikipedia.org/wiki/File:SVM_margin.png` (retrieved on 2021-04-01), CC BY-SA 4.0 <`https://creativecommons.org/licenses/by-sa/4.0`>, via Wikimedia Commons.

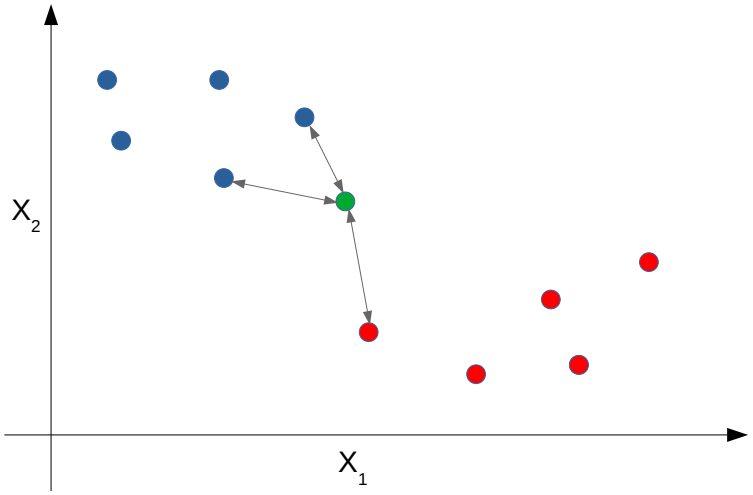


Figure 2.7: A simplified example of k-NN with "blue" and "red" classes of data points. For the new data point in "green", we can determine the most likely class by looking at the classes of its nearest neighbours. In this case, the decision would be “blue” because this is the majority of the k=3 neighbours.

2.5.3 Artificial neural network

Artificial neural network (ANN) is inspired by studies of how biological neural networks processes information [24]. A biological brain utilises a completely different information processing model than traditional digital computers do. The biological brain is a complex and non-linear "computer" which can organise neurons to perform certain tasks, whether this is sight or abstract thinking. The reason for the initial interest in ANN was the realisation that emulating this in digital computers could potentially outperform traditional computing both in performance and ability.

An ANN is targeted and optimised to solve problems in specific areas, e.g. in recognition of patterns or classification of information, and is trained in the same way as humans; by trial and error, assessing achieved results, retrieving stored information and using examples. The ability of ANN to learn enables it to generalise information which allows it to produce reasonable results from any data that were not used in the training phase [29].

The McCulloch-Pitts neuron was an early attempt to model the brain neuron using manually set weights to categorise input data. This was refined by Rosenblatt to a model which could learn weights. Widrow and Hoff further refined this to include regression. The problem with these models is that they rely on linear activation functions, which do not cope well with some complex problems. This caused a hiatus in research into biologically inspired ML models. However, with the introduction of techniques like back-propagation and non-linear activation functions, like ReLu and sigmoid, ANN continues to be relevant in research and applications [16].

A graphical representation of an artificial neuron can be seen in Figure 2.8 and is expressed in Equation (2.14), where $\mathbf{X}$ and $\mathbf{W}$ are the input and weight vectors respectively, which are summed and fed into the activation function f.

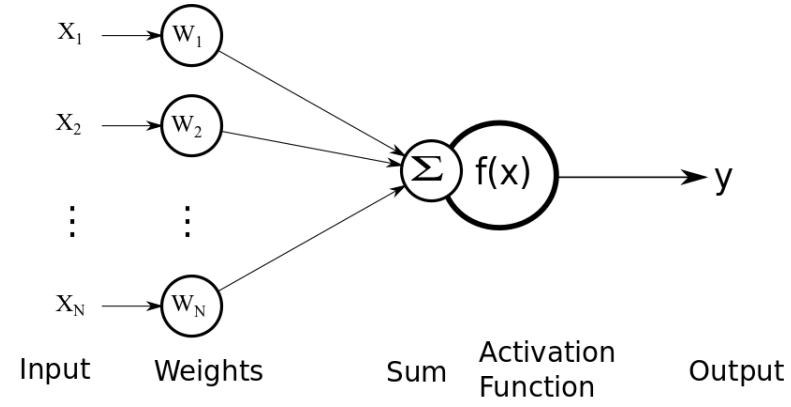


Figure 2.8: A graphical representation of an artificial neuron, also called a perceptron. Inputs and weights are combined into a sum which is acted upon by an activation function that produces an output. A bias is often included in such a representation, but can also be generalised as an ordinary input and wight where the input equals 1.

$$y = f(\sum_{i=1}^{N} X_i W_i), \tag{2.14}$$

The activation functions can be simple, like the one used in the original McCulloch-Pitts neuron; the binary step or Heaviside step function as expressed in Equation (2.15),

$$f(x) = \begin{cases} 0 \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases}, \tag{2.15}$$

or more complex like the softmax function in Equation (2.16),

$$f(\mathbf{X})_i = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \text{ for i=1, ..., J,} \tag{2.16}$$

which outputs a vector of normalised elements, which sums up to 1, based on an input vector.

However, a single artificial neuron is usually not enough to solve real-world problems. A key element of the ANN paradigm is a system with many artificial neurons which are interconnected in a network configuration and work in unison to solve a particular problem.

The variation in activation function, the size of the network, the network topology and selection of training algorithm are used to define the ANN model [30]. A simplified network can look like the one in Figure 2.9. It is a standard feed-forward configuration with two hidden layers. This might have been a good enough model for a simple problem, but contemporary models normally have either many hidden layers, often referred to as *deep* ANN, or many nodes per layer, *wide* ANN. Often, one of the inputs for all non-input nodes is reserved as a bias to have control of the offset of the hyperplane to use all of the available dynamics of the space it occupies.

2.5.4 Ensemble trees

Extremely randomised trees are ensemble learning methods for classification and regression which are as accurate or more accurate than some other decision tree

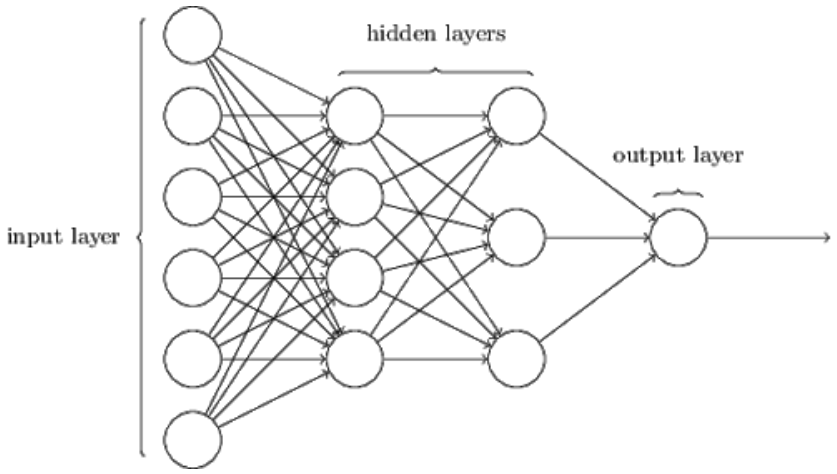


Figure 2.9: A model of an ANN where each circle represents an artificial neuron (or node) as described in Figure 2.8. The normal topology consists of one input layer (to the left), one output layer (to the right) and one or more hidden layers (in the middle). Each node in one layer is connected to all the nodes in the next layer in a feed-forward configuration. Image source: Michael Nielsen [31].

algorithms, particularly on classification problems. They work by combining decision trees, with explicit randomised cut-points and attribute choices at the decision nodes, and averaging the prediction of an ensemble of trees [32].

Gradient boosting is another ensemble method that combines the principle of boosting, converting weak learners into strong, with gradient descent to minimise loss when adding trees to the model. One implementation is XGBoost (extreme gradient boosting) by Chen and Guestrin [33]. It has gained a reputation as having generally very good predictive performance and is used in many ML competitions. It has also found its way into time-series applications [34].

C5.0, an early ML framework for training decisions trees, is also a classification model which employs an ensemble of trees with boosting and pruning [26]. It has multi-path splitting using the principle of gain. In information theory, gain is defined using the Shannon entropy[4]:

$$\mathrm{H(X)} = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i), \tag{2.17}$$

where X is a set of training examples $x_1 \cdots x_n$, $\mathrm{P}(x_i)$ is the probability of value x_i and n is the number of possible values. The information gain IG can then be calculated for X given the feature Y:

$$\mathrm{IG(X,Y)} = \mathrm{H(X)} - \mathrm{H(X|Y)}, \tag{2.18}$$

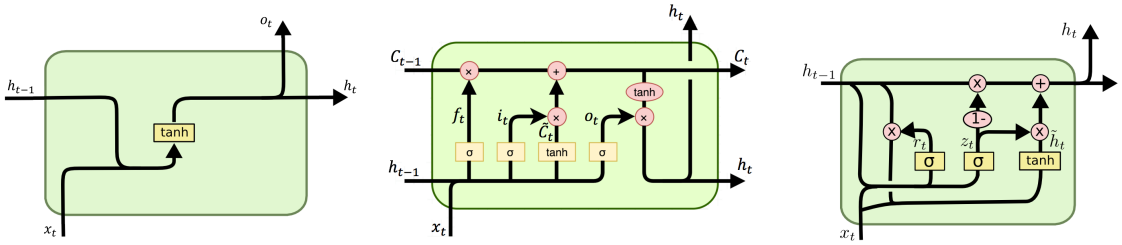
where

$$\mathrm{H(X|Y)} = -\sum_{x\epsilon X, y\epsilon Y} P(x|y) log \frac{P(x|y)}{P(x)}. \tag{2.19}$$

[4]Here we have used base 2 logarithms as this maps to units of bits, but other bases are also possible to use.

2.5.5 Recurrent neural networks

The main difference between traditional feed-forward ANNs and specialised time-aware algorithms like recurrent neural networks (RNNs) is that in the former input data is processed in fixed chunks all at once, while the latter processes input data one at a time and in a sequence where the output of one "step" (one time step) is the input of the next "step", see Figure 2.10 (a).



(a) vanilla RNN (b) LSTM (c) GRU

Figure 2.10: This figure shows the information processing and data flow in vanilla RNN, LSTM and GRU cells. The vanilla RNN (a) only has a single tanh activation function, LSTM (b) consists of three gates: the forget, input and output gates in addition to the cell state and GRU (c) consists only of the reset and update gates and the cell state. One or two, in the case of LSTM, of the outputs act as inputs to the next cell. Image source: dProgrammer lopez[5].

This recursion can be expanded as a standard feed-forward ANN and we can then apply traditional back-propagation, see Figure 2.11, to use existing ML algorithms for training. This technique is often referred to as *back-propagation through time (BPTT)*. Inside each cell, there are one or more *hidden states* which make an RNN specialised for processing a sequence of values distributed in time [16].

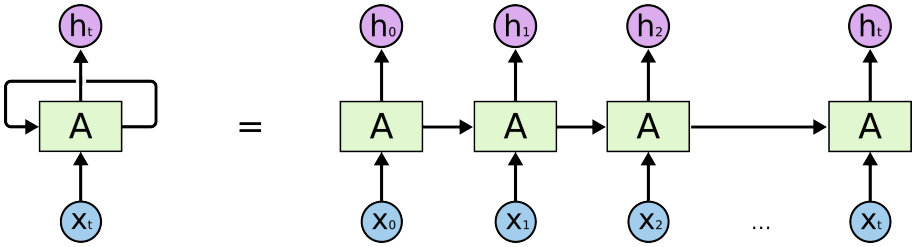


Figure 2.11: An illustration of how recursive BPTT in RNN (on the left of the equal sign) can be unrolled (on the right of the equal sign). This can then be processed in existing ANN algorithms. **A** represents a node as described in Figure 2.10, X_t is the input and h_t the output. The data back-propagated through time can be h_t, but can also be some other hidden state within the cell. Image source: Christopher Olah[6].

Because vanilla RNN suffers from vanishing gradient, causing earlier time steps to become quickly "forgotten", similar to the way ANN also suffer from this, variants like LSTM and GRU were developed to mitigate this issue. However, research by Mikolov et

[5]From: `http://dprogrammer.org/wp-content/uploads/2019/04/RNN-vs-LSTM-vs-GRU-1024x308.png` (retrieved on 2021-04-01).

[6]From: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/RNN-unrolled.png` (retrieved on 2021-04-01).

al. [35] have shown that it's possible to get near LSTM performance in some cases using a structurally constrained RNN.

Both GRU and LSTM use a mechanism called *gates*; additional tensor operations to modify how the hidden state is used within each cell to regulate how they "remember" and "forget" to control the time frame of the BPTT as well as for numerical stability to avoid the vanishing gradient problem [16]. GRU is generally less complicated than LSTM as it has fewer tensor operations and lacks an output gate, see Figure 2.10 (c) and (b), respectively.

Chung et al. [36] cannot conclude which of LSTM and GRU performs better on sequence modelling of polyphonic music and speech data, which is similar to the findings of Bahdanau et al. [37] on machine translations. However, Yang et al. [38] find that GRU can perform better for some problems with smaller datasets and long sequences, possibly due to the lower complexity of the algorithm and hence less overfitting. For all other scenarios, LSTM is superior to GRU.

Westhuizen and Lasenby have demonstrated that LSTM can also be simplified to only a single gate, the forget gate, and give better results than LSTM with three gates, on five different tasks [39]. We will therefore also consider both LSTM and GRU in our research.

2.5.6 Resampling

Resampling in ML are general statistical methods for estimating the model performance of a population without using distribution tables, e.g. normal distribution, either for estimating performance or for validating the model. For model validation, this is typically achieved by using a subset of the original dataset to train a model and the remainder of the dataset to evaluate the model. This process is repeated multiple times and the results are aggregated [26]. In this way, the entire dataset can be used in evaluating a particular algorithm.

The reason for this additional computation step for model validation is to mitigate the *variance-bias trade-off* where too simple models produce results with low variance and high bias, so-called *under-fitting* models, while too complex models produce results with low bias and high variance, so-called *over-fitting* models. The ideal model gives low bias, the ability for a model to capture the true relationship in the data, and low variance, the ability to generalise on the data, but in real-world problems, there is nearly always a trade-off between variance and bias.

K-fold cross-validation is a specific resampling technique where the dataset is randomly split into k subsets, called *folds*. The model is then trained on the complete dataset except for the first fold. This first fold is then used to estimate the performance of the model. This is repeated for the second fold and so on until the k estimates can be summarised. The higher the value of k, the lower the general bias of the model, since the training set increases, while the variance increases [26, 40].

A variant is called *stratified* k-fold cross-validation, usually used for classification, where random sampling is performed within the respective classes. This maintains the class distribution for the outcome of the model, thereby lowering the bias of the model further.

Both of these cross-validation techniques can be repeated with different random splits, thereby also reducing the variance further. Hastie et al. [41] assert that this repetition gives a good estimate of the true accuracy, hence can be used without a separate validation dataset. This is particularly important when the dataset to be used is small.

The misclassification error for cross-validation is calculated similarly to Equation (2.7) as:

$$err_{cv} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f_{-\kappa(i)}(x_i)), \tag{2.20}$$

where κ is the index function which indicates the fold where sample i is present and $f_{-\kappa(i)}(x_i)$ is the prediction function using data not in the set denoted by $\kappa(i)$.

Kohavi [40] recommends ten folds, i.e. $k = 10$, since it strikes a good balance between bias and variance, compared to more computational expensive configurations like twenty fold or leave-one-out cross-validation. There is no general recommendation on the number of repeats as far as we have been able to determine.

Another resampling technique is bootstrap aggregating, also called *bagging*. The bootstrap is a collection of samples taken with replacement from the original dataset of the same size. This means that a particular sample can be represented zero, one or more times in the bootstrap [26]. Usually, bagging is combined with an ML algorithm, e.g. a tree model, where the bootstrap is used for training and the samples not in the bootstrap, referred to as *out-of-bag* samples, are used for testing. This process is then repeated, or *aggregated*, to estimate the model's performance.

The misclassification error and SD of the bagging are then calculated as:

$$err_{boot} = \frac{1}{B}\sum_{i=1}^{B} \overline{err}_i \text{ and} \tag{2.21}$$

$$\sigma_{boot} = \sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\overline{err}_i - err_{boot})^2}, \tag{2.22}$$

where B is the number of replications and err_i is the error from the ith bootstrap repetition as calculated by Equation (2.7).

Another property of bagging is that it can be used to estimate the SD, and hence the confidence interval (CI) of an existing classification or regression effort, so-called *case resampling* [42]. An important aspect of this method is the number of replications, B in Equation (2.22). Davison and Hinkley [42] recommend a sufficiently large B without being specific while Efron and Tibshirani [43] suggest B between 50 and 200. However, the latter also suggests several methods for constructing the CI claiming that the standard normal intervals can be inaccurate when used with bootstraps and that these methods are more accurate for small datasets and a small number of replications, without being specific about what is "small". They suggest $B \geq 500$ for a good estimate of the CI; a higher number of replications for higher confidence levels. Manly [44] suggest looking at the variance over several repetitions of the bootstrap process and suggests at least 1000 bootstrap replications for a 95 % CI.

2.5.7 Hyper-parameter tuning

All the models discussed earlier have at least one parameter which can be tuned to control how the model fits the data. Poor choice of values for these hyper-parameters can potentially decrease the usefulness of a model. For instance, some hyper-parameters can

control the complexity of a model, but a too complex model can result in over-fitting and a too simple model can result in under-fitting.

Traditionally, grid search and random search, have been the most widely used strategies for hyper-parameter optimisation [45]. Over time, other strategies have emerged, like Bayesian optimisation [46] and evolutionary computation [47].

However, these approaches require that the model is trained and evaluated on each set of hyper-parameters permutations, causing the computation time to increase proportionally depending on the number of hyper-parameters to tune and how well the tuning process converges.

Jaderberg et. al have proposed a novel Lamarckian algorithm, population-based training (PBT), which reduces the tuning time by both training and tuning the model simultaneously [48]. It uses a population of "cooperating workers", each optimising one model with randomly initialised hyper-parameters. At regular intervals, the entire population is evaluated and all individuals can have their weights and hyper-parameters updated externally by two methods; *exploit* and *explore*. The former method replaces poorly performing models (bottom quantile) with better performing models (top quantile). The latter method randomly resamples or perturbs the hyper-parameters of exploited individuals. This way, good models can continue to learn while poor models are replaced with variants of good models.

This principle of exploiting and exploring is analogous to inheritance and mutation, respectively, found in evolutionary computation (EC). Since crossover is not part of PBT; weights and hyper-parameters are inherited, but only hyper-parameters are mutated. Learning can in principle be performed by any ML algorithm as the rate of mutation is decoupled from the rate of learning.

The effect is that PBT generates a schedule of hyper-parameters; multiple sets of hyper-parameters that change at intervals during training. This is in contrast to the norm; a single set of hyper-parameters is used for the entire training. See Table 5.5 for an example schedule found during our research.

Jaderberg et al. show that PBT gives faster convergence time and higher performance for several experiments compared to their baseline models. They have used large datasets for their work. Tennebø and Geitle have shown that the technique can also be applied to achieve comparable or state-of-the-art results on some smaller datasets [49].

The problem of optimising hyper-parameters instead becomes a problem of optimising the meta-hyper-parameters of the PBT method like the size of worker population, the size of top and bottom quantiles, the perturbation interval and the resample probability. However, the default values of these are reasonable for most problems, and a conservative approach, which we have chosen, is to only configure the size of the worker population and the perturbation interval.

2.5.8 Data augmentation

The best way to improve the generalisation performance of an ML model is to train it on more data [16]. However, as described in Section 4.1, there are some challenges with our data which includes uncertainties about the quality and quantity of the available data. *Data augmentation* can reduce the problem by increasing the diversity and amount of data by generating new, synthetic data based on existing data.

A simple way to illustrate data augmentation is to consider images of objects, human faces, animals, etc., where each image is transformed using geometrical operations such as

translation, cropping, rotation or shearing to create multiple, similar images to the original image. In addition, it is possible to adjust chroma components of images to get more or less colourful or darker or lighter images. The advantage is that we already have the correct class for the original image, and the number of combinations of geometrical operations on a single image is huge.

Cubuk et al. [50] have automated this principle with their *AutoAugment* algorithm. However, this algorithm uses a fixed augmentation policy and is computer-intensive which has prompted Ho et al. [51] to propose an algorithm called *population-based augmentation (PBA)*. Its principles are based on PBT described in Section 2.5.7 and is a general augmentation policy optimiser that adapts the augmentation operations used in training to generate a complete augmentation policy schedule, instead of a fixed policy, significantly reducing the computational costs while matching the performance of *AutoAugment*. These methods are based on linear augmentations of images.

A potential more powerful strategy is called *elastic deformation* introduced by Castro et al. [52], where a random stress field is used to deform the image in both vertical and horizontal directions and then smoothed by a Gaussian filter.

However, geometrical image transformations are not directly suitable for our data, which is strictly numerical sensor data, see Section 4.1. We can take some of the ideas above and apply them to our dataset, making sure that the augmented dataset retains the ballistic properties of the original. Some of the ideas we will consider are:

- Windowing, where a long trajectory is sectioned into multiple shorter trajectories of varying length.
- Recentering, where a trajectory is moved in a radial direction from its origin.
- Melding, where multiple similar trajectories are aligned in time and individual plots are sampled from those to create new trajectories.
- Noise, where Gaussian or uniform noise is added within the range for each feature.
- Bayesian, where e.g. a Monte Carlo simulation is used to create additional trajectories.

Synthetic minority over-sampling technique (SMOTE) is another data augmentation method and was created to mitigate the problem with model over-fitting from imbalanced datasets [53]. Arsland et al. [54] have reported good results using SMOTE to create additional synthetic data regardless of class imbalance. It works by selecting randomly interpolating features from "close" exiting minority examples to create new minority examples.

Another possibility we have is to use an in-house developed ballistic simulator model to generate enough trajectories. However, this model contains only a limited set of realistic projectile models and might introduce unwanted bias compared to real-world scenarios.

2.5.9 Regularisation

Another important concept when dealing with ML is regularisation; any modification to the learning process that reduces the generalisation error, but not the training error [16]. In other words, techniques to prevent over- and under-fitting. Neyshabur et al. [55] group regularisation into explicit and implicit regularisation. The former involves an explicit penalty to the complexity of the model to reward simpler models and include techniques

like weight decay[7] an additional term to the loss function, or dropout, where part of the structure of the model is removed randomly. Implicit regularisation includes convolutional layers in convolutional neural networks, batch normalisation and also data augmentation.

In their research, Hernández-García and König [56] show that data augmentation results in the same or better results using fewer computing resources compared to weight decay and dropout regularisation for visual object classification using ANN. They conclude further that these two explicit regularisation techniques can dramatically reduce the performance of the model unless the hyper-parameters are carefully optimised for the network architecture and the data set, which is not necessary for data augmentation techniques.

It is worth noting that some algorithms have built-in regularisation, e.g. XGBoost [33].

2.6 Machine learning in radar applications

Various techniques for classifying radar targets have been proposed for ships [57] and theatre ballistic missiles (TBMs)[8] [58], using RCS for both types. The former extracts an additional feature from RCS, angular-diversity RCS, using the maximum scatter difference (MSD) algorithm to improve the accuracy of classification compared to other methods. It is unclear whether this approach can be used for projectiles classification, but this could be a topic of future research. The latter attempts to discriminate between re-entry vehicles and spent tanks from a TBM. However, the RCS of re-entry vehicles and spent tanks are uncorrelated leading to a simpler classification task. However, none of them touches on ballistic projectile classification.

Kobashigawa et al. [59] have looked into detecting unexploded ordnance using ground-penetrating radar, which compares genetic programming (GP) and ANNs. They conclude that for radar imaging, GP outperforms ANN. Pisane [60] has worked on non-cooperating aeroplanes using a bi-static radar configuration, which uses extremely randomised decision trees (extra-trees). Using RCS or complex RCS, but in a bi-static setting, he concludes that extra-trees performance is promising and comparable to automatic recognition of radar images. Again, these methods have not been evaluated on ballistic projectile classification, but particularly extra-trees looks promising in a radar setting.

Rosa et al. [61] have used multiple ML algorithms to identify individual birds from radar images. They use multiple binary classifiers to separate birds from clutter and identify four different types of birds. Random Forest is the best performing classifier with consistent accuracy of >0.8, however other classifiers also performs well and they argue that ensembles of classifiers might give better results. Bird clutter is a problem for WLR since birds can steal capacity and create false targets. Hence, a direct application of this research can be to filter out birds as false targets.

Recent research into also using RCS in conjunction with a fuzzy classifier to classify ballistic projectiles has been performed by Lee et al. [62]. They used simulated RCS values and simulated trajectories for the evaluation, not real data. The fact that clutter and noise will influence the quality of the RCS estimation is only mentioned sparingly, but is simulated in the final result. They also only use one type of artillery gun and two types of mortars limiting the classification space significantly. They conclude that RCS can be used for the classification of projectiles for the Firefinder radar. Since the principles of all

[7] The most common forms of weight decay are L^1 and L^2 regularisation, also called LASSO and ridge regression [16].

[8] TBMs are ballistic missiles with a range of less than 3,500 km.

radars are mostly the same, there is reason to believe that the result should be applicable also for real data from other radars. And as the authors also conclude, additional features, such as drag coefficient, estimated firing velocity and estimated firing elevation, can be subject to fuzzy classifiers and combined to provide a more accurate classification. While the fuzzy logic as described in this research is not ML, we include it for completeness as the only public research we found to attempt to classify ballistic projectiles.

Our work involves data that is organised in time series; plots of tracks are recorded in sequential order. To utilise the data well, a different algorithmic approach is then often needed. Time-series are well known in forecasting [63] and machine learning methods have also been exploited in this area, such as autoregressive neural networks which is a hybrid method combining autoregressive integrated moving average model with neural networks [64].

Saab has also sponsored other similar work related to ML in radar applications. Andersson and Luong [65] have used plots, including features like dimensions and velocities in three dimensions, RCS and micro-Doppler, to distinguish between birds and UAVs. They used filtered and smoothed behavioural data with LSTM and GRU networks as well as a hybrid GRU/naive Bayes classifier. They compared their work with previous work also sponsored by Saab and found that their work yielded better results than time-series classifications using naive Bayes, logistic regression, SVM and ANN by Strandberg [66] and comparable to elastic and Kohonen self-organising maps by Rosén [67]. Similar to Rosa et al. [61], this research can also be used to separate birds from targets of interest.

2.7 Summary

Correct projectile classification from targets detected by weapon locating radars can provide substantial benefits, both in terms of cost, resources spent and lives saved, equally for wartime and peacekeeping scenarios. However, classification has proven to be challenging, mostly because of the number of different weapon systems and projectiles in use, real-world effects like rain, fog or wind, and the number of geometries involved. Research into this field is also sparse.

Based on the quality and quantity of our dataset, we might need to augment the size of it. To do this, a thorough understanding of the physical properties of the measured and any synthetic features which are involved in the augmentation is necessary. Furthermore, the time-series sensitive ML algorithms and hyper-parameter tuning algorithms will need further investigation.

Existing research in this specific field of artillery projectile classification is limited, but particularly the consistent use of RCS in related research is interesting and confirms Saab's historic use of this feature.

Chapter 3

Ethics

"No nation was ever so virtuous as each believes itself, and none was ever so wicked as each believes the other."
— Bertrand Russel, *Justice in War-Time*

This chapter will discuss some general research ethics and specific ethics related to military systems, through five questions followed by a summary. Where applicable, we will refer to autonomous weapons systems (AWS) as analogous to the application of ML in military radar systems.

3.1 Introduction

The Internet Encyclopedia of Philosophy defines that the field of ethics "...involves systematizing, defending, and recommending concepts of right and wrong behavior." and equals this with "moral philosophy" [68]. This interpretation will, in turn, depend on personal and collective beliefs, values and norms.

The Norwegian National Committee for Research Ethics in Science and Technology (NENT) defines "research ethics" as a broad set of standards, values, and institutional arrangements that contribute to and regulate research activities [69]. It lists 23 guidelines on how to perform ethical research grouped into these topics:

- The obligations of research to society.
- Integrity, truthfulness, and accountability.
- Uncertainty, risk, and the precautionary principle.
- Protection of research subjects.
- Protection of animals used in research.
- The relationship between research and other knowledge-bearers and forms of knowledge.
- Commissioned research, openness, and conflicts of interest.
- Whistle-blowing and ethical responsibility.
- Dissemination of research to the general public.

No participants were subject to this research, nor were any animals involved, hence these elements can be disregarded. Most of the others are more general and most concerned with *how* research is performed and not *what* is researched. This applies equally to research in general. However, the special consideration in this research is that it involves weapons,

albeit indirectly, which means that we should investigate the special ethical implication this raises, specifically the obligation to society. NENT lists three guidelines related to this topic:

1. Research has an independent responsibility for the role it plays in social developments.
2. Research should be compatible with sustainable development.
3. Research has a responsibility to contribute to greater global justice.

While the second guideline is relevant in the broader sense of the development of weapon systems, we consider it outside the scope of this research. The other two are relevant also for the use of military radar systems, and we will discuss these further.

The former of these deals with promoting peace and avoiding breaches in international conventions and that the research should benefit society. The latter deals mostly in how the research is performed and shared for 'greater global justice', but we will include international law related to military actions into this.

In addition, since the author is an employer of Saab and as this research is facilitated by the use of Saab proprietary data under the supervision of Saab, we have adhered to Saab's *Code of Conduct* [70].

3.2 Ethics in military applications

Since a radar cannot apply force by itself, we will in this discussion consider WLR in a bigger context which includes a C2 system and weapon systems, see Figure 3.1. The role of the radar in this context is to gather intelligence to the military and political leaders in a conflict, about enemy (and friendly) ballistic weapons, much in the same way as a forward observer, soldiers in the field, but is nevertheless classified as military materiel. In a combat scenario, the WLR is typically deployed behind the forward line of own troops and is continuously moved as the battlefronts move. In a camp protection scenario, the WLR is located inside one of the camps it is tasked to protect. Hence, it is very rarely involved in hostilities directly.

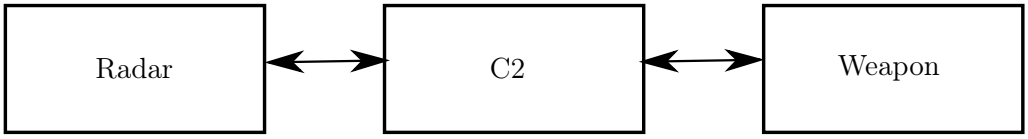


Figure 3.1: This is a very simplified view of a (military) radar, in our example, a WLR, in a setting with a C2 system and one or more weapon systems communicating with each other. They are bidirectionally connected to show that information can also travel from the weapon to the radar to command other missions than pure WL missions.

A WLR can estimate POO, POI and the classification of ballistic projectiles and report this information, either for warning in the case of POI or for the potential of return fire in the case of POO. It is typically able to send reports either manually or semi-automatically, that is with humans in-the-loop, or fully automatic, with no humans in-the-loop, within seconds of detection. With reference to Figure 3.1, there can in other words be different degrees of automation between the elements involved; the radar, the C2 system and the weapon systems. The automation can be further fine-tuned to only trigger if a projectile

is launched from within a designated zone[1], lands within a different designated zone, the estimated POO has an accuracy error below a certain threshold or a combination of these or other criteria. Hence, the WLR itself cannot directly engage an enemy, but provides intelligence that can be used by weapon systems.

There is little publicly available research on the ethical implications of augmentation of military radar systems using ML in general which we have been able to find. However, some research exists in the field of AWS and we will use this analogy even though our research does not touch directly on autonomy.

The laws of war (LOW) are encoded in a collection of international conventions describing what is and what is not acceptable behaviour before, during and after armed conflicts. In addition, national regulations or law is usually also included. *Jus ad bellum*, the right *to* war, prohibits the use of force, except for the right to individual or collective self-defence. This is regulated by the United Nations (UN) Charter Article 51 [71], and for Security Council enforcement measures, *ibid.* Article 42 [72].

Jus in bello, the right *in* war, governs the conduct of military operations. It is also referred to as International Humanitarian Law and is based on the Hague Conventions and the Geneva Conventions including their additional protocols [73]. Also, Article 36 of the 1977 Additional Protocol to the 1949 Geneva Conventions requires states to determine whether new weapons may be prohibited by international law [74], more specifically the rule against *inherently indiscriminate*[2] *weapons* and the rule against weapons that cause *unnecessary suffering or superfluous injury.*

Such weapons include biological weapons which are regulated by the United Nations Biological and Toxin Weapons Convention [75] which bans the development, production and stockpiling of biological weapons by the signatory states. The Geneva Protocol from 1925 [76] regulates the *use* of biological weapons, amongst other methods.

In contrast, there is no similar convention regarding the use of nuclear weapons. At the time of the nuclear bombing of Hiroshima and Nagasaki in Japan, President Truman believed that it was the only way to stop the war with Japan [77], but General Eisenhower claimed that Japan was already defeated before these two attacks [78]. In an interview published in the New York Times in 1946, Einstein claimed that he was sure that President Roosevelt would have forbidden the atomic bombing of Hiroshima had he been alive, and that it was probably carried out to end the Pacific war before the Soviet Union could participate [79]. Einstein later wrote that he always condemned the use of the atomic bomb against Japan [80]. Recently it has come to light that the US military leaders knew, through intercepted communications Between Japan and the Soviet Union, that Japan was negotiating a surrender before the bombing [81].

The use of ML in military systems is also not regulated in any internationally binding way. For instance, the European Commission's proposal for regulating ML, the Artificial Intelligence Act, explicitly exclude systems for military purposes from the proposal [82]. However, this area is being worked on, both by private companies and organisations. For instance, both Capgemini's *Code of Ethics for AI* [83] and IBM's *Everyday Ethics for Artificial Intelligence* [84] are examples of in-company guidelines for developing applications in the ML domain. Similarly, organisations as diverse as the Centre for Humanitarian Dialogue [85] and AI Alliance Russia [86] have developed their respective ethics codes for

[1] A zone in a military context is a geographical area defined by a polygon or circle.

[2] An indiscriminate weapon system is a weapons system that cannot be aimed at a specific target and which is as likely to hit civilians as combatants

the ML domain. We expect that also intergovernmental organisations like the UN will follow suit and provide some regulations in this domain in the future.

In a modern interpretation, *jus ad bellum* and *jus in bello* both operate during hostilities. The former involves the use of force and must be justified by referencing the principles of necessity and proportionality. The latter also involves the same principles, guaranteeing that victims on both sides of a conflict are equally worthy of protection, protecting combatants and civilians from unnecessary suffering, and also deals with how prisoners of war are treated. However, there is no clear distinction between what is considered *necessary* and *proportionally* and it is up to the state, the military command and finally the individual soldier to make the distinction, often during stress [73]. History has shown that adherence to these LOW sometimes breaks down in these situations.

In the following discussion we have formulated five questions to discuss the ethics of our work:

1. Is it ethical to use improved technology to gather intelligence?
2. Is it ethical to use this intelligence to issue warnings?
3. Is it ethical to use this intelligence to cause harm to other human beings or property?
4. Is it ethical to let technology perform lethal or non-lethal force on people autonomously?
5. Is it ethical to develop weapon systems that are capable of performing lethal or non-lethal force on people?

Since a WLR is primarily a defence system, we will not consider the ethics of intelligence in the setting of espionage, covert operations or theft of information as described by Omand and Phythian [87] and Beck [88]. Instead, we will in the following discussion consider the WLR's use of remote observations and their implications.

3.3 Ethical questions

Question 1: Is it ethical to use improved technology to gather intelligence?

Intelligence gathering is an important activity for states to maintain their sovereignty. There has been a huge technological leap from the spyglass to a modern, long-range air surveillance radar - very few questions the legitimacy for states to acquire more advanced radar technology. However, this happens from time to time, e.g. when Russia objected to the planned upgrade of the GLOBUS II radar located in Vardø, near the Norwegian border to Russia [89].

Does this mean that *all* technological advances in intelligence gathering are ethically sound? And is it clear cut what is ethical and what is not? Not necessarily. This is exemplified by the controversy when Edward Snowden leaked highly classified documents in 2013; some see him as a whistle-blower who uncovered a global surveillance program while others felt he is a traitor [87].

What makes the question more complicated is that intelligence gathering can occur during peacetime, during hostilities or during times of "neither peace nor war." Different international regulations apply for these different scenarios: During times of war, both *jus in bello* and *jus ad bellum* are applicable, while only the latter applies to the events leading up to hostilities. However, during peacetime other types of international and national

laws and regulations are in effect, e.g. the UN Charter in its entirety and numerous other treaties and other international agreements [90]. One of the main principles of international law is the promotion of peace [91]. This is also reflected in the NENT ethics guidelines. It follows from this that aggression should be avoided. But what is aggression on this level?

An analogous example can be that of two neighbours where one is peeking "over the fence" on the other's property from afar, perhaps with a pair of binoculars. Can this be considered illegal aggression? Perhaps it is in some jurisdictions, but what if we add that the first neighbour lives in a different country than the second neighbour and does this peeking across the border? Or what if the first neighbour suspects the second neighbour is doing something criminal? Would it be legal without the binoculars? These are questions that are not easy to answer from a strictly ethical perspective. What some will consider aggression, others will dismiss.

An important principle of the LOW is to prevent unnecessary suffering and avoid collateral damage. To distinguish lawful targets from civilian targets, established means of identifying military objects and personnel include distinctive marking, i.e. wearing a uniform and painting vehicles in specific colours. Another important discriminator is the collection of intelligence about the location of combatants to avoid harming civilians and damaging civilian property [92]. Hence, it can be argued that some form of WL ability is required for all states and that it should be optimised for accuracy.

In the case of a radar, the intelligence gathering is non-intrusive; no "curtains" are present and the information is freely available to anyone with the capability to "see" it. Therefore, we consider this to be ethical according to both the perspective of social developments and global justice.

Additionally, it can also act as a deterrent if other nations know the intelligence-gathering capabilities of a particular nation. Hence, we believe this also counts as preventing unnecessary suffering as per *jus in bello*.

Question 2: Is it ethical to use this intelligence to issue warnings?

Ballistic weapons have a high degree of destructive force around the POI. It is therefore critical to warn both civilians and own troops to evacuate this area as quickly as possible before an impending attack. To do that it is necessary to know to a certain degree of confidence which area is affected to route the warning to the correct location. With current WLR technology, this is already performed today, hence this is already established practice. But depending on accuracy, false warnings may reduce the trust in these systems [93]. With improved projectile classification, the accuracy and confidence should increase, hence providing earlier and more accurate warnings as well as reducing the number of false warnings.

As discussed above, this might even be required behaviour. Again, we believe this falls within *jus in bello* and as such is ethical according to both the perspective of social developments and global justice.

Question 3: Is it ethical to use this intelligence to cause harm to other human beings or to property?

As discussed, *jus ad bellum* opens for self-defence from armed attack. To do so effectively, states use systems of varying degrees of technological complexity to gather intelligence, e.g WLR to locate POO and POI of hostile RAM weapons. When the POO is known with a

high enough degree of confidence according to the doctrine of the defending country, it may be entitled to use force back. With a higher degree of confidence from better classification, unnecessary force is possible to avoid since the amount of munition used for counter fire is inversely proportional to this confidence. This relation is described as circular error probability (CEP), the smallest radius of a circle containing the true target coordinates to a certain probability, e.g. 50 % for CEP50 [94]. Higher confidence gives a smaller target area.

This has two immediate effects in a conflict. Firstly, the area being targeted becomes as small as possible, limiting the force used and thereby reducing the destruction wrought and hence, the number of casualties. Secondly, given that the responses are carried out early enough, before the aggressor can move their weapon systems, they can act as a deterrent - reducing the total amount of attacks. Therefore, early and correct projectile classification is essential to prevent unnecessary suffering and collateral damage.

But what if it is a case of first aggression? As already discussed, this is illegal according to *jus ad bellum*, independently of the type of technology used. But even the use of certain weapons in self-defence can be illegal. Such weapons include biological weapons which are regulated by the United Nations Biological and Toxin Weapons Convention [75], which bans the development, production and stockpiling of biological weapons by the signatory states. The Geneva Protocol from 1925 [76] regulates the *use* of biological weapons, amongst other methods.

We believe that intelligence can be used both ethically and unethically in a combat situation. As long as any research is open and available the opportunity that it might be misused is always present. As such, we do not condone the use of force, but accept that there might be circumstances where both defence and attack as defined by the LOW might be necessary. We believe that our research may help to balance the *necessity* and *proportionality* of an attack, and as such is ethical according to both the perspective of social developments and global justice.

Question 4: Is it ethical to let technology perform lethal or non-lethal force on people autonomously?

As to the question about autonomy, the latest technology is often introduced in military applications early and aggressively, also so with ML. One such example is the FOCUS programme out of the Defense Advanced Research Projects Agency (DARPA) where autonomous drones can identify and potentially apply lethal force to human targets [95].

Bode and Huelss [96] discuss AWSs and their implications on norms in international relations. They argue that the development of AWS outpaces public, governmental and legal considerations. These *fundamental norms* which are derived from international law, human rights, and other formal sources, may become second to more *procedural norms*. These latter types of norms are defined from what is more functional or effective in practice in situations of warfare, and they consider that this in turn is leading the development of fundamental norms. Instead of focusing on how AWS can be regulated, they suggest that the perspective should be reversed: How AWS may create the regulation. They claim that per definition, artificial intelligence does not perform (human) ethical judgement in unstructured and complex environments and that actions of AWS are unpredictable. They mirror the US drone program where the use of drones, which has set new, "appropriate" standards for ending human life without there being regulation from international law, have created a public accepted norm which is now difficult to return to *status quo* from.

However, they neglect to discuss that also human actions are highly unpredictable, even in a command structure found in the armed forces. We can consider the problem from another angle. Arkin [97] has suggested that soldiers, through their conditioning and training, can be thought of as robots (or AWS) and that robots could be more humane than humans. He argues that despite their training, when soldiers are faced with the horror of the battlefield, it becomes unrealistic that they would abide by the LOW. He admits that a fully everyday or professional ethics "machine" is unrealistic, but puts forward that a battlefield ethics subset would be more clear-cut and precise and hence feasible to implement. He goes on to describe a prototype implementation of such a battlefield ethics architecture.

In a later work [98], Arkin goes further and argues that AWS could on the battlefield potentially perform more ethically than human soldiers and lists six ethical considerations on why AWS may, at some point, be better able to effectively discriminate whether or not a target is legitimate:

1. The ability to act conservatively, not having to factor in self-preservation.
2. The eventual development and use of a broad range of robotic sensors better equipped for battlefield observations than humans currently possess.
3. Unmanned robotic systems can be designed without emotions that cloud their judgement or result in anger and frustration with ongoing battlefield events.
4. Avoidance of the human psychological problem of "scenario fulfilment," i.e. where humans use new incoming information in ways that only fit their pre-existing belief patterns.
5. They can integrate more information from more sources far faster, before responding with lethal force than a human possibly could in real-time.
6. Their potential capability to monitor and record ethical behaviour alone might lead to a reduction in human ethical infractions when working in teams with humans.

It is quite clear that this is an ambitious goal. Arkin acknowledges that this is currently not possible and we believe there are many technical challenges needed to be solved before this can become reality: The current state-of-the-art ML needs significant updates; the ethical problem of selecting features and training AWS in realistic environments; the problem of measuring data related to human behaviour, e.g. how can human deception be measured? There is also the economic aspect of balancing AWSs conservatism vs. their rate of loss.

One moderator in human behaviour is the possibility of being prosecuted for a crime. This is also relevant for crimes during wartime. Both individual soldiers and officers have been prosecuted and convicted for war crimes in the past. But what about AWS?

Today, and for the foreseeable future, the elements which define AWS; the degrees of autonomy, agency, human control and predictability of result lie on a spectrum. This leads to problems on how to define AWS - what makes an AWS too autonomous? This lack of delineation makes it problematic for a general ban on AWS, but also for negotiating a legal, international framework. Bode and Huelss [96] also note that the idea of ceding control to AWS in military organisations, which are structured around control and accountability, is highly contested. There is presently, to the best of our knowledge, no international regulation on the use of ML in warfare.

Our research attempts to provide increased accuracy of the classification result and does not touch on autonomy, agency nor human control. The issue of human oversight of the final result is always challenging with ML, so also for our research.

Question 5: Is it ethical to develop weapon systems that are capable of performing lethal or non-lethal force on people?

The ethical distinction between the development and the use of weapons, autonomous or not, is pronounced. This is exemplified by Einstein, who was against the use of nuclear weapons, but also signed the letter warning President Roosevelt about the German nuclear program in August 1939, which reputedly prompted the US nuclear weapons programme [99]. More relevantly, autonomous weapons is a controversial topic also at Google, where thousands of employees reportedly signed a letter to pull out of research in ML used for drones employed by the US military [100] and several have since resigned [101]. There are now several worldwide movements that are working for a full ban on autonomous weapons, like the coalition of international and national organisations, *Campaign to Stop Killer Robots*[3], and national petitions like the Norwegian *Lytt til oss*[4].

However, in their essay *Law and Ethics for Autonomous Weapon Systems: Why a Ban Won't Work and How the Laws of War Can*, Anderson and Waxman [102] compare the contemporary controversy of AWS with the, at the time, controversy of the introduction of the submarine and military aviation. They argue that the feature of autonomy, rather than a human being making the final targeting decision, would not in itself render the system indiscriminate, and as such would not automatically be banned by the LOW, as long as lawful targeting information can be ensured and that the autonomous selection of the target does not violate the rule to not cause unnecessary suffering or superfluous injury. They also discuss that over time, for some purposes and forms of attack or defence, autonomous weapons may be able to be more discriminating and precise than human beings, which echo Arkin's argument from above.

Conversely, prominent people in academia and industry like Elon Musk and the late Stephen Hawking have involved themselves in the *Future of Life Institute* which works for research on how to make AI systems robust and beneficial[5]. Specifically for AWS, they raise several areas of research like *Can lethal autonomous weapons be made to comply with humanitarian law?* and *Are autonomous weapons likely to reduce political aversion to conflict, or perhaps result in accidental battles or wars?* [103].

3.4 Summary

This discussion has only scratched the surface on the question about the ethics and morale of ML in weapon systems. Even though we believe this research in total is ethically sound, anyone implementing the results of this research and any nation using weapons that are based on this research will have to make an independent, case-by-case judgement whether they are themselves within the applicable international treaties and laws.

[3] `https://www.stopkillerrobots.org/`
[4] `https://lytt-til-oss.no/`
[5] `https://futureoflife.org/ai-open-letter/`

Chapter 4

Methods

"To be sure of hitting the target, shoot first, and, whatever you hit, call it the target."

— Ashleigh Brilliant, *Pot-Shots #0572*

This chapter will describe the proprietary dataset provided by Saab and the work involved in selecting the final subset of data used for our experiments. We will also discuss the process of evaluating our dataset for the different experiments we have conducted to answer our research questions from Section 1.2.

4.1 Dataset

The data used for this research was provided by Saab Technologies Norway AS, the Norwegian subsidiary of Saab AB of Sweden. It consists of more than 10 GB of raw data, approximately 12 000 track files with over 200 000 individual tracks from various locations around the world. One sample in this context is one track consisting of several radar settings, tactical information, environmental information like weather data and varying numbers of associated *plots*, sampled several times every second. Each plot, in turn, consists of radar measurements, parameters used to control the measurements and processed measurements; semi-synthetic features, 26 features in all. These include positional information in three dimensions; range, azimuth and elevation angles, timing information, signal strength and phase information. In addition, synthetic features like SNR are included. For the overall track, synthetic features like estimated firing elevation and firing velocity, coordinates for POO and POI and original projectile classification are also recorded, in total 41 synthetic features including some redundant information.

The track files have been collected using different radars over many years and in various stages of development, hence with different characteristics. This data needed to be systematised and categorised to associate correct truth data with their respective track. This work was complicated due to many factors.

Firstly, some track files contain recordings of projectiles with truth data mixed with unwanted activities like birds, helicopters and weather clutter. These unwanted activities can sometimes generate false targets which need to be separated from the true targets.

Secondly, sometimes tracks without truth data are also included in the recordings. They need to be identified and excluded.

Thirdly, it is also possible that several projectiles might be in the air at the same time and that the radar does not consistently separate the radar samples from each other or correlate the radar sample to the same physical projectile. This makes it difficult to reliably analyse them. It is, therefore, necessary to screen track files for irregularities even though they appear to classify correctly.

Fourthly, many track files exist in multiple copies, often with different names and even slightly modified for other purposes. These tracks needed to be filtered out as duplicates.

Fifthly, live firing measurements can be inaccurate, either because of internal factors like the internal performance or of the radar itself or external factors like the weather; a combination of systematic errors and stochastic noise. This can give ambiguities in how the data should be interpreted and hence categorised.

Finally, the format used for recording these track files has over time been extended to add features to aid the development of ARTHUR. This made the automatic parsing of the files complicated. We, therefore, concentrated on files from the third generation ARTHUR since the format is more stable within a single generation and because the recording of truth data for this generation has been more consistent than for previous generations.

After this vetting of the raw data, we ended up with only 2 259 unique tracks, just over 1 % of the total. This is significantly fewer tracks than expected and likely not enough for the time-aware ML algorithms discussed in Section 2.5.5. The class distribution of these tracks are given in Table 4.1 and as we can see, the dataset is highly imbalanced. Hence, we decided to first investigate our data with less data-intensive algorithms like C5.0 and XGBoost. As reported by Masini et al. [104], tree models have previously been used for time series prediction to good effect.

We will use regular accuracy scores along with SD to present our results..

Table 4.1: Total number of individual track files where we have truth data, with the distribution into their respective projectile types classes. Percentages are also given.

	Mortar		Gun		Rocket		Sum
	Light	Heavy	Light	Heavy	Light	Heavy	
No. of samples	192	836	37	892	286	16	2259
Percentage of total	8.5 %	37.0 %	1.6 %	39.5 %	12.7 %	0.7 %	100 %

As a benchmark, we performed a case resampling, as described in Section 2.5.6, of the dataset for the generation 1 and 2 algorithms. Several runs with increasing numbers of replications, up to 2000, were performed, to monitor the stability of the result. The results can be seen in Table 4.2. As we can see, the results are stable even at relative few replications. This indicates a good estimate of the true accuracy of these two algorithms. The respective confusion matrices for 2 000 replications can be seen in Table 4.3 and Table 4.4.

Table 4.2: Accuracy of the existing benchmarks for Saab's generation 1 and 2 algorithms using case resampling for increasing number of replications with SD (in parenthesis).

Replications	10	20	50	100	200	500	1000	2000
Generation 1	0.904 (0.007)	0.902 (0.006)	0.902 (0.006)	0.903 (0.006)	0.903 (0.006)	0.902 (0.006)	0.902 (0.006)	0.902 (0.006)
Generation 2	0.856 (0.008)	0.850 (0.008)	0.853 (0.007)	0.852 (0.007)	0.851 (0.007)	0.852 (0.008)	0.852 (0.007)	0.852 (0.007)

Table 4.3: Confusion matrix for Saab's generation 1 classifier with accuracy and SD (only for correct predictions, in parenthesis) using case resampling with 2 000 replications. The predicted class is given horizontally and the true class vertically.

Predicted class	M	G	R
True class			
M	0.984 (0.012)	0.016	0
G	0.004	0.825 (0.004)	0.171
R	0.030	0.109	0.861 (0.020)

Predicted

Table 4.4: Confusion matrix for Saab's generation 2 classifier with accuracy and SD (only for correct predictions, in parenthesis) using case resampling with 2 000 replications. The Predicted class is given horizontally and the true class vertically. Note that this classifier can produce the class 'None' when it fails to discern the class. All in all, 57 projectiles are classified as "None" in this dataset.

Predicted class	ML	MH	GL	GH	RL	RH	None
True class							
ML	0.566 (0.035)	0.366	0	0	0.005	0	0.063
MH	0.147	0.822 (0.013)	0	0.002	0.001	0	0.027
GL	0	0	0.837 (0.062)	0.163	0	0	0
GH	0.002	0.003	0.007	0.961 (0.007)	0.012	0	0.013
RL	0	0.003	0.108	0.014	0.843 (0.021)	0	0.003
RH	0	0	0	0	0.939	0	0.061

4.2 Experiments

4.2.1 Synthetic track features

We collected the overall synthetic track features, one line for each track. From over 40 synthetic features, we selected a sub-set of only four for this experiment, to make sure the model is independent of positional data. These features are firing elevation, velocity and RCS, also used in the original algorithms, as well as acceleration. We have no missing values in our dataset.

We have selected five different algorithms to evaluate our dataset on, in addition to our two original algorithms: SVM, k-NN, C5.0, ANN and XGBoost. For all of these, except C5.0 and XGBoost, a grid search was used to find the best selection of hyper-parameters. For C5.0 we only invoked boosting with one hundred trees and for XGBoost we used the

default hyper-parameter settings, which also include one hundred trees amongst others. For ANN we opted to use 25 epochs to roughly match the computational complexity of the other ML methods. This could still probably be too many epochs, but this model still did not converge. We have therefore included the results for a converging ANN model. Each fold then required just under 100 epochs on average to converge.

Due to the imbalanced dataset, we used stratified K-fold cross-validation, a method of partitioning the dataset uniquely across all folds while keeping the overall distribution between the classes, for the classes to be proportionally represented in both training and testing in all folds. This avoids any class being over- or under-represented in any fold, e.g. excluded completely.

As mentioned in Section 2.5.6, Kohavi recommends ten folds [40], but we also need to consider the dataset. With ten folds, GL and RH would be represented with only 3.7 and 1.6 samples on average in the test set, respectively. Hence, for all classifiers, we used five repeats, each with five folds.

These results, along with the original first- and second-generation classification algorithm accuracies, are presented in Table 5.2.

4.2.2 Complete track plots

From the results in Table 5.2, we see that two of the extended projectile types, GL and RH, have lower accuracy and much higher SD than the rest. Comparing this with the number of samples for each projectile type in Table 4.1, we see that these two are also the projectile types with the fewest samples. However, as the sample size is relatively low, the likelihood of SD being accurate is also relatively low [105]. To reduce the SD and potentially increase the accuracy, more data is necessary.

To complement the dataset, data augmentation was considered as discussed in Section 2.5.8. One possibility would be to over-sample GL and RH, but as they are from two and one series of rounds respectively, this would only add very similar synthetic data compared to what we already have. Undersampling the other classes is not an option as that would likely result in worse results.

We, therefore, decided to instead use the already available additional data in the form of individual plots from each track. We use a stratified *group* K-fold cross-validation to split the dataset. By grouping the plots to their respective tracks and stratifying that, we avoid splitting plots from one track across multiple folds, thereby avoiding having plots from the same track being part of both training and testing in the same fold. This effectively means that we are still treating each track as one sample. A total of 189,125 individual plots are available this way. The class distribution of these is given in Table 4.5. As we can see, there are relatively more GL and RH plots here than for individual tracks in Table 4.1, particularly for RH. However, the real advantage is the almost two magnitudes increase in the number of data points for all classes.

Table 4.5: Total number of plots in all track files where we have truth data, with the distribution into their respective projectile types classes. Percentages are also shown.

	Mortar		Gun		Rocket		Sum
	Light	Heavy	Light	Heavy	Light	Heavy	
No. of samples	12009	81408	3261	64254	25622	2571	189125
Percentage of total	6.3 %	43.0 %	1.7 %	34.0 %	13.5 %	1.4 %	100 %

From the 26 features available for plots, we selected a sub-set of seven which again also are independent of positional information, leaving out features such as range and bearing to the target. These include the elevation and velocity of the plot, the relative strength of the measurement and the SNR. Again we have no missing values in our dataset.

C5.0 does not have the functionality to group data when stratifying and since XGBoost gave the best initial results, see results in Table 5.2, we will focus this experiment and the rest of our research solely on XGBoost. These results are given in Table 5.3.

4.2.3 Partial track plots

As we discussed in Section 3.3, Q2, it is advantageous to get a prediction of POO and POI as early as possible. Normally, ARTHUR will on regular intervals calculate the POO and POI. Because of the connection between the projectile type and the ballistic model, as discussed in Section 2.2, it is therefore important to understand when the accuracy of the classification is good enough to also get accurate estimates of POO and POI.

Hence, we split all the tracks into groups with increasing lengths in steps of five plots, on average one second track time, to match the calculation intervals of the POO and POI, and otherwise performed the same experiment as described in Section 4.2.2 for each group. All in all 20 models were trained and evaluated: The first model uses one second of plot data for all tracks. Since the minimum length of tracks is between one and two seconds, this includes all 2 259 tracks. The second model uses two seconds of plot data for all tracks of at least this length, 2 252 tracks in total. And so on.

Of the 2 259 available tracks, the average length of one track is 84 plots and the median 85 plots. The minimum and the maximum number of plots is 6 and 328, more than 65 seconds, respectively. However, using more than 20 seconds to determine the projectile type would normally be considered too late. We, therefore, limit the maximum length of a track to 20 seconds in this experiment.

Tracks that are shorter than the required number of seconds are discarded for that model. This will gradually reduce the number of included tracks. The results can be seen in Table 5.4.

4.2.4 Hyper-parameter optimisation

Based on the results from Table 5.3, we performed a more thorough hyper-parameter optimisation of both problems in Section 4.2.1 and Section 4.2.2; synthetic track features and complete track plots. We selected PBT as the optimisation method since we have already had experience with it for smaller datasets [49].

As we discussed in Section 2.5.7, PBT can itself be configured with several meta-hyper-parameters. For our work, we opted to use 20 workers for a maximum of 200 iterations each with a perturbation interval of 10 iterations. For the remaining parameters, we used the default values provided for PBT as described in [48].

We then ran this configuration with both synthetic track features and complete track plots. Each worker used 75 % of the dataset for training the PBT schedule and the rest for testing it. PBT can in principle tune any number of hyper-parameters, but the complexity and run-time of the tuning process can increase as the hyper-parameter space increases. However, this is independent of the complexity of the learning process of the XGBoost model itself, which is highly dependent on the selected hyper-parameters of the model, e.g. *max_ depth* which controls the maximum depth of a tree in an XGBoost model.

Informal tests on our dataset both with synthetic track features and full track plots suggested that tuning fewer hyper-parameters from the XGBoost framework, results in better-performing models, meaning that the default values for the hyper-parameters are quite good for XGBoost. Hence, for this dataset we allowed PBT to only tune three hyper-parameters: *max_ depth* between 1 and 9, *min_ child_ wight* between 1 and 3 and *eta* (analogous to learning rate in some other ML algorithms) between 1e-5 and 1e-1.

After the tuning process was completed, the resulting hyper-parameter schedule was validated with a stratified and group stratified K-fold cross-validation with five folds and five repeats - identical to what is described in Section 4.2.1 and Section 4.2.2.

Chapter 5

Results

"When you can measure what you are speaking about, and express it in numbers, you know something about it."
— Lord Kelvin, 1883

In this chapter, we will review the results and compare the performance of Saab's existing non-probabilistic methods for projectile classification with our probabilistic models using ML.

Not only do we show that all of our ML models outperform Saab's proprietary generation 1 and 2 algorithms, but also can do so after the first second of track time using plot features. The best result was achieved using a PBT hyper-parameters optimised XGBoost model with plot data of complete tracks. The resulting accuracy of 0.984 ± 0 for classification into six projectile classes, is significantly better within the 95 % confidence level, than 0.902 ± 0 for the generation 1 algorithm, for only three classes, and 0.852 ± 0 for the generation 2 algorithm, for six classes, since the 95 % CIs do not overlap. A summary of the best results can be seen in Table 5.1.

Table 5.1: Summary of the best results achieved: Saab's first- and second-generation, XGBoost using synthetic features and hyper-parameter optimised XGBoost using plot features. Saab's generation 1 algorithm only classifies into the three classes mortar, gun and rocket, while all the other algorithms also classify into the calibres light and heavy; six classes in total.

Algorithm	Accuracy
Saab's generation 1	0.902 ± 0
Saab's generation 2	0.852 ± 0
XGBoost synthetic features	0.979 ± 0.003
XGBoost plot features, optimised	0.984 ± 0

5.1 Synthetic track features

To run our first experiment, we collected the overall synthetic features, four in total, one line for each track. We then ran this through our selected ML methods[1].

As already discussed in Section 4.1, the dataset is highly imbalanced and it is, therefore, relevant to investigate the relative accuracy distributed on the projectile types. These results, along with the original classification accuracies, both for the first- and second-generation algorithms, are presented in Table 5.2.

Table 5.2: The accuracy and SD (in parenthesis, where available) for C5.0, five classifiers with five folds, and SVM, k-NN, ANN with 25 epochs and converged ANN, and XGBoost, all with five repeats with five folds, using compared to the results of the original first and second generation algorithms, for track data using four synthetic features as discussed in Section 4.2.1. The results are broken down into the three main classes mortar, gun and rocket and the sub-classes light and heavy. 95 % CI is included for the overall results and the generation 2 algorithm. Note that the generation 1 algorithm does not classify into sub-classes light and heavy, only in the main classes gun, mortar and rocket, which partly explains the higher accuracy compared to generation 2.

Class	Gen. 1	Gen. 2	SVM	k-NN
ML	0.984 (0.012)	0.566 (0.035) ±0.002	0.844 (0.058)	0.900 (0.038)
MH		0.822 (0.013) ±0.001	0.966 (0.014)	0.970 (0.016)
GL	0.825 (0.004)	0.837 (0.062) ±0.003	0.933 (0.130)	0.880 (0.130)
GH		0.961 (0.007) ±0.000	0.980 (0.012)	0.973 (0.013)
RL	0.861 (0.020)	0.843 (0.021) ±0.001	0.948 (0.033)	0.948 (0.035)
RH		0	0.937 (0.130)	0.937 (0.130)
Overall	0.902 (0.006) ±0.000	0.852 (0.007) ±0.000	0.958 (0.009) ±0.004	0.961 (0.008) ±0.003

Class	ANN (25 epochs)	ANN (conv.)	XGBoost	C5.0
ML	0.503 (0.196)	0.768 (0.089)	0.969 (0.031)	0.974
MH	0.968 (0.016)	0.959 (0.022)	0.984 (0.011)	0.981
GL	0.028 (0.057)	0.398 (0.244)	0.940 (0.081)	0.973
GH	0.944 (0.035)	0.957 (0.021)	0.986 (0.010)	0.985
RL	0.743 (0.095)	0.799 (0.073)	0.960 (0.035)	0.965
RH	0.353 (0.441)	0.820 (0.234)	0.823 (0.177)	0.813
Overall	0.871 (0.024) ±0.009	0.911 (0.017) ±0.007	0.979 (0.007) ±0.003	0.979 (0.008) ±0.003

The overall results for the two proprietary Saab algorithms, see Section 4.1, are accuracies of 0.902 ± 0 and 0.852 ± 0 within the 95 % confidence level for the generation 1 and 2 algorithms respectively. Note that the generation 1 algorithm does not classify into the sub-classes light and heavy which is the main reason for its better performance. This is evident when inspecting the confusion matrix for the original generation 2 algorithm, see Table 4.4.

We also see that the total results for C5.0 and XGBoost are practically equivalent. However, the SD for C5.0 is slightly higher than for XGBoost, which translates to a slightly more uncertain result. These two methods are significantly better than the

[1]We used Rulequest's implementation of C5.0 (`https://www.rulequest.com/see5-info.html`) and scikit-learn (`https://scikit-learn.org/`) for the rest of the methods.

original algorithms. The other methods, SVM, k-NN and the converged ANN, all have significant lower accuracies than C5.0 and XGBoost, but also these are significantly better than the original algorithms. Additionally, the non-converged ANN is significantly better than the generation 2 algorithm.

The accuracy for both C5.0 and XGBoost is 0.979 ± 0.003, between 0.976 and 0.982, within a 95 % confidence level. All the results and their confidence intervals can be seen in Figure 5.1. Note that the CI for generation 1 and 2 algorithms are calculated from a bootstrap of 2 000 replication, while for the other methods, it is calculated based on five repeats of five folds each, for a total of 25 folds.

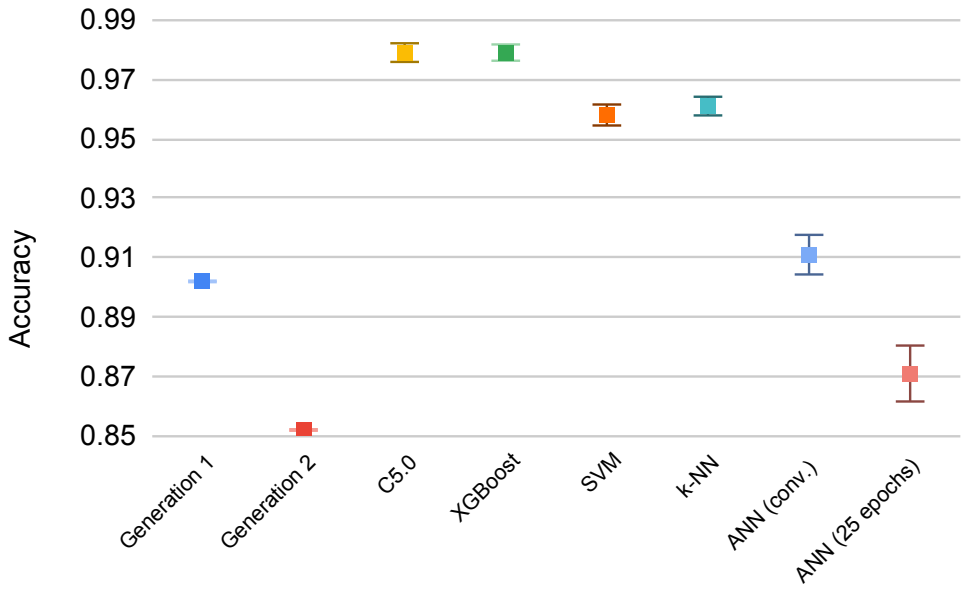


Figure 5.1: Interval plot with the 95 % CI of our results; generation 1 and 2 using case resampling with 2 000 replications, and the ML methods C5.0, XGBoost, SVM, k-NN, converged ANN and ANN with 25 epochs. All of the ML methods are significantly better, with a 95 % confidence level, than both the original Saab algorithms. C5.0 and XGBoost are significantly better, with a 95 % confidence level, than the rest.

5.2 Complete track plots

For our second experiment, we collected the plot features, measured and measurement control features; seven in total, one line for each plot. We then ran this only through XGBoost.

From the results in Table 5.3, the overall accuracy here is significantly lower within the 95 % confidence level compared to using synthetic track features. The individual class accuracies are practically the same within the 95 % confidence level, except for RL, which is significantly lower. More importantly, the SD for all classes and overall is lower using plot features than using synthetic features. This indicates that we may have traded higher bias for lower variance using plot features, hence it should be possible to decrease the bias, thereby improving the results further, by hyper-parameter optimisation.

Table 5.3: The accuracy and SD (in parenthesis) from XGBoost, five repeats with five folds, using plot features as discussed in Section 4.2.2 and the accuracy using synthetic track features, identical to Table 5.2 for convenience of comparison. 95 % CI is also included for both.

Class	XGBoost - tracks		XGBoost - plots	
	Accuracy (SD)	95 % CI	Accuracy (SD)	95 % CI
ML	0.969 (0.031)	±0.012	0.968 (0.011)	±0.004
MH	0.984 (0.011)	±0.004	0.986 (0.004)	±0.002
GL	0.940 (0.081)	±0.032	0.931 (0.034)	±0.013
GH	0.986 (0.010)	±0.004	0.978 (0.007)	±0.003
RL	0.960 (0.035)	±0.014	0.921 (0.017)	±0.007
RH	0.823 (0.177)	±0.070	0.913 (0.062)	±0.024
Overall	0.979 (0.007)	±0.003	0.972 (0.003)	±0.001

5.3 Partial track plots

Similarly to Section 5.2, for our third experiment, we used the same seven features, one line for each plot. We then ran this through XGBoost in groups of increasing length, in steps of one second.

In Table 5.4 we have listed the total accuracy and individual class accuracy for the 20 one-second intervals. We also include the complete dataset using plot features for easy reference of the overall accuracies. The results have been summarised in Figure 5.2 and Figure 5.3.

As expected, we see that the accuracy generally increases as the length of the included tracks increase. For ML, MH, RL, RH and the overall accuracy, the best result is achieved from having a track length of 20 seconds, i.e. the maximum track length we tested. However, for GL and GH, the best result is after six to nine seconds, after which the accuracy starts to decrease.

We also observe that for all classes the accuracies are better than the original Saab generation 2 algorithm as seen in Table 5.2 already after the first second within the 95 % confidence level.

5.4 Hyper-parameter optimisation

Tuning and training on the synthetic track features and complete track plot features concluded with the best schedules giving an accuracy of 1.0 and 0.999 respectively. This type of over-fitting is expected since the optimisation performed by PBT[2]. is quite aggressive. A typical schedule profile from PBT can look like Table 5.5. The schedule describes how the selected hyper-parameters are inherited and perturbed during the training of a model. This schedule, typically for the best performing model, can then be used to re-train new models using different data, e.g. for validating the model.

The results from the cross-validation of these two models are shown in Table 5.6. For synthetic track features, the results are very similar to the original unoptimised results. However, for complete track plots, we see a significant increase in accuracy, from 0.972 ± 0.001 to 0.984 ± 0, both within 95 % CI.

[2]We used Ray's implementation of PBT.

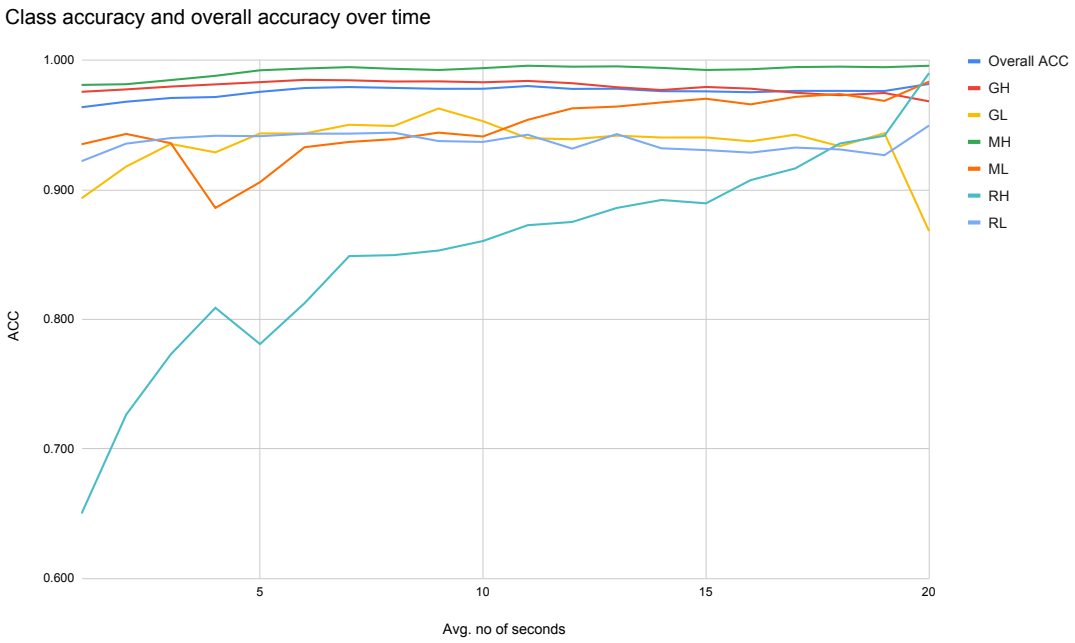


Figure 5.2: The development in class accuracy and overall accuracy using partial track plots in time steps from one to 20 seconds. RH sees a notable increase.

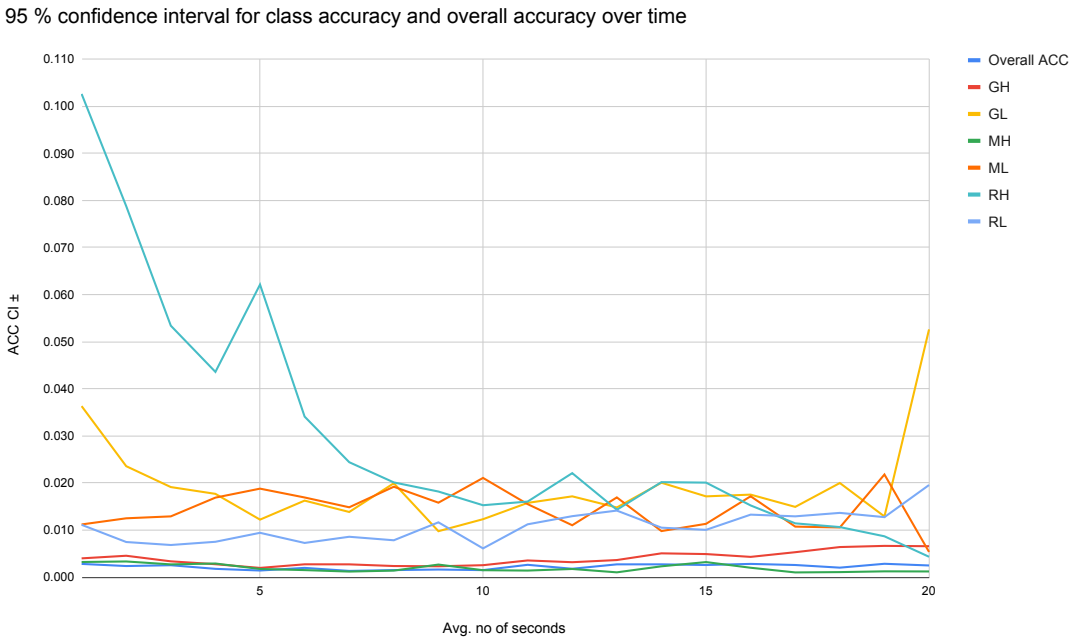


Figure 5.3: The development in the 95 % CI of class accuracy and overall accuracy using partial track plots in time steps from one to 20 seconds. RH sees a notable decrease in CI.

Table 5.4: The accuracy from XGBoost, five repeats with five folds, using tracks with an increasing number of plots. Each row represents the results using the accumulated series of plots from individual tracks in steps of five plots, about one second. In addition to the total accuracy, including the 95 % CI, and the accuracy per class with 95 % CI for the first row, we also show the number of unique tracks containing the required number of plots and the total number of plots included in the training and testing. The results using the complete dataset from Table 5.2 are included at the bottom for easy reference. The best accuracy scores have been highlighted for ease of reading.

Avg. no. of sec.	No. of tracks	No. of plots	Total		Mortar		Gun		Rocket	
			ACC	95 % CI	ML	MH	GL	GH	RL	RH
1	2259	11295	0.964	±0.003	0.935 ±0.011	0.981 ±0.003	0.893 ±0.036	0.976 ±0.004	0.922 ±0.011	0.650 ±0.103
2	2252	22520	0.968	±0.002	0.943	0.981	0.918	0.977	0.936	0.726
3	2214	33210	0.971	±0.003	0.936	0.985	0.935	0.980	0.940	0.773
4	2135	42700	0.972	±0.002	0.886	0.988	0.929	0.981	0.942	0.809
5	2074	51850	0.976	±0.001	0.906	0.992	0.943	0.983	0.941	0.781
6	2025	60750	0.979	±0.002	0.933	0.994	0.943	0.985	0.943	0.812
7	1983	69405	0.979	±0.001	0.937	0.995	0.950	0.985	0.943	0.849
8	1945	77800	0.979	±0.001	0.939	0.993	0.949	0.984	0.944	0.850
9	1889	85005	0.978	±0.002	0.944	0.992	0.963	0.984	0.938	0.853
10	1798	89900	0.978	±0.001	0.941	0.994	0.953	0.983	0.937	0.860
11	1685	92675	0.980	±0.003	0.954	0.996	0.940	0.984	0.943	0.873
12	1547	92820	0.978	±0.002	0.963	0.995	0.939	0.982	0.932	0.875
13	1363	88595	0.978	±0.003	0.964	0.995	0.942	0.979	0.943	0.886
14	1264	88480	0.976	±0.003	0.967	0.994	0.940	0.977	0.932	0.892
15	1204	90300	0.976	±0.003	0.970	0.992	0.940	0.979	0.931	0.890
16	1161	92880	0.975	±0.003	0.966	0.993	0.937	0.978	0.929	0.907
17	1126	95710	0.976	±0.003	0.972	0.995	0.942	0.975	0.933	0.916
18	1048	94320	0.976	±0.002	0.974	0.995	0.934	0.973	0.931	0.936
19	949	90155	0.976	±0.003	0.969	0.995	0.944	0.975	0.927	0.942
20	847	84700	0.982	±0.002	0.984	0.996	0.868	0.968	0.950	0.990
Overall	2259	189125	0.972	±0.001	0.968	0.986	0.931	0.978	0.921	0.913

Table 5.5: This PBT schedule for XGBoost trained using the synthetic track feature dataset to an accuracy of 1.0. It contains four perturbations after the initial perturbation; the first is where worker 17 is replaced with values from worker 13 at time step 10 and the hyper-parameters perturbed to the values 5 for *max_ depth,* 1 for *min_ child_ weight,* and 0.03505 for *eta.* Worker 0 is then replaced with the values from worker 17 at time step 20 and perturbed again etc. Note that the weights of the cloned worker are retained.

Time	Initial worker config				Clone from worker				Perturbed config		
	no.	eta	max depth	min child	no.	eta	max depth	min child	eta	max depth	min child
10	17	0.01803	1	3	13	0.05912	7	3	0.03505	5	1
20	0	0.09621	4	2	17	0.03505	5	1	0.05193	6	1
70	12	0.07653	1	3	0	0.05193	6	1	0.04760	4	2
80	0	0.09621	4	2	12	0.04760	4	2	0.05712	4	1

Table 5.6: The accuracy and SD (in parenthesis) from XGBoost hyper-parameter optimised with the PBT schedules, five repeats with five folds, using synthetic track features and plot features from all tracks. 95 % CI is also included for both. This table is comparable to Table 5.3, but with a schedule instead of a fixed set of hyper-parameters.

Class	XGBoost - tracks		XGBoost - plots	
	Accuracy (SD)	95 % CI	Accuracy (SD)	95 % CI
ML	0.969 (0.027)	±0.011	0.977 (0.003)	±0.001
MH	0.983 (0.010)	±0.004	0.993 (0.001)	±0.000
GL	0.934 (0.091)	±0.036	0.959 (0.007)	±0.003
GH	0.986 (0.009)	±0.004	0.988 (0.001)	±0.000
RL	0.959 (0.036)	±0.014	0.956 (0.002)	±0.001
RH	0.823 (0.177)	±0.069	0.958 (0.010)	±0.004
Overall	0.978 (0.007)	±0.003	0.984 (0.000)	±0.000

Chapter 6

Discussion

"...all knowledge, and especially the weightiest knowledge of the truth, to which only a brief triumph is allotted between the two long periods in which it is condemned as paradoxical or disparaged as trivial. "
— Arthur Schopenhauer, *The World as Will and Representation*

The aim of this research has been to evaluate how well ML algorithms perform at classifying ballistic non-cooperating radar targets (i.e. artillery projectiles) to provide better accuracy in POO and POI, hence providing better intelligence about hostile weapon systems. This intelligence can in turn provide more effective and timely countermeasures. We have compared two of Saab's original algorithms with several ML algorithms, particularly XGBoost. We have also investigated which features provide better classification power and how early in a projectile's trajectory an assessment can be made.

Based on the results in Chapter 5, we will answer the research questions put forward in Section 1.2:

RQ 1 *How good accuracy can be achieved using machine learning techniques compared to Saab's original algorithms?*

Based on the results in Section 5.1 we can conclude that the five ML methods we selected, C5.0, XGBoost, SVM, k-NN and both ANNs all outperforms Saab's original algorithms using synthetic features. The best results were achieved with C5.0 and XGBoost with 0.979 ± 0.003 for six classes for both methods, compared to 0.902 ± 0 for generation 1, for only three classes, and 0.852 ± 0 for generation 2, for six classes. The 95 % CIs do not overlap which is a significant result. Additional hyper-parameter optimisation, discussed in Section 5.4, did not yield improved results for synthetic features. This is discussed further in Section 6.1.1 and Section 6.1.4.

RQ 2 *How can measured features improve the classification power?*

Based on the results in Section 5.2, we observe that using plot features, measured and measurement control features of single plots, does not give a significant improvement in the unoptimised results. However, using PBT hyper-parameter optimisation, as discussed in Section 5.4, improves the best score from 0.979 ± 0.003 to 0.984 ± 0, which is a significant

improvement since the 95 % CIs do not overlap. This is discussed further in Section 6.1.2 and Section 6.1.4.

RQ 3 *How does the accuracy of the classification evolve over time for a track?*

Based on the results in Section 5.3, we observe that the overall classification accuracy using regular intervals of plot features increases over time for a track. We can also see a particularly pronounced increase in accuracy for RH. However, for gun artillery, the best accuracy is achieved at around six to seven seconds for GH and nine seconds for GL. The overall accuracy after one second is 0.964 ± 0.003 which is significantly better than both Saab's original generation 1 and 2 algorithms. The accuracies for individual classes are also significantly better than Saab's original algorithms after one second. This is discussed further in Section 6.1.3.

The rest of this chapter will discuss the results further, the challenges and limitations with this work as well as possible future work.

6.1 Experiments

6.1.1 Synthetic track features

Using synthetic track features, both C5.0 and XGBoost perform significantly better than the original first- and second-generation Saab algorithms, as seen in Table 5.2 and Figure 5.1.

Since both C5.0 and XGBoost are tree-based algorithms, it is relatively easy to investigate their internal workings. One of the 100 decision trees from our C5.0 run is presented in Listing 6.1.

The structure in Listing 6.1 looks similar to the simplified algorithm in Listing 1.1, which is based on the generation 1 algorithm by Saab. If we also consider the example limits in Table 2.1, we see that the similarity is strong with the first two decision levels looking very similar, with some differences in the thresholds and the use of the *acceleration* feature in the C5.0 model. The similarities are somewhat obscured by the fact that the original generation 1 algorithm by Saab only classifies into the three main projectile classes, while we have now also classified into light and heavy for each of the main projectile classes.

One interesting observation is that C5.0 uses different thresholds for *firing_velocity* based on whether the firing elevation is above or below 45.4°. If we compare the two, the first node in the C5.0 example tree, **firing_elevation** $<=$ **45.4**, is more similar to Saab's generation 1 algorithm than the second node, **firing_elevation** $>$ **45.4**; the latter being more complex than the first, indicating that high-angle firing[1] is more complex. Hence, Saab's generation 1 algorithm is possibly too simplistic.

C5.0 is using RCS to discriminate between guns and rockets, which is very reasonable and more or less identical to how it is done in Saab's algorithms since rockets are physically bigger and should therefore result in higher RCS values. But C5.0 is also using RCS to select between using firing elevation or firing velocity to separate between light and heavy mortars. This is less obvious as both firing elevation and velocity in theory should be independent of the apparent size of the projectile. We also see that acceleration, which Saab has not used, is used in two places in the decision tree. This indicates that not

[1] High-angle fire is artillery with a firing elevation at 45°and above, and low-angle is below 45°.

Listing 6.1: Example of a decision tree generated by C5.0. Each leaf is presented with a feature, a threshold value, the resulting class and the number of classifications and misclassifications, if present. All of the projectile classes are represented in this tree except GL. This is one tree from an ensemble so other trees may include GL.

```
firing_elevation <= 45.4:
:...firing_velocity <= 684: GH (767/126.8)
:   firing_velocity > 684:
:   :...rcs < =47.331: GH (63.7/32.2)
:       rcs > 47.331:
:       :...firing_elevation <= 18.2: RH (51/17.3)
:           firing_elevation > 18.2: RL (164.7/0.8)
firing_elevation > 45.4:
:...firing_velocity <= 144:
    :...acceleration <= 7: GH (81.6/6.8)
    :   acceleration > 7: MH (26.2/1.5)
    firing_velocity > 144:
    :...firing_velocity > 603: GH (49.4)
        firing_velocity <= 603:
        :...firing_velocity > 264:
            :...acceleration <= 39: MH (475.5)
            :   acceleration > 39: ML (24.7)
            firing_velocity <= 264:
            :...firing_elevation <= 53.4: MH (37.6)
                firing_elevation > 53.4:
                :...rcs <= 147.334:
                    :...firing_elevation <= 61.2: ML (46.5/1.5)
                    :   firing_elevation > 61.2: MH (144.1/20.3)
                    rcs > 147.334:
                    :...firing_velocity <= 233: ML (90.3/3)
                        firing_velocity > 233: MH (10.5)
```

only can C5.0 extract much of the same information automatically as what has been done empirically by Saab in the past, but also seemingly improve on this, even for classes with relatively few samples; GL and RH.

If we contrast this with the decision trees from the XGBoost classifier in Listing 6.2 through Listing 6.7, for ML, MH, GL, GH, RL and RH respectively, we see that they are binary classifiers where each tree predicts one of the six classes. That means that with one hundred trees per class, 600 trees are produced in total. Each leaf contributes a fraction to the overall result. This fraction is also incorporated into the *feature importance*, the relative usefulness of each feature in the model, where the sum for all features equals 1. In Listing 6.9, we see that the relative feature importance for XGBoost is similar to the feature usage in C5.0, how many of the cases each feature is in use, in Listing 6.8.

We also see that the XGBoost examples are generally much more complex than the one for C5.0, but given the similarity in results, it is likely that similar learning and generalisation to C5.0 is being performed also by XGBoost.

As briefly discussed in Section 2.5.4, XGBoost is an ensemble learning technique where the results of each base-learner are combined to generate the final probabilities. XGBoost treats a classification problem as a logistic regression problem and the raw leaf values seen in our examples are log-odds, **not** probabilities. To get probabilities, the odds need to be summed using the logistic function, which is the inverse of the logit function and then transformed into a probability [41]. This cannot be performed using only parts of

Listing 6.2: Example of a booster tree generated by XGBoost for ML using synthetic features. The tree is a binary classifier and each node is presented with a feature, a threshold value and the decision path further. Each leaf gives information on how much it contributes to the overall result.

```
booster[3]:
0:[firing_velocity <232.5] yes=1,no=2,missing=1
  1:[acceleration <3.5] yes=3,no=4,missing=3
    3:[firing_elevation <65.3500061] yes=7,no=8,missing=7
      7:[firing_elevation <65.1499939] yes=15,no=16,missing=15
        15:[acceleration <2.5] yes=21,no=22,missing=21
          21:leaf=-0.0562642515
          22:[firing_velocity <176.5] yes=25,no=26,missing=25
            25:leaf=0.0141740954
            26:leaf=-0.0525275357
        16:leaf=0.0156661049
      8:[firing_elevation <71.6499939] yes=17,no=18,missing=17
        17:leaf=0.269895345
        18:leaf=-0.0346111655
    4:[firing_velocity <158.5] yes=9,no=10,missing=9
      9:leaf=-0.0372070074
      10:[rcs <23.6464996] yes=19,no=20,missing=19
        19:[acceleration <4.5] yes=23,no=24,missing=23
          23:[firing_elevation <62.25] yes=27,no=28,missing=27
            27:leaf=-0.0372070074
            28:leaf=0.0626644269
        24:leaf=0.175331637
      20:leaf=0.280902416
  2:[rcs <0.685500026] yes=5,no=6,missing=5
    5:[firing_velocity <331.5] yes=11,no=12,missing=11
      11:leaf=0.0626644269
      12:leaf=-0.0410560071
    6:[rcs <4208.80518] yes=13,no=14,missing=13
      13:leaf=-0.0593901761
      14:leaf=-0.00561615312
```

the ensemble, hence we can't get a meaningful probability from single examples, only the example's contribution to the odds.

The complexity of the trees in the model between the different projectile classes varies widely. From a minimum of three nodes, basically a single if-then-else statement, to a maximum of 51 nodes. Our examples included here attempts to reflect the general tree complexities of the various classes. What we see is that in general, the complexity reflects the number of samples in each class. GL (Listing 6.4) and RH (Listing 6.7) are represented with relatively small trees with 8 to 10 nodes respectively, but also have decidedly the lowest number of samples from the dataset, 37 and 16 respectively, from a total of 2 259. Compare that with the trees for the other classes, which have 26 to 30 nodes, but roughly one magnitude more samples. We can reason that the size of the trees is roughly proportional to the number of samples used to generate them. This is not unreasonable as fewer data points of a particular class could result in a less complex algorithm for that class provided the samples used reflects a similar physical model with limited variance.

The question is whether this is a sign of under-fitting for GL and RH? Even though we perform cross-validation, the number of samples are so small for these classes that we cannot be categorical. This is also reflected in the SD presented in Table 5.2 for XGBoost, 0.081 and 0.177 for GL and RH respectively. We do not have SD for individual classes in

Listing 6.3: Example of a booster tree generated by XGBoost for MH using synthetic features. The tree is a binary classifier and each node is presented with a feature, a threshold value and the decision path further. Each leaf gives information on how much it contributes to the overall result.

```
booster[2]:
0:[firing_elevation<46] yes=1,no=2,missing=1
  1:[rcs<1.32500005] yes=3,no=4,missing=3
    3:[acceleration<24.5] yes=7,no=8,missing=7
      7:leaf=-0.0313322134
      8:leaf=0.140073419
    4:[acceleration<209.5] yes=9,no=10,missing=9
      9:leaf=-0.0589417927
      10:leaf=0.0156661049
  2:[firing_velocity<232.5] yes=5,no=6,missing=5
    5:[acceleration<3.5] yes=11,no=12,missing=11
      11:[firing_elevation<65.3500061] yes=15,no=16,missing=15
        15:leaf=0.273978859
        16:[firing_elevation<71.8999939] yes=19,no=20,missing=19
          19:leaf=-0.0541192815
          20:leaf=0.0156661049
      12:[rcs<88.4654999] yes=17,no=18,missing=17
        17:[acceleration<4.5] yes=21,no=22,missing=21
          21:[firing_elevation<62.4000015] yes=23,no=24,missing=23
            23:leaf=0.184263259
            24:leaf=0.0156661049
          22:[firing_elevation<69.1999969] yes=25,no=26,missing=25
            25:leaf=-0.0485969074
            26:leaf=0.0626644269
        18:leaf=-0.057357654
    6:[acceleration<13.5] yes=13,no=14,missing=13
      13:leaf=0.295709848
      14:leaf=0.0744140074
```

C5.0 since the tool does not report this for the individual folds, only for the summary.

It is difficult to compare and contrast individual C5.0 and XGBoost models, partly because they both are ensemble models. However, there are notable differences. Where the C5.0 tree in Listing 6.1 uses low and high relative values of *firing_elevation* to distinguish between RL and RH, XGBoost uses high *firing_velocity* combined with medium *rcs* (node 11 in Listing 6.6: Leaf value of 0.285) and medium *firing_velocity* combined with medium *firing_elevation* (node 21 in Listing 6.6: Leaf value of 0.254) to select for RL and high *acceleration* combined with high *rcs* (node 6 in booster 4 in Listing 6.7: Leaf value of 0.206) to select for RH. The delineation between RL and RH concerning *rcs* in our two XGBoost examples is exactly 889.815552. This can be seen in node 6 of Listing 6.6 and node 2 in booster 4 of Listing 6.7. Being an ensemble of learners, we have no way of validating that value or anything else from these trees, but from a manual inspection of the 16 RH samples, this value seems sensible: Only one RL is above and only two RH are below this limit.

While there are some similarities in the description above, *firing_elevation* being used by both, it is clear that the XGBoost models are much more complex. Yet, C5.0 and XGBoost score more or less the same. We cannot explain this apparent inconsistency. The XGBoost models appear not to be prone to overfitting judging by the SD.

We have not analysed the results of the SVM, k-NN and ANN models as they initially

Listing 6.4: Example of three booster trees generated by XGBoost for GL using synthetic features. Two of the trees are more or less identical with only small differences in the leaf values, while the third also adds RCS as a condition. They are much smaller and less complex than the other projectile classes except RH. The tree is a binary classifier and each node is presented with a feature, a threshold value and the decision path further. Each leaf gives information on how much it contributes to the overall result.

```
booster[1]:
0:[firing_elevation<27.3499985] yes=1,no=2,missing=1
  1:[firing_velocity<316] yes=3,no=4,missing=3
    3:[acceleration<4.5] yes=5,no=6,missing=5
      5:leaf=-0.0538314134
      6:[firing_elevation<19.7999992] yes=7,no=8,missing=7
        7:leaf=-0.0313322134
        8:leaf=0.256362706
    4:leaf=-0.0588730015
  2:leaf=-0.0593802407
booster[7]:
0:[firing_elevation<27.3499985] yes=1,no=2,missing=1
  1:[firing_velocity<316] yes=3,no=4,missing=3
    3:[acceleration<4.5] yes=5,no=6,missing=5
      5:leaf=-0.0529800467
      6:[firing_elevation<19.7999992] yes=7,no=8,missing=7
        7:leaf=-0.031398233
        8:leaf=0.203315571
    4:leaf=-0.058095891
  2:leaf=-0.0586097091
booster[61]:
0:[firing_elevation<27.3499985] yes=1,no=2,missing=1
  1:[firing_velocity<316] yes=3,no=4,missing=3
    3:[acceleration<4.5] yes=5,no=6,missing=5
      5:leaf=-0.0459649973
      6:[rcs<11.5494995] yes=7,no=8,missing=7
        7:leaf=-0.00487041473
        8:[firing_elevation<22.6500015] yes=9,no=10,missing=9
          9:leaf=0.00924922246
          10:leaf=0.0834095553
    4:leaf=-0.0528791547
  2:leaf=-0.0536592118
```

perform worse than C5.0 and XGBoost and are more complicated compared to evaluate than decision tree models. Particularly ANN with its mimic of neurological processes is obtuse. This lack of transparency of some ML algorithms is also limiting the usefulness of their adoptions in some areas due to the lack of trust [106]. This is especially the case in the defence industry where accountability is an important factor, see Section 3.3.

As seen in Listing 6.9, 85.5 % of the decisions are controlled by the three features *firing_elevation*, *firing_velocity* and *rcs* - the same three features used by Saab's generation 1 algorithm. By also utilising the *acceleration* feature, accounting for 14.5 % of the decision power for the XGBoost model, and modelling an ensemble of 100 learners we increase the accuracy from 0.902 ± 0 to 0.979 ± 0.003, while also discriminating into light and heavy calibres. The original result for Saab's generation 2 algorithm, which already discriminates between light and heavy calibres, is 0.852 ± 0. This is a significant improvement compared to both legacy Saab algorithms, see Figure 5.1.

Listing 6.5: Example of a booster tree generated by XGBoost for GH using synthetic features. The tree is a binary classifier and each node is presented with a feature, a threshold value and the decision path further. Each leaf gives information on how much it contributes to the overall result.

```
booster[0]:
0:[firing_elevation <36.5499992] yes=1,no=2,missing=1
  1:[firing_velocity <700] yes=3,no=4,missing=3
    3:[firing_velocity <463.5] yes=7,no=8,missing=7
      7:[acceleration <25.5] yes=15,no=16,missing=15
        15:[firing_velocity <312.5] yes=25,no=26,missing=25
          25:[acceleration <4.5] yes=29,no=30,missing=29
            29:leaf=0.269157022
            30:leaf=-0.0169648547
          26:leaf=0.280767739
        16:[firing_elevation <27.4500008] yes=27,no=28,missing=27
          27:[rcs <1.46950006] yes=31,no=32,missing=31
            31:leaf=0.0156661049
            32:leaf=-0.0520377681
          28:leaf=0.0626644269
      8:[acceleration <35.5] yes=17,no=18,missing=17
        17:leaf=0.295744568
        18:leaf=0.111621007
    4:[rcs <47.723999] yes=9,no=10,missing=9
      9:[firing_velocity <945.5] yes=19,no=20,missing=19
        19:leaf=0.173055828
        20:leaf=0.0156661049
      10:leaf=-0.0571657307
  2:[rcs <15.5895004] yes=5,no=6,missing=5
    5:[firing_elevation <47.4500008] yes=11,no=12,missing=11
      11:[rcs <4.33899975] yes=21,no=22,missing=21
        21:leaf=0.0156661049
        22:leaf=0.255798131
      12:leaf=-0.0551214851
    6:[firing_velocity <145] yes=13,no=14,missing=13
      13:[rcs <276.780518] yes=23,no=24,missing=23
        23:leaf=-0.0313322134
        24:leaf=0.0484556332
      14:leaf=-0.0588880628
```

6.1.2 Complete track plots

It came as a surprise to us that using plot features gave such strong results compared to synthetic track features, 0.972 ± 0.001 vs. 0.979 ± 0.003, see Table 5.3. The result is significantly lower, within a 95 % confidence level since the CIs do not overlap, but is still significantly better than the existing Saab algorithms. What is surprising is that using plots directly, without considering it a time series, is within 0.007 points of the result where we have used features extracted from the entire length of the track.

It is not obvious to us how this result is achieved. Is the problem easier than expected? Does XGBoost perform feature extraction which is not obvious? Or are we just lucky because rounds follow a narrow feature space?

Inspecting the feature importances for the plot features can give us some insight. They are given in Listing 6.10 and we observe that the most important feature by far accounting for 46.1 % of the "decision power", how important a feature is, is the *radial_velocity* feature. Radial velocity is the rate of change as a vector of the distance between a point,

Listing 6.6: Example of a booster tree generated by XGBoost for RL using synthetic features. The tree is a binary classifier and each node is presented with a feature, a threshold value and the decision path further. Each leaf gives information on how much it contributes to the overall result.

```
booster[5]:
0:[firing_velocity<685.5] yes=1,no=2,missing=1
  1:[firing_elevation<20.25] yes=3,no=4,missing=3
    3:[firing_velocity<409.5] yes=7,no=8,missing=7
      7:[firing_elevation<19.0499992] yes=13,no=14,missing=13
        13:leaf=-0.0546157882
        14:leaf=0.0156661049
      8:[firing_elevation<18.4000015] yes=15,no=16,missing=15
        15:leaf=0.0626644269
        16:[firing_velocity<462.5] yes=21,no=22,missing=21
          21:leaf=0.254096627
          22:leaf=0.0626644269
    4:[firing_velocity<134.5] yes=9,no=10,missing=9
      9:leaf=0.140073419
      10:[acceleration<37.5] yes=17,no=18,missing=17
        17:[firing_elevation<20.4500008] yes=23,no=24,missing=23
          23:[firing_velocity<572] yes=27,no=28,missing=27
            27:leaf=0.0156661049
            28:leaf=-0.0372070074
          24:leaf=-0.0591478087
        18:[firing_velocity<501] yes=25,no=26,missing=25
          25:leaf=-0.0346111655
          26:leaf=0.0626644269
  2:[rcs<39.6094971] yes=5,no=6,missing=5
    5:leaf=-0.0448522829
    6:[rcs<889.815552] yes=11,no=12,missing=11
      11:leaf=0.285002619
      12:[firing_elevation<18.1500015] yes=19,no=20,missing=19
        19:leaf=-0.0425222926
        20:leaf=0.0156661049
```

our WLR, and an object, the projectile in question. In other words, the speed at which a target is moving away from the radar. An object can be approaching, negative radial velocity, receding, positive radial velocity, or stationary relative to the radar, zero radial velocity. On its own, it is not a very useful feature as it is highly dependent on the geometry of the firing, and can in principle be the same irrespectively of the projectile type and calibre, at least for a single measurement. However, as we see from a typical decision tree example produced by XGBoost using plot features in Listing 6.11 for RH, this feature is heavily involved with other features like *snr* and str^2 (nodes 1, 2, 4, 11, 19, 20, 23, 33 and 38 in Listing 6.11). This indicates that XGBoost has managed to perform feature extraction which is not evident for a human or that the limited number of RHs have similar geometries relative to the radar. This situation is also similar for the other features.

As for the other questions posed above, we cannot conclude in any direction. The likely answer is that it is a combination of several factors which leads to these positive results.

The decision trees produced by XGBoost when using plot features are much more

[2]STR is a proprietary Saab measure of signal strength irrespectively of attenuations and gains in the signal path.

Listing 6.7: Example of two booster trees generated by XGBoost for RH using synthetic features. Since these two are for the same class they are quite similar. Other trees can be much more complicated, some with up to fifty leaves. The tree is a binary classifier and each node is presented with a feature, a threshold value and the decision path further. Each leaf gives information on how much it contributes to the overall result.

```
booster[4]
0:[acceleration<88.5] yes=1,no=2,missing=1
  1:[firing_elevation<17] yes=3,no=4,missing=3
    3:[firing_elevation<13.8999996] yes=7,no=8,missing=7
      7:leaf=-0.039313063
      8:leaf=0.0626644269
    4:leaf=-0.059409827
  2:[rcs<889.815552] yes=5,no=6,missing=5
    5:leaf=-0.0511596315
    6:leaf=0.206069544
booster[70]:
0:[acceleration<70.5] yes=1,no=2,missing=1
  1:leaf=-0.0533642098
  2:[firing_velocity<862.5] yes=3,no=4,missing=3
    3:[firing_elevation<12.75] yes=5,no=6,missing=5
      5:leaf=-0.032286115
      6:[acceleration<96.5] yes=7,no=8,missing=7
        7:leaf=0.0901749209
        8:leaf=0.0285347607
    4:leaf=-0.0475433432
```

Listing 6.8: The feature usage of the four track features from the model generated by C5.0. The percentages describe the feature's use in the total decision path for all the trees.

```
100 %  firing_elevation
100 %  firing_velocity
 55 %  acceleration
 24 %  rcs
```

Listing 6.9: The feature importances of the same four track features from the XGBoost model. The numbers are fractions of cases determined by that feature for the whole ensemble of trees, summing up to 1.

```
0.493  firing_elevation
0.297  firing_velocity
0.145  acceleration
0.065  rcs
```

complex than when using synthetic features. The example in Listing 6.11 for RH contains 75 nodes, compared to eight for a tree using synthetic features in Listing 6.7. Part of this increase in complexity can be explained by the increase in dimensionality; from four to seven features, about a factor of $(7/4)^2 \approx 3.0$ in the complexity of a binary tree. The "remaining" increase is likely to come from the increase in samples, from 16 of 2 259, or 0.7 %, to 2571 of 189 125, or 1.4 %, two magnitudes more total samples and a doubling in the relative number of samples, for RH.

The complexity of the trees in the model across the different projectile classes varies, from a minimum of 53 nodes to a maximum of 127 nodes. This is only a doubling of

Listing 6.10: The feature importances of the seven features from the XGBoost model using complete track plots. The numbers are fractions of cases determined by that feature for the whole ensemble of trees, summing up to 1.

```
0.461  radial_velocity
0.196  upper_pri
0.131  snr
0.074  elevation
0.051  lower_pri
0.050  str
0.034  phase
```

complexity difference compared to more than a magnitude for the model using synthetic features. A higher maximum complexity indicates a more complex problem with higher dimensionality. Since the relative difference between the maximum and minimum complexity is lower, this could potentially indicate better model fitting of the data, i.e. less over- or under-fitting. This is supported by the lower variance for plot features compared to synthetic features.

6.1.3 Partial track plots

From the chart in Figure 5.2, it is clear that there is a marked improvement in accuracy for RH with longer tracks. For the other classes and overall accuracy, there are fewer variations with the increase in the length of the tracks. Similarly, the 95 % CI is following an inverse trend compared to the accuracy, as seen in Figure 5.3.

Using plot features in groups of regular intervals of one second gives a general increase in accuracy as the tracks get longer without affecting the 95 % CI much, see Table 5.4. As the tracks get longer, the number of tracks of this minimum length from our dataset, get fewer. However, the total number of plots used for training one group increases up to more than 92 000 before levelling off at around eleven seconds. Still, for ML, MH, RL and RH, the accuracies continue to increase. This can be explained either with a robust model for these classes or that there is less variation for these longer tracks.

For GL and GH the accuracies increase up to seven seconds for GH and nine seconds for GL. If we zoom in on the accuracy and 95 % CI for just GL and GH, see Figure 6.1 and Figure 6.2, we can conclude that the change in accuracy is significant with the 95 % confidence level. Also note that there is a sudden drop in the accuracy, and a similar increase in CI, for GL for the last group with 20 seconds worth of plot data for each track, which can be explained by very few samples of tracks of this length.

It is worth noting that the models are trained for a specific duration of plot data. Using these models for any other duration of plot data is probably not useful. This is illustrated in Figure 6.3 where we see that the relative feature importances vary over time.

One possible practical application of this discovery is to fix the projectile classification after a certain time *iff* it is one of GL or GH. In our research, this time appears to be around six to nine seconds, but further research is needed to confirm our findings. This could also potentially increase the accuracy for the other projectile classes because of a decrease in the problem complexity if two classes are removed from the remaining classification problem.

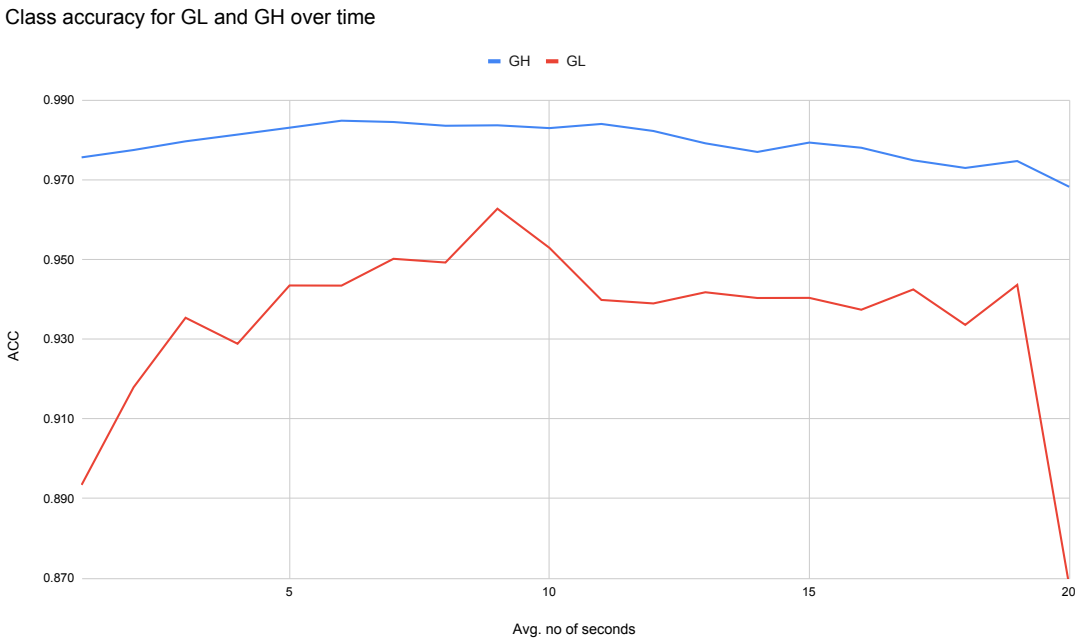


Figure 6.1: The development in class accuracy for GL and GH from the XGBoost model using partial track plots in time steps from one to 20 seconds.

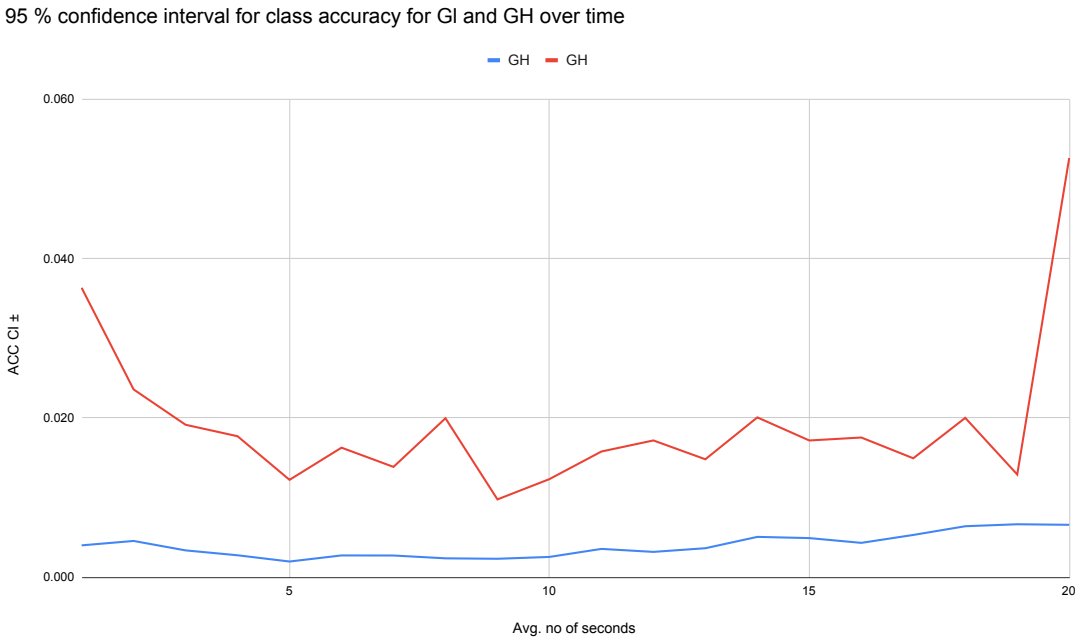


Figure 6.2: The development in the 95 % CI for GL and GH from the XGBoost model using partial track plots in time steps from one to 20 seconds.

Listing 6.11: Example of a booster tree generated by XGBoost for RH using plot features. The tree is a binary classifier and each leaf is presented with a feature, a threshold value and how much each leaf contributes to the overall result.

```
booster[4]:
0:[snr<18.3449993] yes=1,no=2,missing=1
  1:[radial_velocity<-456.100006] yes=3,no=4,missing=3
    3:[str<13.6900005] yes=7,no=8,missing=7
      7:[str<12.8150005] yes=13,no=14,missing=13
        13:[snr<-0.875] yes=25,no=26,missing=25
          25:leaf=-0.17680946
          26:[snr<1.27999997] yes=47,no=48,missing=47
            47:leaf=0.0499999933
            48:leaf=-0.166666657
        14:[snr<4.15999985] yes=27,no=28,missing=27
          27:[phase<151.75] yes=49,no=50,missing=49
            49:leaf=-0.112500004
            50:leaf=0.527586222
          28:[elevation<2.21000004] yes=51,no=52,missing=51
            51:leaf=-0.148543686
            52:leaf=0.0209302269
      8:[pri_upper<433.5] yes=15,no=16,missing=15
        15:leaf=-0.166386545
        16:[str<15.4699993] yes=29,no=30,missing=29
          29:[snr<8.71999931] yes=53,no=54,missing=53
            53:leaf=0.51351887
            54:leaf=-0.138461545
          30:[elevation<2.80999994] yes=55,no=56,missing=55
            55:leaf=0.756054819
            56:leaf=0.0473684184
    4:[radial_velocity<-349.799988] yes=9,no=10,missing=9
      9:[snr<7.59500027] yes=17,no=18,missing=17
        17:[str<12.9099998] yes=31,no=32,missing=31
          31:[snr<4.72000027] yes=57,no=58,missing=57
            57:leaf=-0.178372413
            58:leaf=-0.130668849
          32:[pri_upper<493.5] yes=59,no=60,missing=59
            59:leaf=0.22005856
            60:leaf=-0.176908389
        18:[str<13.9699993] yes=33,no=34,missing=33
          33:[radial_velocity<-437.450012] yes=61,no=62,missing=61
            61:leaf=0.0264705811
            62:leaf=-0.150000006
          34:[pri_upper<375] yes=63,no=64,missing=63
            63:leaf=-0.16578947
            64:leaf=0.580683291
      10:[snr<14.9699993] yes=19,no=20,missing=19
        19:[radial_velocity<-311.200012] yes=35,no=36,missing=35
          35:[snr<12.3450003] yes=65,no=66,missing=65
            65:leaf=-0.177563265
            66:leaf=0.145065784
          36:leaf=-0.17986697
        20:[radial_velocity<-282.700012] yes=37,no=38,missing=37
          37:[pri_upper<359] yes=67,no=68,missing=67
            67:leaf=-0.175149709
            68:leaf=0.766569734
          38:[radial_velocity<-218.350006] yes=69,no=70,missing=69
            69:leaf=-0.140141562
            70:leaf=-0.179855406
  2:[radial_velocity<-215.549988] yes=5,no=6,missing=5
    5:[pri_upper<357.5] yes=11,no=12,missing=11
      11:[radial_velocity<-223.299988] yes=21,no=22,missing=21
        21:[pri_upper<356.5] yes=39,no=40,missing=39
          39:leaf=-0.179216072
          40:leaf=-5.02914199e-09
        22:[elevation<3.71000004] yes=41,no=42,missing=41
          41:leaf=-0.0947368443
          42:leaf=0.33157894
      12:[elevation<4.11499977] yes=23,no=24,missing=23
        23:[radial_velocity<-319.149994] yes=43,no=44,missing=43
          43:[pri_upper<382] yes=71,no=72,missing=71
            71:leaf=-0.124137938
            72:leaf=0.661764741
          44:leaf=-0.176223785
        24:[elevation<6.07999992] yes=45,no=46,missing=45
          45:[phase<216.25] yes=73,no=74,missing=73
            73:leaf=0.866650403
            74:leaf=0.228497401
          46:leaf=-0.17496112
    6:leaf=-0.179932356
```

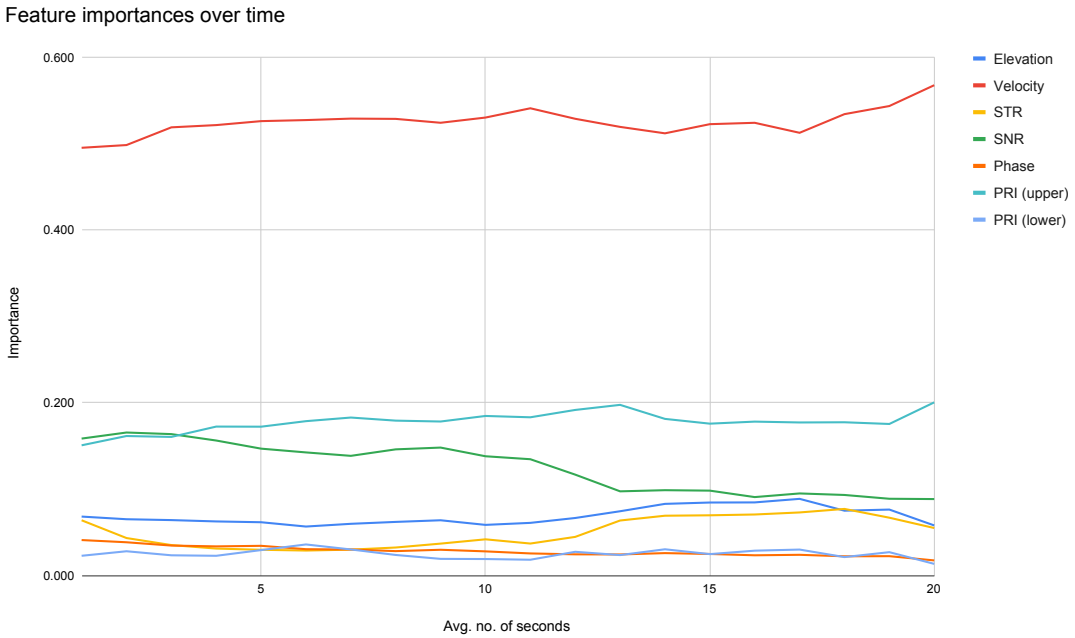


Figure 6.3: The variation of feature importances of the seven features from the XGBoost model using partial track plots in time steps from one to 20 seconds.

6.1.4 Hyper-parameter optimisation

Following up on the discussion from Section 6.1.2 and seeing that the variance is much less for an XGBoost model trained with plot features compared to one trained with synthetic track features, see Table 5.3, we suspected that additional hyper-parameter optimisation would be more beneficial when using plot features. As we can see from Table 5.6, there is very little change in hyper-parameter optimising the synthetic features, while optimising plot features gives a significant increase from 0.972 ± 0.001 to 0.984 ± 0. This is also our best result, significantly better than 0.979 ± 0.003 for unoptimised, synthetic features.

The complexity of the XGBoost trees across the different projectile classes varies even more in the optimised model. From a minimum of 25 nodes to a maximum of 401 nodes. Again we have more than a magnitude in the difference in the complexity between the minimum and maximum.

What happens during PBT hyper-parameter optimisation is that the learning process is perturbed several times during learning and this process is performed for multiple models, 20 in our case. It is therefore probable that the variety in the complexity of individual trees becomes greater as a more random exploration of the hyper-parameter space is performed.

However, this added complexity might be unwanted in a military application where accountability is important. As described in Section 4.2.4, we did not attempt to restrain XGBoost from growing in complexity in this experiment. Particularly *max_depth* is important for the total complexity of each learner, but other hyper-parameters are also available to control the resulting complexity in more detail.

6.2 Challenges and limitations

As described in Section 4.1, the labelling of the tracks is based on truth data which is manually recorded, usually in a notebook or a separate computer file, for each firing session. This can introduce several sources of errors; from the people in charge of the firing, via the people handling the ammunitions and weapons, via the people communicating this information, typically over radio communication and potentially across a language barrier, to a representative from Saab, to the Saab employee who records this information and associate it with the corresponding track file. Also, the final step, which we have performed as a part of this research, of mapping a single track to the correct recorded label to prepare a benchmark, can introduce additional sources of errors.

Normally, one firing session is recorded in multiple track files which can contain zero or more tracks each. The challenge is that the recorded truth data is done per firing session and usually only for the firings of interest within that session. Hence, many types of projectiles can have been fired during the same session, often at the same time, without the specific truth data being associated with a particular track. Also, firings that are not part of the session and false targets from clutter can be detected and recorded without being labelled as such. These tracks needed to be identified and removed.

Discriminating which projectile is represented in which track is, therefore, a time consuming and potentially error-prone task when done retrospectively. The raw dataset provided by Saab contains over 200 000 tracks and through the manual process outlined in Section 4.1, we ended up with 2 259 tracks from 172 sessions, with what we believe is correctly labelled data. During this process, we found several examples of suspicious or wrongly labelled data. These were excluded altogether.

Another factor we need to consider is the background for these recordings. Normally, a firing session is planned well in advance to test a particular feature of the radar or validate the radar performance in a specific scenario. To do this, the geometry of firing and the weather must be within specific parameters. This means that there could be a bias towards particular patterns for particular projectiles, geometries and specific weather conditions which could affect the real-life performance of our work. Naturally, any empirically developed algorithm using the same data will have similar constraints.

As can be seen in Table 4.4, there is mainly one source for the relatively low performance of the generation 2 algorithm; the failure to distinguish between the calibre groups light and heavy. More specifically, the failure to classify RH in general, the misclassification of ML as MH and vice versa and, to a lesser extent, the misclassification of GL as GH and vice versa. This makes sense as it is less complicated to only classify into three main groups than to also classify into two subgroups, six groups altogether. If we adjust for this and only look at the basic projectile types also for generation 2, we get an overall accuracy of 0.969. This is better than for generation 1, but this is not what we desire as this reduces the functionality of the classifier. Hence, the comparison between Saab's generation 1 and 2 performed in this work is not "fair".

As far as we have been able to determine, no research into ballistic projectile classification using machine learning has ever been published before. One of the reasons for this can be that the dataset we have used is based on many years of development, testing and data recording using the ARTHUR WLR. This makes this proprietary dataset potentially unique, which other manufacturers of WLR systems might not have available for their research. Another reason can be that manufacturers have kept their

results proprietary and hence never published them as we do now. This makes it impossible to compare and contrast our results with similar research.

Nevertheless, the results we have are very clear compared to Saab's original proprietary algorithms, that we think they stand on their own. However, the single largest challenge with this work is the quality and amount of data, particularly the low number of samples for the projectile classes GL and RH. Even though the variance for these two classes is also generally low in our hyper-parameter optimised results, as shown in Table 5.6, the data has been collected from only a limited number of firing sessions, rounds of fire with similar geometries; two and one sessions respectively for GL and RH. It is therefore likely that additional samples of these projectile classes from more diverse firing sessions will provide a more challenging problem. We recommend that additional data are sourced, either from additional recordings, or an even more involved examination of the existing data.

As with most ML applications, one weakness in the novel methods for projectile classification we have discussed and used in this work, and hence in the results reported, is that all the samples in the data set are from weapon and projectile types that are known. New projectile types, projectiles that have never been recorded by ARTHUR or new firing geometries might fall outside the "boundaries" defined during the learning process and hence might give incorrect classification or fail to give a majority classification if encountered. This is also a challenge with the original Saab algorithms as they too are based on empirical data. However, for a human engineer, it is possible to take the knowledge of missing knowledge and add fail-safes or extrapolate to probable scenarios and thereby potentially handle these types of situations better than ML algorithms.

As we briefly mentioned in Section 4.2.2, we have not performed any data augmentation, see Section 2.5.8. Nor have we applied any regularisation techniques, see Section 2.5.9, partly based on the initial strong results, but also since there is built-in regularisation in XGBoost. However, these are techniques that could also potentially be beneficial for this dataset irrespectively of the underlying ML framework.

Normally we would run a suitable statistical hypothesis test or calculate the p-value of a given hypothesis test on our results, for a statistically strong result. However, since we use stratified k-fold cross-validation, which breaks the condition of independence, and since the measures of variability are dissimilar, given the differences in replications using bagging to derive the results for Saab's generation 1 and 2 algorithms, and using cross-validation for our ML methods, this is not methodically safe [105]. Instead, we use the 95 % CI directly: Where the confidence intervals do not overlap we can claim that the result is significant within the 95 % confidence level. However, where they do overlap, we cannot claim the converse, i.e. that the result is not significant.

The consequence of this method is that we may fail to detect type II errors, i.e. that we fail to reject a false null hypothesis [105]. In other words, that we fail to detect a significant result where the intervals overlap. To increase the statistical power of our work, a more thorough statistical treatment is required, which is not within the scope of this thesis.

However, for this work, this is less of a problem as we do not have CI overlap for our claims. The main reason for this is that the variances are generally very low, resulting in very small CIs.

6.3 Future work

Since the initial results using decision tree models were quite strong, as described in Section 5.1, we opted to focus our effort in this direction since decision trees are easier to interpret and, as such, more suitable for Saab. We see from Section 5.1 that all the ML methods tested perform better than Saab's algorithms. Hence, there is no reason that other ML algorithms cannot perform equally well or better compared to our best results with the appropriate optimisation. Also, the time-aware RNN algorithms briefly discussed in Section 2.5.5, should be investigated.

We have additional domain knowledge in our track files that have not been used. By including information about the weather, particularly wind, accurate position information of the WLR and individual system corrections, synthetic, precise 3D-position vectors of plots can be calculated. Together with the relative height of the plot, these additional synthetic features, which represent the ballistic trajectory of the projectile when all plots are included, can be considered a behavioural pattern when considered as a time series. They could potentially give any ML method, but perhaps RNN more so, useful additional information to discriminate between the various projectile classes.

The natural next concrete step would be to mix synthetic and plot features and train a mixed model, using e.g. XGBoost. There is no obvious advantage to this based on the existing results; no projectile class performs better using only synthetic features. However, a mixed model could potentially perform more advanced feature extraction which could provide improved performances.

Hybrid models where different models are used for different parts of the problems, as introduced in Section 6.1.3 with one or two models for guns and separate models for rockets and mortars could also provide better overall performance. Another possibility can be seen from Table 5.2 where SVM and k-NN give better performance for the RH class than the other ML methods.

One of the advantages of decision trees like C5.0 and XGBoost is that they are relatively easy to interpret. This is particularly useful in the defence industry where accountability is important. Of course, having an ensemble model with several 10s or 100s of individual decision trees can become very difficult to follow. While the decision trees produced by C5.0 can be quite complex, the one in Listing 6.1 is one of the least complex produced in our experiments, they can be used as inspiration to manually adapt existing algorithms. We have shown that not only can additional features be used to good effect, but additional discriminant nodes can also be added to improve the final accuracy as illustrated in Listing 6.1. The advantage of this approach is that it can be possible to validate the final solution analytically.

Instead of a manual adaption of existing algorithms or using ensembles of decision trees directly from e.g. XGBoost, it could be worthwhile to investigate how GP, a sub-field within genetic algorithms first introduced by Koza [107], could be used on this problem. We envisage that the existing algorithm, possibly optimised by some of our findings, can undergo evolution using any of the GP frameworks that exist, e.g. ADATE [108], IGoR [109] or MagickHaskell [110], and provide valuable insight.

Furthermore, both C5.0 and XGBoost have hyper-parameters that can be used to adjust the complexities of each learner. C5.0 has the concept of *pruning*, where parts of the tree, which are predicted to have a relatively high error rate, are removed, and it is possible to control how aggressive this pruning is. Similarly, the *max_ depth* hyper-parameter in

XGBoost is used to control the maximum depth of a tree, which directly controls the complexity. The default value is six, which can still give quite complex trees. XGBoost also provides hyper-parameters to tune the total number of leaves and how tolerant each node is for incorrect classification. We have on purpose used default values for our experiments. Reducing the ceiling for these hyper-parameters will reduce the complexity of the final model and could make them easier for human analysis.

By changing the focus from a "best possible" model to a "lowest possible complexity while retaining as much accuracy as possible" model, we believe it is possible to create models which are much less complex, both for C5.0 and XGBoost, and still provides better performance than the original Saab algorithms. These can then be better candidates to analyse and understand to validate for accountability, where this is a requirement, or for further research as described above.

A prerequisite for the classification methods used here, both Saab's original and the ML methods developed here, is that false targets like birds, planes or other forms of clutter have been filtered out before the projectile classification. By thresholding the resulting probabilities it could be possible to detect false targets also during the projectile classification. Similarly, the individual probabilities can be used to calculate the level of confidence of the classification, which can be used to determine whether a particular classification should be trusted or not.

Since we have a large amount of unlabelled data, which has not been used in this research, as well as a small amount of labelled data, over 200 000 and 2 259 respectively, the situation lends itself to exploration using semi-supervised methods. Semi-supervised methods can be thought of as a hybrid of supervised and unsupervised learning, which were briefly mentioned in Section 2.5, where the small amount of labelled data augments the unlabelled data [111], and can thereby potentially improve the final result.

Chapter 7

Conclusion

"Only a Sith deals in absolutes."
— Obi-Wan Kenobi, *Revenge of the Sith*

The purpose of this master's thesis is to investigate the possibility of using ML for classifying ballistic artillery projectiles. We have used both synthetic features, features calculated from measurements of a single projectile track, and radar plot features, individual measurement and measurement control features.

The data was provided by Saab Technology Norway AS, a subsidiary of Saab AB, and is completely novel. It consists of more than 10 GB of raw data, approximately 12 000 track files with over 200 000 individual tracks recorded using Saab's ARTHUR WLR from various locations around the world. The filtered dataset usable for our research consists of 2 259 unique tracks, just over 1 % of the total tracks available, with a total of 189 125 individual radar plots.

Three research questions are discussed in this thesis: Firstly, we want to see how well ML can classify our data using synthetic features compared to Saab's original algorithms (RQ1). Secondly, we look at how measured features can improve the classification power (RQ2). Thirdly, we explore how the accuracy of the classification evolve over time for a track (RQ3).

Our work shows that we can significantly outperform Saab's original classification algorithms using ML and synthetic features. We also show that the performance can be further improved by using PBT to optimise the hyper-parameters when using plot features from a track. Finally, we can outperform Saab's original algorithms using only one second of non-optimised plot features from a track. The best results were achieved using XGBoost, an ensemble tree method with gradient boosting. Using our best ML model to classify our data into six projectile classes, the accuracy increases from 0.852, using Saab's original generation 2 algorithm, to 0.984.

Using plot data, instead of synthetic track data, to classify ballistic artillery projectiles into one of the projectile types gun, mortar or rocket as well as the calibres light or heavy, is, as far as we have been able to determine, completely novel. As for synthetic features, we see that, in addition to the natural features elevation, firing velocity and RCS, acceleration has a clear decision power; stronger than RCS. This knowledge can be utilised regardless of what else is taken from this work.

These results can be used to improve the accuracy of the estimation of POO and POI and thereby improve the quality of the warnings given to own troops and civilians about

pending ballistic attacks. They can also be used to improve the effectiveness of counter-attacks on hostile weapons and thereby potentially reduce fatalities, collateral damage and the number of enemy attacks.

Organisations or individuals who choose to use the result from this research, must on an independent basis make the judgement whether they are within the respective national and applicable international treaties, norms and laws.

Chapter 8

Glossary

AESA active electronically steered array

ANN artificial neural network

ARTHUR artillery hunting radar

AWS autonomous weapons systems

BPTT back-propagation through time

C2 command and control

CEP circular error probability

CI confidence interval

DARPA Defense Advanced Research Projects Agency

EC evolutionary computation

EM electromagnetic

GH gun heavy

GL gun light

GP genetic programming

GRU gated recurrent unit

IFF identification friend or foe

LOW laws of war

LSTM long short-term memory

k-NN k-nearest neighbour

MH mortar heavy

ML machine learning

ML mortar light

MLE maximum likelihood estimation

MSD maximum scatter difference

NCTR non-cooperative target recognition

NENT National Committee for Research Ethics in Science and Technology

NN neural network

PBA population-based augmentation

PBT population-based training

PESA passive electronically scanned array

POI point of impact

POO point of origin

PSR primary surveillance radar

RAM rocket, artillery, mortar

RH rocket heavy

RL rocket light

RCS radar cross-section

RNN recurrent neural network

SD standard deviation

SNR signal-to-noise ratio

SMOTE synthetic minority over-sampling technique

SSR secondary surveillance radar

SVM support vector machine

UN United Nations

TBM theatre ballistic missile

WL weapon locating

WLR weapon locating radar

WND will not disclose

Bibliography

[1] "U.S. Army engineers work to create a new longer M777 155mm howitzer under the name M777er 13103162," *Army Recognition*, Mar. 2016.

[2] Col. Liam Collins and Capt. Harrison "Brandon" Morgan, "King of Battle: Russia Breaks Out the Big Guns," *Association of the United States Army*, Jan. 2019.

[3] D. Clarke, *World War I Battlefield Artillery Tactics*. Bloomsbury Publishing, 2014.

[4] Department of the Army, "FM 6-121: Tactics, Techniques, and Procedures for Field Artillery Target Acquisition," Sept. 1990.

[5] W. Van der Kloot, "Lawrence Bragg's role in the development of sound-ranging in World War I," *Notes and Records of the Royal Society*, vol. 59, no. 3, pp. 273–284, 2005.

[6] Saab AB, "Arthur - Weapon Locating System," 2018.

[7] L. Dicander, G. Kihlström, and P. Lundgren, "Framgångsfaktorer i ett lyckat samarbetsprojekt mellan Norge och Sverige," *Försvarets Historiska Telesamlingar, Armén*, Mar. 2009.

[8] Saab AB, "Giraffe AMB - Air Surveillance," 2018.

[9] W. Fishbein, "Firefinder, a Radar Forty Years in the Making," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 2, pp. 817–829, 2008.

[10] H. Meikle, *Modern Radar Systems*. Artech House, second ed., 2008.

[11] PIT-RADWAR SA, "WLR-100 Weapon Locating Radar LIWIEC," 2018.

[12] What Is India Publishers (P) Ltd., "BEL Weapon Locating Radars @ whatisindia.com Thursday, December 14, 2006 19:14:15 IST," Dec. 2006.

[13] OCCAR, "COBRA," 2018.

[14] Israel Aerospace Industries Ltd., "ELM-2084 - MRR," 2018.

[15] V. C. Ravindra, Y. Bar-Shalom, and P. Willett, "Projectile Identification and Impact Point Prediction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 4, pp. 2004–2021, 2010.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[17] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," *Data mining and knowledge discovery handbook*, pp. 875–886, 2009.

[18] C. X. Ling, J. Huang, and H. Zhang, "AUC: A Better Measure than Accuracy in Comparing Learning Algorithms," in *Advances in Artificial Intelligence: 16th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2003, Halifax, Canada, June 11-13, 2003, Proceedings* (Y. Xiang and B. Chaib-draa, eds.), (Berlin, Heidelberg), pp. 329–341, Springer, 2003.

[19] B. Hanczar, J. Hua, C. Sima, J. Weinstein, M. Bittner, and E. R. Dougherty, "Small-sample precision of roc-related estimates," *Bioinformatics*, vol. 26, no. 6, pp. 822–830, 2010.

[20] D. J. Hand, "Measuring classifier performance: a coherent alternative to the area under the ROC curve," *Machine learning*, vol. 77, no. 1, pp. 103–123, 2009.

[21] J. M. Lobo, A. Jiménez-Valverde, and R. Real, "AUC: a misleading measure of the performance of predictive distribution models," *Global Ecology and Biogeography*, vol. 17, no. 2, pp. 145–151, 2008.

[22] P. Simon, *Too Big to Ignore: The Business Case for Big Data*. John Wiley & Sons, Mar. 2013.

[23] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[24] S. Marsland, *Machine Learning: An Algorithmic Perspective*. Chapman and Hall/CRC, 2014.

[25] P. A. Vikhar, "Evolutionary Algorithms: A Critical Review and its Future Prospects," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pp. 261–265, Dec. 2016.

[26] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2013.

[27] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] K. Cho, B. v. Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv preprint arXiv:1406.1078*, 2014.

[29] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1994.

[30] R. Lippmann, "An introduction to computing with neural nets," *IEEE Assp magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[31] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[32] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[33] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data*, pp. 785–794, ACM, Aug. 2016.

[34] Q. Lei, J. Yi, R. Vaculin, L. Wu, and I. S. Dhillon, "Similarity Preserving Representation Learning for Time Series Clustering," *arXiv preprint arXiv:1702.03584*, 2017.

[35] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, "Learning Longer Memory in Recurrent Neural Networks," *arXiv preprint arXiv:1412.7753*, 2014.

[36] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[37] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[38] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU neural network performance comparison study: Taking Yelp review dataset as an example," in *2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, pp. 98–101, IEEE, 2020.

[39] J. Van Der Westhuizen and J. Lasenby, "The unreasonable effectiveness of the forget gate," *arXiv preprint arXiv:1804.04849*, 2018.

[40] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, (San Francisco, CA, USA), pp. 1137–1145, Morgan Kaufmann Publishers Inc., 1995.

[41] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Science & Business Media, 2009.

[42] A. C. Davison and D. V. Hinkley, *Bootstrap Methods and their Application*. Cambridge University Press, 1997.

[43] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

[44] B. F. Manly, *Randomization, Bootstrap and Monte Carlo Methods in Biology*. Chapman & Hall/CRC, Taylor & Francis, 2006.

[45] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[46] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," *arXiv preprint arXiv:1012.2599*, 2010.

[47] J. H. Holland, "Genetic Algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[48] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population Based Training of Neural Networks," *arXiv preprint arXiv:1711.09846*, 2017.

[49] F. Tennebø and M. Geitle, "Evaluating Population Based Training on Small Datasets," in *Norsk IKT-konferanse for forskning og utdanning*, 2019.

[50] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning Augmentation Strategies from Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, IEEE, 2019.

[51] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel, "Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 2731–2741, PMLR, 09–15 Jun 2019.

[52] E. Castro, J. S. Cardoso, and J. C. Pereira, "Elastic Deformations for Data Augmentation in Breast Cancer Mass Detection," in *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pp. 230–234, IEEE, 2018.

[53] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[54] M. Arslan, M. Guzel, M. Demirci, and S. Ozdemir, "SMOTE and Gaussian Noise Based Sensor Data Augmentation," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pp. 1–5, IEEE, 2019.

[55] B. Neyshabur, R. Tomioka, and N. Srebro, "In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning," *arXiv preprint arXiv:1412.6614*, 2014.

[56] A. Hernández-García and P. König, "Data augmentation instead of explicit regularization," *arXiv preprint arXiv:1806.03852*, 2018.

[57] S.-C. Chan and K.-C. Lee, "Radar Target Recognition by MSD Algorithms on Angular-Diversity RCS," *IEEE Antennas and Wireless Propagation Letters*, vol. 12, pp. 937–940, 2013.

[58] A. Register, W. Blair, L. Ehrman, and P. K. Willett, "Using Measured RCS in a Serial, Decentralized Fusion Approach to Radar-Target Classification," in *2008 IEEE Aerospace Conference*, pp. 1–8, IEEE, 2008.

[59] J. S. Kobashigawa, H.-S. Youn, M. F. Iskander, and Z. Yun, "Classification of Buried Targets Using Ground Penetrating Radar: Comparison Between Genetic Programming and Neural Networks," *IEEE Antennas and Wireless Propagation Letters*, vol. 10, pp. 971–974, 2011.

[60] J. Pisane, *Automatic target recognition using passive bistatic radar signals.* PhD thesis, Supélec, Apr. 2013.

[61] I. M. D. Rosa, A. T. Marques, G. Palminha, H. Costa, M. Mascarenhas, C. Fonseca, and J. Bernardino, "Classification success of six machine learning algorithms in radar ornithology," *Ibis*, vol. 158, no. 1, pp. 28–42, 2016.

[62] S.-J. Lee, S.-J. Jeong, B.-S. Kang, H. Kim, S.-M. Chon, H.-G. Na, and K.-T. Kim, "Classification of Shell-Shaped Targets Using RCS and Fuzzy Classifier," *IEEE Transactions on Antennas and Propagation*, vol. 64, no. 4, pp. 1434–1443, 2016.

[63] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[64] T. Taskaya-Temizel and M. C. Casey, "A Comparative Study of Autoregressive Neural Network Hybrids," *Neural Networks*, vol. 18, no. 5-6, pp. 781–789, 2005.

[65] H. Andersson and C. T. Luong, "Classification between birds and UAVs using recurrent neural networks," Master thesis, Chalmers University of Technology, 2020.

[66] J. Strandberg, "Time series classification for identification of radar targets," Master thesis, Chalmers University of Technology, 2015.

[67] J. Rosen, "Behavior Classification based on Sensor Data - Classifying time series using low-dimensional manifold representations," Master thesis, Chalmers University of Technology, 2015.

[68] J. Fieser, "Ethics," in *Internet Encyclopedia of Philosophy*, Internet Encyclopedia of Philosophy, 2021.

[69] Ø. Mikkelsen, S. H. Hartvigsen, K. H. Hauge, S. Nordenson, E. Nordtveit, R. Nydal, M. van der Velden, J. U. Skåre, R. Viste, L. Øvreås, *et al.*, *Guidelines for Research Ethics in Science and Technology*. The National Committee for Research Ethics in Science and Technology, 2. ed., June 2016.

[70] Saab, "Code of Conduct," Aug. 2020.

[71] United Nations, "Charter of the United Nations Article 51," Oct. 1945.

[72] United Nations, "Charter of the United Nations Article 42," Oct. 1945.

[73] J. Moussa, "Can jus ad bellum override jus in bello? Reaffirming the separation of the two bodies of law," *International Review of the Red Cross*, vol. 90, no. 872, pp. 963–990, 2008.

[74] M. Verbruggen and V. Boulanin, "Article 36 reviews: Dealing with the challenges posed by emerging technologies," in *Article 36 reviews and emerging technologies: Exploring the challenges posed by emerging technologies to the legal review of weapons, and means and methods of warfare.*, Stockholm International Peace Research Institute (SIPRI), 2017.

[75] United Nations, *Convention on the Prohibition of the Development, Production and Stockpiling of Bacteriological (Biological) and Toxin Weapons and on their Destruction*. United Nations, 1972.

[76] League of Nations, *Protocol for the Prohibition of the Use in War of Asphyxiating, Poisonous or Other Gases, and of Bacteriological Methods of Warfare.* League of Nations, 1925.

[77] H. S. Truman, "Address in Milwaukee, Wisconsin, October 14, 1948," Oct. 1948.

[78] D. D. Eisenhower, *The White House Years: Mandate for Change, 1953-1956*, vol. 1. Doubleday, 1963.

[79] "Einstein deplores Use of Atom Bomb," *The New York Times*, p. 1, Aug. 1946.

[80] O. Nathan and H. Norden, *Einstein on Peace.* Schocken Books, 1968.

[81] G. Alperovitz, "The War Was Won Before Hiroshima—And the Generals Who Dropped the Bomb Knew It," *The Nation*, Aug. 2015.

[82] European Commission, "Proposal for a regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts," Apr. 2021.

[83] Capgemini, "Code of Ethics for AI," Jan. 2021.

[84] IBM, "Everyday Ethics for Artificial Intelligence," 2019.

[85] Centre for Humanitarian Dialogue, "Code of Conduct on Artificial Intelligence in Military Systems," Aug. 2019.

[86] AI Alliance Russia, "Artificial Intelligence Code of Ethics," Oct. 2021.

[87] D. Omand and M. Phythian, *Principled Spying: The Ethics of Secret Intelligence.* Oxford University Press, 2018.

[88] N. J. Beck, "Espionage and the Law of War," *American Intelligence Journal*, vol. 29, no. 1, pp. 126–136, 2011.

[89] T. O'Connor, "Russia will "take measures" against a U.S. radar nears its border, thought to serve missile defense," *Newsweek*, May 2019.

[90] R. Sims, "How do we understand international law and peace?," Masters thesis, Georgia State University, 2018.

[91] E. V. Carreno, "Address," in *Peace Through International Law: The Role of the International Law Commission. A Colloquium at the Occasion of Its Sixtieth Anniversary* (G. Nolte, ed.), vol. 211, pp. 5–6, Springer Science & Business Media, 2009.

[92] T. Grant, "Building an Ontology for Planning Attacks That Minimize Collateral Damage: Literature Survey," in *14th International Conference on Cyber Warfare and Security: ICCWS 2019*, p. 78, Academic Conferences International Limited, 2019.

[93] D. A. Cafarelli, "Effect of False Alarm Rate on Pilot Use and Trust of Automation Under Conditions of Simulated High Risk," Master thesis, Massachusetts Institute of Technology, 1998.

[94] W. Nelson, "Use of Circular Error Probability in Target Detection," *The MITRE Corporation*, 1988.

[95] M. Rosenberg and J. Markoff, "The Pentagon's 'Terminator Conundrum': Robots That Could Kill on Their Own," *The New York Times*, Oct. 2016.

[96] I. Bode and H. Huelss, "Autonomous weapons systems and changing norms in international relations," *Review of International Studies*, vol. 44, no. 3, pp. 393–413, 2018.

[97] R. C. Arkin, *Governing Lethal Behavior in Autonomous Robots*. Chapman and Hall/CRC, 2009.

[98] R. C. Arkin, "The Case for Ethical Autonomy in Unmanned Systems," *Journal of Military Ethics*, vol. 9, no. 4, pp. 332–341, 2010.

[99] R. G. Hewlett and O. E. Anderson, *The New World 1939/1946: A History of the United States Atomic Energy Commission*, vol. 1. The Pennsylvania State University Press, 1962.

[100] K. Quach, "Googlers revolt over AI military tech contract, brainiacs boycott killer robots, and more," *The Register*, 2018.

[101] K. Conger, "Google Employees Resign in Protest Against Pentagon Contract," *Gizmodo*, May 2018.

[102] K. Anderson and M. C. Waxman, "Law and Ethics for Autonomous Weapon Systems: Why a Ban Won't Work and How the Laws of War Can," tech. rep., Stanford University, The Hoover Institution (Jean Perkins Task Force on National Security and Law Essay Series), 2013.

[103] S. Russell, D. Dewey, and M. Tegmark, "Research Priorities for Robust and Beneficial Artificial Intelligence," *AI Magazine*, vol. 36, no. 4, pp. 105–114, 2015.

[104] R. P. Masini, M. C. Medeiros, and E. F. Mendes, "Machine learning advances for time series forecasting," *arXiv preprint arXiv:2012.12802*, 2020.

[105] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press, 2011.

[106] D. Gunning, "Broad Agency Announcement — Explainable Artificial Intelligence (XAI)," tech. rep., DARPA, 2016.

[107] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT Press, 1992.

[108] R. Olsson, "Inductive functional programming using incremental program transformation," *Artificial intelligence*, vol. 74, no. 1, pp. 55–81, 1995.

[109] M. Hofmann, "Igorii-an analytical inductive functional programming system (tool demo," in *In PEPM*, Citeseer, 2010.

[110] S. Katayama, "Systematic search for lambda expressions.," *Trends in functional programming*, vol. 6, pp. 111–126, 2005.

[111] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.